Two-Round Threshold Lattice Signatures from Threshold Homomorphic Encryption

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

[1] Department of Computer Science
University of Maryland
dgur1@cs.umd.edu, jkatz2@gmail.com

[2] Department of Information Security and Communication Technology
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. based on lattices that support only the full threshold case (i.e., $t = n$).

We show here a two-round threshold signature scheme based on standard lattice assumptions that support arbitrary thresholds $t \leq n$. Estimates of our scheme's performance at the 128-bit security level with a trusted setup show that in the 3-out-of-5 case, we obtain signatures of size 11.5 KB and public keys of size 13.6 KB, with an execution of the signing protocol using roughly 1.5 MB of communication per party. We achieve improved parameters if only a small bounded number of signatures are ever issued with the same key.

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

Keywords: Lattice-Based Cryptography · Threshold Signatures · Threshold Homomorphic Encryption · Zero-Knowledge Proofs · Active Security

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 having

⋆ Portions of this work were done while at Dfns Labs.

⋆⋆ Work done in part while visiting the University of Maryland.

been focused on threshold versions of the ECDSA [GGN16,GG18,LN18,DKLs19, DOK$^+$20, CGG$^+$20, CCL$^+$20, DJN$^+$20] and Schnorr-like schemes [KG20, Lin22, CGRS23, CKM23]. Based in part on this level of interest, NIST has announced their intention [BP23] to standardize threshold cryptosystems.

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 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 have 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 practical t-out-of-n threshold signature scheme based on standard lattice assumptions (Ring-LWE/SIS) that supports arbitrary thresholds $t \leq n$. As in the scheme of Damgård et al., our scheme features a two-round online signing protocol and allows for distributed key generation. Estimates of our scheme's performance at the 128-bit security level with a trusted setup show that in the 3-out-of-5 case, we obtain signatures of size 11.5 KB and public keys of size 13.6 KB, where execution of the signing protocol uses roughly 1.5 MB of communication per party. We can also improve the signature size by roughly a factor of 7.2$\times$ in settings where the number of signatures generated using a single key is bounded in advance; we refer to Section 6 for further details.

Our instantiations are competitive with the trivial solution of concatenating n public keys and t signatures while offering several advantages in practice, e.g., the anonymity of the signing threshold towards the public. Our constructions are also more in line with the formal definition of a threshold signature scheme where there exists one public key and one signature for the threshold rather than a set; therefore, the verification algorithm for the base scheme can be used without knowing whether a single user or multiple users generated the signature.

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 the first actively secure t-out-of-n threshold key-generation and decryption protocols for a lattice-based (linearly) homomorphic encryption scheme. Our construction relies on the BGV homomorphic encryption scheme combined with (verifiable) Shamir secret sharing and recent lattice-based non-interactive zero-knowledge proofs.

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 because this is intended only to provide intuition.

We 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 and 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 the 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 the possibility of aborts, is to increase parameters [GKPV10]. 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 the FSwA approach creates difficulties.

Damgård et al. propose a way to distribute the FSwA version of the above scheme based on the following idea: say there are n signers, and the ith signer holds (short) vector $\mathbf{s}_i$ where $\mathbf{s} = \sum_{i\in[n]} \mathbf{s}_i$ 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 $(c, \boldsymbol{z})$.

[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.

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 t-out-of-n fashion using, e.g., Shamir secret sharing. (Shamir's scheme can be generalized to many commutative rings with identity [ACD+19].) 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 the approach of increasing parameters to prevent 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.

Instead, 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 such scheme and show in Section 3 an instantiation based on standard lattice assumptions obtained by adapting the BGV scheme [BGV12]. We build on a version of the three-message identification protocol described above that uses larger parameters and does not require aborts. The master private key $\mathbf{s}$ and public 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 will all hold encryption $\mathsf{ctx}_{\mathbf{s}} = \mathsf{Enc}(\mathbf{s})$ of $\mathbf{s}$ (with respect to some known public key) and share the corresponding decryption key $\boldsymbol{k}$ in a t-out-of-n fashion. Any set $\mathcal{U} \subseteq [n]$ of t parties can now generate a signature (in the semi-honest setting) as follows:

1. For $i \in \mathcal{U}$, the ith signer chooses a (short) vector $\boldsymbol{r}_i \in R_q^{\ell+k}$ and sends $\boldsymbol{w}_i := \bar{\mathbf{A}}\boldsymbol{r}_i$. 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 sends its threshold decryption share of $\mathsf{ctx}_{\boldsymbol{z}}$.
3. Given decryption shares from all parties in $\mathcal{U}$, each signer can decrypt $\mathsf{ctx}_{\boldsymbol{z}}$ 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, see Section 4, additional work is needed to handle malicious adversaries while achieving a two-round online 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 signature scheme [GPV08]. Their protocol uses generic secure multiparty computation to distributively compute the most expensive part of the scheme (namely, Gaussian sampling [Pei10]), and it seems unlikely to yield a practical solution; moreover, their scheme requires $t - 1 < n/2$. Cozzo and Smart [CS19] 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$^+$18] show a "universal thresholdizer" that can be used to create a threshold version of any signature scheme. The basic idea behind their framework is to encrypt the master private key of the underlying signature scheme using a threshold fully homomorphic encryption (FHE) scheme, evaluate the underlying scheme homomorphically, and then use threshold decryption to recover the signature. In general, because their approach relies on FHE to evaluate the circuit for the signing algorithm, this will not lead to an efficient protocol. Agrawal et al. [ASY22] adapted this approach to the specific signature scheme Dilithium-G [DKL$^+$18] and showed how to tolerate adaptive corruptions; their solution still relies on FHE. Our approach is similar in spirit to these approaches but specialized and optimized for a particular underlying signature scheme. In particular, by moving as many steps as possible outside the homomorphic evaluation, we can base our protocol on threshold *linearly* homomorphic—rather than *fully* homomorphic—encryption; besides the efficiency advantages this confers, this also allows us to distribute key generation (something not achieved in [ASY22, BGG$^+$18]).

We have already mentioned the recent 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 also 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 an n-out-of-n threshold signature scheme and they also show how to reduce the number of rounds for signing using pre-processing. Unfortunately, as with the scheme by Damgård et al., it seems difficult to adapt their scheme to support arbitrary thresholds.

Threshold homomorphic encryption. The threshold homomorphic encryption scheme we construct is based on the work of Aranha et al. [ABGS23], which shows how to achieve malicious security for the Bendlin-Damgård scheme [BD10].

Although the Bendlin-Damgård scheme supports arbitrary thresholds, the subsequent work of Aranha et al. only supports the full threshold case.

The threshold FHE scheme of Boneh et al. [BGG$^+$18] lacks an efficient mechanism for proving the correctness of partial decryptions. Recent work by Boudgoust and Scholl [BS23] is similar in spirit to our threshold encryption scheme but has the same drawback; moreover, their encryption scheme only achieves one-way security. (Note that natural approaches for bootstrapping one-wayness to CPA security ruin the homomorphic property of the scheme and/or make it more difficult to give zero-knowledge proofs of correct encryption.)

Rotaru et al. [RST$^+$22] give an actively secure distributed key-generation protocol for lattice-based encryption, but only for the full threshold case. Viand et al. [VKH23] construct a threshold fully homomorphic encryption scheme using generic zero-knowledge proofs. Gentry et al. [GHL22] build verifiable secret sharing for LWE encryption schemes. Still, the zero-knowledge proofs in their construction rely on discrete-logarithm assumptions, so they are not quantum-resilient.

2 Background

Notation. We use boldface lowercase and uppercase letters to denote vectors and matrices of ring elements, respectively. For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same dimension, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes their inner product. We let $[n] = \{1, \dots, n\}$. We use "$\leftarrow$" to denote probabilistic assignment from a distribution or randomized algorithm and use "$:=$" for deterministic assignment.

Polynomial rings. 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 $\left\{-\frac{q-1}{2}, \dots, \frac{q-1}{2}\right\}$ and then viewing f as a vector over $\mathbb{Z}$; thus,

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

with $\alpha_i \in \left\{-\frac{q-1}{2}, \dots, \frac{q-1}{2}\right\}$. 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., $\|\boldsymbol{f}\|_1 = \max_{i \in \{1, \dots, k\}} \{\|f_i\|_1\}$. 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.

2.1 Shamir Secret Sharing over R_q^ℓ

We describe a natural extension of Shamir secret sharing [Sha79] over R_q. In Shamir's original scheme, a secret $s \in \mathbb{Z}_q$ is split into n shares so that at least t

shares are needed to recover s. This is done via an algorithm Share that samples $t-1$ uniform elements $a_1, \ldots, a_{t-1} \in \mathbb{Z}_q$, defines the polynomial $P(X) = s + a_1 X + a_2 X^2 + \cdots + a_{t-1} X^{t-1} \in \mathbb{Z}_q[X]$ such that $P(0) = s$, and computes share $P(x_i)$ for party $\mathcal{P}_i$, where $x_1, \ldots, x_n \in \mathbb{Z}_q^*$ are distinct. Any subset $\mathcal{U}$ of t parties can reconstruct s using an algorithm Rec that computes the Lagrange coefficients $\lambda_{i,\mathcal{U}} = \prod_{j \in \mathcal{U} \setminus \{i\}} \frac{x_j}{x_j - x_i}$ and then computes $s = \sum_{i \in \mathcal{U}} \lambda_{i,\mathcal{U}} \cdot P(x_i)$. No group of fewer than t participants learns anything about s.

We can extend Shamir's secret-sharing scheme to secrets in R_q by sampling each $P(X)$ coefficient from R_q instead of $\mathbb{Z}_q$. However, we need to be careful with the set of evaluation points $\{x_i\}_{i \in [n]}$. We say $\{x_i\}_{i \in [n]}$ is an *exceptional set* if all pairwise differences are invertible. Shamir's scheme can be used as long as R_q contains an exceptional set of size n [ACD+19]. Note that $\mathbb{Z}_q^* \subset R_q$ is an exceptional set in R_q, and in particular, when $n < q$ we can simply set $x_i = i$. We can share secrets in R_q^ℓ by running this scheme ℓ times independently in parallel.

2.2 Hardness Assumptions

We want security against probabilistic polynomial time (PPT) adversaries and say that a problem is ϵ_X hard for an adversary $\mathcal{A}$ if $\mathcal{A}$ has the advantage (that is, likelihood or probability) ϵ_X of breaking (i.e., finding or distinguishing) an instance of problem X. We implicitly also consider the time it takes to achieve this probability, which will always be some polynomial in the input size.

Informally, the Ring Short Integer Solution problem $\mathsf{R\text{-}SIS}_{k,N,q,\beta}$ over an N-dimensional polynomial ring R_q (in the ℓ_2 norm) is to find a set of short ring elements $\{y_i\}_{i \in [k]}$ satisfying $a_1 y_1 + \cdots + a_\ell y_\ell = 0$, where each $a_i \in R_q$ is uniform.

Definition 1. *$\mathcal{A}$ solves the $\mathsf{R\text{-}SIS}_{k,N,q,\beta}$ problem with advantage $\epsilon_{\mathsf{R\text{-}SIS}}$ if*

$$\Pr\left[\begin{array}{c} \{a_i\}_{i \in [k]} \leftarrow R_q; \\ \{y_i\}_{i \in [k]} \leftarrow \mathcal{A}(\{a_i\}) \end{array} : \begin{array}{c} \forall i\colon \|y_i\|_\infty \leq \beta \ \wedge y_i \neq 0 \\ \wedge \sum\limits_{i \in [k]} a_i y_i = 0 \bmod q \end{array} \right] = \epsilon_{\mathsf{R\text{-}SIS}}.$$

The Decision Ring Learning With Errors problem $\mathsf{R\text{-}LWE}_{k,N,q,\chi}$ is to distinguish k instances of $a_i s + e_i$ from uniform (given $\{a_i\}_{i \in [k]}$), where s and the $\{e_i\}_{i \in [k]}$ are sampled from distribution χ.

Definition 2. *Let χ be a distribution over R_q. $\mathcal{A}$ solves the $\mathsf{R\text{-}LWE}_{k,N,q,\chi}$ problem with advantage $\epsilon_{\mathsf{R\text{-}LWE}}$ if*

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

For both of these problems, we sometimes leave parameters implicit.

2.3 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 secret key share sk_j, a public key pk, and (shared) auxiliary data aux.
- $\mathsf{Sign}_{\mathcal{TS}}$ is an interactive protocol run by a set of t users $\mathcal{U}$. Each party begins holding their secret key share, auxiliary data, 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 or an active 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. 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 and 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 $\mathcal{C}_{\mathcal{U}}$ following an honest execution; 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 (concerning 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 $\mathcal{TS}$.

2.4 Homomorphic Trapdoor Commitment Schemes

Let cpp denote fixed parameters that include sets S_μ and S_ρ, as well as a distribution D_ρ over S_ρ. (We let cpp be 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 $\mu \in S_\mu$, samples randomness $\rho \in S_\rho$ according to D_ρ, and outputs a commitment com.

[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.

- Open takes as input a commitment com, message $\mu \in S_\mu$, 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 $\mu \in S_\mu$

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

Hiding: The scheme is *ϵ-hiding* if for $\mu_0, \mu_1 \in S_\mu$ and any PPT adversary $\mathcal{A}$:

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

Binding: The scheme is *ϵ-binding* if for $\mu_0, \mu_1 \in S_\mu$ and any adversary $\mathcal{A}$:

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

Equivocality: The scheme is *equivocal* if for any $\mu \in S_\mu$, the statistical difference between the following distributions is at most $\epsilon_{\mathsf{equivocal}}$:

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

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

A trapdoor commitment scheme over polynomial rings can be constructed from standard assumption based on R-SIS and R-LWE [DOTT21,GPV08,MP12].

2.5 Non-interactive Zero-knowledge Proofs of Knowledge

We use standard definitions of non-interactive zero-knowledge proofs of knowledge (NIZKPoKs) [DDO+01]. An NIZKPoK for an NP language $\mathcal{L}$ with associated relation $\mathcal{R}_\mathcal{L}$ is a tuple of poly-time algorithms $(\mathsf{Setup},\mathsf{Prove},\mathsf{Vrfy})$ 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, π, outputs a result $b \in \{0,1\}$.
- $\mathsf{Extract}$, on input $\mathsf{crs},\mathsf{td}, x, \pi^*$, outputs a witness w^*.

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

Correctness: We require that 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: For any PPT prover running Prove we have an extractor running $\mathsf{Extract}$ such that

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupTD}; \\ \pi^* \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w); \\ w^* \leftarrow \mathsf{Extract}(\mathsf{crs}, \mathsf{td}, x, \pi^*);\end{array} : \begin{array}{c}\mathsf{Vrfy}(\mathsf{crs}, x, \pi^*) = 1 \\ \wedge\ (x, w^*) \in \mathcal{R}_{\mathcal{L}}\end{array}\right] \geq 1 - \epsilon_{\mathsf{extract}},$$

except for a small failure probability $\epsilon_{\mathsf{extract}}$.

