Olingo: Threshold Lattice Signatures with DKG and Identifiable Abort*

Kamil Doruk Gur[1**], Patrick Hough[2], Jonathan Katz[3], Caroline Sandsbråten[1], and Tjerand Silde[1]

[1] Norwegian University of Science and Technology
{kamil.d.gur, caroline.sandsbraten, tjerand.silde}@ntnu.no
[2] Universität der Bundeswehr München, patrick.hough@unibw.de
[3] Google, jkatz2@gmail.com

Abstract. We present Olingo, a framework for threshold lattice signatures that is the first to offer all desired properties for real-world implementations of quantum-secure threshold signatures: small keys and signatures, low communication and round complexity, non-interactive online signing, distributed key generation (DKG), and identifiable abort.

Our starting point is the framework by Gur, Katz, and Silde (PQCrypto 2024). We change the underlying signature scheme to Raccoon (Katsumata et al, Crypto 2024), remove the trapdoor commitments, extend the scheme from two to three rounds, and then apply numerous improvements and optimizations to achieve all the above properties. We provide detailed proofs of security for our new framework and present concrete parameters and benchmarks.

At the 128-bit security level, for up to 1024 parties and supporting 2^{60} signatures, our scheme has 4.4 KB public keys and 11.3 KB signatures; while signing requires communication of 852 KB per party using the LaBRADOR proof system (Beullens and Seiler, Crypto 2023). An optimistic non-interactive version of our scheme requires only 76 KB communication per party.

Keywords: Threshold Signatures, Lattices, DKG, Identifiable Abort

1 Introduction

In a t-out-of-n threshold scheme [Des88, DF90], a key is split into n shares such that t of those shares are needed to perform some cryptographic operation. Threshold schemes ensure both improved security and resilience: an adversary who corrupts fewer than t parties cannot carry out the cryptographic operation on its own; at the same time, even if $n-t$ parties are unavailable, the remaining t parties can still perform the desired computation.

* This is the full version of the paper published at ACM CCS 2026 [GHK+26].
** Work mainly done at the University of Maryland.

Threshold signatures have seen increased interest in recent years (e.g., [BL22, KG20, DKLs24]) due, in part, to their application to blockchain wallets. Indicative of this interest, NIST recently issued a call for multi-party threshold schemes [BP25]. Most existing threshold signatures are, however, based on classical hardness assumptions, leaving them vulnerable to cryptographically relevant quantum computers. This has recently motivated works on (plausibly) *quantum-secure* threshold signatures, especially constructions based on lattice assumptions [GKS24, DKM+24, EKT24, KRT24, CATZ24, BKL+25, ZT25, PN25, PKN+25]. For practical deployments, however, such schemes should satisfy the following requirements:

- Security based on standard hardness assumptions.
- Scalability to a large number of users n (NIST calls for up to $n = 1024$) with flexible choice of threshold t.
- Key and signature sizes comparable to those of standardized (non-threshold) signature schemes.
- Support for *non-interactive online signing* so that, following an offline, message-independent, pre-processing phase, parties can sign a message in only one round of online communication. The total communication and round complexity should also be low.
- Distributed key generation (DKG) to avoid trusted setup.
- Identifiable abort so a misbehaving party can be identified in case signature generation fails.

Recent work on lattice-based threshold signatures shows schemes satisfying some, but not all, of the above requirements. (See Table 1.) The most efficient schemes [DKM+24, EKT24, KRT24, CATZ24, BKL+25, ZT25, PN25, PKN+25] lack DKG or identifiable abort (usually both), while the only scheme offering both[4] [GKS24] does not have non-interactive online signing and is significantly less efficient.

1.1 Our Contributions

We propose a new framework for lattice-based threshold signatures called Olingo.[5] When applied to the Raccoon signature scheme [dPKPR24], it is concretely efficient and offers DKG and identifiable aborts based on standard lattice assumptions. In more detail:

- Our public keys and signatures have size 4.4 KB and 11.3 KB, respectively, comparable to the original Raccoon scheme [dPKPR24] and the most efficient lattice-based threshold schemes [DKM+24, EKT24, KRT24, BKL+25,

[4] Although the scheme allows for identifying misbehaving parties, the authors do not claim (and no proof has been given) that the scheme satisfies the formal requirements of identifiable abort.

[5] The Olingo is a mammal of the Procyonidae family, which among others includes the Raccoon. Olingos are not masked and are thus *identifiable*.

PN25, PKN+25]. In particular, we achieve the smallest signature sizes of all lattice-based designs as well as the smallest combined size of public key and signature.
– Our signing protocol has three rounds of communication, and offers non-interactive online signing.[6]
– Generating a signature requires 852 KB total (142 KB online) communication when using the LaBRADOR proof system [BS23]. An optimistic version of our scheme only requires 76 KB of communication in the last round.
– Our scheme is compatible with an efficient, 2-round DKG protocol that we describe as part of this work. Our DKG protocol relies on a new scheme for verifiable secret sharing in the lattice setting that we believe is of independent interest.
– Our signing protocol (though not our DKG protocol) has identifiable aborts, so misbehaving parties can be identified and then rebooted and/or excluded from further executions.
– Our implementation of the scheme shows that it is practical, with signing taking $(3.1 + 0.7 \cdot t)$ seconds total[7] (only 1.4 seconds for the non-interactive online phase).

A drawback of Olingo is that it is only proven secure in a sequential setting, in contrast to some more-recent schemes [PKN+25, DKM+24, GKS24, BKL+25] that offer concurrent security. However, our protocol can be made concurrently secure if we sacrifice online signing.

We compare Olingo to prior work in Table 1 and detail the parameters and benchmarks in Section 7. We also provide a more in-depth discussion of related work in Appendix A. We include two versions of Olingo: the complete communication and an optimistic version which is more efficient when all parties are honest. For completeness we also include an optimistic version of [PKN+25].

1.2 Technical Overview

Our starting point is the work of Gur, Katz, and Silde [GKS24] (the GKS-framework), which supports distributed key generation (DKG) and identifiable aborts (IA). While secure, the concrete instantiation in GKS suffers from poor communication efficiency, complicated trapdoor commitments, and that both rounds of interaction requires knowledge of the message. At a high level, we add a single communication round to the signing protocol, which simultaneously enables non-interactive online signing and allows for substantial performance improvements.

The base signature scheme underlying our new construction is Raccoon [dP-KPR24]. Fix a ring R_q. The signing key is a short vector $\mathbf{s} \in R_q^{\ell+k}$, and the

[6] Only the final round of our protocol depends on the message being signed. Our security definition, however, requires that messages being signed are independent of the offline phase. In practical deployments we believe this level of security suffices.

[7] Note that the LaZer library [LSS24] does not yet support larger parameter sets, so our benchmarks are conducted with the LNP [LNP22] proof system instead.

Table 1. Comparison of lattice-based threshold signature schemes. COM is communication per party; all sizes are in KB. "—" means the data is not provided by the authors. The number of rounds is given as "offline + online rounds," so "$X + 1$" rounds means that the scheme has non-interactive online signing. ID-A means identifiable abort. **S** means "sequential" and **C** means "concurrent" for the security model. Items with † are independent, concurrent work.

Scheme	PK	SIG	COM	Rounds	Users	DKG	ID-A	Model
[GKS24]	13.6	46.6	3000	$0+2$	5	✓	(✓)‡	**C**
[DKM+24]	3.9	12.7	28	$0+3$	1024	✗	✗	**C**
[EKT24]	5.5	10.8	538	$1+1$	1024	✗	✗	**S**
[EKT24, ZT25]	8.7	30.9	767	$1+1$	1024	✗	✗	**S**
[KRT24]	3.9	12.7	—	$2+3$	1024	✗	✗	**S**
[CATZ24, ZT25]	42.1	144.5	1240	$1+1$	32	✗	✗	**S**
[BKL+25]	4.5	13.4	629	$1+1$	1024	✗	✗	**C**
[PN25]	2.6	2.7	5.6	$2+1$	8	✗	✗	**S**
[PKN+25]	3.9	12.7	$88+6.4t$	$0+3$	1024	✗	✓	**C**
[PKN+25] (Optimistic)	3.9	12.7	28	$0+3$	1024	✗	✓	**C**
Olingo	4.4	11.3	852	$2+1$ $(0+3)$	1024	✓	✓	**S** (**C**)
Olingo (Optimistic)	4.4	11.3	76	$2+1$ $(0+3)$	1024	✓	✓	**C**
[BCd+25, CdPE+26] †§	1.3	2.4	1050	2+4	6	✓	✗	**C**
[BCd+25, BCP+26] †	2.8	11.3	73	1+1	64	✗	✓	**C**
[dENP25]† Repl. (Ramp)	3.5	11.9	25	0+3	16 (1024)¶	✓	✓	**C**

‡ The scheme allows for identifying aborting parties, but no proof that it satisfies identifiable abort is given.
§ The reported values are for signatures produced with $1/2$ success probability.
¶ The scheme has two variants based on the underlying secret sharing algorithm: Replicated and Ramp. Ramp secret share variant is given in parentheses and has intermediate thresholds which leak secret key share material.

verification key consists of a matrix $\bar{\mathbf{A}} := [\mathbf{A} \mid \mathbf{I}] \in R_q^{k \times (\ell+k)}$ (for uniform $\mathbf{A}$) together with $\mathbf{y} := \bar{\mathbf{A}}\mathbf{s}$. A simplified (interactive) execution of the signature scheme proceeds as follows:

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

A key challenge in distributing this protocol arises from the large coefficients induced by Shamir secret sharing over R_q. The GKS-framework circumvents this issue using a linearly homomorphic encryption (LHE) scheme. Rather than sharing $\mathbf{s}$ explicitly, the parties hold a ciphertext $\mathsf{ctx}_{\mathbf{s}} = \mathsf{Enc}(\mathbf{s})$ under a public encryption key pk, while the corresponding decryption key sk is generated via DKG and shared in a t-out-of-n fashion. Any subset $\mathcal{U} \subseteq [n]$ of t parties can jointly produce a signature as follows:

1. Each party $i \in \mathcal{U}$ samples a bounded vector $\mathbf{r}_i \in R_q^{\ell+k}$, computes $\mathbf{w}_i := \bar{\mathbf{A}}\mathbf{r}_i$, an encryption $\mathsf{ctx}_{\mathbf{r}_i} := \mathsf{Enc}(\mathbf{r}_i)$, and a commitment $\mathsf{com}_i := \mathsf{Com}(\mathbf{w}_i)$, and broadcasts $(\mathsf{ctx}_{\mathbf{r}_i}, \mathsf{com}_i)$.
2. The parties compute $\mathsf{com} := \sum_{i\in\mathcal{U}} \mathsf{com}_i$, derive the challenge $c := H(\mathbf{w})$ for $\mathbf{w} := \sum_i \mathbf{w}_i$, and homomorphically compute $\mathsf{ctx}_{\mathbf{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_i \mathsf{ctx}_{\mathbf{r}_i}$. Each party then broadcasts its threshold decryption share of $\mathsf{ctx}_{\mathbf{z}}$.
3. Given t valid decryption shares, the parties reconstruct $\mathbf{z}$, compress it, and output the signature $(c, \mathbf{z})$.

In GKS, correctness and robustness additionally require opening the commitments in the final round, and the sum of these openings becomes part of the signature, increasing its size and modifying verification. Moreover, to prevent selective abort and related attacks [DEF+19, DOTT21], the commitment key must depend on the message, forcing the message to be fixed before signing and ruling out non-interactive online signing. Finally, zero-knowledge proofs are required throughout to ensure active security.

Our main technical simplification is to remove these trapdoor commitments. This allows us to use the standard (non-rejecting) Raccoon scheme with essentially original parameters [dPKPR24], apply state-of-the-art compression techniques [BG14, LDK+22, DKM+24], and eliminate message-dependent setup. Consequently, the resulting protocol admits a non-interactive online signing phase. We further replace statistical noise-drowning with a refined noise analysis based on the MLWE with Hints assumption [KLSS23] and Rényi divergence [BLR+18], significantly improving efficiency. Finally, we optimize the DKG and signing procedures by tightly integrating efficient secret sharing [Sha79, CD17] with modern zero-knowledge proofs for lattices [BS23, LNP22, BDL+18]. These changes result in a substantially revised framework, for which we provide complete security proofs.

The resulting protocol, Olingo, is a three-round threshold signing framework:

1. Each party $i \in \mathcal{U}$ samples $\mathbf{r}_i \in R_q^{\ell+k}$, computes $\mathbf{w}_i := \bar{\mathbf{A}}\mathbf{r}_i$, and broadcasts a hash $h_i := \mathsf{H}(i, \mathbf{w}_i)$.
2. After receiving all hashes, each party broadcasts $\mathbf{w}_i$ together with $\mathsf{ctx}_{\mathbf{r}_i} := \mathsf{Enc}(\mathbf{r}_i)$.
3. The parties compute $\mathbf{w} := \sum_i \mathbf{w}_i$, derive $c := H(\mathbf{w})$, and homomorphically compute $\mathsf{ctx}_{\mathbf{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_i \mathsf{ctx}_{\mathbf{r}_i}$. Each party broadcasts its threshold decryption share of $\mathsf{ctx}_{\mathbf{z}}$.
4. Given t valid shares, each signer reconstructs $\mathbf{z}$ and outputs the signature $(c, \mathbf{z})$.

Bias resistance of the Fiat–Shamir challenge is ensured by the commit-then-open mechanism on the $\mathbf{w}_i$ values: since each $\mathbf{w}_i$ is fixed before being revealed, no adversarial party can influence the challenge $c := H(\sum_i \mathbf{w}_i)$. To ensure active security, we add zero-knowledge proofs of correct computation in the last two rounds.

Finally, the roles of the main building blocks are as follows. Linearly homomorphic encryption enables joint evaluation of linear signing operations without

revealing $\mathbf{s}$. The DKG ensures that no single party ever learns the decryption key, while still allowing any qualified subset to sign. The identifiable abort mechanism allows honest parties to detect and exclude misbehaving signers, ensuring robustness against denial-of-service attacks. Overall, these design choices yield a significantly more efficient and practical construction, as confirmed by our comparison in Table 1.