Computational Zero-Knowledge: There exists simulator algorithm Sim such that for all PPT adversaries $\mathcal{A}$ the following distributions are computationally indistinguishable except with advantage ϵ_{HVZK}:

$$\left\{\begin{array}{c}\mathsf{crs} \leftarrow \mathsf{Setup};\ x, w \leftarrow \mathcal{A}(\mathsf{crs}); \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{array} : (\mathsf{crs}, x, \pi)\right\}$$

$$\left\{\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupTD};\ x, w \leftarrow \mathcal{A}(\mathsf{crs}); \\ \pi \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, x)\end{array} : (\mathsf{crs}, x, \pi)\right\}$$

where both Prove and Sim output $\perp$ if $(x, w) \notin R_{\mathcal{L}}$.

3 Threshold Homomorphic Encryption

Our threshold signature scheme relies on an underlying threshold homomorphic encryption scheme satisfying several properties. We define the required properties formally and then show how to instantiate a scheme with those properties based on the BGV encryption scheme [BGV12].

3.1 Definitions

A homomorphic encryption scheme $\mathcal{E}$ consists of the following algorithms:

- $\mathsf{KGen}_{\mathcal{E}}$ is a probabilistic algorithm that takes as input a depth bound d and outputs a public encryption key pk and a decryption key sk.
- Enc is a probabilistic algorithm that takes as input a public key pk and a plaintext ptx, and outputs a ciphertext ctx.
- Eval takes as input a circuit F of depth at most d and a list of ciphertexts $\mathsf{ctx}_1, \ldots, \mathsf{ctx}_k$, and outputs either a new ciphertext ctx^* or $\perp$.
- Dec is a deterministic algorithm that takes as input a decryption key sk and ciphertext ctx^*, and outputs either a plaintext ptx or $\perp$.

Notationally, we allow algorithms to take as input a set of plaintexts, ciphertexts, or keys to denote that the algorithms are applied on each input one by one. $\mathcal{E}$ is said to be secure if the following holds:

Correctness: A homomorphic encryption scheme is *correct* with respect to sk if the following holds for all circuits F of depth at most d:

$$\Pr\left[\begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathcal{E}}(d)\\ \{\mathsf{ctx}_i\}_{i\in[k]} \leftarrow \mathsf{Enc}(\mathsf{pk},\{\mathsf{ptx}_i\}_{i\in[k]})\\ \mathsf{ctx}^* \leftarrow \mathsf{Eval}(F,\{\mathsf{ctx}_i\}_{i\in[k]})\end{array} : \mathsf{Dec}(\mathsf{sk},\mathsf{ctx}^*) = F(\{\mathsf{ptx}_i\}_{i\in[k]})\right] = 1.$$

Ciphertext Indistinguishability: A homomorphic encryption scheme is said to be IND-CPA secure if:

$$\Pr\left[\begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathcal{E}}(d)\\ \{\mathsf{ptx}_0,\mathsf{ptx}_1\} \leftarrow \mathcal{A}(\mathsf{pk}); b \leftarrow \{0,1\}\\ \mathsf{ctx} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{ptx}_b); b' \leftarrow \mathcal{A}(\mathsf{ctx})\end{array} : b' = b\right] \leq \frac{1}{2} + \epsilon_{\text{IND-CPA}}.$$

We now extend these notions to accommodate distributed key generation and threshold decryption:

- DKGen is an interactive protocol run by n users, on common input a threshold t and a depth bound d. All honest users abort with output $\bot$, or all outputs a public encryption key pk and a decryption key share sk_j.
- TDec is an interactive protocol run by a set of t users $\mathcal{U}$ in which each user takes as input a ciphertext ctx^* and a decryption key share sk_j, where each party outputs either a partial decryption share ds_j or $\bot$.
- Comb is a deterministic algorithm that takes as input a ciphertext ctx^* and decryption shares $\{\mathsf{ds}_j\}_{j\in\mathcal{U}}$ for $|\mathcal{U}| = t$ and outputs either ptx or $\bot$.

We also assume the existence of an algorithm $\mathsf{Recover}$ that takes as input the public key pk, the threshold t, and a set of decryption key shares $\{\mathsf{sk}_j\}_{j\in\mathcal{U}}$ with $|\mathcal{U}| \geq t$ and outputs either a secret key sk or $\bot$. (This algorithm is never run in the real world but is used in some of our definitions.)

The following definition follows immediately from the non-threshold setting:

Threshold Correctness: A threshold homomorphic encryption scheme is *correct* if the following holds for all circuits F of depth at most d:

$$\Pr\left[\begin{array}{c}(\mathsf{pk},\{\mathsf{sk}_j\}_{j\in[n]}) \leftarrow \mathsf{DKGen}(t,n,d)\\ \mathsf{sk} \leftarrow \mathsf{Recover}(\mathsf{pk},t,\{\mathsf{sk}_j\}_{j\in[n]})\\ \{\mathsf{ctx}_i\}_{i\in[k]} \leftarrow \mathsf{Enc}(\mathsf{pk},\{\mathsf{ptx}_i\}_{i\in[k]})\\ \mathsf{ctx}^* \leftarrow \mathsf{Eval}(F,\{\mathsf{ctx}_i\}_{i\in[k]})\end{array} : \begin{array}{c}\mathsf{Dec}(\mathsf{sk},\mathsf{ctx}^*) = F(\{\mathsf{ptx}_i\}_{i\in[k]}) =\\ \mathsf{Comb}(\mathsf{TDec}(\{\mathsf{sk}_j\}_{j\in\mathcal{U}},\mathsf{ctx}^*,\mathcal{U}))\end{array}\right] \geq 1 - \epsilon_{\mathsf{corr}}.$$

We then define the two key notions of security: *key generation security* and *threshold security*, adapted from [DDE+23]:

Key Generation Security A threshold homomorphic encryption scheme is said to have secure key generation if it securely implements functionality $\mathcal{F}_{\mathsf{DKGen}}$ from Figure 1.

Threshold Security A threshold homomorphic encryption scheme is said to be secure if it securely implements functionality $\mathcal{F}_{\mathsf{TDec}}$ in Figure 2.

```
$\underline{\mathcal{F}_{\mathsf{DKGen}}}$
Execute $\mathsf{pk}, \mathsf{sk} \leftarrow \mathsf{KGen}_{\mathcal{E}}(d)$.
Receive set of shares $\mathsf{sk}_j$ for each corrupted party $j \in \mathcal{C}$.
Construct a complete set of key shares $\mathsf{sk}_i$ for $i \in \mathcal{H}$ based on
    $\mathsf{sk}_j$ such that every subset of $\mathsf{sk}_j$ of size $t$ is equal to $\mathsf{sk}$.
Send $\mathsf{pk}$ to all players, and $\mathsf{sk}_i$ to player $i$.
```

Fig. 1. Functionality $\mathcal{F}_{\mathsf{DKGen}}$.

```
$\underline{\mathcal{F}_{\mathsf{TDec}}}$
Receive a set of users $\mathcal{U}$ where $|\mathcal{U}| \geq t$ and a set of key shares
    $\mathsf{pk}, \mathsf{sk}_j$ and ciphertext $\mathsf{ctx}$ for each party $j$ in $\mathcal{U}$.
Send $\perp$ to every party if not all $\mathsf{pk}$ are the same and abort.
Send $\perp$ to every party if not all $\mathsf{ctx}$ are the same and abort.
Send $\perp$ to every party if $\perp = \mathsf{Recover}(\mathsf{pk}, t, \{\mathsf{sk}_j\}_{j \in \mathcal{U}})$ and abort.
Compute $\mathsf{sk} := \mathsf{Recover}(\mathsf{pk}, t, \{\mathsf{sk}_j\}_{j \in \mathcal{U}})$
Compute $\mathsf{ptx} := \mathsf{Dec}(\mathsf{sk}, \mathsf{ctx})$.
Send $\mathsf{ptx}$ to the adversary.
```

Fig. 2. Functionality $\mathcal{F}_{\mathsf{TDec}}$.

3.2 The BGV Scheme

Our scheme will be based on the BGV encryption scheme [BGV12], augmented with maliciously secure key-generation and threshold decryption protocols. We begin by reviewing the BGV encryption scheme. Let $p \ll q$ be prime numbers, let R_q and R_p be as in Section 2 for some fixed 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 $a \in R_q$ along with $s, e \leftarrow D_{\mathsf{KGen}}$, and output public key $\mathsf{pk} := (a, b) = (a, as + pe)$ and secret key $\mathsf{sk} := s$.
- $\mathsf{Enc}_{\mathsf{BGV}}$: On input a public key $\mathsf{pk} = (a, b)$ and a message $m \in R_p$, sample $r, e', e'' \leftarrow D_{\mathsf{Enc}}$ and output the ciphertext $(u, v) = (ar + pe', br + pe'' + m)$.
- $\mathsf{Dec}_{\mathsf{BGV}}$: On input a secret key $\mathsf{sk} = s$ and a ciphertext (u, v), output the message $\mathsf{ptx} := (v - su \bmod q) \bmod p$.

The scheme is correct if and only if $\|v - su\|_\infty \leq B_{\mathsf{Dec}} < q/2$ and secure against *chosen plaintext attacks* if the $\mathsf{R\text{-}LWE}_{N,q,\beta}$ problem is hard for β depending on the choice of $B_{\mathsf{KGen}}, B_{\mathsf{Enc}}, N, q, p$. Denote $\beta := \beta(B_{\mathsf{KGen}}, B_{\mathsf{Enc}}, N, q, p)$.

3.3 Zero-Knowledge Proofs: From Passive to Active Security

The security against malicious adversaries in our protocol relies on commitments[3] and non-interactive zero-knowledge proofs of knowledge, which we formally define in Sections 2.4 and 2.5. These proofs are explicit for distributed key generation, encryption, and threshold decryption. We describe the required relations for each algorithm:

- The proof π_{sk_i} proves knowledge of a *short* key s'_i and a *short* error e'_i corresponding to some b_i and correct t-out-of-n secret sharing of these secrets. Furthermore, it proves that the short secrets are correct t-out-of-n secret shared. For publicly fixed a, B_{DKGen} and p and public $b_i, \{\bar{b}_{i,j}\}$ and commitments to $s'_i, e'_i, \{\bar{s}_{i,j}, \bar{e}_{i,j}\}_{j\in[n]}$ the relation $\mathcal{R}_{\mathsf{sk}}$ shows: (1) The secrets s'_i, e'_i have norm smaller than B_{DKGen}, (2) b_i is correctly computed with respect to private s'_i and e'_i, (3) $\{\bar{s}_{i,j}\}_{j\in[n]}$ are correct t-out-of-n shares of s'_i (4) $\{\bar{e}_{i,j}\}_{j\in[n]}$ are correct t-out-of-n shares of e'_i (5) $\{b_{i,j}\}_{j\in[n]}$ is correctly computed with respect to $\bar{s}_{i.j}$ and $\bar{e}_{i,j}$. We define the relation $\mathcal{R}_{\mathsf{sk}}$:

$$\mathcal{R}_{\mathsf{sk}} := \left\{ (x,w) \,\middle|\, \begin{array}{c} x := (b_j, \{\bar{b}_{i,j}\}, \mathsf{com}_{s'_j}, \mathsf{com}_{e'_j}, \{\mathsf{com}_{\bar{s}_{i,j}}\}_{i\in[n]}, \{\mathsf{com}_{\bar{e}_{i,j}}\}_{i\in[n]}) \wedge \\ w := (s'_j, \rho_{s'_j}, e'_j, \rho_{e'_j}, \{\bar{s}_{i,j}\}_{i\in[n]}, \{\rho_{i,j}\}_{i\in[n]}, \{\bar{e}_{i,j}\}_{i\in[n]}, \{\bar{\rho}_{i,j}\}_{i\in[n]}) : \\ \\ \|s'_j, e'_j\| \le B_{\mathsf{DKGen}} \wedge b_j = as'_j + pe'_j \wedge \forall i\ \bar{b}_{i,j} = a\bar{s}_{i,j} + p\bar{e}_{i,j} \wedge \\ b_j = \mathsf{Rec}(\{\bar{b}_{i,j}\}) \wedge s'_j = \mathsf{Rec}(\{\bar{s}_{i,j}\}) \wedge e'_j = \mathsf{Rec}(\{\bar{e}_{i,j}\}) \\ \wedge \mathsf{Open}(\mathsf{com}_{s'_j}, s'_j, \rho_{s'_j}) \wedge \mathsf{Open}(\mathsf{com}_{e_j}, e_j, \rho_{e_j}) \wedge \\ \forall i\ \mathsf{Open}(\mathsf{com}_{\bar{s}_{i,j}}, \bar{s}_{i,j}, \rho_{\bar{s}_{i,j}}) \wedge \forall i\ \mathsf{Open}(\mathsf{com}_{\bar{e}_{i,j}}, \bar{e}_{i,j}, \rho_{\bar{e}_{i,j}}) \end{array} \right\}.$$

- The proof π_{ctx} in encryption proves that the given ciphertext $\mathsf{ctx} = (u, v)$ corresponds to a plaintext m of the specific norm and is correctly generated using the encryption algorithm. For publicly fixed a, b, B_{Enc} and p and public commitments to secret m, r, e', e'', the relation $\mathcal{R}_{\mathsf{Enc}}$ shows: (1) The secrets r, e', e'' have norm smaller than B_{Enc}, (2) The message m has a norm smaller than p, (3) u is computed correctly with respect to a, r, e, p, (4) v is computed correctly with respect to b, r, p, e'', m. We define the relation $\mathcal{R}_{\mathsf{Enc}}$:

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

[3] Here, we use the standard definition of commitments without trapdoors. This involves algorithms $\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}$ satisfying *hiding* and *binding*.

– The proof π_{ds_i} of valid threshold decryption proves that the correct secret key share $\mathsf{sk}_i = s_i$ was used to generate the public decryption share ds_i for a given ciphertext $\mathsf{ctx} = (u, v)$. For publicly fixed $a, b, \lambda_i, p, B_{\mathsf{TDec}} := 2^{\mathsf{sec}} B_{\mathsf{Dec}}$ and public commitments to secret s_i, E_i the relation $\mathcal{R}_{\mathsf{ds}}$ shows: (1) The secret error E_i has norm smaller than B_{TDec}, (2) ds_i is correctly computed with respect to $\lambda_i, s_i, u, p, E_i$. We define the relation $\mathcal{R}_{\mathsf{ds}}$:

$$\mathcal{R}_{\mathsf{ds}} := \left\{ (x, w) \;\middle|\; \begin{array}{c} x := (\mathsf{ds}_j, \mathsf{ctx}, \mathsf{com}_{\mathsf{sk}_j}, \mathsf{com}_{E_j}) \;\wedge\; w := (\mathsf{sk}_j, \rho_{\mathsf{sk}_j}, E_j, \rho_{E_j}) : \\ \\ \|E_j\|_\infty \le B_{\mathsf{TDec}} \;\wedge\; \mathsf{ds}_j = \lambda_j s_j u + pE_j \;\wedge \\ \mathsf{Open}(\mathsf{com}_{\mathsf{sk}_j}, \mathsf{sk}_j, \rho_{\mathsf{sk}_j}) \;\wedge\; \mathsf{Open}(\mathsf{com}_{E_j}, E_j, \rho_{E_j}) \end{array} \right\}.$$

NIZK proofs for these relations can be instantiated using the BDLOP commitment scheme [BDL+18] together with its zero-knowledge proofs of opening and linear relation combined with a proof of shortness by Lyubashevsky et al. [BLNS21, LNP22].