1.3 Independent and Concurrent Work

Three independent and concurrent works were recently posted online. del Pino et al. [dENP25] present an efficient threshold signature scheme from lattices with both a DKG and identifiable aborts based on new secret sharing techniques. Whilst [dENP25] achieves very similar signature and public key sizes and impressive communication cost (only 25 KB per party), their design has two main drawbacks compared to Olingo. Firstly, their design uses three message-dependent rounds as compared to Olingo's (optimistic) non-interactive design with only one (final) message-dependent round, making it very attractive for low-latency deployment. Secondly, their 'Ramp' variant, accommodating larger groups (> 16), suffers from statistical key-share leakage from colluding subsets of intermediate size. Moreover, this leakage becomes more severe at higher security levels and thus no parameters are given for Ramp at bit-security levels > 128. Bienstock et al. [BdCE+26] showcase efficient MPC techniques to compute pure ML-DSA [LDK+22] signatures in the honest majority setting. Celi et al. [CdPE+26] design and implement an efficient threshold version of ML-DSA with small signature and public key sizes for up to 6 users and provide a distributed key generation. However, they do not achieve identifiable aborts, they require two offline rounds and 4 online rounds, and their per-party communication cost grows fast even for small groups with around 1MB already for the 6-party setting. Borin et al. [BCP+26] provide a FROST-like threshold signature design based on Raccoon. Their signature and public key sizes are competitive, as is per-party communication cost at 73 KB. They also achieve a single offline round and one online round. Their design accommodates up to 64 users and identifiable aborts but does not support a distributed key generation. We note that initial versions of [CdPE+26] and [BCP+26] were presented in a combined work [BCd+25]; after the split, the threshold ML-DSA scheme [CdPE+26] was accepted at USENIX Security 2026. However, having now split into two works, both schemes have undergone some changes and we present the most up-to-date characteristics in Table 1. Where implementations are given for these works, we compare timing benchmarks in Section 7.2.

2 Background

2.1 Communication Model

We assume a synchronous communication model with private and authenticated links between each pair of parties and a broadcast channel [Fel87, GRR98]. The

synchronous property means that protocols proceed in a sequence of rounds. In each round, a party performs some local computation, sends messages (if any) through the private and authenticated links, and broadcasts some information over the broadcast channel. By the end of the round, it also receives all messages sent or broadcast by the other parties in the same round.

We consider t-bounded, static, active adversaries. That is, adversaries who may corrupt up to t parties, from the outset of a protocol, with the ability to read all incoming messages and internal values of that party and determine its output messages. We always assume a *rushing* adversary; i.e., in any round the malicious parties receive the messages sent by the honest parties before deciding their own.

2.2 Cyclotomic Rings, Norms, and Rounding

Consider the cyclotomic rings $R = \mathbb{Z}[x]/\phi$ and $R_q = \mathbb{Z}_q[x]/\phi$, where $\phi = (x^N + 1)$ for N an integer power of 2 and q a prime. Elements in both rings are polynomials of degree at most $N-1$, with those in the latter ring having coefficients between $-(q-1)/2$ and $(q-1)/2$. We denote elements of $\mathbb{Z}$ and R by lower-case letters, vectors in R^k by bold lower-case letters, and matrices in $R^{(k\times\ell)}$ by bold upper-case letters. We use the standard notation $\|\cdot\|$ for norms of ring elements or vectors, specifying with a prefix which norm is meant (1, 2, or ∞). For a matrix $\mathbf{A} \in R^{k\times\ell}$, $\|\mathbf{A}\| \leq B$ denotes that *each entry* of the matrix has norm bounded by B.

We let $\mathcal{C}_\nu = \{c \in R_q : \|c\|_\infty = 1, \|c\|_1 = \nu\} \subset R_q$ to be the set of polynomials with coefficients in $\{-1, 0, 1\}$ and exactly ν non-zero coefficients, and define $\bar{\mathcal{C}}_\nu = \{c - c' : c, c' \in \mathcal{C}_\nu, c \neq c'\}$. Let $\lfloor\cdot\rceil : \mathbb{R} \to \mathbb{Z}$ denote the rounding-up operation. We define the lower κ bit dropping operation $\lfloor\cdot\rceil_{q'} : \mathbb{Z}_q \to \mathbb{Z}_{q'}$ such that $\lfloor x\rceil_{q'} := \lfloor x/2^\kappa\rceil + q'\mathbb{Z}$ where $q' = \lfloor q/2^\kappa\rfloor$ for $\kappa \neq 0 \in \mathbb{N}$. Like other ring operations, the bit dropping operation extends into vectors naturally by applying everything coefficient-wise. We also define the map coefficient embedding by the map $\Phi : R_q \to \mathbb{Z}_q^N$ which maps elements of R_q to their coefficient vectors. This naturally extends to vectors and matrices of elements in R_q.

2.3 Lattice Assumptions

The security of our scheme relies on computational problems over structured lattices, namely Module Short Integer Solution (MSIS), Self-Target Module Integer Short Solution (SelfTargetMSIS), Module Learning with Errors (MLWE), and Hint Module Learning with Errors (H-MLWE) over some fixed polynomial ring R_q implicitly defining the parameters q and d. We present SelfTargetMSIS and H-MLWE here, and refer to Appendix B.3 for MSIS and MLWE.

Definition 1 (SelfTargetMSIS [DKL+18]). *Let k, ℓ be positive integers and $0 < \eta \ll q$. Let $\mathsf{H} \colon R_q^k \times \{0,1\}^* \to \mathcal{C}_\nu$ be a cryptographically secure hash function modeled as a random oracle. Then, given $(\mathbf{A}, \mathbf{t}) \in R_q^{k\times\ell} \times R_q^k$, the Self-Target MSIS problem asks an adversary $\mathcal{A}$ to find $(\mathbf{z}, \mu) \in R_q^\ell \times \{0,1\}^*$ such that $0 <$*

$\|\mathbf{z}\|_2 \leq \eta$ *and* $\mathsf{H}([\mathbf{A} \mid \mathbf{t}] \cdot \mathbf{r}, \mu) = c$*, where* $\mathbf{r} = [\mathbf{z}\ c]^\top$*.* $\mathcal{A}$ *is said to have advantage* $\epsilon_{\mathsf{STMSIS}}$ *in solving* $\mathsf{SelfTargetMSIS}_{k,\ell,\eta}$ *if*

$$\Pr\left[\begin{matrix}\mathsf{H}([\mathbf{A} \mid \mathbf{t}] \cdot \mathbf{r}, \mu) = c \\ \wedge\ 0 < \|\mathbf{z}\|_2 \leq \eta\end{matrix} \;\middle|\; \mathbf{A} \leftarrow R_q^{k \times \ell};\ \left(\mathbf{r} = \begin{bmatrix}\mathbf{z} \\ c\end{bmatrix}, \mu\right) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathbf{A}, \mathbf{t})\right] \geq \epsilon_{\mathsf{STMSIS}}.$$

Definition 2 (H-MLWE **[KLSS23]**)**.** *Let* k, ℓ, Q *be positive integers,* χ_1 *and* χ_2 *be probability distributions over* R_q*, and* $\mathcal{C}$ *be a subset of* R_q*. The Hint-MLWE problem* $\mathsf{H\text{-}MLWE}_{k,\ell,\chi_1,\chi_2,Q}$ *then asks an adversary* $\mathcal{A}$ *to distinguish between the following two cases:*

1. $(\mathbf{A}, \mathbf{As}, (c_i, \mathbf{z}_i)_{i\in[Q]})$ *for* $\mathbf{A} \leftarrow R_q^{k\times(\ell+k)}$*,*
2. $(\mathbf{A}, \mathbf{b}, (c_i, \mathbf{z}_i)_{i\in[Q]})$ *for* $\mathbf{A} \leftarrow R_q^{k\times(\ell+k)}$*,* $\mathbf{b} \leftarrow R_q^k$*,*

where $\mathbf{s} \leftarrow \chi_1^{\ell+k}$*,* $c_i \leftarrow \mathcal{C}$ *for* $i \in [Q]$*, and* $\mathbf{z}_i := c_i \cdot \mathbf{s} + \mathbf{y}_i$ *where* $\mathbf{y}_i \leftarrow \chi_2^{\ell+k}$ *for* $i \in [Q]$*.*

$\mathcal{A}$ *has advantage* $\epsilon_{\mathsf{H\text{-}MLWE}}$ *in solving* $\mathsf{H\text{-}MLWE}_{k,\ell,\chi_1,\chi_2,Q}$ *if*

$$\begin{aligned}\Big|\Pr\left[b = 1 \mid \begin{matrix}\mathbf{A} \leftarrow R_q^{k\times(\ell+k)}; \mathbf{s} \leftarrow \chi_1^{\ell+k}; c_i \leftarrow \mathcal{C}; \\ \mathbf{y}_i \leftarrow \chi_2^{\ell+k}; \mathbf{z}_i := c_i\mathbf{s} + \mathbf{y}_i \text{ for } i \in [Q]; \\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{As}, (c_i, \mathbf{z}_i)_{i\in[Q]})\end{matrix}\right] \\ - \Pr\left[b = 1 \mid \begin{matrix}\mathbf{A} \leftarrow R_q^{k\times(\ell+k)}; \mathbf{b} \leftarrow R_q^k; \\ \mathbf{s} \leftarrow \chi_1^{\ell+k}; c_i \leftarrow \mathcal{C}; \mathbf{y}_i \leftarrow \chi_2^{\ell+k} \\ \mathbf{z}_i := c_i\mathbf{s} + \mathbf{y}_i \text{ for } i \in [Q]; \\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b}, (c_i, \mathbf{z}_i)_{i\in[Q]})\end{matrix}\right]\Big| \geq \epsilon_{\mathsf{H\text{-}MLWE}}.\end{aligned}$$

2.4 The BDLOP Commitment Scheme

We recall the commitment scheme introduced in [BDL+18] which allows one to commit to vectors $\mathbf{m} \in R_q^{\ell_m}$ of 'fully fledged' elements in R_q. The scheme is parameterized by the system parameters $\mathsf{sp}_{\mathsf{Com}}$ containing the natural numbers $k_{\mathsf{MSIS}}, k_{\mathsf{MLWE}}, \ell_m, q, \nu \in \mathbb{N}$ and real number $\sigma_{\mathsf{Com}} \in \mathbb{R}^+$.

Challenge Space. Let ν be such that $\binom{N}{\nu} \cdot 2^\nu > 2^\lambda$. We define the challenge space $\mathcal{C}_\nu := \{c \in R_q \mid \|c\|_\infty \leq 1 \wedge \|c\|_1 \leq \nu\}$ and its difference set $\bar{\mathcal{C}}_\nu := \{c - c' \mid c, c' \in \mathcal{C}_\nu \wedge c \neq c'\}$.

$\mathsf{BDLOP.Setup}(\mathsf{sp}_{\mathsf{Com}})$: On input parameters $\mathsf{sp}_{\mathsf{Com}}$, samples uniform $\mathbf{A}_1, \mathbf{A}_2 \in R_q^{k_{\mathsf{MSIS}} \times (k_{\mathsf{MSIS}}+k_{\mathsf{MLWE}}+\ell_m)} \times R_q^{\ell_m \times (k_{\mathsf{MSIS}}+k_{\mathsf{MLWE}}+\ell_m)}$ outputs the public commitment key $\mathsf{pk}_{\mathsf{Com}}$ defined as:

$$\mathsf{pk}_{\mathsf{Com}} := \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{bmatrix}$$

$\mathsf{BDLOP.Com}(\mathsf{pk}_{\mathsf{Com}}, \mathbf{m})$: On input a public commitment key $\mathsf{pk}_{\mathsf{Com}}$ and a vector $\mathbf{m}$ in $R_q^{\ell_m}$, samples a vector $r_{\mathbf{m}} \leftarrow \mathcal{D}_{\sigma_{\mathsf{Com}}}^{(k_{\mathsf{MSIS}}+k_{\mathsf{MLWE}}+\ell_m)}$ and computes the commitment:

$$\mathsf{com}_{\mathbf{m}} := \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} r_{\mathbf{m}} + \begin{bmatrix} \mathbf{0} \\ \mathbf{m} \end{bmatrix} \in R_q^{k_{\mathsf{MSIS}}+\ell_m}.$$

It then outputs $\mathsf{com}_{\mathbf{m}}$ and the opening $\mathsf{d} = (\mathbf{m}, r_{\mathbf{m}}, 1)$.

$\mathsf{BDLOP.Open}(\mathsf{pk}_{\mathsf{Com}}, \mathsf{com}, \mathsf{d})$: On input a public commitment key $\mathsf{pk}_{\mathsf{Com}}$, the commitment com and the opening $\mathsf{d} = (\mathbf{m}, r_{\mathbf{m}}, f)$ where $f \in \bar{\mathcal{C}}$. It verifies:

$$f \cdot \mathsf{com} \stackrel{?}{=} \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} r_{\mathbf{m}} + \begin{bmatrix} \mathbf{0} \\ f \cdot \mathbf{m} \end{bmatrix} \in R_q^{k_{\mathsf{MSIS}}+\ell_m}.$$

and $\|r_{\mathbf{m}}\|_2 \leq t_2 \cdot \sigma_{\mathsf{Com}} \cdot \sqrt{N}$ (norms are calculated entry-wise). It outputs 1 if the relations hold and 0 otherwise.

Lemma 1 (Commitment Security [BDL+18]). *Define* $k := k_{\mathsf{MSIS}}+\ell_m$, $\ell := k_{\mathsf{MSIS}}+k_{\mathsf{MLWE}}+\ell_m$, *and* $\beta := 16 \cdot \sigma_{\mathsf{Com}} \cdot \sqrt{\nu \cdot N}$. *The BDLOP commitment scheme is hiding if* $\mathsf{MLWE}_{k,\ell,\mathcal{D}_{\sigma_{\mathsf{Com}}}}$ *is hard and it is binding if* $\mathsf{MSIS}_{k_{\mathsf{MSIS}},\ell,\beta}$ *is hard.*

2.5 Threshold Signatures

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

- $\mathsf{KGen}_{\mathcal{S}}$ is an interactive protocol run by n users that takes as input n and a threshold t. Each user either aborts or outputs a (common) public key $\mathsf{pk}_{\mathcal{S}}$, (common) auxiliary data aux, and a secret key share $\mathsf{sk}_{\mathcal{S}}^{(i)}$.
- $\mathsf{Sign}_{\mathcal{S}}$ is an interactive protocol run by a set $\mathcal{U}$ of t users. Each party begins holding their secret key share, auxiliary data aux, and a message μ. Each user either aborts or outputs a signature Ψ.
- $\mathsf{Vf}_{\mathcal{S}}$ takes as input the public key $\mathsf{pk}_{\mathcal{S}}$, a message μ, and a signature Ψ, and outputs 1 iff the signature is valid.

Threshold Correctness We say that a threshold signature scheme $\mathcal{S}$ is correct if for all thresholds (t, n) and security parameter λ we have that

- for all $\left(\mathsf{pk}_{\mathcal{S}}, \{\mathsf{aux}_i, \mathsf{sk}_{\mathcal{S}}^{(i)}\}_{i\in[n]}\right) \leftarrow \mathsf{KGen}_{\mathcal{S}}(t, n, \lambda)$
- for all $\Psi \leftarrow \mathsf{Sign}_{\mathcal{S}}\left(\mathcal{U}, \mu, \mathsf{pk}_{\mathcal{S}}, \{\mathsf{aux}_j, \mathsf{sk}_{\mathcal{S}}^{(j)}\}_{j\in\mathcal{U}}\right)$, where $\mathcal{U} \subset [n]$ and $|\mathcal{U}| \geq t$,

we have that $1 \leftarrow \mathsf{Vf}_{\mathcal{S}}(\mathsf{pk}_{\mathcal{S}}, \mu, \Psi)$.