To achieve concurrent security for distributed key generation and threshold decryption, we need zero-knowledge proofs that are straight-line extractable. We can extend the proof systems for proving linear relations or of bounded values using the generic but efficient transform by Katsumata [Kat21] to go from rewindable proof systems to proofs that achieve straight-line extractability.

3.4 Distributed Key Generation for BGV

We propose a new t-out-of-n distributed key generation protocol using Shamir secret sharing secure against a malicious adversary corrupting $|\mathcal{C}| \le t-1$ parties.

The intuition is that our protocol is similar to publicly verifiable secret sharing (PVSS) based on discrete logarithms. To t-out-of-n secret share a decryption key s between n parties, each party individually t-out-of-n secret shares their individual s_i and e_i and derives the final sharing of s based on shares received by other parties. Unlike traditional PVSS however, we rely on commitments and zero-knowledge proofs in a commit-then-prove fashion to verify each share is correctly computed. We describe the protocol depicted in Figure 3 from the viewpoint of party $\mathcal{P}_i$ for some $i \in [n]$:

1. $\mathcal{P}_i$ samples an element a_i uniformly from R_q and computes the hash $h_{a_i} := H_0(a_i)$ as a commitment and broadcasts the hash. After receiving h_{a_j} from all $j \ne i$, it broadcasts a_i. After receiving a_j from all $j \ne i$, it verifies a_j using h_{a_j}, and aborts with output j if $h_{a_j} \ne H_0(a_j)$. Otherwise, it defines $a := \sum a_j$ as a part of the public key.

2. $\mathcal{P}_i$ samples decryption key share s'_i and noise share e'_i from a distribution D_{KGen}. $\mathcal{P}_i$ then sets the key $b_i := as'_i + pe'_i$. $\mathcal{P}_i$ commits to b_i as the hash $h_{b_i} := H_1(b_i)$, broadcasts it, and receives h_{b_j} from all $j \ne i$.

```
$\underline{\mathsf{DKGen}(t, n, d)}$
$\boldsymbol{a}_i \leftarrow R_q, \quad h_{a_i} := H_0(a_i)$
                                        $\xrightarrow{\quad h_{a_i} \quad}$
                                        $\xleftarrow{\quad \{h_{a_j}\}_{j \neq i} \quad}$
                                        $\xrightarrow{\quad a_i \quad}$
                                        $\xleftarrow{\quad \{a_j\}_{j \neq i} \quad}$
**if any** $h_{a_j} \neq H_0(a_j)$: **abort** $(j)$
$a := \sum_{j \in [n]} a_j, \quad s'_i, e'_i \leftarrow D_{\mathsf{KGen}},$
$b_i = as'_i + pe'_i \quad h_{b_i} := H_1(b_i)$
                                        $\xrightarrow{\quad h_{b_i} \quad}$
                                        $\xleftarrow{\quad \{h_{b_j}\}_{j \neq i} \quad}$
$\rho_{s'_i}, \rho_{e'_i}, \rho_{\bar{s}_{i,j}}, \rho_{\bar{e}_{i,j}} \leftarrow S_\rho$
$\{\bar{s}_{i,j}\}_{j \in [n]} \leftarrow \mathsf{Share}_{t,n}(s'_i), \{\bar{e}_{i,j}\}_{j \in [n]} \leftarrow \mathsf{Share}_{t,n}(e'_i)$
$\bar{b}_{i,j} := a\bar{s}_{i,j} + p\bar{e}_{i,j}$
$\mathsf{com}_{\bar{s}_{i,j}} := \mathsf{Com}(\bar{s}_{i,j}; \rho_{\bar{s}_{i,j}}), \mathsf{com}_{\bar{e}_{i,j}} := \mathsf{Com}(\bar{e}_{i,j}; \rho_{\bar{e}_{i,j}})$
Compute $\pi_{\mathsf{sk}_j}$ according to Section 3.3
                                        $(\mathsf{com}_{s'_i}, \mathsf{com}_{e'_i}, \{\mathsf{com}_{\bar{s}_{i,j}}, \mathsf{com}_{\bar{e}_{i,j}}\}_{i \in [n]},$
                                        $\xrightarrow{\quad b_i, \{\bar{b}_{i,j}\}_{j \in [n]}, [\bar{s}_{i,j}, \rho_{\bar{s}_{i,j}}]_j) \quad}$
                                        $\{(\mathsf{com}_{s'_j}, \mathsf{com}_{e'_j}, \{\mathsf{com}_{\bar{s}_{j,k}}, \mathsf{com}_{\bar{e}_{j,k}}\}_{k \in [n]},$
                                        $\xleftarrow{\quad b_j, \{\bar{b}_{j,k}\}_{k \in [n]}, [\bar{s}_{j,i}, \rho_{\bar{s}_{j,i}}]_j)\}_{j \neq i} \quad}$
**if any** $h_{b_j} \neq H_1(b_j)$: **abort** $(j)$
**if any** $b_j \neq \mathsf{Rec}_{t,n}(\{\bar{b}_{j,k}\}_{k \in [n]})$: **abort** $(j)$
**if any** $0 = \mathsf{Open}(\mathsf{com}_{\bar{s}_{j,i}}, \bar{s}_{j,i}, \rho_{\bar{s}_{j,i}})$: **abort** $(j)$
**if any** $\pi_{\mathsf{sk}_j}$ **is invalid**: **abort** $(j)$
$b := \sum_{j \in [n]} b_j, \quad s_i := \sum_{j \in [n]} \bar{s}_{j,i},$
$\rho_{s_i} := \sum_{j \in [n]} \rho_{\bar{s}_{j,i}}, \quad \mathsf{com}_{sk_j} := \sum_{j \in [n]} \mathsf{com}_{\bar{s}_{j,i}}$
**return** $\mathsf{pk} := (a, b, \{\mathsf{com}_{\mathsf{sk}_j}\}), \quad \mathsf{sk}_i := (s_i, \rho_i)$
```

Fig. 3. Actively secure key generation protocol for signer $\mathcal{S}_i$. The elements in square brackets with subscript j denote they are sent to $\mathcal{P}_j$ through private channels.

3. $\mathcal{P}_i$ commits to s'_i as $\mathsf{com}_{s'_i}$ using randomness $\rho_{s'_i}$ and to e'_i as $\mathsf{com}_{e'_i}$ using randomness $\rho_{e'_i}$. $\mathcal{P}_i$ then secret shares s'_i and e'_i using t-out-of-n secret sharing and computes $\bar{b}_{i,j} := a\bar{s}_{i,j} + p\bar{e}_{i,j}$ for all $j \in [n]$.

4. $\mathcal{P}_i$ also commits to all individual key shares as $\mathsf{com}_{\bar{s}_{i,j}} := \mathsf{Com}(\bar{s}_{i,j}; \rho_{\bar{s}_{i,j}})$ and $\mathsf{com}_{\bar{e}_{i,j}} := \mathsf{Com}(\bar{e}_{i,j}; \rho_{\bar{e}_{i,j}})$ for all $j \in [n]$.

5. $\mathcal{P}_i$ then computes a proof π_{sk_j} according to the relation $\mathcal{R}_{\mathsf{sk}}$ and broadcasts the proof π_{sk_j} and all $\mathsf{com}_{s'_i}, \mathsf{com}_{e'_i}, \{\mathsf{com}_{\bar{s}_{i,j}}\}, \{\mathsf{com}_{\bar{e}_{i,j}}\}$ together with b_i and $\{\bar{b}_{i,j}\}_{j\in[n]}$ for all $j \in [n]$.

6. For each j, $\mathcal{P}_i$ sends $\bar{s}_{i,j}$ and $\rho_{\bar{s}_{i,j}}$ to party $\mathcal{P}_j$ over a secure channel.

7. Upon receiving $b_j, h_{b_j}, \pi_{\mathsf{sk}_j}, \{\bar{b}_{j,k}\}, \bar{s}_{j,i}$ and all the commitments from all $j \neq i$, $\mathcal{P}_i$ verifies that all $h_{b_j} = H_1(b_j)$ and $b_j = \mathsf{Rec}_{t,n}(\{\bar{b}_{j,k}\}_{k\in[n]})$ where Rec is the reconstruction algorithm for the secret sharing in Section 2.1, and aborts with output j otherwise.

8. $\mathcal{P}_i$ then verifies all proofs π_{sk_j}, checks that $\mathsf{Open}(\mathsf{com}_{j,i}, \bar{s}_{j,i}, \rho_{j,i})$ is valid, and aborts with output j if any of them fails.

9. Otherwise $\mathcal{P}_i$ computes $b := \sum b_j$, $s_i = \sum \bar{s}_{j,i}$, $\rho_{s_i} = \sum \rho_{s_{j,i}}, \mathsf{com}_{sk_j} := \sum \mathsf{com}_{\bar{s}_{j,i}}$ for all $j \in [n]$. $\mathcal{P}_i$ outputs $\mathsf{pk} = (a, b, \{\mathsf{com}_{\mathsf{sk}_j}\})$ and $\mathsf{sk}_i = (s_i, \rho_{s_i})$.

Using a simulation argument, we now prove that the distributed key generation is secure against a static active adversary $\mathcal{A}$ corrupting at most $|\mathcal{C}| \leq t-1$ parties. We show that the protocol securely implements the functionality $\mathcal{F}_{\mathsf{DKGen}}$ and that the difference in distribution between the communication in the protocol and the simulation reduces to the security of the underlying proofs and hardness assumptions. We assume a rushing adversary where honest users always publish their messages first.

Theorem 1 (Key Generation Security). *Let the* $\mathsf{R\text{-}LWE}_{N,q,\beta}$ *be* $\epsilon_{\mathsf{R\text{-}LWE}}$*-hard for* $\beta = \beta(B_{\mathsf{DKGen}}, N, q, p)$*. For* $i = 0, 1$ *let* Q_{H_i} *denote the number of queries made to the random oracle* H_i *and let* ℓ_i *be the bit-length of the oracle output. Let* Com *be* $\epsilon_{\mathsf{hiding}}$ *hiding. Finally, let proof system* Π_{sk} *for the relation* $\mathcal{R}_{\mathsf{sk}}$ *be* $\epsilon_{\mathsf{HVZK}_{\mathsf{sk}}}$ *and* $\epsilon_{\mathsf{extract}_{\mathsf{sk}}}$ *secure. Then, the scheme implements* $\mathcal{F}_{\mathsf{DKGen}}$ *with computational security in the programmable random oracle model against a static active adversary* $\mathcal{A}$ *corrupting at most* $t-1$ *parties, and the advantage of* $\mathcal{A}$ *is:*

$$\epsilon_{\mathsf{DKGen}} \leq \epsilon_{\mathsf{HVZK}_{\mathsf{sk}}} + 2 \cdot (|\mathcal{H}| + 1) \cdot \epsilon_{\mathsf{hiding}} + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract}_{\mathsf{sk}}} + \epsilon_{\mathsf{R\text{-}LWE}} + \frac{Q_{H_0}}{2^{\ell_0}} + \frac{Q_{H_1}}{2^{\ell_1}}.$$

Proof. We define the simulator $\mathsf{Sim}_{\mathsf{DKGen}}$ interacting with $\mathcal{A}$ and controlling the honest parties as follows:

1. Simulator receives a set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| < t$.

2. Simulator receives a BGV public key (a, b), and chooses $i' \in \mathcal{H}$. For $i \neq i' \in \mathcal{H}$, the simulator executes honest parties like the protocol execution.
3. Simulator samples $h_{a_{i'}} \leftarrow \{0,1\}^{\ell_0}$ and sends out h_{a_i} for $i \in \mathcal{H}$. After receiving h_{a_j} for $j \in \mathcal{C}$ from the adversary, simulator searches for a_j such that $h_{a_j} = H_0(a_j)$. Then simulator sets $a_{i'} := \sum\limits_{i \neq i' \in [n]} a_i$ and programs H_0 such that $H_0(a_{i'}) = h_{a_{i'}}$. If programming fails, the simulator aborts. Otherwise, it sends a_i for $i \in \mathcal{H}$.
4. After receiving a_j for $j \in \mathcal{C}$ from the adversary, if $h_{a_j} \neq H_0(a_j)$ for any j, simulator aborts with output j. Simulator then samples $h_{b_{i'}} \leftarrow \{0,1\}^{\ell_1}$ and sends out h_{b_i} for $i \in \mathcal{H}$. After receiving h_{b_j} for $j \in \mathcal{C}$ from the adversary, simulator searches for b_j such that $h_{b_j} = H_1(b_j)$. Simulator then derives $b_{i'} := \sum\limits_{i \neq i' \in [n]} b_i$ and programs H_1 such that $H_1(b_{i'}) = h_{b_{i'}}$. If programming fails, the simulator aborts. Otherwise it t-out-of-n secret shares $b_{i'}$ to obtain $\bar{b}_{i',j}$ for $j \in [n]$. For each $j \in \mathcal{C}$, the simulator derives random $\bar{s}_{i,j}$ and $\bar{e}_{i,j}$ such that $a\bar{s}_{i,j} + p\bar{e}_{i,j}$.
5. Simulator commits to each $\bar{s}_{i,j}$,$\bar{e}_{i,j}$ using randomness $\rho_{\bar{s}_{i,j}}$,$\rho_{\bar{e}_{i,j}}$ for $j \in \mathcal{C}$. For i in $\mathcal{H} \setminus \{i'\}$ and commitment to $s'_{i'}, e'_{i'}$, the simulator samples random commitments from the commitment space. Then, calls the simulator $\mathsf{Sim}_{\mathsf{sk}}$ to obtain a proof $\pi_{\mathsf{sk}_{i'}}$. Sends b_i, h_{b_i},$\{\bar{b}_{i,j}\}_{j \in [n]}$, $\{\bar{s}_i, j, \rho_{\bar{s}_{i,j}}\}_{j \in \mathcal{C}}$ for $i \in \mathcal{H}$ to the adversary.
6. After receiving the corresponding information, the simulator does the final checks as in the protocol. The simulator calls the extractor $\mathsf{Extract}_{\mathsf{sk}}$ to extract s'_j from π_{sk_j} for $j \in \mathcal{C}$ and aborts if any of them fails. Otherwise secret shares s'_j according to protocol to derive $\{\bar{s}_{j,i}\}_{i \in [n]}$ and derive s_j for $j \in \mathcal{C}$. The simulator finally outputs (a, b) as the public key.

Correctness of the simulator is immediate as the final public key a, b is retrieved from BGV as output, and since there are $n > t$ shares total for s'_j, $j \in \mathcal{C}$ it is possible to reconstruct and re-share s'_j to obtain s_j. The security of the protocol follows by showing the output of the simulator is computationally indistinguishable from the actual protocol execution. We show this through a series of hybrid experiments.