The security notion we consider is threshold unforgeability:

Threshold Unforgeability $\mathcal{S}$ is (T, ϵ)-threshold unforgeable if the probability of any adversary $\mathcal{A}$ running in time T succeeding in experiment $\mathsf{Exp}_{\mathcal{S}}^{\mathsf{ts\text{-}uf\text{-}cma}}(\mathcal{A})$ as depicted in Figure 1 is at most ϵ.

$\mathsf{Exp}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{S}}(\mathcal{A})$

1. $\mathcal{A}$ outputs a set of corrupted users $\mathcal{C}$ of size at most $t-1$. Let $\mathcal{H} := [n] \setminus \mathcal{C}$, and $\mathcal{M} = \emptyset$.
2. $\mathcal{A}$ (controlling parties in $\mathcal{C}$) runs $\mathsf{KGen}_{\mathcal{S}}$ with the parties in $\mathcal{H}$. The view of the honest parties defines a public verification key $\mathsf{pk}_{\mathcal{S}}$ and keys $\{\mathsf{sk}^{(i)}_{\mathcal{S}}\}_{i\in\mathcal{H}}$ and auxiliary data aux.
3. $\mathcal{A}$ may make an arbitrary number of "signing queries" of the form $(\{\mathcal{U}^{(\tau)}\}, \{\mu^{(\tau)}\})$, answered as follows:
   - $\mathcal{M} := \mathcal{M} \cup \{\mu^{(\tau)}\}$
   - Let $\mathcal{H}_{\mathcal{U}} := \mathcal{H} \cap \mathcal{U}^{(\tau)}$. For $i \in \mathcal{H}_{\mathcal{U}}$, compute $\Psi^{(\tau)} \leftarrow \mathsf{Sign}_{\mathcal{S}}(\mathcal{U}^{(\tau)}, \mathsf{aux}, \mu^{(\tau)}, \mathsf{sk}^{(i)}_{\mathcal{S}})$ or abort
   - Return the signature $\Psi^{(\tau)}$ to $\mathcal{A}$.
4. $\mathcal{A}$ finally outputs a message-signature pair (μ^*, Ψ^*) and succeeds if $1 = \mathsf{Vf}_{\mathcal{S}}(\mathsf{pk}_{\mathcal{S}}, \mu^*, \Psi^*)$ and $\mu^* \notin \mathcal{M}$

Fig. 1. Experiment $\mathsf{Exp}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{S}}(\mathcal{A})$ parameterized by threshold t and total number of users n.

Moreover, we make the security against identifiable aborts explicit based on the definition of [IOZ14]:

Security with Identifiable Aborts $\mathcal{S}$ has (T, ϵ)-security with identifiable aborts if the probability of any adversary $\mathcal{A}$ running in time T succeeding in experiment $\mathsf{Exp}^{\mathsf{ID\text{-}AB}}_{\mathcal{S}}(\mathcal{A})$ as depicted in Figure 2 is at most ϵ.

We remark that this definition of security with identifiable aborts considers only the signing protocol $\mathsf{Sign}_{\mathcal{S}}$, and not the distributed key generation protocol $\mathsf{KGen}_{\mathcal{S}}$.[8]

2.6 Coding Theory

Here we recall the basic notions and results from coding theory needed in our verifiable secret sharing.

Definition 3 (Linear Error Correcting Code). *We define an $[n, k, d]$ code $\mathcal{C}$ is a linear error correcting code over $\mathbb{Z}_q$ of length n, dimension k and minimum distance d. Its dual code $\mathcal{C}^{\perp}$ is the vector space consisting of all vectors $c^{\perp} \in \mathbb{Z}_q^n$ such that $\langle c, c^{\perp} \rangle = 0$ for all $c \in \mathcal{C}$.*

We will focus on Reed-Solomon codes as following.

Definition 4 (Reed-Solomon (RS) Code). *A code $\mathcal{C}$ is a Reed-Solomon (RS) code if it is of the following form*

$$\mathcal{C} := \{(p(1), p(2), \ldots, p(n)) \ : \ p(x) \in \mathbb{Z}_q[x], \deg(p(x)) \leq k-1\}$$

where $p(x)$ is any polynomial in $\mathbb{Z}_q[x]$ of degree at least $k-1$.

[8] This will also be the case when we introduce a similar notion for the homomorphic encryption scheme.

$\mathsf{Exp}^{\mathsf{ID\text{-}AB}}_{\mathcal{S}}(\mathcal{A})$

1. $\mathcal{A}$ outputs a set of corrupted users $\mathcal{C}$ of size at most $t-1$. Let $\mathcal{H} := [n] \setminus \mathcal{C}$, and $\mathcal{M} = \emptyset$.
2. $\mathcal{A}$ (controlling parties in $\mathcal{C}$) runs $\mathsf{KGen}_{\mathcal{S}}$ with the parties in $\mathcal{H}$. The view of the honest parties defines a public verification key $\mathsf{pk}_{\mathcal{S}}$ and keys $\{\mathsf{sk}^{(i)}_{\mathcal{S}}\}_{i\in\mathcal{H}}$ and auxiliary data aux.
3. $\mathcal{A}$ may make an arbitrary number of "signing queries" of the form $(\mathcal{U}^{(\tau)}, \{\mu^{(\tau)}\})$, answered as follows:
   - Let $\mathcal{H}_{\mathcal{U}} := \mathcal{H} \cap \mathcal{U}^{(\tau)}$. For $i \in \mathcal{H}_{\mathcal{U}}$, compute $\Psi^{(\tau)} \leftarrow \mathsf{Sign}_{\mathcal{S}}(\mathcal{U}^{(\tau)}, \mathsf{aux}, \mu^{(\tau)}, \mathsf{sk}^{(i)}_{\mathcal{S}})$
   - Return the signature $\Psi^{(\tau)}$ to $\mathcal{A}$.
4. At any time $\mathcal{A}$ can make a subset of corrupt parties $\mathcal{CS}^{(\tau)} \subseteq \mathcal{C}$ arbitrarily deviate from protocol execution.
5. Each honest party i then either continues with the protocol execution or aborts and outputs a set of parties $\mathcal{CS}^{(\tau)}_i$ that they think are cheating.
6. $\mathcal{A}$ succeeds if any of the following is true:
   - $\exists i, i' \in \mathcal{H}$ s.t. $\mathcal{CS}^{(\tau)}_i \neq \mathcal{CS}^{(\tau)}_{i'}$.
   - $\exists i \in \mathcal{H}$ s.t. $\mathcal{CS}^{(\tau)}_i \not\subseteq \mathcal{C}$ or $i' \in \mathcal{CS}^{(\tau)}_i$ but $i' \notin \mathcal{CS}^{(\tau)}$.
   - $\exists i \in \mathcal{H}, \exists i' \in \mathcal{C}$ s.t. $i' \notin \mathcal{CS}_i$ but $i' \in \mathcal{CS}^{(\tau)}$.
   - $\mathsf{Vf}_{\mathcal{S}}(\mathsf{pk}_{\mathcal{S}}, \mu^{(\tau)}, \Psi^{(\tau)}) = 0$ but $\mathcal{CS}^{(\tau)} = \emptyset$.

Fig. 2. Experiment $\mathsf{Exp}^{\mathsf{ID\text{-}AB}}_{\mathcal{S}}(\mathcal{A})$ parameterized by threshold t and total number of users n.

We naturally extend this notation to accommodate polynomials $\mathbf{P} \in R_q^{m_1 \times m_2}[x]$. That is, polynomials whose coefficients are matrices of elements in R_q and whose domain is $\mathbb{N}$ and range is $R_q^{m_1 \times m_2}$. The we can define an $[n, k, s]$ RS code $\mathcal{C}$ as

$$\mathcal{C} := \{(\mathbf{P}(1), \ldots, \mathbf{P}(n)) \ : \ \mathbf{P}(x) \in R_q^{m_1 \times m_2}[x], \deg(\mathbf{P}(x)) \leq k-1\} \tag{1}$$

Note, Equation (1) can be viewed as capturing $N \cdot m_1 \cdot m_2$ codes with a more compact notation. We will use the following (adapted) lemma from [CD17]. Note that the dual code remains defined over $\mathbb{Z}_q^n$.

Lemma 2 ([CD17] (adapted)). *If $\mathbf{S}_1, ..., \mathbf{S}_n \in R_q^{m_1 \times m_2} \setminus \mathcal{C}$, and $c^{\perp}$ is chosen randomly in $\mathcal{C}^{\perp}$ then the following holds:*

$$\Pr\left[\sum_{i=1}^{n} c^{\perp}\mathbf{S}_i = \mathbf{0}\right] = \frac{N \cdot m_1 \cdot m_2}{q}.$$

3 Verifiable Secret Sharing

Verifiable secret sharing (VSS), introduced by Chor et al. [CGMA85], can be viewed as an interactive protocol for secret sharing that is secure against malicious behavior by all parties (including the dealer) during both the sharing and reconstruction phases. One of the most common designs combines the distribution (by a dealer) of committed (or encrypted) shares along with zero-knowledge

proofs that the commitments contain correct shares. Our lattice-based design follows this framework and is thus closer in design to the constructions of Feldman [Fel87] and Pedersen [Ped92] in the discrete logarithm setting.

Gentry et al. [GHL22] proposed a publicly-verifiable secret sharing (PVSS) scheme from lattice-based encryption and Bulletproofs [BBB+18]. Here, publicly verifiable means that anyone can verify that shares have been generated and distributed correctly. While lattice-based encryption provides post-quantum secrecy with respect to the ciphertexts, Bulletproofs are built from classical assumptions, so the proof aspect opens the door to a quantum adversary. Recently, Espitau et al. [ENP24] proposed a fully lattice-based VSS. Their work introduced a new 'flood-and-submerse' technique to build a robust VSS. Unfortunately, security is only guaranteed for an honest majority ($n \geq 2t + 1$) and is thus unsuitable in our setting.

We propose an efficient (fully) lattice-based VSS, secure for arbitrary thresholds ($t < n$). Our design takes inspiration from the above schemes, whereby a dealer commits to the set of shares and broadcasts these with a non-interactive proof of opening. They then distribute the shares (openings) among the shareholders via secure channels. Shareholders can identify a faulty dealing by checking that their share is a valid opening for their commitment.

In order to make the non-interactive proof of opening efficient, we employ an observation of McEliece and Sarwate [MS81]: Shamir secret sharing [Sha79] is equivalent to encoding the secret using a Reed-Solomon error correcting code. In particular, the vector of secret shares forms a codeword of a Reed-Solomon code. As observed by Cascudo and David in [CD17], to prove a Shamir secret sharing, it suffices to prove that the inner product of this codeword with a randomly selected word from the dual code is zero. In [CD17], the discrete log setting lends itself naturally to allowing anyone to compute the inner product of a random codeword of their choice with the set of committed shares. This naturally preserves the secrecy of shares as the check is performed in the exponent. In the lattice setting, however, there is no such structure. The solution we propose, is to demonstrate the inner product check in zero-knowledge where the random codeword is provided as part of the random challenge of the proof protocol. Anyone verifying such a proof can be confident with overwhelming likelihood that the committed shares are correct.

3.1 VSS Model and Definitions

We assume the existence of secure channels between the dealer and each shareholder. We also assume, for generality, the dealer's input lies in $R_q^{m_1 \times m_2}$ where $\mathbb{Z}_q$ contains $\{1, ..., n\}$.

We present a 1-round (non-interactive) VSS design with a slight relaxation that the output of the honest parties can be a pre-defined abort symbol. This variant is called *weak* verifiable secret sharing (WSS) [KKK08, Rab94, RB89]. Looking ahead, our threshold signature scheme uses a 1-round DKG for which robustness cannot be guaranteed; that is, a well-formed output cannot be guaranteed for honest parties. Thus, WSS is sufficient in our setting.

We define a weak verifiable secret sharing (WSS) scheme.

Definition 5 (Weak verifiable secret sharing). *A two-phase (*sharing *and* reconstruction*) protocol involving n parties $\mathcal{P} := \{\mathcal{P}_1, ..., \mathcal{P}_n\}$ in which a dealer $D \in \mathcal{P}$ holds an initial input $\mathbf{S} \in R_q^{m_1 \times m_2}$, is a* WSS *tolerating $t-1$ malicious parties if the following properties hold with overwhelming probability in the presence of any (PPT) adversary controlling at most $t-1$ parties:*

Privacy *If the dealer is honest at the end of the sharing phase, then at the end of this phase the joint view of the malicious parties is independent of the dealer's input $\mathbf{S}$.*

Correctness *If the dealer is honest, with input $\mathbf{S}$, then at the end of the sharing phase each honest party $\mathcal{P}_i$ outputs a value $\mathbf{S}_i \in R_q^{m_1 \times m_2}$ with $\mathbf{S}_i \neq \perp$. If $\geq t$ honest parties subsequently run the reconstruction phase, all output $\mathbf{S}$.*

Weak commitment *For a malicious dealer, at the end of the sharing phase the joint view of the honest parties defines a value $\mathbf{S}^* \in R_q^{m_1 \times m_2} \cup \perp$ (which can be computed in polynomial time from this view) such that if any set of $\geq t$ parties subsequently runs the reconstruction phase, either every honest party outputs $\mathbf{S}^*$ or every honest party outputs $\perp$. Moreover, assuming each honest party outputs some $\mathbf{S}_i \neq \perp$ at the end of the sharing phase, there exists a polynomial $\mathbf{P}(x) \in R_q^{m_1 \times m_2}[X]$ of degree at most t such that:*

1. *$\mathbf{S}_i = \mathbf{P}(i)$ for every honest party $\mathcal{P}_i$.*
2. *if $\mathbf{S}^* \neq \perp$, then $\mathbf{S}^* = \mathbf{P}(0)$.*

3.2 Overview of our VSS Construction

Our (P)VSS construction relies on three main building blocks.

Shamir Secret Sharing. Shamir Secret Sharing [Sha79] is used by the dealer to generate shares of its input $\mathbf{S}$.

BDLOP Commitment. The dealer broadcasts commitments to the shares using the lattice-based BDLOP [BDL+18] commitment scheme described in Section 2.4.

Non-interactive Zero Knowledge. We employ a novel combination of zero-knowledge proofs to enforce the verifiability property of the secret sharing. The dealer begins by committing to the secret $\mathbf{S}$, as well as all shares $\mathbf{S}_i$ using the BDLOP commitment scheme. The dealer then proves that

π_1 $\mathbf{S}$ has small norm using the LNP framework in [LNP22]

π_2 The set of shares $\{\mathbf{S}_i\}$ forms a codeword of a corresponding Reed-Solomon code. This is done using the Π_{LIN} protocol from [BDL+18]. Explicitly, we prove that for a random element $c^\perp = (c_1, \ldots, c_n) \in \mathbb{Z}_q^n$ of the dual code,

$$\sum_{i=1}^{n} c_i^\perp \cdot \mathbf{S}_i = 0.$$

π_3 A t-sized subset of the shares reconstructs correctly to the secret. Again, this uses Π_{LIN}. Explicitly, given a set $\mathcal{S} \subset [n]$ of size t, we prove that

$$\mathbf{S} = \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} \mathbf{S}_i,$$

where the $\lambda_i^{\mathcal{S}}$ are the Lagrange coefficients.

Crucially, we note that proofs π_2 and π_3 ensure that the committed objects form a correct sharing. Note that proof π_2 argues that the $\{\mathbf{S}_i\}$ are a correct sharing of *something* while proof π_3 argues that some t-sized subset of the shares correctly reconstructs $\mathbf{S}$. Crucially, the two together prove that *any* t-sized subset of the shares will reconstruct the secret.

We note that our design could be made publicly verifiable if needed. In particular, the dealer could employ a PKE scheme such as CRYSTALS-Kyber [SAB+22] to encrypt shares to shareholders. These ciphertexts could be broadcast along with proof that the underlying plaintexts satisfy a correct sharing. This verifiable encryption design was used in [GHL22]. For our applications, a commit-and-prove design is sufficient and makes for a simpler presentation.

We do however allow for an additional 'complaint' round in which parties may broadcast an abort symbol should they find that their received share is not a valid opening of the dealer's corresponding commitment.

We present the VSS scheme Π_{VSS} in Figure 3.

3.3 Security of VSS

We now show the theorem our WSS scheme provides and provide proofs for this claim in three lemmas in Appendix C.1.

Theorem 1 (VSS Security). *The* WSS *scheme* Π_{VSS} *offers correctness, privacy, and weak commitment.*

Protocol Π_{VSS}

Let $\mathcal{C}$ be an error-correcting code corresponding to the t-out-of-n-threshold Shamir Secret Sharing scheme and let $\mathcal{C}^{\perp}$ be its dual code. Let H be a random oracle mapping $\{0,1\}^*$ to $\mathcal{C}^{\perp}$. Protocol Π_{VSS} is run between n parties $\mathcal{P}_1, ..., \mathcal{P}_n$, a dealer D and an external verifier V. Let $\mathcal{D}_{\mathbf{B}}$ be a distribution over $R_q^{m_1 \times m_2}$.

Setup: The $\mathsf{BDLOP.Setup}(\mathsf{sp}_{\mathsf{Com}})$ commitment setup algorithm is run to generate the public commitment key $\mathsf{pk}_{\mathsf{Com}}$ of the BDLOP commitment scheme, in practice generated from a seed.

Distribution: The dealer D first samples the secret $\mathbf{S} \leftarrow \mathcal{D}_{\mathbf{B}}$ and does the following

- Sample $\mathbf{P} \leftarrow R_q^{m_1 \times m_2}[X]$ of degree $< t$, with $\mathbf{P}(0) = \mathbf{S}$.
- For $i \in [n]$, compute the i'th share as $[\![\mathbf{S}]\!]_i := \mathbf{P}(i)$.
- For $i \in [n]$, $(\mathsf{com}_{[\![\mathbf{S}]\!]_i}, \mathsf{d}) \leftarrow \mathsf{BDLOP.Com}(\mathsf{pk}_{\mathsf{Com}}, [\![\mathbf{S}]\!]_i)$[a].
- Compute $c^{\perp} := \mathsf{H}(\{(\mathsf{com}_{[\![\mathbf{S}]\!]_i})\}_{i \in [n]})$.
- Generate $\pi_{\mathsf{VSS}} := (\pi_1, \pi_2, \pi_3)$, where π_2 is computed w.r.t. $c^{\perp}$ and π_3 is generated w.r.t. $\mathcal{S} := \{1, ..., t\}$ (w.l.o.g).
- Broadcast π_{VSS} and send $([\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i})$ to party i over a secure channel.

Upon (secure) receipt of $\mathsf{d} := ([\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i}, 1)$, party i checks that $c^{\perp} := \mathsf{H}(\{(\mathsf{com}_{[\![\mathbf{S}]\!]_i})\}_{i \in [n]})$ and then verifies the proof π_{VSS} (each constituent proof) and that d is a correct opening of $\mathsf{com}_{[\![\mathbf{S}]\!]_i}$, outputting abort ($\perp$) if either fails.

Reconstruction: Suppose a subset $\mathcal{S}$ of size t wish to reconstruct the secret $\mathbf{S}$. If verification of π_{VSS} failed in the sharing phase, $\mathcal{P}_i$ will output $\perp$ here also. Assuming no party outputs $\perp$, each party $\mathcal{P}_i \in \mathcal{S}$ broadcasts $([\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i})$. Upon receiving all shares and openings from parties $\mathcal{P}_j$ ($j \neq i$), $\mathcal{P}_i$ checks that $\mathsf{Open}(\mathsf{com}_{[\![\mathbf{S}]\!]_j}, \mathsf{d} := ([\![\mathbf{S}]\!]_j, r_{[\![\mathbf{S}]\!]_j}, 1)) = 1$ for all $j \in \mathcal{S} \setminus \{i\}$, aborting otherwise. The secret $\mathbf{S}$ is then reconstructed as

$$\mathbf{S} = \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} [\![\mathbf{S}]\!]_i.$$

[a] $\mathsf{com}_{[\![\mathbf{S}]\!]_i}$ is the horizontal concatenation of commitments to the columns of $[\![\mathbf{S}]\!]_i$

Fig. 3. Verifiable secret sharing protocol Π_{VSS}

4 Threshold Homomorphic Encryption

We present the underlying threshold linearly homomorphic encryption scheme. This follows a similar structure to the one used in [GKS24] but is now defined over module lattices with the aim of enabling more efficient parameters later on. Since we employ a distributed key generation and (non-interactive) threshold decryption, our encryption scheme is defined by the tuple of algorithms $(\mathsf{KGen}_{\mathcal{E}}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{TDec}, \mathsf{Comb})$. We begin by describing the distributed key generation algorithm $\mathsf{KGen}_{\mathcal{E}}$ before defining the remaining algorithms, with the omission of Eval which we will define according to its specific function in our threshold signature scheme.

4.1 Distributed Key Generation

We define a t-out-of-n distributed key generation protocol based on a module lattice encryption scheme based on CRYSTALS-Kyber [SAB+22] and BGV [BGV12], closely following the construction by Gur et al. [GKS24]. Due to this similarity, we present the key generation algorithm in its full, actively secure form.

Our threshold homomorphic encryption scheme is defined with respect to the public parameters $\mathsf{sp} = (k, \ell, \hat{k}, \hat{\ell}, m, q, \hat{q}, N, \sigma_{\mathsf{LWE}})$, where $k, \ell, \hat{k}, \hat{\ell}, m, q, \hat{q}, N \in \mathbb{N}$, $\sigma_{\mathsf{LWE}} \in \mathbb{R}^+$, and σ_{LWE} defines the discrete Gaussian distributions $\mathcal{D}_{\mathbf{B}}$ and $\mathcal{D}_{\mathsf{ctx}}$ (used in the secret sharing). It is helpful to think of (k, ℓ, q) as being smaller and defining an 'inner layer' of signing keys (and plaintext space) and $(\hat{k}, \hat{\ell}, m, \hat{q})$ as being larger and defining an 'outer layer' for a ciphertext space.

We assume all parties agree on a common $\mathbf{A}_{\mathcal{E}} \in R_{\hat{q}}^{\hat{k} \times \hat{\ell}}$ which could be the output of a random oracle on some session ID or nonce. Let H be a hash function acting as a random oracle where $\mathsf{H} : \mathbb{N} \times R_{\hat{q}}^{\hat{k} \times m} \to \{0,1\}^{\kappa}$. We describe the distributed key generation algorithm from the point of view of a party $\mathcal{P}_i$;

1. $\mathcal{P}_i$ samples short $\mathbf{S}_i \leftarrow \mathcal{D}_{\mathbf{B}}^{\hat{\ell} \times m}$ and $\mathbf{E}_i \leftarrow \mathcal{D}_{\mathbf{B}}^{\hat{k} \times m}$, computes $\mathbf{B}_i := \mathbf{A}_{\mathcal{E}} \mathbf{S}_i + q\mathbf{E}_i$ and $h_i \leftarrow \mathsf{H}(i, \mathbf{B}_i)$, and broadcasts h_i.
2. On receipt of $\{h_j\}$ for $j \neq i$, $\mathcal{P}_i$ generates t-out-of-n Shamir secret shares $\{\mathbf{S}_{i,j}\}_{j \in [n]} := \mathsf{Share}_{t,n}(\mathbf{S}_i)$ and $\{\mathbf{E}_{i,j}\}_{j \in [n]} := \mathsf{Share}_{t,n}(\mathbf{E}_i)$, and sets $\mathbf{B}_{i,j} := \mathbf{A}_{\mathcal{E}} \mathbf{S}_{i,j} + q\mathbf{E}_{i,j}$, for $j \in [n]$.
3. $\mathcal{P}_i$ now creates a zero-knowledge proof $\pi^{(i)}_{\mathsf{KGen}_{\mathcal{E}}}$ of witness

$$\mathsf{W} := \left(\mathbf{S}_i, \mathbf{E}_i, \{\mathbf{S}_{i,j}\}_{j \in n}, \{\mathbf{E}_{i,j}\}_{j \in [n]}\right),$$

such that (1) $\mathbf{B}_i$ and $\{\mathbf{B}_{i,j}\}_{j \in [n]}$ are constructed correctly w.r.t. W, (2) $\{\mathbf{S}_{i,j}\}_{j \in n}$ (resp. $\{\mathbf{E}_{i,j}\}_{j \in [n]}$) are a correct t-out-of-n secret sharing of $\mathbf{S}_i$ (resp. $\mathbf{E}_i$), and (3) $\|(\mathbf{S}_i)\|_2, \|(\mathbf{E}_i)\|_2 < t_2 \cdot \mathbf{B} \cdot \sqrt{N}$. (1) is performed using the Π_{LIN} protocol from [BDL+18] whilst (2) and (3) are performed by the proof π_{VSS} as described in Section 3.2. We denote by $\pi^{(i)}_{\mathsf{KGen}_{\mathcal{E}}}$ the collection of these proofs. $\mathcal{P}_i$ samples commitment randomness $\boldsymbol{\rho}_{i,j} \leftarrow \chi_{\rho}$ and commits to $\mathbf{S}_{i,j}$

as $\mathsf{com}_{i,j} := \mathsf{Com}(\mathbf{S}_{i,j}, \boldsymbol{\rho}_{i,j})$.[9] $\mathcal{P}_i$ sends $[\mathbf{S}_{i,j}, \mathbf{E}_{i,j}, \boldsymbol{\rho}_{i,j}]_{j\in[n]}$ to $\mathcal{P}_j$ (via a secure channel) and broadcasts $(\mathbf{B}_i, \{\mathbf{B}_{i,j}, \mathsf{com}_{i,j}\}_{j\in[n]}, \pi^{(i)}_{\mathsf{KGen}_\mathcal{E}})$.

4. $\mathcal{P}_i$ defines $\mathcal{J}_\mathbf{B} := \{j \in \mathcal{U} \setminus \{i\} : h_j \neq \mathsf{H}(j, \mathbf{B}_j) \vee \mathsf{Open}(\mathsf{com}_{j,i}) \neq (\mathbf{S}_{j,i}, \boldsymbol{\rho}_{j,i}) \vee \pi^{(j)}_{\mathsf{KGen}_\mathcal{E}}$ is invalid$\}$. If $\mathcal{J}_\mathbf{B} \neq \emptyset$, $\mathcal{P}_i$ aborts with output $\mathcal{J}_\mathbf{B}$ (and subsequently all parties abort)[10]. Assuming no party aborts, $\mathcal{P}_i$ computes $\mathbf{B}_\varepsilon := \sum_j \mathbf{B}_j$, $\mathbf{S}^{(i)}_\mathcal{E} := \sum_j \mathbf{S}_{j,i}$, and $\boldsymbol{\rho}_i := \sum_j \boldsymbol{\rho}_{i,j}$. $\mathcal{P}_i$ then computes the public commitments $\mathsf{com}_j := \sum_{k\in[n]} \mathsf{com}_{j,k}$. The public key is $\mathsf{pk}_\mathcal{E} := (\mathbf{A}_\varepsilon, \mathbf{B}_\varepsilon, \{\mathsf{com}_j\}_{j\in[n]})$ and secret key $\mathsf{sk}^{(i)}_\mathcal{E} := (\mathbf{S}^{(i)}_\mathcal{E}, \boldsymbol{\rho}_i)$.

4.2 Threshold Decryption

We now describe how threshold decryption is performed. Inheriting the parameter set from the previous section, we additionally define the Gaussian parameter σ_{TDec}. Figure 5 displays the threshold decryption and combination algorithms TDec and Comb respectively. We also include the BGV-style encryption algorithm Enc, encrypting a vector of length m to make ciphertext structure explicit.

4.3 Threshold Security

For completeness we first show correctness and then prove indistinguishability for the module variant of the encryption scheme.

Theorem 2. *Let* $\hat{q}/2 \geq B_{\mathsf{Dec}} + t_\infty \cdot t \cdot \sigma_{\mathsf{TDec}}$ *and* $\sigma_{\mathsf{TDec}} \geq \sqrt{\frac{Q \cdot B_2^2}{q^2}}$ *where* B_2 *and* B_{Dec} *are the* ℓ_2 *and* ℓ_∞ *norm bounds for a correctly generated ciphertext* ctx*. Then the linearly homomorphic threshold decryption described in Section 4 correctly decrypts* ctx*.*

Theorem 3. *Let* $a \in (1, +\infty)$ *and* $\sigma_{\mathsf{TDec}} \geq \sqrt{\frac{Q \cdot B_2^2}{q^2}}$ *where* B_2 *is the* ℓ_2 *norm bound of the noise in a correctly generated ciphertext* ctx*. Let* $\mathsf{MLWE}_{N,q,\sigma_{\mathsf{LWE}}}$ *be* ϵ_{MLWE} *hard, and the module variant of BGV be* $\epsilon_{\text{IND-CPA}}$ *secure. Let the NIZKAoK* $\Pi_{\mathsf{KGen}_\mathcal{E}}$ *be* ϵ_{ZK}*- and* $\epsilon_{\mathsf{extract}}$*-secure, and let* Q_H *denote the number of random oracle queries made by the adversary. Then the linearly homomorphic threshold encryption scheme* $\mathcal{E}$ *described in Section 4 is indistinguishable. More formally, for any PPT adversary* $\mathcal{A}$ *corrupting up to* $t-1$ *parties the advantage in the indistinguishability game can be bounded as:*

$$\mathsf{Adv}^{\text{IND}}_\mathcal{E}(\mathcal{A}) \leq |\mathcal{H}| \cdot \epsilon_{\mathsf{ZK}} + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract}} + \left(2^{\frac{a}{a-1}+2} \cdot e^{\pi a} \cdot \epsilon'\right)^{\frac{a-1}{2a-1}},$$

where $\epsilon' := (2|\mathcal{H}| + |\mathcal{H}_\mathcal{U}| - 1) \cdot \epsilon_{\mathsf{MLWE}} + Q_H/2^\lambda + \epsilon_{\text{IND-CPA}}$.