$\boldsymbol{G}_0$*:* The first experiment corresponds to the real world:

1. $\mathcal{A}$ sends a set of corrupted parties $\mathcal{C}$.
2. For $i \in [n] \backslash \mathcal{C} = \mathcal{H}$, compute a_i, h_{a_i} according to the protocol, send $\{h_{a_i}\}_{i \in \mathcal{H}}$ to $\mathcal{A}$. Receive $\{h_{a_j}\}_{j \in \mathcal{C}}$ from $\mathcal{A}$. Send $\{a_i\}_{i \in \mathcal{H}}$ to $\mathcal{A}$.
3. Receive $\{a_j\}_{j \in \mathcal{C}}$ from $\mathcal{A}$, if $h_{a_j} \neq H_0(a_j)$ abort. Otherwise, compute a according to the protocol.
4. Sample s'_i, e'_i, $\rho_{s'_i}$ and $\rho_{e'_i}$, and compute b_i, h_{b_i} according to the protocol. Send $\{h_{b_i}\}_{i \in \mathcal{H}}$ to $\mathcal{A}$.
5. Receive $\{h_{b_j}\}_{j \in \mathcal{C}}$ from $\mathcal{A}$. Sample $\rho_{\bar{s}_{i,j}}$, $\rho_{\bar{e}_{i,j}}$ and compute $\bar{s}_{i,j}$, $\bar{e}_{i,j}$, $\bar{b}_{i,j}$, $\mathsf{com}_{s'_i}$, $\mathsf{com}_{e'_i}$, $\mathsf{com}_{\bar{s}_{i,j}}$, $\mathsf{com}_{\bar{e}_{i,j}}$, π_{sk_i} according to the protocol. Send $\{\bar{s}_{i,j}, \rho_{\bar{s}_{i,j}}, \mathsf{com}_{s'_i}, \mathsf{com}_{e'_i}, \pi_{\mathsf{sk}_i}, b_i\}_{i \in \mathcal{H}, j \in \mathcal{C}}$, $\{\bar{b}_{i,j}, \mathsf{com}_{\bar{s}_{i,j}}, \mathsf{com}_{\bar{e}_{i,j}}\}_{i \in \mathcal{H}, j \in [n]}$ to $\mathcal{A}$.

6. Receive $\{\bar{s}_{j,i}, \rho_{\bar{s}_{j,i}}, \mathsf{com}_{s'_j}, \mathsf{com}_{e'_j}, \pi_{\mathsf{sk}_j}, b_j\}_{j\in\mathcal{C},i\in\mathcal{H}}$, $\{\bar{b}_{j,u}, \mathsf{com}_{\bar{s}_{j,i}}, \mathsf{com}_{\bar{e}_{j,i}}\}_{j\in\mathcal{C},i\in[n]}$. If any $h_{b_j} \neq H_0(b_j)$, $b_j \neq \mathsf{Rec}_{t,n}(\{\bar{b}_{j,k}\}_{k\in[n]})$, $0 = \mathsf{Open}(\mathsf{com}_{j,i}, \bar{s}_{j,i}, \rho_{\bar{s}_{j,i}})$ or π_{sk_j} is invalid, abort. Otherwise compute b, s_i, ρ_{s_i} and $\mathsf{com}_{\mathsf{sk}_j}$ according to the protocol.

The output is then $\mathcal{C}$, $(a, b, \{s_i\}_{i\in\mathcal{H}})$ and the output of $\mathcal{A}$.

G$_1$*:* We now change one proof. Fix an index $i' \in \mathcal{H}$, in step 5 of the experiment, π_{sk_i} is now computed by the simulator $\mathsf{Sim}_{\mathsf{sk}}$ of Π_{sk}. The rest of the experiment remains the same. $\mathbf{G}_1$ is then indistinguishable from $\mathbf{G}_0$ by the zero-knowledge property of Π_{sk} and the distinguishing advantage of $\mathcal{A}$ is $\epsilon_{\mathsf{HVZK}_{\mathsf{sk}}}$.

G$_2$*:* For i' we now replace the unopened commitments. For commitments that has opening sent to $\mathcal{A}$ i.e. $\mathsf{com}_{\bar{s}_{i',j}}$, the commitments are computed as in $\mathbf{G}_1$. Other commitments in step 5 are randomly sampled from the commitment space, and the rest of the experiment remains the same. $\mathbf{G}_2$ is then indistinguishable from $\mathbf{G}_1$ by the hiding property of the commitment scheme. For i' the commitments to $s'_{i'}, e'_{i'}, \{\mathsf{com}_{\bar{s}_{i',i}}, \mathsf{com}_{\bar{e}_{i',i}}\}_{i\in\mathcal{H}}$ are never opened, which bounds $\mathcal{A}$'s distinguishing advantage to $2\cdot(|\mathcal{H}|+1)\cdot\epsilon_{\mathsf{hiding}}$ by a cumulative bound.

G$_3$*:* We now derive the keys of $\mathcal{A}$. In step 6, call $\mathsf{Extract}_{\mathsf{sk}}$ to extract s'_j from π_{sk_j} for $j \in \mathcal{C}$ and abort if any of them fails. Otherwise secret share s'_j according to protocol to derive $\{\bar{s}_{j,i}\}_{i\in[n]}$ and derive s_j for $j \in \mathcal{C}$. The rest of the experiment remains the same. $\mathbf{G}_3$ is then indistinguishable from $\mathbf{G}_2$ 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 $|\mathcal{C}|\cdot\epsilon_{\mathsf{extract}_{\mathsf{sk}}}$.

G$_4$*:* 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 $\{\bar{b}_{i,k}\}_{k\in[n]}$. Then derive $\bar{s}_{i,k}$,$\bar{e}_{i.k}$ from $\bar{b}_{i,k}$. The rest of the experiment remains the same.

By the $\mathsf{R\text{-}LWE}$ assumption, $b_{i'}$ is computationally indistinguishable from $b_{i'} = as_{i'} + pe_{i'}$. Since $b_{i'}$ is t-out-of-n secret shared, each $\bar{b}_{i',j}$ 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 $\bar{s}_{i',j}$, $\bar{e}_{i,j}$ and consequently $\bar{b}_{i'} = as'_{i'} + pe'_{i'}$ is uniform in R_q. $\mathbf{G}_4$ is then indistinguishable from $\mathbf{G}_3$ by $\mathsf{R\text{-}LWE}$ assumption and $\mathcal{A}$'s distinguishing advantage can be bounded by $\epsilon_{\mathsf{R\text{-}LWE}}$.

G$_5$*:* We now derive $b_{i'}$ a posteriori. Before step 1, receive a BGV key pair (a, b). In step 3, sample a random $h_{b_{i'}} \leftarrow \{0,1\}^{\ell_1}$. After receiving h_{b_j} for $j \in \mathcal{C}$ from the adversary find b_j such that $h_{b_j} = H_1(b_j)$. Then derives $b_{i'} := \sum_{i\neq i'\in[n]} b_i$ and program H_1 such that $H_1(b_{i'}) = h_{b_{i'}}$. If programming fails, abort. The rest of the experiment is the same as before.

The distribution of $h_{b_{i'}}$ is statistically indistinguishable from the output of H_1 by the random oracle model. By the $\mathsf{R\text{-}LWE}$ assumption, b is indistinguishable from uniform b is indistinguishable from uniform. $\mathbf{G}_5$ and $\mathbf{G}_6$ is then 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 $\frac{Q_{H_1}}{2^{\ell_1}}$.

$\boldsymbol{G}_6$*:* Finally we repeat the same process for a_i, in step 2 sample a random $h_{a_{i'}} \leftarrow \{0,1\}^{\ell_0}$. After receiving h_{a_j} for $j \in \mathcal{C}$ from $\mathcal{A}$ find a_j such that $h_{a_j} = H_0(a_j)$. Then derives $a_{i'} := \sum\limits_{i \neq i' \in [n]} a_i$ and program H_0 such that $H_0(a_{i'}) = h_{a_{i'}}$. If programming fails, abort. The rest of the experiment is the same as before. Each a_i and obtained a are uniformly random in R_q thus the derived $a_{i'}$ is also statistically indistinguishable from uniform. Using a similar argument to $\mathbf{G}_5$, $\mathbf{G}_5$ and $\mathbf{G}_6$ are indistinguishable and the advantage of $\mathcal{A}$ is $\frac{Q_{H_0}}{2^{\ell_0}}$.

We now see that $\mathbf{G}_6$ is indistinguishable from the ideal world with $\mathsf{Sim}_{\mathsf{DKGen}}$ simulating the behavior of the honest parties. This concludes the proof. □

3.5 Encryption, Evaluation, and Aborting Decryption

For completeness, here we describe the BGV threshold homomorphic encryption algorithms Enc, Eval and Dec corresponding to a public circuit F of max depth d in the abstract definition above.

Enc On input $\mathsf{pk} = (a, b)$ and $\mathsf{ptx} = m$, it runs $\mathsf{ctx} := \mathsf{Enc}_{\mathsf{BGV}}(\mathsf{pk}, \mathsf{ptx})$ as described above while committing to r, e', e'', m using randomnesses $\rho_r, \rho_{e'}, \rho_{e''}, \rho_m$ and computes proof π_{ctx} according to relation $\mathcal{R}_{\mathsf{Enc}}$. Enc outputs $\mathsf{ctx}^* = (\mathsf{ctx}, \pi_{\mathsf{ctx}})$.

Eval On input a function F of depth at most d and set of ciphertexts $\{\mathsf{ctx}^*_i = (\mathsf{ctx}_i, \pi_{\mathsf{ctx}_i})\}_{i \in [k]}$, it outputs $\mathsf{ctx}^* := F(\{\mathsf{ctx}_i\}_{i \in [k]})$ if π_{ctx_i} verifies for all i, and otherwise outputs $\perp$. Since F is public and does not require an evaluation key, ctx^* can be verified by simply checking if ctx^* is equal to $F(\{\mathsf{ctx}_i\}_{i \in [k]})$.

Dec On input sk and $\mathsf{ctx}^* = (\mathsf{ctx}, \pi_{\mathsf{ctx}})$ it outputs $\perp$ if π_{ctx} does not verify and $\mathsf{ptx} := \mathsf{Dec}_{\mathsf{BGV}}(\mathsf{sk}, \mathsf{ctx})$ otherwise.

Note that although we use $\mathsf{Enc}_{\mathsf{BGV}}$ for Enc, the standard IND-CPA security of BGV is not directly applicable as an adversary $\mathcal{A}$ that was part of the distributed key generation has information about sk and can derive it's partial decryption shares. We briefly show that IND-CPA still holds in the distributed setting.

Lemma 1. *Let* DKGen *described in Section 3.4 be* $\epsilon_{\mathsf{DKGen}}$*-secure, proof system* $\mathit{\Pi}_{\mathsf{ctx}}$ *for the relation* $\mathcal{R}_{\mathsf{ctx}}$ *be* $\epsilon_{\mathsf{HVZK}_{\mathsf{ctx}}}$*-secure, and BGV encryption scheme be* $\epsilon_{\text{IND-CPA}}$ *secure. Then, for a static active adversary* $\mathcal{A}$ *corrupting at most* $t-1$ *parties during* DKGen*,* Enc *is distributed IND-CPA secure.*

Proof. We show this by defining a challenger $\mathcal{B}$ against IND-CPA security of BGV, which interacts with an encryption oracle for BGV and uses $\mathcal{A}$ as a subroutine to answer IND-CPA queries. Whenever $\mathcal{B}$ receives a public key (a, b) for BGV, it initiates $\mathsf{Sim}_{\mathsf{DKGen}}$ with $\mathcal{A}$ where the received (a, b) is used in the simulator. When $\mathcal{A}$ provides two challenge plaintexts ptx_0 and ptx_1, $\mathcal{B}$ submits these as challenges to BGV oracle and obtains ctx. $\mathcal{B}$ then calls the simulator for $\mathit{\Pi}_{\mathsf{ctx}}$ and obtains π_{ctx}, then sends $\mathsf{ctx}^* = (\mathsf{ctx}, \pi_{\mathsf{ctx}})$ to $\mathcal{A}$. When $\mathcal{A}$ outputs its answer, $\mathcal{B}$ forwards the answer to the oracle as its response to the IND-CPA game.

Since DKGen is secure, the public key $\mathcal{A}$ generates is the same (a, b) as $\mathcal{B}$ received from the oracle as a challenge. Since the proof system also has HVZK property, the simulated π_{ctx} is indistinguishable from a real one. Hence, ctx, therefore ctx^* received by $\mathcal{A}$ is indistinguishable from ctx^* received 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. $\square$

3.6 Threshold Decryption for BGV

We first define the actively secure threshold decryption algorithm TDec and the decryption share combination algorithm Comb depicted in Figure 4, and then prove it secure.

```
$\underline{\mathsf{TDec}(\mathsf{ctx}^* = (u, v), \mathsf{sk}_i = (s_i, \rho_{\mathsf{sk}_i}), \mathcal{U})}$
**if** $\mathsf{ctx}^* = \perp$: **return** $\perp$
$m_i := \lambda_i s_i u, E_i \leftarrow R_q$
$d_i \leftarrow m_i + pE_i, \rho_{E_i} \leftarrow \{0,1\}^*$
$\mathsf{com}_{E_i} := \mathsf{Com}(E_i; \rho_{E_i})$
Compute $\pi_{\mathsf{ds}_i}$ according to Section 3.3
**return** $\mathsf{ds}_i := (d_i, \pi_{\mathsf{ds}_i}, \mathsf{com}_{E_i}, \mathsf{com}_{\mathsf{sk}_i} = \mathsf{Com}(s_i; \rho_{\mathsf{sk}_i}))$

$\underline{\mathsf{Comb}(\mathsf{ctx}^*, \{\mathsf{ds}_j = (d_j, \pi_{\mathsf{ds}_j}, \mathsf{com}_{E_j}, \mathsf{com}_{\mathsf{sk}_j})\}_{j \in \mathcal{U}})}$
**if any** $\pi_{\mathsf{ds}_j}$ **is invalid**: **abort** $(j)$
$\mathsf{ptx} := v - \sum_{j \in \mathcal{U}} d_j \mod p$
**return** $\mathsf{ptx}$
```

Fig. 4. Threshold decryption & share combination algorithms for party $\mathcal{S}_i$.

TDec On input a ciphertext $\mathsf{ctx}^* = (\mathsf{ctx}, \pi_{\mathsf{ctx}}) = ((u, v), \pi_{\mathsf{ctx}}) =$ produced by Enc or Eval, a decryption key share $\mathsf{sk}_i = (s_i, \rho_{\mathsf{sk}_i})$, and a set of users $\mathcal{U}$ of size t, if $\mathsf{ctx} = \perp$ or π_{ctx} does not verify it outputs $\perp$. Otherwise it computes $m_i := \lambda_i s_i u$ where λ_i is the Lagrange coefficient for party i with respect to $\mathcal{U}$, samples uniform noise $E_i \leftarrow R_q$ such that $\|E_i\|_\infty \leq 2^{\mathsf{sec}} B_{\mathsf{Dec}}$ for statistical security parameter sec and noise-bound B_{Dec}, then computes $d_i := m_i + pE_i$. $\mathcal{P}_i$ then computes $(\pi_{\mathsf{ds}_i}, \mathsf{com}_{\mathsf{sk}_i}, \mathsf{com}_{E_i})$ where $\mathsf{com}_{E_i} := \mathsf{Com}(E_i; \rho_{E_i})$ is a commitment to E_i and π_{ds_i} is a proof of the relation $\mathcal{R}_{\mathsf{ds}}$ with respect to s_i

and E_i. $\mathcal{P}_i$ finally outputs $\mathsf{ds}_i := (d_i, \pi_{\mathsf{ds}_i}, \mathsf{com}_{E_i}, \mathsf{com}_{\mathsf{sk}_i})$.

Comb On input a ciphertext $\mathsf{ctx}^* = (\mathsf{ctx}, \pi_{\mathsf{ctx}}) = ((u, v), \pi_{\mathsf{ctx}}) =$ produced by Enc or Eval and partial decryption shares $\{\mathsf{ds}_j = (d_j, \pi_{\mathsf{ds}_j}, \mathsf{com}_{E_j}, \mathsf{com}_{\mathsf{sk}_j})\}_{j\in\mathcal{U}}$, it verifies all π_{ds_j} and aborts with output $\perp$ if any of them fails. Otherwise it outputs $\mathsf{ptx} := (v - \sum_{j\in\mathcal{U}} d_j) \mod p$.

Since $\|E_j\|_\infty \le 2^{\mathsf{sec}} B_{\mathsf{Dec}}$ for $j \in \mathcal{U}$, we have $\left\|\sum_{j\in\mathcal{U}} E_j\right\|_\infty \le t \cdot 2^{\mathsf{sec}} \cdot B_{\mathsf{Dec}}$. Then the requirement $\left\|v - \sum_{j\in\mathcal{U}} d_j\right\|_\infty \le (1 + 2^{\mathsf{sec}})B_{\mathsf{Dec}} < \lfloor q/2 \rfloor$ implies that $(\sum_{j\in\mathcal{U}} \lambda_j s_j)u \mod q = su \mod q$, and it follows that the threshold decryption is correct as long as the underlying BGV scheme is correct.

Using a simulation argument, we show that the threshold decryption is secure against an active adversary $\mathcal{A}$.

Theorem 2 (Threshold Decryption Security). *Let the* $\mathsf{R\text{-}LWE}_{N,q,\beta}$ *be* $\epsilon_{\mathsf{R\text{-}LWE}}$*-hard for* $\beta = \beta(B_{\mathsf{TDec}}, N, q, p)$*, Let* Com *be* $\epsilon_{\mathsf{hiding}}$ *hiding, and let proof system* Π_{ds} *for the relation* $\mathcal{R}_{\mathsf{ds}}$ *be* $\epsilon_{\mathsf{HVZK}_{\mathsf{ds}}}$ *secure. Then, the threshold decryption algorithm implements* $\mathcal{F}_{\mathsf{TDec}}$ *with computational security in the* $\mathcal{F}_{\mathsf{DKGen}}$*-hybrid model against a static active adversary* $\mathcal{A}$ *corrupting at most* $|\mathcal{C}| \le t - 1$ *parties, and the advantage of* $\mathcal{A}$ *is:*

$$\epsilon_{\mathsf{TDec}} := \mathsf{Adv}_{\mathsf{TDec}}(\mathcal{A}) \le |\mathcal{H}_{\mathcal{U}}| \cdot (\epsilon_{\mathsf{R\text{-}LWE}} + 2 \cdot \epsilon_{\mathsf{hiding}} + \epsilon_{\mathsf{HVZK}_{\mathsf{ds}}}).$$

Proof. We define the simulator $\mathsf{Sim}_{\mathsf{TDec}}$ as follows:

1. It runs $\mathcal{F}_{\mathsf{DKGen}}$ to obtain $\mathsf{pk}, \mathsf{sk}, \{\mathsf{sk}_j = (s_j, \rho_j)\}$, and chooses $i' \in \mathcal{H}_{\mathcal{U}}$.
2. When $\mathsf{ctx} = (u, v)$ arrives, the simulator decrypts $m = \mathsf{Dec}_{\mathsf{BGV}}(\mathsf{sk}, \mathsf{ctx})$. It then samples a random E_j for $j \in \mathcal{C}_{\mathcal{U}}$ such that $\|E_i\|_\infty \le 2^{\mathsf{sec}} B_{\mathsf{Dec}}$, and computes $d_j := \lambda_j u s_j + pE_j$ for $j \in \mathcal{C}_{\mathcal{U}}$. The simulator samples random d_i for $i \ne i' \in \mathcal{H}_{\mathcal{U}}$ then computes the partial decryption share for i' as $d_{i'} := v - m - \sum\limits_{j\in\mathcal{U}, j\ne i'} d_j \mod q$. Simulator then calls $\mathsf{Sim}_{\mathsf{ds}}$ to obtain the proof $\{\pi_{\mathsf{ds}_i}\}_{i\in\mathcal{H}_{\mathcal{U}}}$ and sends $\{\mathsf{ds}_i\}_{i\in\mathcal{H}_{\mathcal{U}}}$ to the adversary.
3. Simulator verifies the same share as the protocol and aborts with j if the proof does not match the decryption share. Otherwise, it outputs m as the decryption result.

Correctness of the simulator follows from the correct decryption for BGV and the appropriate noise bounds chosen for $t - 1$ shares. The security of the protocol follows by showing the output of the simulator is computationally indistinguishable from the actual protocol execution, which we again show through a set of hybrid experiments.

$\boldsymbol{G}_0$*:* The first experiment corresponds to the real world:

1. $\mathcal{A}$ sends a set of corrupted parties that participate in the decryption $\mathcal{C}_{\mathcal{U}}$.
2. Receive $\mathsf{pk}, \mathsf{sk}, \{\mathsf{sk}_j\}_{\mathcal{U}}, \mathsf{ctx}^*$.

3. For $i \in \mathcal{H}_\mathcal{U}$, sample E_i, ρ_{E_i} and compute m_i, d_i, cmt_{sk_i},com_{E_i},π_{ds_i} according to protocol. Send $\{\mathsf{ds}_i = (d_i, \pi_{\mathsf{ds}_i}, \mathsf{com}_{E_i}, \mathsf{com}_{\mathsf{sk}_i})\}_{i \in \mathcal{H}_\mathcal{U}}$ to $\mathcal{A}$.
4. Receive $\{\mathsf{ds}_j\}_{j \in \mathcal{C}_\mathcal{U}}$ from $\mathcal{A}$. If any π_{ds_j} is invalid abort. Otherwise, compute ptx according to the protocol.

The output is ptx and the output of $\mathcal{A}$.

$\mathbf{G}_1$: We now change how proofs are calculated. In step 2 of the experiment, π_{ds_i} is now computed by the simulator $\mathsf{Sim}_{\mathsf{ds}}$ of Π_{ds}. The rest of the experiment remains the same. $\mathbf{G}_1$ is then indistinguishable from $\mathbf{G}_0$ by the zero-knowledge property of Π_{ds} and the distinguishing advantage of $\mathcal{A}$ is $|\mathcal{H}_\mathcal{U}| \cdot \epsilon_{\mathsf{HVZK}_{\mathsf{ds}}}$ by a cumulative bound.

$\mathbf{G}_2$: We now replace the unopened commitments, which are only used for π_{ds_i}. The commitments for the honest parties are now sampled from the commitment space, and the rest of the experiment remains the same. $\mathbf{G}_2$ is then indistinguishable from $\mathbf{G}_1$ by the hiding property of the commitment scheme, which bounds $\mathcal{A}$ distinguishing advantage to $2 \cdot |\mathcal{H}_\mathcal{U}| \cdot \epsilon_{\mathsf{hiding}}$ by a cumulative bound.

$\mathbf{G}_3$: Finally, we replace how partial decryptions are computed. In step 2, decrypt $m = \mathsf{Dec}_{\mathsf{BGV}}(\mathsf{sk}, \mathsf{ctx})$, then sample a random E_j for $j \in \mathcal{C}_\mathcal{U}$ such that $\|E_i\|_\infty \leq 2^{\mathsf{sec}} B_{\mathsf{Dec}}$, and compute $d_j := \lambda_j u s_j + pE_j$ for $j \in \mathcal{C}_\mathcal{U}$. Fix an index $i' \in \mathcal{H}_\mathcal{U}$, sample random d_i for $i \neq i' \in \mathcal{H}_\mathcal{U}$ then compute the partial decryption share for i' as $d_{i'} := v - m - \sum_{j \in \mathcal{U}, j \neq i'} d_j \mod q$. The rest of the experiment is the same.

In $\mathbf{G}_2$ each partial decryption share is calculated as $d_i := \lambda_i u \mathbf{s}_i + pE_i$. While the threshold decryption shares is not an $\mathsf{R\text{-}LWE}$ sample as defined in Section 2.2, since each λ_j is in $\mathbb{Z}_q$ and invertible, and u having uniform coefficients, this implies that $\lambda_j us + pE$ also is a $\mathsf{R\text{-}LWE}$ sample[4]. $\mathbf{G}_3$ is then indistinguishable from $\mathbf{G}_2$ by the $\mathsf{R\text{-}LWE}$ assumption and the advantage of $\mathcal{A}$ is the cumulative bound on distinguishing $\{d_i\}_{i \in \mathcal{H}_\mathcal{U}}$, which is $|\mathcal{H}_\mathcal{U}| \cdot \epsilon_{\mathsf{R\text{-}LWE}}$.

We now see that $\mathbf{G}_3$ is indistinguishable from the ideal world with $\mathsf{Sim}_{\mathsf{TDec}}$ simulating the behavior of the honest parties. This concludes the proof. □

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

We demonstrate a simple, passively secure version of our t-out-of-n threshold signature protocol $\mathcal{TS}$. For figure depictions of semi-honest protocols, see Figures 5 and 6. For brevity, we omit the exact bounds and parameters for the underlying lattice problems as this section serves mainly as a warm-up. $\mathcal{E} = (\mathsf{DKGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{TDec}, \mathsf{Comb})$ is a threshold homomorphic encryption scheme (cf. Section 3).

[4] We can see this since for uniform a, short s and e in R_q, and fixed invertible p, p' and λ in $\mathbb{Z}_q$, we have that $\lambda as + pe \approx as + \lambda^{-1}pe \approx as + p'e \approx as + e$ from $\mathsf{R\text{-}LWE}$.

4.1 Dilithium without Aborts

We use a slightly adjusted ring version of Dilithium, also denoted a "Lyubashevsky-type signature", without rejection sampling extending [ASY22] as the underlying signature scheme. The scheme in discussion is also similar in nature to Raccoon [dPEK+23], a recent submission to NIST's additional digital signature candidates [NIS22]. 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_D 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_D Takes as input $(\mathsf{sk}, \mathsf{pk}, \mu)$, samples $r_1, r_2 \leftarrow D$ and set $\boldsymbol{r} := \begin{bmatrix} r_1 \ r_2 \end{bmatrix}$. Computes $w := \langle \boldsymbol{a}, \boldsymbol{r} \rangle$ and derive the challenge $c := H(w, \mathsf{pk}, \mu) \in \mathcal{C}_\nu$. Outputs the signature $(c, \boldsymbol{z})$ where $\boldsymbol{z} := c\mathbf{s} + \boldsymbol{r}$.

Vrfy_D Takes as input $(\mathsf{pk}, (\boldsymbol{z}, c), \mu)$ and checks that $\|\boldsymbol{z}\|_2 \leq B = (\sigma + \eta\nu)\sqrt{2N}$ 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 correctness and security of this scheme. In particular, they show:

Lemma 2. *Let $\beta = 2B + 2\eta\nu\sqrt{2N}$ and $\sigma \geq \eta\nu\sqrt{2NQ}$, where Q is the maximum number of signing queries an adversary can make, and let the hash function 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).

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 and depth d where d defines the set of all circuits consisting of one multiplication with an element from $\mathcal{C}_\nu$ and t additions and all circuits consisting of n additions. As a result, $\mathcal{S}_i$ learns the public encryption key $\mathsf{pk}_\mathcal{E}$ and its decryption key share sk_i. Each party $\mathcal{S}_j$ then chooses a uniform ring element $a_j \in R_q$ and broadcasts.

2. The parties define $a := \sum_{j \in [n]} a_j$ and $\boldsymbol{a} := \begin{bmatrix} a \ 1 \end{bmatrix}$. Then $\mathcal{S}_i$ samples short $s_{i,1}, s_{i,2}$ and sets $\mathbf{s}_i = \begin{bmatrix} s_{i,1} \ s_{i,2} \end{bmatrix}$ $y_i := \langle \boldsymbol{a}, \mathbf{s}_i \rangle$. It computes the ciphertext $\mathsf{ctx}_{\mathbf{s}_i} := \mathsf{Enc}(\mathsf{pk}_\mathcal{E}, \mathbf{s}_i)$ and broadcasts that ciphertext along with y_i.

```
$\underline{\mathsf{KGen}_{\mathcal{TS}}}$
                                        $\underline{\mathsf{DKGen}(t, n, d)}$
                                        $\longrightarrow$
                                        $\longleftarrow$
                                        return $\mathsf{sk}_i, \mathsf{pk}_{\mathcal{E}}$
$a_i \leftarrow R_q$
                                        $\xrightarrow{a_i}$
                                        $\xleftarrow{\{a_j\}_{j \neq i}}$
$a := \sum_{j \in [n]} a_j, \quad \boldsymbol{a} := \begin{bmatrix} a & 1 \end{bmatrix}$
$s_{i,1}, s_{i,2} \leftarrow D, \quad \mathbf{s}_i := \begin{bmatrix} s_{i,1} & s_{i,2} \end{bmatrix}$
$y_i := \langle \boldsymbol{a}, \mathbf{s}_i \rangle, \quad \mathsf{ctx}_{\mathbf{s}_i} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i)$
                                        $\xrightarrow{y_i, \mathsf{ctx}_{\mathbf{s}_i}}$
                                        $\xleftarrow{\{(y_j, \mathsf{ctx}_{\mathbf{s}_j})\}_{j \neq i}}$
$y := \sum_{j \in [n]} y_j, \quad \mathsf{ctx}_{\mathbf{s}} := \sum_{j \in [n]} \mathsf{ctx}_{\mathbf{s}_j}$
return $\mathsf{pk} := (\boldsymbol{a}, y), \quad \mathsf{sk}_i, \quad \mathsf{aux} := (\mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$
```

Fig. 5. Passively secure key-generation protocol for signer $\mathcal{S}_i$.

3. The parties compute $\mathsf{ctx} := \sum \mathsf{ctx}_{\mathbf{s}_j} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s})$ and $y := \sum y_j = \langle \boldsymbol{a}, \mathbf{s} \rangle$, where $\mathbf{s} = \sum \mathbf{s}_j$, and define the public verification key to be $\mathsf{pk} := (\boldsymbol{a}, \boldsymbol{y})$. 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 $\mathsf{Sign}_{\mathcal{TS}}$ goes as follows:

1. To sign a message μ, party $\mathcal{S}_i$ first samples a short ring elements $r_{i,1}$,$r_{i,2}$ and defines vector $\boldsymbol{r}_i := \begin{bmatrix} r_{i,1} & r_{i,2} \end{bmatrix}$. Signer $\mathcal{S}_i$ then computes $w_i := \langle \boldsymbol{a}_i, \boldsymbol{r}_i \rangle$ 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. Next, the signers compute $w := \sum_{j \in \mathcal{U}} w_j$ and $c := H(w, \mathsf{pk}, \mu) \in \mathcal{C}_\nu$, 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 then 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 $\mathsf{pk} = (\boldsymbol{a}, y)$ if (1) $\boldsymbol{z}$ is short and (2) $H(\langle \boldsymbol{a}, \boldsymbol{z} \rangle - cy, \mathsf{pk}, \mu) = c$.