[9] In practice, $\pi^{(i)}_{\mathsf{KGen}_\mathcal{E}}$ contains such a commitment but we make it explicit in the protocol description since it will be used in our signature scheme. We also note that commitment to a matrix here can be done by committing to the vector of polynomials formed by the vertical concatenation if its columns.

[10] We note that while this provides some level of identifiable aborts, it does not completely satisfy the ID-A property. However, with the use of verifiable encryption we believe this to be possible, though that is left for future work.

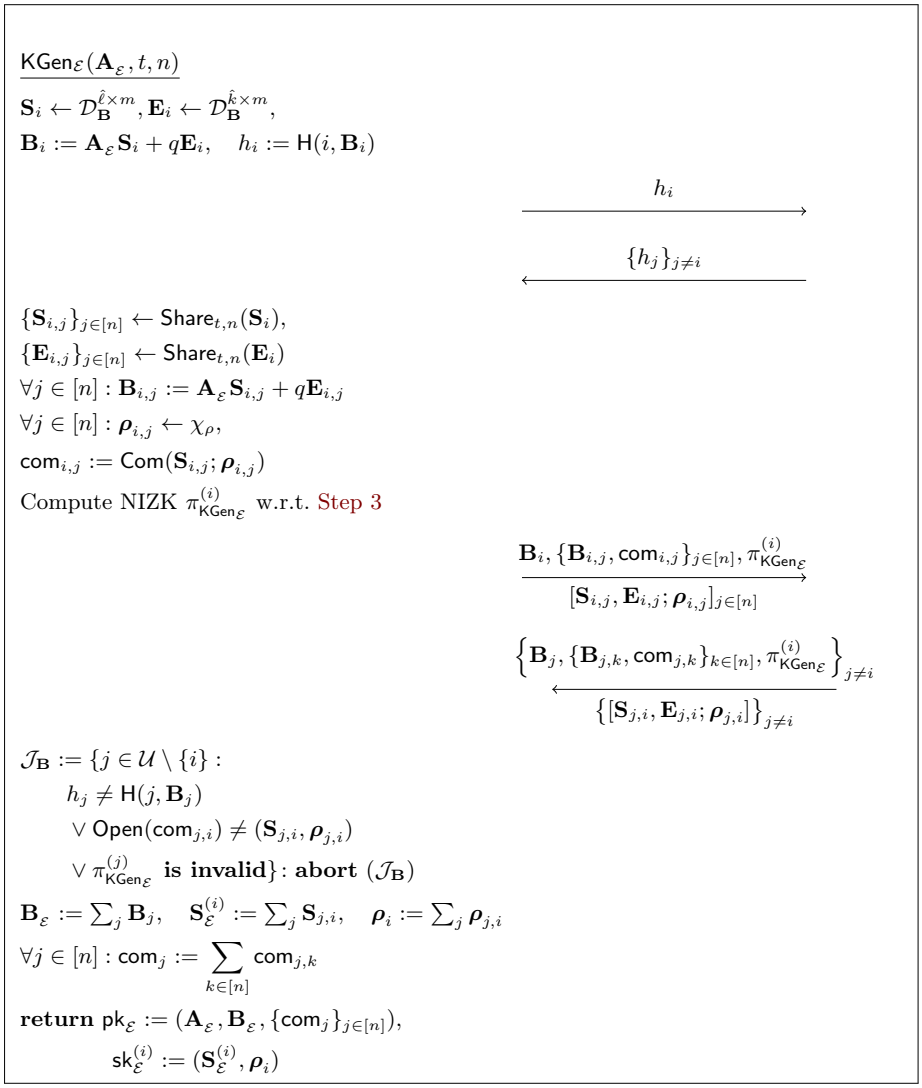

Fig. 4. Actively secure key-generation protocol, from the point of view of $\mathcal{P}_i$. The elements in square brackets with subscript j are sent to $\mathcal{P}_j$ over a private channel.

Encryption $\mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{m})$. Given public key $\mathsf{pk}_{\mathcal{E}} = (\mathbf{A}_{\mathcal{E}}, \mathbf{B}_{\mathcal{E}}) \in R_{\hat{q}}^{\hat{k}\times\hat{\ell}} \times R_{\hat{q}}^{\hat{k}\times m}$ and message $\mathbf{m} \in R_q^m$:

1. Sample randomness $\mathbf{r} \leftarrow \mathcal{D}_{\mathsf{ctx}}^{\hat{k}}$ and noise terms $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathsf{ctx}}^{\hat{\ell}}$ and $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathsf{ctx}}^{m}$.
2. Compute $\mathbf{u} := \mathbf{A}_{\mathcal{E}}^{\top}\mathbf{r} + q \cdot \mathbf{e}_1$ and $\mathbf{v} := \mathbf{B}_{\mathcal{E}}^{\top}\mathbf{r} + q \cdot \mathbf{e}_2 + \mathbf{m}$.
3. Return ciphertext $\mathsf{ctx} = (\mathbf{u}, \mathbf{v})$.

Threshold Decryption $\mathsf{TDec}(\mathsf{sk}_{\mathcal{E}}^{(i)}, \mathsf{ctx}, \mathcal{U})$. Given decryption key share $\mathsf{sk}_{\mathcal{E}}^{(i)} = (\mathbf{S}_{\mathcal{E}}^{(i)}, \rho_i)$ and ciphertext $\mathsf{ctx} = (\mathbf{u}, \mathbf{v})$:

1. Sample noise drowning term $E_i \leftarrow \mathcal{D}_{\sigma_{\mathsf{TDec}}}^{m}$.
2. Compute Lagrange coefficient λ_i for party $\mathcal{P}_i$ w.r.t. $\mathcal{U}$.
3. Compute $\mathsf{ds}_i := \lambda_i {\mathbf{S}_{\mathcal{E}}^{(i)}}^{\top}\mathbf{u} + q \cdot E_i$.
4. Return decryption share ds_i.

Combine $\mathsf{Comb}(\{\mathsf{ds}_i\}_{i\in\mathcal{U}}, \mathsf{ctx})$. Given a set of decryption shares and a ciphertext:

1. Compute $\mathbf{m} := (\mathbf{v} - \sum_{i\in\mathcal{U}} \mathsf{ds}_i) \mod \hat{q} \mod q$.
2. Return plaintext message $\mathbf{m}$.

Fig. 5. The threshold decryption scheme MLWE-Encrypt.

We show the proof in Appendix C.2.

The following theorem asserts that the depicted protocol has security with identifiable aborts. This will be helpful later when we show our threshold signature scheme has security with identifiable aborts. Looking ahead for the signature scheme we assume each ds_i also comes with a proof $\pi_{\mathsf{ds},i}$ that proves correct partial decryption.

Theorem 4. *Let Π_x for language $\mathcal{R}_x$ be NIZKAoKs. Let* Com *be a binding commitment scheme and* H *be a hash function modeled as a random oracle. Then the actively secure threshold homomorphic encryption scheme $\mathcal{E}$ has security with identifiable aborts in the synchronous model with access to a broadcast channel.*

The proof is given in Appendix C.3.

5 Threshold Signature Protocol

5.1 Underlying Rejection-Free Signatures

For the underlying signature we use a variant of the base signature of Threshold Raccoon [DKM+24]:

KGen Samples a uniform $\mathbf{A}_{\mathcal{S}} \in R_q^{k\times\ell}$, a Gaussian secret key $\mathbf{s} \leftarrow \mathcal{D}_{\mathbf{y}}^{\ell}$ and an error term $\mathbf{e} \leftarrow \mathcal{D}_{\mathbf{y}}^{k}$. Finally, computes $\mathbf{y} := \mathbf{A}_{\mathcal{S}}\mathbf{s} + \mathbf{e}$ and rounded version $\mathbf{y}' := \lfloor \mathbf{y} \rceil_{q_{\mathbf{y}}}$ where $q_{\mathbf{y}} = \lfloor q/2^{\kappa_{\mathbf{y}}} \rfloor$, and outputs $\mathsf{sk}_{\mathcal{S}} := \mathbf{s}$ and $\mathsf{pk}_{\mathcal{S}} := (\mathbf{A}_{\mathcal{S}}, \mathbf{y}')$.

Sign Takes as input $(\mathsf{sk}_{\mathcal{S}}, \mathsf{pk}_{\mathcal{S}}, \mu)$ and samples Gaussian masks $\mathbf{r} \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell}$, $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbf{w}}^{k}$. It then computes $\mathbf{w} := \mathbf{A}_{\mathcal{S}}\mathbf{r} + \mathbf{e}'$ alongside a rounded version $\mathbf{w}' := \lfloor \mathbf{w} \rceil_{q_{\mathbf{w}}}$ where $q_{\mathbf{w}} = \lfloor q/2^{\kappa_{\mathbf{w}}} \rfloor$, and derive the challenge $c := \mathsf{H}(\mathsf{pk}_{\mathcal{S}}, \mathbf{w}', \mu) \in \mathcal{C}_{\nu}$.

Finally, it outputs the signature $\Psi := (c, \mathbf{z}, \mathbf{h})$ where $\mathbf{z} := c\mathbf{s} + \mathbf{r}$, $\mathbf{h} := \mathbf{w}' - \mathbf{t}$, and $\mathbf{t} := \lfloor \mathbf{A}_{\mathcal{S}}\mathbf{z} - 2^{\kappa_{\mathbf{y}}} c\mathbf{y}' \rceil_{q_{\mathbf{w}}}$.

Vf Takes as input $(\mathsf{pk}_{\mathcal{S}}, \Psi := (c, \mathbf{z}, \mathbf{h}), \mu)$ and checks that $0 < \|\mathbf{z}, 2^{\kappa_{\mathbf{w}}} \cdot \mathbf{h}\|_2 \leq B$ for some bound B and that $c = \mathsf{H}(\mathsf{pk}_{\mathcal{S}}, \lfloor \mathbf{A}_{\mathcal{S}}\mathbf{z} - 2^{\kappa_{\mathbf{y}}} c\mathbf{y}' \rceil_{q_{\mathbf{w}}} + \mathbf{h}, \mu)$. It outputs 1 if both hold and otherwise outputs 0.

It is easy to see that the signature above is the immediate realization of a digital signature using H-MLWE based zero-knowledge proofs [KLSS23] and the Fiat–Shamir transform [FS87] utilizing generic rounding based [BG14] approaches. Unforgeability of the signature scheme follows from a similar analysis as for the base signature of Threshold Raccoon [DKM+24]. For completeness, we recite the theorem:

Theorem 5 ([DKM+24, Theorem 5.1]). *Let q, N, k, ℓ, $\kappa_{\mathbf{y}}$, $\kappa_{\mathbf{w}}$ be integers, Q be number of signature queries the adversary can perform, and B, B_{MSIS}, $B_{\mathsf{ST\text{-}MSIS}}, B_{\mathsf{H\text{-}MLWE}}$ be bounds where $\sigma \geq \sqrt{\ell + k} \cdot \omega(\sqrt{\log N})$, $B = e^{1/4} \cdot (\nu\sigma_{\mathbf{y}} + \sigma_{\mathbf{w}})\sqrt{N(k+\ell)} + (\nu \cdot 2^{\kappa_{\mathbf{y}}} + 2^{\kappa_{\mathbf{w}}+1}) \cdot \sqrt{Nk}$, $B_{\mathsf{H\text{-}MLWE}} = Q \cdot \nu \cdot \left(1 + \frac{N(\lambda+1+2\log(N))}{\sqrt{Q}}\right)$, $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{n \cdot \sigma_{\mathbf{y}}^2} + \frac{B_{\mathsf{H\text{-}MLWE}}}{t \cdot \sigma_{\mathbf{w}}^2}\right)$, $\sigma_{\mathbf{y}} = 2\sqrt{\ell + k} \cdot \log N$, $\sigma_{\mathbf{w}} = 2\sqrt{B_{\mathsf{H\text{-}MLWE}} \cdot (\ell + k)} \cdot \log N$, $q > B_{\mathsf{MSIS}}\sqrt{Nk} \cdot \omega(\log(Nk))$, $B_{\mathsf{MSIS}} = 2B_{\mathsf{ST\text{-}MSIS}}$, and $B_{\mathsf{ST\text{-}MSIS}} = B + \sqrt{\nu} + (\nu \cdot 2^{\kappa_{\mathbf{y}}} + 2^{\kappa_{\mathbf{w}}+1}) \cdot \sqrt{Nk}$. Then the signature scheme is correct and unforgeable under the $\mathsf{SelfTargetMSIS}_{k,\ell+1,\nu,B_{\mathsf{ST\text{-}MSIS}}}$ and $\mathsf{H\text{-}MLWE}_{k,\ell,\sigma_{\mathbf{y}},\sigma_{\mathbf{w}},Q}$ assumptions. More formally, for a PPT adversary $\mathcal{A}$ against the unforgeability game, Q_{H} and Q_{Sign} queries to random H and signing oracles respectively, there exists adversaries $\mathcal{B}$ and $\mathcal{B}'$ against $\mathsf{H\text{-}MLWE}_{k,\ell,\sigma_{\mathbf{y}},\sigma_{\mathbf{w}},Q}$ and $\mathsf{SelfTargetMSIS}_{k,\ell+1,\nu,B_{\mathsf{ST\text{-}MSIS}}}$ such that the advantage of $\mathcal{A}$ can be bounded as:*

$$\mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{H\text{-}MLWE}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{ST\text{-}MSIS}}(\mathcal{B}') + \frac{(Q_{\mathsf{H}} + 1) \cdot Q_{\mathsf{Sign}}}{2^{N-1}}.$$

5.2 Actively Secure Threshold Signatures

We now present the actively secure, threshold signing protocol, constructed by bootstrapping the passively-secure protocol described in Section 1.2. Let $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ be hash functions acting as random oracles where $\mathsf{H}_0, \mathsf{H}_1 : \mathbb{N} \times R_q^k \to \{0,1\}^{\kappa}$ and H_2 is the same as H in Section 5.1. The key generation and signing protocols are depicted in Figure 6 and Figure 7, respectively.

We modify the design of [GKS24], removing the need for trapdoor commitments at the cost of an extra round of communication and add rounding to the public key and the signature. The motivation for this is to reduce the overall size of resulting signatures.