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

```
$\underline{\mathsf{Sign}_{\mathcal{TS}}(\mathsf{sk}_i, \mathsf{aux}, \mathcal{U}, \mu)}$
$r_{i,1}\ r_{i,2} \leftarrow D_r, \quad \boldsymbol{r}_i := \left[r_{i,1}\ r_{i,2}\right]$
$w_i := \langle \boldsymbol{a}, \boldsymbol{r}_i \rangle, \quad \mathsf{ctx}_{\boldsymbol{r}_i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i)$
                                        $\xrightarrow{w_i, \mathsf{ctx}_{\boldsymbol{r}_i}}$
                                        $\xleftarrow{\{(w_j, \mathsf{ctx}_{\boldsymbol{r}_j})\}_{j \in \mathcal{U} \setminus \{i\}}}$
$w := \sum_{j \in \mathcal{U}} w_j, \quad c := H(w, \mathsf{pk}, \mu)$
$\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r}_j}$
$\mathsf{ds}_i := \mathsf{TDec}(\mathsf{ctx}_{\boldsymbol{z}}, \mathsf{sk}_i, \mathcal{U})$
                                        $\xrightarrow{\mathsf{ds}_i}$
                                        $\xleftarrow{\{\mathsf{ds}_j\}_{j \in \mathcal{U} \setminus \{i\}}}$
$\boldsymbol{z} := \mathsf{Comb}(\mathsf{ctx}_{\boldsymbol{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}})$
**return** $\sigma := (c, \boldsymbol{z})$
```

Fig. 6. Passively secure t-out-of-n threshold signing protocol for signer $\mathcal{S}_i$.

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

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

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

4.3 Proof of Security

Theorem 3 (Informal). *The threshold signature scheme $\mathcal{TS}$ is threshold existentially unforgeable under chosen message attacks (*ts-uf-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 key generation secure, threshold secure and secure against chosen-plaintext attacks, and the* R-LWE *assumption is secure.*

Proof. 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 $\mathsf{pk} = (\boldsymbol{a}, y)$. 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 $a_j \leftarrow R_q$ and give $\{a_j\}_{i\in[n]}$ to $\mathcal{A}$. Set $\boldsymbol{a} = [\sum_{j\in[n]} a_j \ 1]$.

3. For $j \in [n]$, sample $s_{j,1}, s_{j,2} \leftarrow D$ and compute $\mathbf{s}_j := [s_{j,1} \ s_{j_2}]$, $y_j := \langle \boldsymbol{a}, \mathbf{s}_j \rangle$, 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, pk, μ) 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 $\boldsymbol{r}_j := [r_{j,1} \ r_{j,2}]$, $w_j := \langle \boldsymbol{a}, \boldsymbol{r}_j \rangle$ and $\mathsf{ctx}_{\boldsymbol{r}_j} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_j)$. Then $\mathcal{A}$ is given $\{\boldsymbol{r}_j\}_{j\in\mathcal{C}_{\mathcal{U}}}$ and $\{(w_j, \mathsf{ctx}_{\boldsymbol{r}_j})\}_{j\in\mathcal{U}}$. Set $w := \sum_{j\in\mathcal{U}} w_j$, $c := H(w, \mathsf{pk}, \mu)$, 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

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

$\boldsymbol{G}_1$*:* In this game $\mathcal{B}$ uses simulators for the distributed key generation and threshold decryption, and replaces public encryption key, distributed decryption keys as well as partial decryptions with simulated shares.

The experiment is the same as $\mathbf{G}_0$ except that in step 1 of $\mathsf{KGen}_{\mathcal{TS}}$, $\mathcal{B}$ runs $\mathsf{pk}_{\mathcal{E}}, \{\mathsf{sk}_j\}_{j\in\mathcal{C}} \leftarrow \mathsf{Sim}_{\mathsf{DKGen}}$ so that the output $\mathsf{view}_{\mathcal{C}}$ consists of $\mathcal{U}, \{\mathsf{sk}_j\}_{j\in\mathcal{C}}$ and $\mathsf{pk}_{\mathcal{E}}$, which is given to $\mathcal{A}$.

Then the final step uses simulated decryption instead of shares from honest parties. Specifically, in step 2 of the signing protocol $\mathsf{Sign}_{\mathcal{TS}}$, $\mathcal{B}$ computes $\{\mathsf{ds}_i\}_{\mathcal{H}_{\mathcal{U}}}$ using $\mathsf{Sim}_{\mathsf{TDec}}$ based on $\{\mathsf{sk}_i\}_{i\in\mathcal{C}_{\mathcal{U}}}, \mathsf{ctx}_{\boldsymbol{z}}$, and gives $\{\mathsf{ds}_i\}_{i\in\mathcal{U}}$ to $\mathcal{A}$.

By key generation security of $\mathcal{E}$, the view of each corrupted party should be computationally indistinguishable from actual protocol for all but a negligible probability $\epsilon_{\mathsf{DKGen}}$. Similarly, since the distributed key generation of $\mathcal{E}$ implements the functionality $\mathcal{F}_{\mathsf{DKGen}}$ securely, the view of corrupted parties during threshold decryption is computationally indistinguishable from actual protocol outside a negligible probability ϵ_{TDec}. Then $\mathbf{G}_0$ and $\mathbf{G}_1$ are computationally indistinguishable as long as key generation and threshold security of $\mathcal{E}$ holds:

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0]| \leq \epsilon_{\mathsf{DKGen}} + \epsilon_{\mathsf{TDec}}.$$

$\boldsymbol{G}_2$: 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 $\mathbf{G}_2$ and $\mathbf{G}_1$, then it is possible to use $\mathcal{A}$ to break IND-CPA security 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 $\mathbf{G}_2$ is exactly the same as $\mathbf{G}_1$, 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 $\mathcal{E}$. Thus we conclude that $\mathbf{G}_1$ and $\mathbf{G}_2$ are indistinguishable as long as IND-CPA security of $\mathcal{E}$ holds:

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1]| \leq \epsilon_{\text{IND-CPA}}.$$

$\boldsymbol{G}_3$: 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 $(\boldsymbol{a}, y)$ 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 of key generation, for every $j \in [n], j \neq i'$, a_j is sampled in the same way as earlier. $a_{i'}$ is then fixed as $a_{i'} := a - \sum_{j \in [n], j \neq i'} a_j$. During step 3, 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 $i \in \mathcal{C}_{\mathcal{U}}$, sample $r_{i,1}, r_{i,2} \leftarrow D_r$ and set $\boldsymbol{r}_i := [r_{i,1}\ r_{i,2}]$ and $w_i := \langle \boldsymbol{a}, \boldsymbol{r}_i \rangle$ as before. Letting i' denote some index in $\mathcal{H}_{\mathcal{U}}$, sample $w_i \leftarrow R_q$ uniformly for $i \in \mathcal{H}_{\mathcal{U}} \setminus \{i'\}$, and set $w_{i'} := w - \sum_{i \in \mathcal{U} \setminus \{i'\}} w_i$, where w is computed from $(c, \boldsymbol{z})$ as $w := \langle \boldsymbol{a}, \boldsymbol{z} \rangle - cy$. $\mathcal{B}$ 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 a is uniform in R_q and $y = \langle \boldsymbol{a}, \mathbf{s}' \rangle$ for an unknown $\mathbf{s}'$ with $\|\mathbf{s}'\| \leq tB$. Since

$\{a_i\}_{i\in[n]\setminus\{i'\}}$ is sampled honestly from a uniform distribution, then $a_{i'} := a - \sum_{i\in[n]\setminus\{i'\}} a_i$ is also uniform in $R_q^{k\times\ell}$. Consequently $y_{i'} := y - \sum_{i\in[n]\setminus\{i'\}} y_i = \langle \boldsymbol{a}, \mathbf{s}'_{i'}\rangle$ 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 $\{w_i\}_{i\in\mathcal{H}_\mathcal{U}}$ 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 $a, a_{i'}, y, y_{i'}$ 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 $\mathbf{G}_2$ and $\mathbf{G}_3$.

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 $\boldsymbol{r}$, $\mathcal{B}$ obtains $(\boldsymbol{a}', u')$ and sets $\boldsymbol{a} := \boldsymbol{a}'$ and $w := u'$. $\mathcal{B}$ computes w_i for $i \neq i' \in \mathcal{U}$ according to the protocol and derives $w_{i'} := w - \sum_{i\in\mathcal{U}\setminus\{i'\}}$ and sends $\{w_i\}_{i\in\mathcal{U}}$ 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 $\mathbf{G}_2$ and $\mathbf{G}_3$ are indistinguishable as long as R-LWE holds:

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| \leq \epsilon_{\mathsf{R\text{-}LWE}}.$$

We remark that the combined signature in $\mathbf{G}_3$ now has the same distribution for $(\boldsymbol{a}, y, c, \boldsymbol{z})$ as the underlying signature for combined parameters, since $(\boldsymbol{a}, y, c, \boldsymbol{z})$ 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:

$$\Pr[\mathbf{G}_3] = \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A}).$$

This concludes the proof. □

Since we remove the reliance on aborts, each signature may leak information about the secret key as discussed by Lyubashesvky [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 6.

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 to this paper: the actively secure threshold signature scheme. We bootstrap the passively secure protocol described in Section 4 to a protocol secure against a rushing and active adversary that may behave arbitrarily. The key generation and signing protocols are depicted in Figure 7 and Figure 8, 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 Extended Zero-Knowledge Proofs

Our distributed $\mathsf{KGen}_{\mathcal{TS}}$ and $\mathsf{Sign}_{\mathcal{TS}}$ routines need two zero-knowledge proofs that extend the relations given about the encryption scheme above. We give the exact relations each proof has to prove below:

The proof $\pi_{\mathbf{s}_i}$ during $\mathsf{KGen}_{\mathcal{TS}}$ 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 $\boldsymbol{a}$, $B_{\mathbf{s}}$ and public y_i, $\mathsf{pk}_{\mathcal{E}}$, and the commitment to the secret $\mathbf{s}_i$, the relation $\mathcal{R}_{\mathbf{s}}$ shows:(1) The 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 $\mathsf{pk}_{\mathcal{E}}$.

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

The proof $\pi_{\boldsymbol{r}_i}$ during $\mathsf{Sign}_{\mathcal{TS}}$ 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 publicly fixed $\boldsymbol{a}$, $B_{\boldsymbol{r}}$ and public com_i,$\mathsf{pk}_{\mathcal{E}}$,$\mathsf{ctx}_{\boldsymbol{r}_i}$ for secret w_i,$\boldsymbol{r}_i$, ρ_i the relation $\mathcal{R}_{\boldsymbol{r}}$ shows: (1) The 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 $\boldsymbol{r}_i$ using $\mathsf{pk}_{\mathcal{E}}$.

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

In both relations, (3) is the proof of of knowledge of plaintext, which is done as part of $\mathcal{E}$. The rest of the proofs can be instantiated similar to the proofs for $\mathcal{E}$ discussed in Section 3.3.

5.2 Key Generation and Signing Protocols

We retain our notation and viewpoint from the passive protocol, in addition to introducing homomorphic commitments and non-interactive zero-knowledge proofs. Note that we change the signatures slightly 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.

```
$\mathsf{KGen}_{\mathcal{TS}}$
                                   $\mathsf{DKGen}(t, n, d)$
                                   ┌──────────────────────────────────────────┐
                                   │                 $\longrightarrow$        │
                                   │                 $\longleftarrow$         │
                                   │ return $\mathsf{sk}_i = \mathsf{sk}_{\mathcal{E},i}, \mathsf{pk}_{\mathcal{E}}, \mathsf{aux}_{\mathcal{E}}$ │
                                   └──────────────────────────────────────────┘
$a_i \leftarrow R_q, \quad h_{a_i} := H_0(a_i)$
                                   $\xrightarrow{h_{a_i}}$
                                   $\xleftarrow{\{h_{a_j}\}_{j \neq i}}$
                                   $\xrightarrow{a_i}$
                                   $\xleftarrow{\{a_j\}_{j \neq i}}$
if any $h_{a_j} \neq H_0(a_j)$: abort $(j)$
$a := \sum_{j \in [n]} a_j, \quad \boldsymbol{a} := [a \ 1]$
$s_{i,1}, s_{i,2} \leftarrow D_s, \quad \mathbf{s}_i := [s_{i,1} \ s_{i,2}]$
$y_i = \langle \boldsymbol{a}, \mathbf{s}_i \rangle, \quad h_{y_i} := H_1(y_i)$
                                   $\xrightarrow{h_{y_i}}$
                                   $\xleftarrow{\{h_{y_j}\}_{j \neq i}}$
$\mathsf{ctx}_{\mathbf{s}_i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i)$
Compute $\pi_{\mathbf{s}_i}$ according Section 5.1
                                   $\xrightarrow{(y_i, \mathsf{ctx}_{\mathbf{s}_i}, \pi_{\mathbf{s}_i})}$
                                   $\xleftarrow{\{(y_j, \mathsf{ctx}_{\mathbf{s}_j}, \pi_{\mathbf{s}_j})\}_{j \neq i}}$
if any $h_{y_j} \neq H_1(y_j)$: abort $(j)$
if any $\pi_{\mathbf{s}_j}$ is invalid: abort $(j)$
$y := \sum_{j \in [n]} y_j, \quad \mathsf{ctx}_{\mathbf{s}} := \sum_{j \in [n]} \mathsf{ctx}_{\mathbf{s}_j}$
return $\mathsf{pk} := (\boldsymbol{a}, y), \quad \mathsf{sk}_i, \quad \mathsf{aux} := (\mathsf{aux}_{\mathcal{E}}, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$
```

Fig. 7. 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,n, and d 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 associated with $\mathcal{E}$.

2. $\mathcal{S}_i$ samples a uniform a_i from R_q, computes $h_{a_i} = H_0(a_i)$ as a commitment and broadcasts h_{a_i}. After receiving h_{a_j} for all $j \neq i$, it broadcasts a_i. After receiving a_j for all $j \neq i$, it verifies h_{a_j}, and aborts with output j if $h_{a_j} \neq H_0(a_j)$. Otherwise define $a = \sum a_j$ and $\boldsymbol{a} = [a\ 1]$ for $j \in [n]$.

3. $\mathcal{S}_i$ then samples short signing key pieces $s_{i,1}, s_{i,2}$ sets $\mathbf{s}_i := [s_{i,1}\ s_{i,2}]$, $y_i := \langle \boldsymbol{a}, \mathbf{s}_i \rangle$, hash $h_{y_i} := H_1(y_i)$ 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 $\pi_{\mathbf{s}_i}$ according to $\mathcal{R}_{\mathbf{s}}$, then broadcasts $(y_i, \mathsf{ctx}_{\mathbf{s}_i}, \pi_{\mathbf{s}_i})$.

4. 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 $h_{y_j} = H_1(y_j)$ 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 $\mathsf{pk} = (\boldsymbol{a}, y)$, secret key sk_i, and auxiliary information $\mathsf{aux} = (\mathsf{aux}_{\mathcal{E}}, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$ where $y := \sum y_j = \langle \boldsymbol{a}, \mathbf{s} \rangle$, $\mathsf{ctx}_{\mathbf{s}} := \sum \mathsf{ctx}_{\mathbf{s}_j}$.

$\mathsf{Sign}_{\mathcal{TS}}$ 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 $r_{i,1}, r_{i,2} \leftarrow D_r$ and commitment randomness $\rho_i \leftarrow \chi$, derives per message commitment key $\mathsf{ck} = H_2(\mathsf{pk}, \mu)$ and computes $\boldsymbol{r}_i := [r_{i,1}\ r_{i,2}]$, $w_i := \langle \boldsymbol{a}, \boldsymbol{r}_i \rangle$ 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 $\pi_{\boldsymbol{r}_i}$ according to $\mathcal{R}_{\boldsymbol{r}}$. $\mathcal{S}_i$ 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 $c := H_3(\mathsf{com}, \mathsf{pk}, \mu)$. 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, the verifier checks that $\|\boldsymbol{z}\| \leq B_{\boldsymbol{z}}$ and $\|\rho\| \leq B_\rho$, computes $w^* := \langle \boldsymbol{a}, \boldsymbol{z} \rangle - c^* y$, derives $c^* := H_3(\mathsf{Com}(w^*; \rho), \mathsf{pk}, \mu)$, then finally outputs 1 if and only if checks hold and $c = c^*$, and 0 otherwise.

```
$\underline{\mathsf{Sign}_{\mathcal{TS}}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{aux}, \mathcal{U}, \mu)}$
$\mathsf{ck} := H_2(\mathsf{pk}, \mu), \quad r_{i,1}, r_{i,2} \leftarrow D_r, \quad , \boldsymbol{r}_i := [r_{i,1} \ r_{i,2}]$
$\rho_i \leftarrow \chi, w_i := \langle \boldsymbol{a}, \boldsymbol{r}_i \rangle$
$\mathsf{com}_i := \mathsf{Com}_{\mathsf{ck}}(w_i, \rho_i), \mathsf{ctx}_{\boldsymbol{r}_i} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i)$
Compute $\pi_{\boldsymbol{r}_i}$ according to Section 5.1          $\xrightarrow{(\mathsf{ctx}_{\boldsymbol{r}_i}, \mathsf{com}_i, \pi_{\boldsymbol{r}_i})}$
                                                                  $\xleftarrow{\{(\mathsf{ctx}_{\boldsymbol{r}_j}, \mathsf{com}_j, \pi_{\boldsymbol{r}_j})\}_{j \in \mathcal{U} \setminus \{i\}}}$
$\textbf{if any } \pi_{\boldsymbol{r}_j} \textbf{ is invalid}: \textbf{abort } (j)$
$\mathsf{com} := \sum_{j \in \mathcal{U}} \mathsf{com}_j, \quad c := H_3(\mathsf{com}, \mathsf{pk}, \mu)$
$\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r}_j}$
$\mathsf{ds}_i := \mathsf{TDec}(\mathsf{ctx}_{\boldsymbol{z}}, \mathsf{sk}_i, \mathcal{U})$
                                                                  $\xrightarrow{(w_i, \rho_i, \mathsf{ds}_i)}$
                                                                  $\xleftarrow{\{(w_j, \rho_j, \mathsf{ds}_j)\}_{j \in \mathcal{U} \setminus \{i\}}}$
$\textbf{if any } \mathsf{Open}(\mathsf{com}_j, w_j, \rho_j) = 0: \textbf{abort } (j)$
$\boldsymbol{z} := \mathsf{Comb}(\mathsf{ctx}_{\boldsymbol{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}})$
$\textbf{if } \boldsymbol{z} = \perp \text{ with } \textbf{abort } (j): \textbf{abort } (j)$
$\rho := \sum_{j \in \mathcal{U}} \rho_j$
$\textbf{return } \sigma := (c, \boldsymbol{z}, \rho)$
```

Fig. 8. Actively secure 2-round t-out-of-n threshold signature protocol for signer $\mathcal{S}_i$.

5.3 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 $[-B, B]$, then the sum will be in the interval $[-tB, tB]$. However, if we sample from a discrete Gaussian distribution of standard deviation σ, then 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 size of the per-signature randomness $\boldsymbol{r}$ or by 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 then sends two hashes of size ℓ_0 and ℓ_1 bits respectively each and 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 $N \log_2 q$.

5.4 Security Proof

Similar to the case with a passive adversary, we now prove the security of our protocol by constructing an algorithm $\mathcal{B}$ interacting with an active adversary $\mathcal{A}$. Unlike the passive case however, we assume a rushing adversary where honest users always publish their messages first. If decryption shares are published by honest parties, then we say that the message is signed.

Theorem 4. *Let $\mathsf{R\text{-}LWE}_{N,B}$ be $\epsilon_{\mathsf{R\text{-}LWE}}$-hard. Let Q_S, Q_H denote the number of signing queries and total number of queries made to H_0, H_1, H_2, and H_3 respectively. Let ℓ_0, ℓ_1 be the bit-length of the output of H_0 and H_1. Let $\mathcal{E}$ be $\epsilon_{\text{IND-CPA}}$, $\epsilon_{\mathsf{DKGen}}$ and ϵ_{TDec} secure. Finally, let proof system $\Pi_{\boldsymbol{x}}$ for the relation $\mathcal{R}_{\boldsymbol{x}}$ be $\epsilon_{\mathsf{HVZK}_{\boldsymbol{x}}}$ 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 8, is* ts-uf-cma *secure when H_0 and H_1 are modeled as programmable random oracles and H_2 and H_3 are modeled as random oracles. The advantage of adversary $\mathcal{A}$ is:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{TS}}(\mathcal{A}) \leq e(Q_H + Q_S + 1)\big(&|\mathcal{H}| \cdot \epsilon_{\mathsf{HVZK}_{\mathbf{s}}} + |\mathcal{H}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{HVZK}_{\boldsymbol{r}}} + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract}_{\mathbf{s}}} \\ &+ |\mathcal{C}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{HVZK}_{\boldsymbol{r}}} + \epsilon_{\mathsf{DKGen}} + \epsilon_{\mathsf{TDec}} \\ &+ \frac{|\mathcal{C}|(Q_H + Q_S)}{2^{\ell_0}} + \frac{|\mathcal{C}|(Q_H + Q_S)}{2^{\ell_1}} + (Q_H + Q_S) \cdot \epsilon_{\mathsf{td}} \\ &+ 2 \cdot \epsilon_{\text{IND-CPA}} + \epsilon_{\mathsf{R\text{-}LWE}} + \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A})\big).\end{aligned}$$

Proof. We prove this through a series of hybrids written out in full detail. The simulator $\mathcal{B}$ again has access to the 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 $\sigma = (c, \boldsymbol{z}, \rho)$.

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, H_2, H_3 are simulated using tables $\mathcal{HT}_0$, $\mathcal{HT}_1$, $\mathcal{HT}_2$, and $\mathcal{HT}_3$. 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 message-dependent commitment key $\mathsf{ck}^* := H_2(\mathsf{pk}, \mu^*)$, computes $w^* := \langle \boldsymbol{a}, \boldsymbol{z}^* \rangle - c^* y$ and $\mathsf{com}^* := \mathsf{com}_{\mathsf{ck}^*}(w^*; \rho^*)$. Then, if the challenge $c^* \neq H_3(\mathsf{com}^*, \mathsf{pk}, \mu^*)$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ halts with the output (σ^*, μ^*) and $\mathcal{A}$ is successful.

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 $\pi_{\mathbf{s}_i} := \mathsf{Sim}(\boldsymbol{a}, y_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}_i}, B_{\mathbf{s}})$ and $\pi_{\boldsymbol{r}_i} := \mathsf{Sim}(\boldsymbol{a}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\boldsymbol{r}_i}, B_{\boldsymbol{r}})$. $\mathcal{B}$ follows the remaining parts of the protocol honestly. $\mathbf{G}_0$ and $\mathbf{G}_1$ is indistinguishable by HVZK of the underlying NIZKs:

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

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 $\mathbf{s}_j := \mathsf{Extract}_{\mathbf{s}}(\pi_{\mathbf{s}_j}; \boldsymbol{a}, y_j, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}_j}, B_{\mathbf{s}})$. Similarly, during signing after receiving $\pi_{\boldsymbol{r}_i}$, $\mathcal{B}$ computes $\boldsymbol{r}_i := \mathsf{Extract}_{\boldsymbol{r}}(\pi_{\boldsymbol{r}_i}; \boldsymbol{a}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\boldsymbol{r}_i}, B_{\boldsymbol{r}})$. 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 $\mathcal{A}$. Thus we bound $\mathcal{A}$'s advantage in $\mathbf{G}_3$ by the sum of independent probabilities of 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$_3$*:* In this hybrid $\mathcal{B}$ simulates the distributed decryption shares and signing keys instead of computing them similar to $\mathbf{G}_1$ in the passive case. During key generation, $\mathcal{B}$ uses the simulator $\mathsf{Sim}_{\mathsf{DKGen}}$ for $\mathcal{E}$ to interact with $\mathcal{A}$ to obtain key shares $\{\mathsf{sk}_j\}_{j \in \mathcal{C}}$, then proceeds with the rest of the protocol until the second round of signing. During the second round of signing after computing $\mathsf{ctx}_{\boldsymbol{z}}$, $\mathcal{B}$ knows the plaintext $\boldsymbol{z} = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctx}_{\boldsymbol{z}})$ since it already have extracted all $\mathbf{s}_j$ and $\boldsymbol{r}_j$ for $j \in \mathcal{C}_{\mathcal{U}}$ and invokes the distributed decryption simulator $\mathsf{Sim}_{\mathsf{TDec}}$ of $\mathcal{E}$ to obtain $\{\mathsf{ds}_i\}_{i \in \mathcal{H}_{\mathcal{U}}}$ from $\mathsf{Sim}_{\mathsf{TDec}}$ using $\{\mathsf{sk}_j\}_{j \in \mathcal{C}}, \boldsymbol{z}, \mathcal{U}$, and sends them as their decryption shares instead. The rest of the signing proceeds as $\mathbf{G}_2$.

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. Similar to the $\mathbf{G}_1$ in the passive case, we can now reduce to the security of $\mathcal{E}$ during key generation and threshold decryption since DKGen and TDec are secure against a static active adversary corrupting up to $t-1$ parties:

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| \leq \epsilon_{\mathsf{DKGen}} + \epsilon_{\mathsf{TDec}}.$$

$\mathbf{G}_4$: $\mathbf{G}_4$ is exactly the same as $\mathbf{G}_3$ except $\mathcal{B}$ embeds a trapdoor to the commitment key ck with high probability. Then the forgery by $\mathcal{A}$ must be with respect to 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_2 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_2, $\mathcal{B}$ sets the flag *bad* and aborts if $\mathcal{TDT}[\mathsf{pk}, \mu] = \perp$. 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 it is in $\mathbf{G}_3$. 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 $\mathbf{G}_3$. At the final step, if $\mathcal{TDT}[\mathsf{pk}, \mu^*] \neq \perp$, $\mathcal{B}$ aborts.

If $\mathcal{B}$ does not abort, then $\mathsf{ck}^* = H_2(\mathsf{pk}, \mu^*)$ 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_2 and uses the predefined ck for the one associated with forgery, i.e., *bad* is not set. Based on the security of the trapdoor commitment scheme defined in Section 2.4:

$$\Pr[\mathbf{G}_4] \geq \phi^{Q_H+Q_S} \cdot (1-\phi) \cdot \Pr[\mathbf{G}_3] - (Q_H + Q_S) \cdot \epsilon_{\mathsf{td}}.$$

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

$$\Pr[\mathbf{G}_4] \geq \frac{\Pr[\mathbf{G}_3]}{e(Q_H + Q_S + 1)} - (Q_H + Q_S) \cdot \epsilon_{\mathsf{td}},$$

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

$\mathbf{G}_5$: In this hybrid $\mathcal{B}$ changes how ciphertexts are computed similar to $\mathbf{G}_2$ 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 $\mathbf{G}_4$.

Note that since the decryption shares are simulated starting from $\mathbf{G}_3$, any inconsistency between $\mathsf{ctx}_{\boldsymbol{z}}$ and $\boldsymbol{z}$ is handled by the simulator. Hence, the only difference between $\mathbf{G}_4$ and $\mathbf{G}_5$ is $\mathcal{A}$'s view on $\mathsf{ctx}_{\mathbf{s}_{i'}}$ and $\mathsf{ctx}_{\boldsymbol{r}_{i'}}$. Following the same argument in $\mathbf{G}_2$ in the passive case, we have:

$$|\Pr[\mathbf{G}_5] - \Pr[\mathbf{G}_4]| \leq 2 \cdot \epsilon_{\text{IND-CPA}}.$$

$\boldsymbol{G}_6$: In this hybrid $\mathcal{B}$ changes how $h_{a_{i'}}$ and $h_{y_{i'}}$ are computed. $\mathcal{B}$ first samples random $h_{a_{i'}} \in \{0,1\}^{\ell_0}$ on the first round and sends it as the commitment to $a_{i'}$. After receiving h_{a_j} for all $j \in \mathcal{C}$, $\mathcal{B}$ extracts a_j from recorded entries in $\mathcal{HT}_0$. Then samples a random $a \leftarrow R_q$, derives $a_{i'} = a - \sum\limits_{j\in[n]\setminus\{i'\}} a_j$ and programs H_0 so that $H_0(a_{i'}) = h_{a_{i'}}$. $\mathcal{B}$ then sends a random $h_{y_{i'}} \in \{0,1\}^{\ell_1}$ as the commitment $y_{i'}$ then programs H_1 such that $H_1(y_{i'}) = h_{y_{i'}}$. If programming fails, $\mathcal{B}$ aborts.

Furthermore, a_i in $\mathbf{G}_5$ are sampled uniformly from R_q, hence $a := \sum_i a_i$ is statistically indistinguishable from uniform in R_q. Since a is sampled from uniform in $\mathbf{G}_6$, the two are statistically indistinguishable. Consequently, the derived $a_{i'}$ is also statistically indistinguishable from uniform which means it is indistinguishable from $a_{i'}$ in $\mathbf{G}_5$. Then the only difference in $\mathcal{A}'s$ view is on how $h_{a_{i'}}$ and $h_{y_{i'}}$ are computed. By the oracle assumption, the distributions is indistinguishable between these two games, hence from the view of the adversary $\mathcal{A}$, $\mathbf{G}_5$ and $\mathbf{G}_6$ are indistinguishable as long as $\mathcal{B}$ does not abort during programming. Thus we can bound the advantage of $\mathcal{A}$ in distinguishing $\mathbf{G}_5$ and $\mathbf{G}_6$ by cumulative failure probability of independent programming for H_0 and H_1:

$$|\Pr[\mathbf{G}_6] - \Pr[\mathbf{G}_5]| \leq \frac{|\mathcal{C}|(Q_H + Q_S)}{2^{\ell_0}} + \frac{|\mathcal{C}|(Q_H + Q_S)}{2^{\ell_1}}.$$

$\boldsymbol{G}_7$: 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 $(\boldsymbol{a}, y)$, then replaces $\boldsymbol{a}$ in $\mathbf{G}_6$ with the received $\boldsymbol{a}$. Instead of honestly computing $y_{i'}$, $\mathcal{B}$ derives $y_{i'} = y - \sum\limits_{j\in[n]\setminus\{i'\}} y_j$ and continues the rest of key generation as in $\mathbf{G}_6$.