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

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

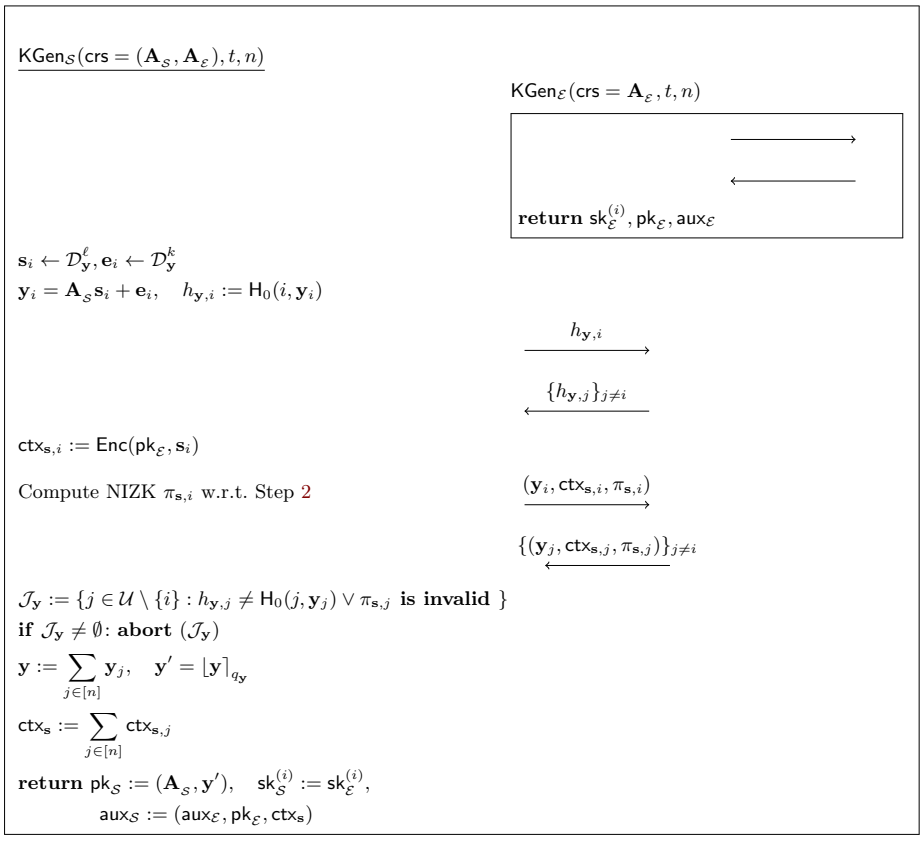

Fig. 6. Actively secure key generation protocol with bit dropping for signer $\mathcal{P}_i$.

2. $\mathcal{P}_i$ then samples short signing key $\mathbf{s}_i \leftarrow D^{\ell}_{\mathbf{y}}$ alongside an error $\mathbf{e}_i \leftarrow D^{k}_{\mathbf{y}}$ and sets $\mathbf{y}_i := \mathbf{A}_{\mathcal{S}}\mathbf{s}_i + \mathbf{e}_i$, hash $h_{\mathbf{y},i} := \mathsf{H}_0(i, \mathbf{y}_i)$ as a commitment to $\mathbf{y}_i$, and broadcasts $h_{\mathbf{y},i}$. Upon receiving $h_{\mathbf{y},j}$ for all $j \neq i$, $\mathcal{P}_i$ encrypts $\mathbf{s}_i$ as $\mathsf{ctx}_{\mathbf{s},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i)$ and computes a NIZK proof $\pi_{\mathbf{s},i}$ to prove that the correct $\mathbf{s}_i$ was used for $\mathbf{y}_i$ and $\mathsf{ctx}_{\mathbf{s},i}$ (see Appendix D for the formal relations), then broadcasts $(\mathbf{y}_i, \mathsf{ctx}_{\mathbf{s},i}, \pi_{\mathbf{s},i})$.
3. Finally, upon receiving $\mathsf{ctx}_{\mathbf{s},j}$, $\pi_{\mathbf{s},j}$ and $\mathbf{y}_j$ from each $j \neq i$, $\mathcal{P}_i$, $\mathcal{P}_i$ defines $\mathcal{J}_{\mathbf{y}} := \{j \in \mathcal{U} \setminus \{i\} : h_{\mathbf{y},j} \neq \mathsf{H}_0(j, \mathbf{y}_j) \vee \pi_{\mathbf{s},j}$ invalid w.r.t $\mathsf{ctx}_{\mathbf{s},j}$ and $\mathbf{y}_j\}$. If $\mathcal{J}_{\mathbf{y}} \neq \emptyset$, $\mathcal{P}_i$ aborts with output $\mathcal{J}_{\mathbf{y}}$. Otherwise, it defines the public key $\mathsf{pk}_{\mathcal{S}} := (\mathbf{A}_{\mathcal{S}}, \mathbf{y}')$, secret key $\mathsf{sk}^{(i)}_{\mathcal{S}} := \mathsf{sk}^{(i)}_{\mathcal{E}}$, and auxiliary information $\mathsf{aux}_{\mathcal{S}} = (\mathsf{aux}_{\mathcal{E}}, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$ where $\mathbf{y}' := \left\lfloor \sum_{j\in\mathcal{U}} \mathbf{y}_j \right\rceil_{q_{\mathbf{y}}}$ for $q_{\mathbf{y}} = \lfloor q/2^{\kappa_{\mathbf{y}}} \rfloor$, $\mathsf{ctx}_{\mathbf{s}} := \sum_{j\in\mathcal{U}} \mathsf{ctx}_{\mathbf{s},j}$.

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

1. Let $\mathcal{P}_i$ be one out of t parties in the set $\mathcal{U}$. Upon receiving the message μ to be signed, and $\mathcal{P}_i$ samples per signatures randomness $\mathbf{r}_i \leftarrow D^{\ell}_{\mathbf{w}}$ and $\mathbf{e}'_i \leftarrow D^{k}_{\mathbf{w}}$. It computes $\mathbf{w}_i := \mathbf{A}_{\mathcal{S}}\mathbf{r}_i + \mathbf{e}'_i$ and $\mathcal{P}_i$ and $h_{\mathbf{w}_i} := \mathsf{H}_1(i, \mathbf{w}_i)$, and broadcasts $h_{\mathbf{w}_i}$.
2. Upon receiving $h_{\mathbf{w}_j}$ for all $j \neq i$, $\mathcal{P}_i$ encrypts the randomness $\mathbf{r}_i$ with the encryption key $\mathsf{pk}_{\mathcal{E}}$ as $\mathsf{ctx}_{\mathbf{r},i} \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{r}_i)$, and computes a proof $\pi_{\mathbf{r},i}$ to prove that $\mathbf{r}_i$ was used for $\mathsf{ctx}_{\mathbf{r},i}$ and $\mathbf{w}_i$. $\mathcal{P}_i$ then sends $(\mathbf{w}_i, \mathsf{ctx}_{\mathbf{r},i}, \pi_{\mathbf{r},i})$ to all $j \in \mathcal{U} \setminus \{i\}$.
3. Upon receiving $(\mathbf{w}_j, \mathsf{ctx}_{\mathbf{r},j}, \pi_{\mathbf{r},j})$ for each $j \in \mathcal{U} \setminus \{i\}$, $\mathcal{P}_i$ defines $\mathcal{J}_{\mathbf{r}} := \{j \in \mathcal{U} \setminus \{i\} : h_{\mathbf{w}_j} \neq \mathsf{H}_1(j, \mathbf{w}_j) \wedge \pi_{\mathbf{r},j}$ invalid w.r.t. $\mathsf{ctx}_{\mathbf{r},j}$ and $\mathbf{w}_j\}$. If $\mathcal{J}_{\mathbf{r}} \neq \emptyset$, $\mathcal{P}_i$ aborts with output $\mathcal{J}_{\mathbf{r}}$. Otherwise it computes $\mathbf{w} := \sum_{i\in\mathcal{U}} \mathbf{w}_i$, consequently $\mathbf{w}' := \lfloor \mathbf{w} \rceil_{q_{\mathbf{w}}}$ where $q_{\mathbf{w}} = \lfloor q/2^{\kappa_{\mathbf{w}}} \rfloor$, and computes the challenge $c := \mathsf{H}_2(\mathsf{pk}_{\mathcal{S}}, \mathbf{w}', \mu)$. It then computes the encryption of the signature as $\mathsf{ctx}_{\mathbf{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j\in\mathcal{U}} \mathsf{ctx}_{\mathbf{r},j}$ and decrypts its share as $\mathsf{ds}_i \leftarrow \mathsf{TDec}(\mathsf{ctx}_{\mathbf{z}}, \mathsf{sk}_i, \mathcal{U})$ and sends ds_i to the parties in $\mathcal{U}$ alongside a proof $\pi_{\mathsf{ds},i}$ of correct decryption showing the correct key share is used.
4. Upon receiving ds_j for all $j \in \mathcal{U} \setminus \{i\}$, $\mathcal{P}_i$ combines the decryptions as $\mathbf{z} := \mathsf{Comb}(\mathsf{ctx}_{\mathbf{z}}, \{\mathsf{ds}_j\}_{j\in\mathcal{U}})$ and defines $\mathcal{J}_{\mathsf{ds}} := \{j \in \mathcal{U} \setminus \{i\} : \mathbf{z} = \perp \vee \pi_{\mathsf{ds},j}$ invalid w.r.t $\mathsf{ctx}_{\mathbf{z}}$ and $\mathsf{ds}_j\}$. If $\mathcal{J}_{\mathsf{ds}} \neq \emptyset$, $\mathcal{P}_i$ aborts with output $\mathcal{J}_{\mathsf{ds}}$. $\mathcal{P}_i$ prepares hint $\mathbf{h} := \mathbf{w}' - \mathbf{t}$ where $\mathbf{t} := \lfloor \mathbf{A}_{\mathcal{S}}\mathbf{z} - 2^{\kappa_{\mathbf{y}}} c\mathbf{y}' \rceil_{q_{\mathbf{w}}}$, and outputs the signature $\Psi := (c, \mathbf{z}, \mathbf{h})$.

$\mathsf{Vf}_{\mathcal{S}}$: Upon receiving $\psi := (c, \mathbf{z}, \mathbf{h})$ and μ, checks that $\|\mathbf{z}, 2^{\kappa_{\mathbf{w}}} \cdot \mathbf{h}\| \leq B_{\mathbf{z}}$, computes $\mathbf{w}^* := \mathbf{h} + \lfloor \mathbf{A}_{\mathcal{S}}\mathbf{z} - 2^{\kappa_{\mathbf{y}}} c\mathbf{y}' \rceil_{q_{\mathbf{w}}}$, derives $c^* := \mathsf{H}_2(\mathsf{pk}_{\mathcal{S}}, \mathbf{w}^*, \mu)$, then finally outputs 1 if and only if checks hold and $c = c^*$, and 0 otherwise.

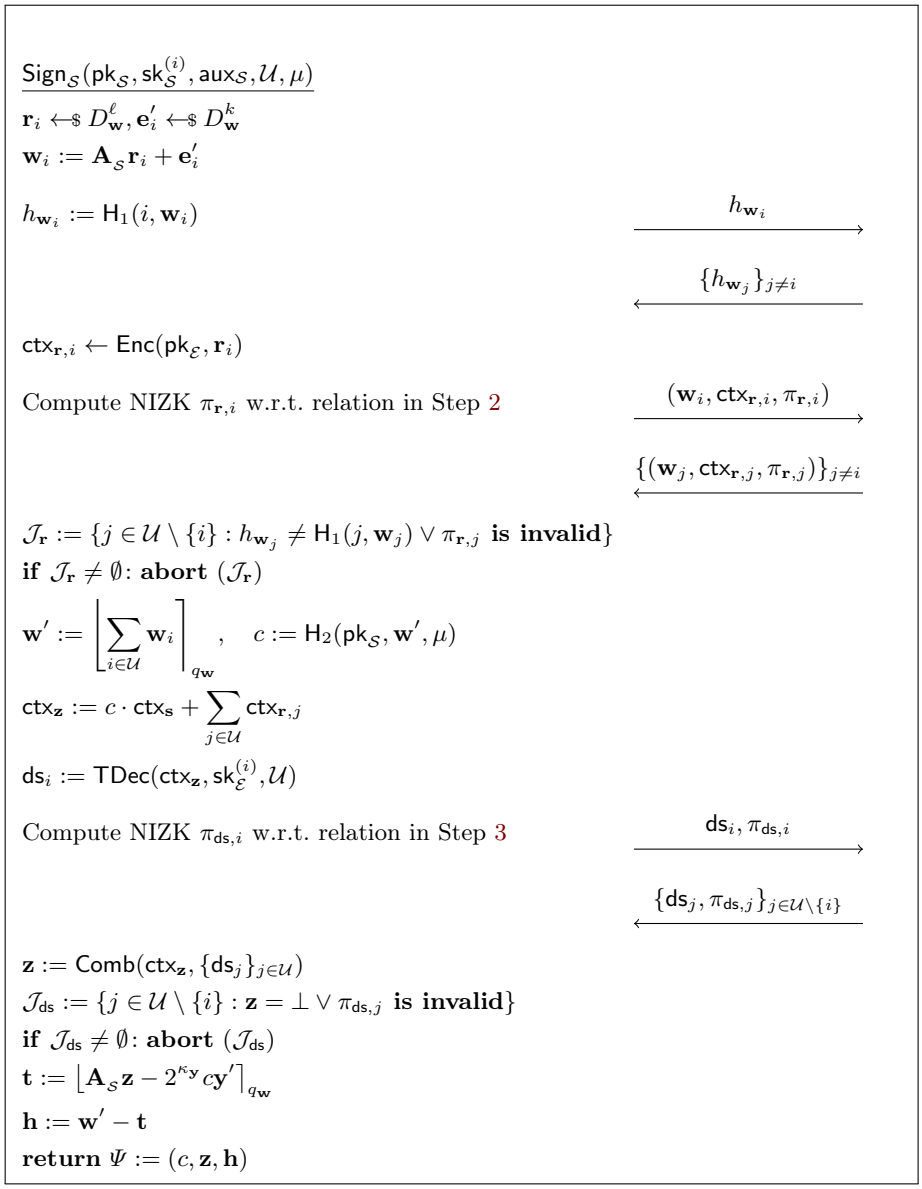

Fig. 7. Actively secure, 3-round, t-out-of-n threshold signature protocol with bit dropping for party $\mathcal{P}_i$.

6 Security

Here we show correctness, unforgeability, and security with identifiable aborts for our signature scheme. The proofs are given in Appendices C.4 to C.6.

Theorem 6 (Correctness). *The threshold signature scheme described in Figure 7 is correct if* $\kappa_{\mathbf{w}} \geq 4$,

$$B_{2,\mathbf{z}} = e^{1/4} \cdot (n\nu\sigma_{\mathbf{y}} + t\sigma_{\mathbf{w}})\sqrt{N(k+\ell)} + (\nu \cdot 2^{\kappa_{\mathbf{y}}} + 2^{(\kappa_{\mathbf{w}}+1)}) \cdot \sqrt{Nk}, \quad (2)$$

and the homomorphic threshold decryption correctness condition is satisfied.

Theorem 7 (Unforgeability). *Assume that the* H-MLWE *problem with parameters* $(k, \ell, \chi_1, \chi_2, Q)$ *is* $\epsilon_{\mathsf{H\text{-}MLWE}}$*-hard. Let* Q_S, Q_H *denote the number of signing queries and the total queries made to* $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ *respectively. Let* ℓ_0, ℓ_1 *be the bit-length of the output of* H_0 *and* H_1 *respectively. Let* $\mathcal{E}$ *be an* ϵ_{IND} *secure encryption scheme. Let* $|\mathcal{C}_{\mathcal{U}}|$ *denote the number of corrupted users. Let proof system* $\Pi_{\mathbf{x}}$ *for the relations* $\mathcal{R}_{\mathbf{x}}$ *be* $\epsilon_{\mathsf{ZK},\mathbf{x}}$*, and* $\epsilon_{\mathsf{extract},\mathbf{x}}$*-secure. Finally, let* $\mathcal{B}$ *be an adversary against the unforgeability of the underlying Raccoon signature scheme. Then, the actively secure* t*-out-of-*n *threshold signature scheme* $\mathcal{S}$*, is* ts-uf-cma*-secure when* H_0 *and* H_1 *are modeled as programmable random oracles and* H_2 *is modeled as a random oracle. The advantage of adversary* $\mathcal{A}$ *is*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{S}}(\mathcal{A}) \leq\ & \frac{|\mathcal{H}|(Q_H+Q_S)}{2^{\ell_0}} + \frac{|\mathcal{H}_{\mathcal{U}}|(Q_H+Q_S)}{2^{\ell_1}} \\ & + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract},\mathbf{s}} + |\mathcal{C}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{extract},\mathbf{r}} \\ & + |\mathcal{H}| \cdot \epsilon_{\mathsf{ZK},\mathbf{s}} + |\mathcal{H}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{ZK},\mathbf{r}} \\ & + \epsilon_{\mathrm{IND}} + \epsilon_{\mathsf{H\text{-}MLWE}} + \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{B}).\end{aligned}$$

Remark 1. Note that while computation in the first two rounds of our protocol is independent of the message to be signed, our proof of Theorem 7 assumes the message is known to the simulator at the start of the second round. We also remark that if the message is included in the hash during the first round, it binds the adversary to the transcript and we achieve a concurrently secure protocol.

Theorem 8 (Security with Identifiable Aborts). *Let* $\Pi_{\mathbf{x}}$ *be a NIZKAoK corresponding to the relation* $\mathcal{R}_{\mathbf{x}}$*. Let* $\mathcal{E}$ *be a correct linear homomorphic encryption scheme with identifiable aborts. Let* H_0, H_1 *be hash functions modeled as random oracles. Then the actively secure threshold signature scheme* $\mathcal{S}$ *has security with identifiable aborts in the synchronous model with access to a broadcast channel.*

7 Performance

7.1 Parameters and Sizes

In Table 2 we collect the parameters governing our scheme and the constraints that apply to them. Table 3 displays the concrete values that satisfy each of the three security levels.

Table 2. Parameters of the signature scheme

Parameter	Description	Constraints (if any)
(t, n)	# of threshold signing parties and total # of signing parties	$t \leq n$
N	Dimension of polynomial ring	$\mathsf{poly}(\lambda)$
(k, ℓ)	Dimensions of public signing matrix $\mathbf{A}_{\mathcal{S}}$	Set so that MLWE and MSIS hard
$(\hat{k}, \hat{\ell})$	Dimensions of public encryption matrix $\mathbf{A}_{\mathcal{E}}$	Set so that MLWE and MSIS hard
Q	max # of signatures issued	$Q \approx 2^{60}$
$(\kappa_{\mathbf{y}}, \kappa_{\mathbf{w}})$	# of bits dropped for verification key $\mathbf{y}$ and the commitment $\mathbf{w}$	$O(\log \lambda)$, $\kappa_{\mathbf{w}} \geq 4$
ν	ℓ_1 norm bound for the signature challenge c from $\mathcal{C}_\nu$	$\lvert\mathcal{C}_\nu\rvert \geq 2^\lambda$
$B_{\mathsf{H\text{-}MLWE}}$	Bound required for the H-MLWE to MLWE reduction	$B_{\mathsf{H\text{-}MLWE}} = Q \cdot \nu \cdot \left(1 + \frac{N(\lambda+1+2\log(N))}{\sqrt{Q}}\right)$
σ_{LWE}	Gaussian width of encryption secret keys and encryption randomness	Set so that MLWE and MSIS are hard
$\sigma_{\mathbf{y}}$	Gaussian width of user signing key secrets in $\mathbf{y}$	$\sigma_{\mathbf{y}} = 2\sqrt{\ell + k} \cdot \log N$
$\sigma_{\mathbf{w}}$	Gaussian width of the signature randomness in $\mathbf{w}$	$\sigma_{\mathbf{w}} = 2\sqrt{B_{\mathsf{H\text{-}MLWE}} \cdot (\ell + k)} \cdot \log N$
σ	Gaussian width for the H-MLWE to MLWE reduction	$\sigma \geq \sqrt{\ell + k} \cdot \omega(\sqrt{\log N})$ $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{n \cdot \sigma_{\mathbf{y}}^2} + \frac{B_{\mathsf{H\text{-}MLWE}}}{t \cdot \sigma_{\mathbf{w}}^2}\right)$
$B_{2,\mathbf{z}}$	Bound for the signature $\mathbf{z}$	$B_{2,\mathbf{z}} =$ $e^{1/4} \cdot (n\nu\sigma_{\mathbf{y}} + t\sigma_{\mathbf{w}})\sqrt{N(k+\ell)}$ $+ (\nu \cdot 2^{\kappa_{\mathbf{y}}} + 2^{(\kappa_{\mathbf{w}}+1)}) \cdot \sqrt{Nk}$
$B_{\mathsf{ST\text{-}MSIS}}$	Bound required for the ST-MSIS security	$B_{\mathsf{ST\text{-}MSIS}} = B_{2,\mathbf{z}} + \sqrt{\nu}$ $+ (\nu \cdot 2^{\kappa_{\mathbf{y}}} + 2^{(\kappa_{\mathbf{w}}+1)}) \cdot \sqrt{Nk}$
B_{MSIS}	Bound required for the ST-MSIS to MSIS reduction	$B_{\mathsf{MSIS}} = 2B_{\mathsf{ST\text{-}MSIS}}$
q	Signature modulus	$q > B_{\mathsf{MSIS}}\sqrt{Nk} \cdot \omega(\log(Nk))$

Table 3. Parameter values for Olingo for thresholds up to $(t, n) = (1023, 1024)$ and a maximum of $Q = 2^{60}$ signatures. When $\sigma_{\mathsf{LWE}} = \sqrt{2/3}$ we sample from a binary distribution.

Param	λ	N	q	$\hat{q}$	ℓ	$\hat{\ell}$	k	$\hat{k}$	$\kappa_{\mathbf{y}}$	$\kappa_{\mathbf{w}}$	ν	σ_{LWE}	$\sigma_{\mathbf{y}}$	$\sigma_{\mathbf{w}}$
Level I	128	256	2^{46}	2^{114}	8	21	8	21	29	32	23	$\sqrt{2/3}$	2^{10}	2^{29}
Level II	192	256	2^{47}	2^{116}	11	29	11	29	29	33	39	$\sqrt{2/3}$	2^{10}	2^{30}
Level III	256	256	2^{48}	2^{118}	14	37	14	37	29	33	60	$\sqrt{2/3}$	2^{10}	2^{30}

In Table 4 we present the communication costs of Olingo's signing procedure as compared to TRaccoon [DKM+24], Ringtail [BKL+25], and concurrent works [CdPE+26], [BCP+26], and [dENP25]. Total communication includes all rounds and the online communication refers to messages sent during message-dependent rounds. Our final NIZK proof $\pi_{\mathsf{ds},i}$ is only sent if the final signature fails to verify in the optimistic setting, where we assume that most signature executions go well. See Table 4 for a comparison of these sizes to a wider list of existing works, chosen here since they currently are the only implemented schemes.

We used the *lattice-estimator*[11] by Albrecht et al. [APS15] to estimate the security of our parameters and ensure the relations in Table 2 are satisfied. We remark that binary secrets are commonly used for encryption schemes based on lattices [BCC+24], providing the same security with respect to the learning with errors assumption as for other distributions [Mic18], also ensuring security against the Arora-Ge combinatorial attack [AG11] for up to 2^{60} samples in the module setting.

To ensure correctness is maintained, our parameter script also includes checks that ensure the relation in Theorem 6 is satisfied, bit dropping parameters as in Section 2.2 are carefully chosen, and the ciphertext modulus is chosen according to Theorem 2 to allow correct plaintext reconstruction. All parameter sets are chosen to ensure correctness for any number of users and threshold up to $(n, t) = (1024, 1023)$.

Whilst all benchmarks and sizes were computed with respect to parameters that would provide security and correctness up to $(n, t) = (1024, 1023)$, we did compute parameter sets for smaller values of (n, t). For example, for $(n, t) = (16, 15)$ we have a 41-bit signature modulus (vs. 46 bit at $(n, t) = (1024, 1023)$) and overall we get public key and signatures sizes of 3.7 KB and 9.7 KB (vs. 4.4 KB and 11.3 KB resp.). Communication per party becomes 802(64) KB (vs. 852(76) KB). Moreover, the change in these sizes is even less pronounced at intermediate thresholds and since the signature and ciphertext moduli remain within the same word-size across all thresholds as do lattice dimensions, benchmarks are also largely unchanged and so we decided to present 'worst-case' parameters and benchmarks as an upper bound on the efficiency of our design rather than give a range of parameter sets for different thresholds. Of course, one can

[11] Lattice-estimator: https://github.com/malb/lattice-estimator

compute exact parameters for any given threshold should they wish, using our parameter script.

Table 4. Comparison with TRaccoon [DKM+24], Ringtail [BKL+25], and concurrent works [CdPE+26, BCP+26, dENP25]. Sizes are given in KB per signer. Communication costs are total and online communication, respectively. For schemes that do not support $(t, n) = (1024, 1024)$, we report the concrete parameters at their maximum supported scale; "–" means the data is not reported. The optimistic communication is without NIZKs in the online phase, which are only required for identifiable abort. For Threshold ML-DSA, communication is reported per party and averaged over successful signing attempts; since offline communication is negligible, total and online communication are essentially the same. With the use of LaBRADOR, the online phase of Olingo can be reduced to the optimistic cost plus 66 KB. Concurrent work is denoted by †.

Scheme	Users	λ	$\mathsf{pk}_{\mathcal{S}}$	Ψ	Total COM	Online COM (Optimistic)
TRaccoon	1024	128	3.9	12.7	41	41
Ringtail	1024	128	4.5	13.4	613	11
Hermine [BCP+26]†	64	128	2.8	11.3	73	73
Threshold ML-DSA [CdPE+26]†	6	128	1.3	2.4	1050	1050
Replicated [dENP25]†	16	128	3.9	10.3	20	20
Ramp [dENP25]†	1024	128	3.5	11.9	25	25
Olingo	**1024**	**128**	**4.4**	**11.3**	**852**	**563 (76)**
TRaccoon	1024	192	5.8	18.9	59	59
Ringtail	1024	192	6.4	19.9	–	–
Hermine [BCP+26]†	64	192	4.2	16.9	153	153
Threshold ML-DSA [CdPE+26]†	6	192	2.0	3.3	9139	9139
Olingo	**1024**	**192**	**6.4**	**15.8**	**1254**	**860 (108)**
TRaccoon	1024	256	7.2	21.6	69	69
Ringtail	1024	256	8.5	27.3	–	–
Hermine [BCP+26]†	64	256	4.9	22.0	223	223
Threshold ML-DSA [CdPE+26]†	6	256	2.6	4.6	3059	3059
Replicated [dENP25]†	16	256	6.8	18.4	36	36
Ramp [dENP25]†	16	256	6.8	18.4	36	36
Olingo	**1024**	**256**	**8.4**	**20.1**	**1702**	**1208 (139)**

7.2 Implementation and Benchmarks

Our implementation can be accessed at https://github.com/carrosa/Olingo_Benchmarks, and it recycles components for polynomial multiplication

and NTT from the Dilithium C implementation[12] and the discrete Gaussian sampler[13] by Zhao, Steinfeld and Sakzad [ZSS20]. Moreover we use the LaZer Library[14] [LSS24] for the execution of NIZK proofs of exact norms. We adapt the linearity proof used in [ABGS23][15] for our linearity proofs. We have modified these components to be compatible with larger integers using the Gnu Multiple Precision library [Gt12].

Benchmarks are run on a desktop computer with an Intel i9-14900 CPU, 64 GB RAM running Ubuntu 24.04.2. The timings for the DKG are averaged over 10 runs (except for $\mathsf{KGen}_{\mathcal{E}}$ using $n = 128$). The timings for signing and verification are averaged over 10 runs, and the benchmarks of the NIZKs are also averaged over 10 runs. To simplify our analysis, where possible, we simulate only one user and ignore all potential communication latency[16].

We chose to use the LNP proofs [LNP22] of the LaZer library [LSS24] for implementing the zero-knowledge proofs, as it is, at the time of writing, the easiest to use and, for many use cases, the most efficient available library for proving exact norms over lattices with online extractability. We note that the authors claim a ~30% overhead when using the Python wrapper for their library. In addition, the LaZer library requires that we decompose vectors with ring elements whose coefficients are sampled from a large Gaussian, since it is not specifically designed for our parameters, leading to slower proof generation and verification times and a larger proof size in the final signing round in $\mathsf{Sign}_{\mathcal{S}}$. In a real-world implementation, these proofs would likely be faster than showcased in our benchmarks.

We would have preferred to use LaBRADOR for the final signing-round proof, where online extractability is not needed, owing to its succinctness (66 KB). However, the LaZer codebase does not currently support parameters of our magnitude, and we use LNP instead for the benchmarks. The linearity proofs used in $\mathsf{KGen}_{\mathcal{E}}$ and in the final signing round of $\mathsf{Sign}_{\mathcal{S}}$ use the Π_{LIN} proof from [BDL+18].

Table 5 shows the running time of our distributed key generation $\mathsf{KGen}_{\mathcal{S}}$ for various thresholds $(n, t = n - 1)$. The first column indicates the time to run the distributed encryption key generation $\mathsf{KGen}_{\mathcal{E}}$, where we provide both the time to run the (passively secure) base protocol and the time to generate the NIZKs needed for active security. In the second columns we do the same for the whole distributed signing key generation $\mathsf{KGen}_{\mathcal{S}}$ (which internally runs $\mathsf{KGen}_{\mathcal{E}}$). The final column shows the time to verify $\mathsf{KGen}_{\mathcal{S}}$, i.e., the various NIZKs produced throughout the protocol. To obtain these benchmarks, we focus only on the running time of a single party, assuming that all offline computations performed by other parties run in parallel. To further avoid unnecessary memory usage, we simulate the inputs from other parties using 'dummy' objects. We note that

[12] Dilithium: https://github.com/pq-crystals/dilithium
[13] Gaussian sampler: https://gitlab.com/raykzhao/gaussian
[14] LaZer: https://github.com/lazer-crypto/lazer
[15] Proofs of Linearity: https://github.com/dfaranha/lattice-verifiable-mixnet
[16] We exclude the broadcast channel from all experiments and focus solely on computational overhead.

these are chosen to reflect the cost of computing on 'real' inputs. We observe that, whilst the key generation could not be called fast, the timings are perfectly acceptable for a one-time setup computation. Indeed, even up to 64 signers, the total key generation can be done in under 3 minutes. Table 6 shows the communication in the distributed key generation protocol.