For $i \in \mathcal{H}_{\mathcal{U}} \setminus \{i'\}$, signing starts as $\mathbf{G}_6$. 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 $w := \langle \boldsymbol{a}, \boldsymbol{z}\rangle - cy$ 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 $\mathbf{G}_6$.

The second round of communication in the key generation is identical to experiment $\mathbf{G}_6$ outside $\boldsymbol{a}$. Since $\boldsymbol{a}$ received from $\mathcal{D}$ is honestly computed, a is uniform and the distribution of $\boldsymbol{a}$ in these games is the same. Similarly, since y received from $\mathcal{D}$ is a valid $\mathsf{R\text{-}LWE}$ instance, the distribution of $y_{i'}$ in $\mathbf{G}_7$ is same as $\mathbf{G}_6$ by linearity of operations the derived $y_{i'} = \langle \boldsymbol{a}, \mathbf{s}'_{i'}\rangle$ for an unknown $\mathbf{s}'_{i'}$.

$\mathcal{B}$ now has to fix the signature and its shares in a way that is 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 $w := \langle \boldsymbol{a}, \boldsymbol{z} \rangle - cy$. 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 $\mathbf{G}_6$ and $\mathbf{G}_7$ can be used as a distinguisher for R-LWE. We then have:

$$|\Pr[\mathbf{G}_7] - \Pr[\mathbf{G}_6]| \leq \epsilon_{\mathsf{R\text{-}LWE}}.$$

Now the signatures are independent of all secret key material 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:

$$|\Pr[\mathbf{G}_7]| \leq \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A}).$$

This concludes the proof. $\square$

6 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 (σ_1), (2) where a signature is produced at most β times (σ_β), for some moderate β, and (3) where a signature is produced essentially an unlimited number of times (σ_∞). This has been an ongoing discussion at the NIST emailing lists [5] and in a recent note about SPHINCS+ [Kö22], showing interests in schemes with a limited number of signatures. Following NIST recommendations, the third parameter set should allow for up to 2^{64} signatures.

For simplicity, we let the distribution D be the uniform ternary distribution in each of the three cases. This can naturally be extended to higher module dimensions and use module variants of these problems[6] for more flexibility, and there are several flavors of secret and noise distributions offering other trade-offs between security and compactness. We use the BGV scheme described in Section 3 for the underlying additive homomorphic encryption scheme, and the homomorphic trapdoor commitment scheme by Damgård et al. [DOTT21] to commit to the shares in the first round of the signing protocol. Finally, one would have to use the Katsumata transform [Kat21] to achieve straight-line extractability in the protocol, which adds essentially two ring elements per proof, a small overhead compared to the total amount of communication.

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 small signatures and public

[5] See `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/LUczQNCw7HA`.
[6] See [LS15] for more details about the hardness of problems over module lattices

Comm.	σ_1	y_1	Π_1	σ_β	y_β	Π_β
Size	1.6 KB	2.6 KB	$\approx$ 750 KB	4.4 KB	6.4 KB	$\approx$ 750 KB

Comm.	σ_∞	y_∞	Π_∞	σ_{triv}	y_{triv}	Π_{triv}
Size	11.5 KB	13.6 KB	$\approx$ 1.5 MB	7.3 KB	6.6 KB	2.4 KB

Table 1. Estimated sizes of $(3, 5)$-threshold signatures σ, public keys y and per party communication in the interactive signing protocol Π for settings: 1) where the a signature σ_1 is only produced once, 2) where a signature σ_β is produced at most $\beta = 365$ times, and 3) where a signature σ_∞ can be produced essentially an unlimited number of times. These parameters achieve at least 128 bits of R-SIS and R-LWE security. We compare to the trivial case where a public key y_{triv} consists of five Dilithium NIST level *II* [LDK+20] public keys of roughly 1.3 KB each and σ_{triv} consists of three signatures of roughly 2.4 KB each. We note that in an optimistic setting, we can reduce the communication by more than 50% when all signers are honest; see Section 7 for details.

keys using our techniques, while also acknowledging that the total amount of communication in the protocol is quite large. Furthermore, in this section, we assume a trusted setup and only focus on the signing protocol, not the key generation. We summarize the results in Table 1.

6.1 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 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_r with standard deviation $\sigma_r = \nu \cdot \sqrt{N}$, which is a common choice of parameters, see e.g. [LNS21]. Then, with high probability, the ℓ_∞ norm of $\boldsymbol{r}$ are bounded by $B_r = 4 \cdot \sigma_r$. Each signer needs to 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 $B_z = 2 \cdot t \cdot B_r \cdot \sqrt{N}$ in our signature scheme. We finally set $\nu = 14$ (so that $|\bar{\mathcal{C}}| > 2^{128}$) to get $\sigma_r \approx 2^{8.8}$, $B_{\boldsymbol{z}} \approx 2^{18.4}$ and $q \approx 2^{20}$. This leads to more than 128 bits of $\mathsf{R\text{-}SIS}$ security when inserting the parameters into the equation above and more than 128 bits of $\mathsf{R\text{-}LWE}$ security according to the LWE-estimator [APS15] when only revealing one signature per key.

The absolute norm of each coefficient in $\boldsymbol{z}$ is bounded by $4 \cdot t \cdot \sigma_r$, and this leads to a signature size of $\boldsymbol{z}$ of at most $N \log_2(4 \cdot t \cdot \sigma_r)$ bits, which is roughly 1.6 KB with the given parameters. The public verification key y is of size $N \log_2 q$ bits (we assume that $\boldsymbol{a}$ can be generated by a random oracle from a short seed, as done by Dilithium [DKL+18]), which result in a key of size 2.6 KB. We ignore the cost of ρ by assuming that each ρ_j can be generated from a short seed, and that sending t such seeds (instead of sum ρ) is small compared to $\boldsymbol{z}$.

What remains is to estimate the size of the 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], and instantiate this with module dimension one to match our security assumptions. Then the randomness consists of three short elements in R_q and a commitment consists of two uniform elements in R_q. We conclude that com_i is of size 7.7 KB.

For the encryption scheme, we need q to be the plaintext modulus and need to choose a larger modulus Q for the ciphertexts. We also increase the dimension to $N' = 4096$ to achieve proper security. We still use ternary secrets and noise values, and follow the analysis from Aranha et al. [ABGS23, Appendix F] which gives us the correctness requirement $B_{\mathsf{Dec}} + B_{\mathsf{TDec}} < Q/2$, where B_{Dec} is the noise-bound in the ciphertext $\mathsf{ctx}_{\boldsymbol{z}}$ when the encrypted per-signature randomness shares are combined with the challenge and the encrypted signing key, and B_{TDec}, is the bound of the noise added during the distributed decryption procedure. We use a zero-knowledge proof to prove that the randomness used to generate the BGV ciphertexts is bounded, and we can again use the exact proofs by Lyubashevsky et al. [BLNS21, LNP22] to achieve this. Then the noise in $\mathsf{ctx}_{\boldsymbol{z}}$ is bounded by $B_{\mathsf{Dec}} = q \cdot (\nu + t) \cdot (2 \cdot N' + 1) \approx 2^{37}$. Furthermore, $B_{\mathsf{TDec}} = q \cdot t \cdot B_E$ where E is the noise added to the partial decryptions and B_E is the bound we can prove in a zero-knowledge proof to ensure that E is bounded (and that we do not get any decryption error when we use Comb to extract the signature). It follows that $B_E = ||E||_\infty = B_{\mathsf{Dec}} \cdot 2^{\mathsf{sec}}$ and some statistical security parameter sec. A common choice is $\mathsf{sec} = 40$. This leads to $B_E \approx 2^{77}$. Then we get $B_{\mathsf{TDec}} \approx 2^{99}$ and can set $Q \approx 2^{100}$, which gives more than 128 bits of $\mathsf{R\text{-}SIS}$ and $\mathsf{R\text{-}LWE}$ security. We conclude that $\mathsf{ctx}_{\boldsymbol{r}}$ is of size $2 \cdot N' \log_2 Q$ bits, that is, 103 KB, and that decryption shares ds are $N' \log_2 Q$ bits, or 52 KB.

The ciphertext $\mathsf{ctx}_{\boldsymbol{r}}$ consists of three noise values that are bounded with respect to B_r, so the proof $\pi_{\boldsymbol{r}}$ is of size ≈ 200 KB using exact proofs [LNP22].

The decryption proof π_{ds} consists of a commitment, a proof of linearity (see Aranha et al. [ABGS23, Appendix B.1]) and a proof of boundedness (see Hough et al. [HSS23] or [LNP22]). The commitment over R_Q is of size as a ciphertext (153 KB), the proof of linearity is of size $2N' \log_2 B_r$ bits, that is, 12 KB and the proof of boundedness is roughly of size ≈ 200 KB. Then π_{ds} is of 565 KB.

The total amount of communication per party in the protocol is ≈ 750 KB.

6.2 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. One concrete example is FIDO login[7] where the signing key is secret shared over several devices to ensure both that the user can log in in the case of lost devices and that no one can impersonate the user even with a limited number of stolen devices.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 $\sigma_r = \nu \cdot \sqrt{N \cdot 365}$ (see [ASY22, Theorem 4.1]) to ensure that the signature does not leak too much information when producing at most 365 signatures. We furthermore set $N = 2048$, and get $\sigma_r \approx 2^{13.6}$, $\mathbf{B}_z \approx 2^{23.7}$ and $q \approx 2^{25}$ to ensure at least 128 bits of security with respect to the hardness of $\mathsf{R\text{-}SIS}$ and $\mathsf{R\text{-}LWE}$. The elements $\boldsymbol{z}$ are of size 4.4 KB and the verification key y is of size 6.4 KB.

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

6.3 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 $\sigma_r = \nu \cdot \sqrt{N \cdot 2^{64}} \approx 2^{41.3}$ when $N = 2048$ and $B_{\boldsymbol{z}} \approx 2^{51.4}$, and hence, we need to set $q \approx 2^{53}$ to get exactly 128 bits of security. We get signatures of size 11.5 KB, and verification keys of size 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 twice compared to the previous settings.

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

7 Extensions

Concrete instantiation and implementation. We have only given a rough estimate of public key and signature sizes and a conservative communication estimate in this paper to show that we can achieve practical signatures in a real-world scenario. We leave it as follow-up work to concretely instantiate the key generation protocol and the proof systems, and to implement the scheme.

Compression. The most efficient lattice-based public keys, ciphertexts, signatures, and zero-knowledge proofs 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.

Pre-processing keys. Although some bits of the signing key $\mathbf{s}$ leak with each signature, we do not need to run the whole KGen at each key renewal. The decryption key shares sk_i are bound by the security guarantees of $\mathcal{E}$, and hence, they can be reused even if $\mathbf{s}$ needs to be refreshed. This also allows $\mathbf{s}$ to be generated in batches and replaced independently from sk_i.

Reducing communication cost via optimistic approach. Assuming non-malicious behavior we can omit sending w_j and π_{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 π_{ds_j} in a third round as proof of correct computation. In an honest execution, this saves $|\mathcal{U}|(|w_j|+|\pi_{\mathsf{ds}_j})|$ bits per party, significantly reducing the overall communication. For one-time signatures, 395 KB of 750 KB per party communication is due to π_{ds}, which can be removed with this approach.

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. We leave this as future work.

Adding robustness. Our protocol does not guarantee *robustness* where t honest users are able to produce a verifying signature regardless of the malicious attempts of remaining signers. We rely on the availability of a broadcast channel like robust Schnorr signatures. This by itself cannot guarantee robustness as the broadcast channel is often trickier in practice. It is possible to replace the broadcast channel with alternative solutions like the recent work by Ruffing et al. [RRJ+22] which is a generic wrapper that adds robustness to any semi-interactive threshold signature with the help of a semi-trusted coordinator.

References

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

ACD+19. Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via galois rings. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 471–501. Springer, Heidelberg, December 2019.

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*, volume 5978 of *LNCS*, pages 201–218. Springer, Heidelberg, February 2010.

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*, volume 11035 of *LNCS*, pages 368–385. Springer, Heidelberg, September 2018.

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, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

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*, volume 7954 of *LNCS*, pages 218–236. Springer, Heidelberg, June 2013.

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, Part II*, volume 12973 of *LNCS*, pages 608–627. Springer, Heidelberg, October 2021.

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. Cryptology ePrint Archive, Paper 2023/016, 2023. `https://eprint.iacr.org/2023/016`.

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, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Heidelberg, August 2022.

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, Part II*, volume 12111 of *LNCS*, pages 266–296. Springer, Heidelberg, May 2020.

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 Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

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*, pages 743–773, Cham, 2023. Springer Nature Switzerland.

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*, pages 716–747, Cham, 2023. Springer Nature Switzerland.

CKM23. Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 678–709, Cham, 2023. Springer Nature Switzerland.

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 *LNCS*, pages 128–153. Springer, Heidelberg, December 2019.

DDE$^+$23. Morten Dahl, Daniel Demmler, Sarah Elkazdadi, Arthur Meyre, Jean-Baptiste Orfila, Dragos Rotaru, Nigel P. Smart, Samuel Tap, and Michael Walter. Noah's ark: Efficient threshold-fhe using noise flooding. Cryptology ePrint Archive, Paper 2023/815, 2023. `https://eprint.iacr.org/2023/815`.

DDO$^+$01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

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*, volume 12238 of *LNCS*, pages 382–400. Springer, Heidelberg, September 2020.

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 TCHES*, 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. IEEE Computer Society Press, May 2019.

DOK$^+$20. Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 654–673. Springer, Heidelberg, September 2020.

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, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Heidelberg, May 2021.

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.

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*, volume 12505 of *LNCS*, pages 45–64. Springer, Heidelberg, November / December 2020.

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*, pages 1109–1123. ACM Press, November 2022.

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*, pages 1179–1194. ACM Press, October 2018.

GGN16. Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 156–174. Springer, Heidelberg, June 2016.

GHL22. Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, Heidelberg, May / June 2022.

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*, pages 230–240. Tsinghua University Press, January 2010.

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 ACM STOC*, pages 197–206. ACM Press, May 2008.

HSS23. Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. Concrete ntru security and advances in practical lattice-based electronic voting. Cryptology ePrint Archive, Paper 2023/933, 2023. `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, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 580–610, Virtual Event, August 2021. Springer, Heidelberg.

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*, volume 12804 of *LNCS*, pages 34–65. Springer, Heidelberg, October 2020.

Kö22. Stefan Kölbl. A note on SPHINCS$^+$ parameter sets. Cryptology ePrint Archive, Paper 2022/1725, 2022.

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/round-3-submissions`.

Lin17. Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 613–644. Springer, Heidelberg, August 2017.

Lin22. Yehuda Lindell. Simple three-round multiparty Schnorr signing with full simulatability. Cryptology ePrint Archive, Paper 2022/374, 2022.

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*, pages 1837–1854. ACM Press, October 2018.

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, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Heidelberg, August 2022.

LNS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 215–241. Springer, Heidelberg, May 2021.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, June 2015.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

MR09. Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

NIS22. NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process, September 2022.

Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Heidelberg, August 2010.

RRJ$^+$22. Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. Roast: Robust asynchronous schnorr threshold signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2551–2564, New York, NY, USA, 2022. Association for Computing Machinery.

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/round-3-submissions`.

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

VKH23. Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption, 2023.