Table 7 shows the signing times of the main signing protocol Sign. We have split the timings of $\mathsf{Sign}_{\mathcal{S}}$ into three parts: $\mathsf{Sign}_{\mathcal{S}_1}$, $\mathsf{Sign}_{\mathcal{S}_2}$ and $\mathsf{Sign}_{\mathcal{S}_3}$, corresponding to rounds 1, 2 and 3 in Figure 7. The numbers for $\mathsf{Sign}_{\mathcal{S}_2}$ are presented as a sum, with the second and third terms representing proof generation and proof verification, respectively. We emphasize that for real-world thresholds (small n and t), the t-dependence in verification would likely be acceptable, and per-proof verification would be smaller owing to the LaZer Python wrapper overhead. Furthermore, this verification could be parallelized. We note that only $\mathsf{Sign}_{\mathcal{S}_3}$ is considered to be an 'online' round, and the NIZK in that round needs only to be sent if there is a dispute. In the case of such a dispute, our benchmarks show a verification time of roughly $700 \cdot t$ ms, which again, could likely be reduced in practice via parallelization and a clean LNP implementation. We include, for comparison, the timings for Threshold Raccoon [DKM+24] and Ringtail [BKL+25]. We wish to emphasize that neither Threshold Raccoon nor Ringtail offers a distributed key generation or achieves identifiable aborts. Whilst the addition of these makes our design very attractive from a security perspective, Table 7 demonstrates its efficiency comparable to those less fully-featured designs.

Regarding concurrent works, Threshold ML-DSA [CdPE+26] achieves an average successful signing time of approximately 36 ms for $(n, t) = (6, 5)$, the maximum threshold admissible in that work. While the authors give running times for a trusted setup, they do not provide timings for their DKG. Verification is very fast, at 0.03 ms, because the verification algorithm is that of standard ML-DSA.

Hermine [BCP+26] achieves a total signing time of 58 ms for its maximum supported threshold of $(n, t) = (64, 64)$. Its verification time is 0.65 ms. No DKG is given in that paper.

Table 5. Timing provided in seconds for our distributed signing key generation $\mathsf{KGen}_{\mathcal{S}}$. The timings are given as a sum; the second component represents the time taken to generate the relevant NIZKs, and the first component represents the time to run the rest of the protocol.

n	$\mathsf{KGen}_{\mathcal{E}}$	$\mathsf{KGen}_{\mathcal{S}}$ (total)	$\mathsf{KGen}_{\mathcal{S}}$ (verify)
16	19.45 = 10.60 + 8.85	21.42 = 11.10 + 10.32	17.30
32	43.32 = 33.86 + 9.46	45.29 = 34.36 + 10.93	41.37
64	172.08 = 161.34 + 10.74	174.05 = 161.84 + 12.21	107.60
128	1494.14 = 1480.74 + 13.40	1496.11 = 1481.24 + 14.87	317.15

Table 6. Communication cost given in MB. The cost is given as a sum; the second component represents the cost of the relevant NIZKs and the first represents the cost of the rest of the protocol. These values are computed for $(n, t) = (1024, 1023)$ and are thus upper bounds on the communication cost at lower thresholds.

λ	$\mathsf{KGen}_{\mathcal{E}}$	$\mathsf{KGen}_{\mathcal{S}}$ (total)
128	$2003.5 = 2002.1 + 1.4$	$2003.8 = 2002.3 + 1.5$
192	$3910.7 = 3908.6 + 2.1$	$3911.1 = 3908.8 + 2.3$
256	$6439.3 = 6436.2 + 3.1$	$6439.7 = 6436.4 + 3.3$

Table 7. Signing timings for our scheme, TRaccoon [DKM+24] and Ringtail [BKL+25]. All timings are listed in milliseconds, computed by dividing the number of CPU cycles by the CPU frequency. We did not include the time needed to allocate space on the heap for shared variables. The summations in rounds 2 and 3 timings reflect the additional cost of generating the NIZKs and thus achieving IA security. Note that all timings are set according to the same parameter set, which ensures 128-bit security up to the maximum thresholds $(n, t) = (1024, 1023)$; in practice, timings are likely to be faster for smaller thresholds.

Scheme	t	$\mathsf{Sign}_{\mathcal{S}_1}$	$\mathsf{Sign}_{\mathcal{S}_2}$	$\mathsf{Sign}_{\mathcal{S}_3}$	Comb	Verify
Olingo	4	1.304	$22.544 + 1682.00 + 703 \cdot t$	$2.840 + 1365.61$	0.810	1.046
Olingo	16	1.304	$22.544 + 1682.00 + 879 \cdot t$	$3.920 + 1365.61$	1.030	1.046
Olingo	64	1.304	$22.544 + 1682.00 + 923 \cdot t$	$8.280 + 1365.61$	1.820	1.046
Olingo	256	1.304	$22.544 + 1682.00 + 934 \cdot t$	$25.020 + 1365.61$	4.940	1.046
Olingo	1023	1.304	$22.544 + 1682.00 + 936 \cdot t$	$92.790 + 1365.61$	17.500	1.046
TRaccoon	4	20.092	0.539	1.588	1.128	1.094
TRaccoon	16	20.076	2.102	5.559	1.209	1.093
TRaccoon	64	21.830	8.216	21.350	1.579	1.100
TRaccoon	256	33.549	32.788	84.333	3.186	1.095
TRaccoon	1024	67.213	131.887	338.614	11.571	1.106
Ringtail	4	18.995	131.944	4.601	0.143	1.361
Ringtail	16	29.084	164.646	13.904	0.179	1.331
Ringtail	64	73.684	313.938	52.036	0.339	1.627
Ringtail	256	254.253	829.033	201.272	0.821	1.515
Ringtail	1024	978.182	2949.853	794.288	2.706	1.434

7.3 Non-Interactive Signing

Our signing protocol, depicted in Figure 7, has three rounds of communication, compared to the two rounds of [GKS24]. This increase in the number of rounds comes with an advantage, as the first two rounds of our protocol are independent of the message to be signed. Hence, it is possible to bootstrap our scheme to a non-interactive threshold signature scheme.

Instead of running the first two rounds of signing in an interactive manner, each $\mathcal{U}$ set can run a "presigning" round to determine $\mathbf{w}^{(\tau)}$, $\mathsf{ctx}_{\mathbf{r}}^{(\tau)}$ to use in τ-th time they have to sign a message. Then whenever $\mathcal{U}$ has to sign $\mu^{(\tau)}$, $\mathcal{U}$ can use $\mathbf{w}^{(\tau)}$ to derive $c^{(\tau)}$ and consecutively $\mathsf{ctx}_{\mathbf{r}}^{(\tau)}$.

With pre-signing, there is only one round of communication, which then includes the partial decryption shares ds_i. However, this also means that the final round of communication can be avoided. Combining decryption shares does not require interaction; hence, each decryption share can be published individually for an external party to combine. This makes the protocol with presigning *non-interactive*[17].

Another advantage of non-interactivity is the capability of batching NIZKs. The first two rounds of signing that can be pre-processed contain a proof of plaintext knowledge for signature randomness $\mathbf{r}$. While the value of $\mathbf{r}$ and consequently $\mathbf{w}$ changes for different signing queries, $\mathbf{A}_{\mathcal{S}}$ remains the same. Hence, instead of sending Q proofs per party in the first round, we can send a single proof of increased witness size. Considering we need to prove a set of inner products, we can use LaBRADOR [BS23] that allows succinct proofs for inner product relations in the ring $R'_q = \mathbb{Z}_q[X]/(X^{64}+1)$. This way $\pi_{\mathbf{r}}$ for Q values of $\mathbf{r}$ constitutes a single proof of an increased witness size of $\lceil N/64 \rceil \cdot (Q \cdot \ell)$.

7.4 Optimistic Approach

In addition to making the scheme non-interactive, we can utilize the "optimistic" approach taken in [GKS24] to reduce communication bandwidth. Instead of sending a NIZK alongside ds_i, each signer only sends ds_i. If the combined signature verifies, the protocol is completed. If not, each signer can ask for the corresponding NIZKs to verify that each ds_i was correctly computed. In an optimistic scenario where the final signature verifies, this reduces the final communication size to 76 KB.

TRaccoon-IA [PKN+25] also provides an optimistic approach for their threshold signature scheme reducing their communication cost to 28.2 KB, independent of the threshold. This is less than half of our optimistic communication cost. However, Olingo provides a distributed key generation which is not provided in [PKN+25]. Both Olingo's optimistic approach and that of [PKN+25] would require an additional round of communication to send the NIZKs in case of a dispute to achieve identifiable aborts, narrowing the gap in communication costs between the two schemes for larger thresholds.

[17] Again, we assume that the participating set of users $\mathcal{U}$ is known a priori.

Acknowledgments

We thank Patrick Steuer and Vadim Lyubashevsky for help with the LaZer library, Diego F. Aranha for input on optimizing the implementation, and Ignacio Cascudo for pointers on improving the verifiable secret sharing scheme. We also thank Sascha Roth and Matthieu Rambaud for discussions on the security model and Guilhem Niot for input regarding the parameter sets. The work of Kamil Doruk Gur was partially supported by the European Union and the European Cybersecurity Competence Centre under Grant Agreement No. 101225691 (QARC).

References

ABGS23. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 1467–1481, Copenhagen, Denmark, November 26–30, 2023. ACM Press. https://doi.org/10.1145/3576915.3616683

ADP24. Nabil Alkeilani Alkadri, Nico Döttling, and Sihang Pu. Practical lattice-based distributed signatures for a small number of signers. In Christina Pöpper and Lejla Batina, editors, *ACNS 2024: 22nd International Conference on Applied Cryptography and Network Security, Part I*, volume 14583 of *Lecture Notes in Computer Science*, pages 376–402, Abu Dhabi, UAE, March 5–8, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-54770-6_15

AG11. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011: 38th International Colloquium on Automata, Languages and Programming, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415, Zurich, Switzerland, July 4–8, 2011. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-642-22006-7_34

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. https://doi.org/10.1515/jmc-2015-0016

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. https://doi.org/10.4230/LIPIcs.ICALP.2022.8

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press. https://doi.org/10.1109/SP.2018.00020

BCC+24. Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xianhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Yuriy Polyakov, Luis Antonio Ruiz Lopez, Yongsoo Song, and Donggeon Yhee. Security guidelines for implementing homomorphic encryption. *IACR Communications in Cryptology (CiC)*, 1(4):26, 2024. https://doi.org/10.62056/anxra69p1

BCd+25. Giacomo Borin, Sofía Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Threshold signatures reloaded: ML-DSA and enhanced raccoon with identifiable aborts. Cryptology ePrint Archive, Report 2025/1166, 2025. https://eprint.iacr.org/2025/1166

BCP+26. Giacomo Borin, Sofía Celi, Rafaël Del Pino, Thomas Espitau, Shuichi Katsumata, Guilhem Niot, Thomas Prest, and Kaoru Takemure. Hermine: An efficient lattice-based FROST-like threshold signature. *IACR Cryptol. ePrint Arch.*, 2026:419, 2026. https://eprint.iacr.org/2026/419

BdCE+26. Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. Quorus: Efficient, scalable threshold ML-DSA signatures from MPC. In *USENIX Security 2026: 35th USENIX Security Symposium*, Baltimore, MD, USA, August 12–14, 2026. USENIX Association. Full version: https://eprint.iacr.org/2025/1163.

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385, Amalfi, Italy, September 5–7, 2018. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-98113-0_20

BG14. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47, San Francisco, CA, USA, February 25–28, 2014. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-04852-9_2

BGG+18. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-96884-1_19

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery. https://doi.org/10.1145/2090236.2090262

BKL+25. Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. Ringtail: Practical two-round threshold signatures from learning with errors. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 149–164, San Francisco, CA, USA, May 12–15, 2025. IEEE Computer Society Press. https://doi.org/10.1109/SP61157.2025.00070

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 2013: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 218–236, Banff, AB, Canada, June 25–28, 2013. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-642-38980-1_14

BL90. Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35, Santa Barbara, CA, USA, August 21–25, 1990. Springer, New York, USA. https://doi.org/10.1007/0-387-34799-2_3

BL22. Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 193–207, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. https://doi.org/10.1145/3548606.3560656

BLNS23. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 384–417, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-38545-2_13

BLR+18. Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, April 2018. https://doi.org/10.1007/s00145-017-9265-9

BP25. Luís T. A. N. Brandão and René Peralta. NIST first call for multi-party threshold schemes. NIST internal report (NISTIR) 8214c 2pd (second public draft). https://doi.org/10.6028/NIST.IR.8214C.2pd, 2025.

BS23. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 518–548, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-38554-4_17

BTT22. Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 276–305, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-15979-4_10

CATZ24. Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part IV*, volume 15487 of *Lecture Notes in Computer Sci-*

ence, pages 268–302, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore. https://doi.org/10.1007/978-981-96-0894-2_9

CD17. Ignacio Cascudo and Bernardo David. SCRAPE: Scalable randomness attested by public entities. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 2017: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 537–556, Kanazawa, Japan, July 10–12, 2017. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-61204-1_27

CdPE+26. Sofía Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Efficient threshold ML-DSA. In *USENIX Security 2026: 35th USENIX Security Symposium*, Baltimore, MD, USA, August 12–14, 2026. USENIX Association. Full version: https://eprint.iacr.org/2026/013.

CGMA85. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. https://doi.org/10.1109/SFCS.1985.64

Che23. Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 716–747, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-38554-4_23

CS19. Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *Lecture Notes in Computer Science*, pages 128–153, Oxford, UK, December 16–18, 2019. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-35199-1_7

DEF+19. Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press. https://doi.org/10.1109/SP.2019.00050

dENP25. Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Simple and efficient lattice threshold signatures with identifiable aborts. Cryptology ePrint Archive, Report 2025/871, 2025. https://eprint.iacr.org/2025/871

Des88. Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127, Santa Barbara, CA, USA, August 16–20, 1988. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/3-540-48184-2_8

DF90. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, New York, USA. https://doi.org/10.1007/0-387-34805-0_28

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A

lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, 2018. https://doi.org/10.13154/tches.v2018.i1.238-268

DKLs24. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Threshold ECDSA in three rounds. In *2024 IEEE Symposium on Security and Privacy*, pages 3053–3071, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press. https://doi.org/10.1109/SP54263.2024.00178

DKM+24. Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 219–248, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-58723-8_8

DOTT21. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 99–130, Virtual Event, May 10–13, 2021. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-75245-3_5

dPKPR24. Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 409–444, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-68376-3_13

EKT24. Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 387–424, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-68394-7_13

ENP24. Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 425–458, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-68394-7_14

Fel87. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437, Los Angeles, CA, USA, October 12–14, 1987. IEEE Computer Society Press. https://doi.org/10.1109/SFCS.1987.4

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

pore, November 29 – December 1, 2020. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-62576-4_3

FHSZ23. Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 386–400, Copenhagen, Denmark, November 26–30, 2023. ACM Press. https://doi.org/10.1145/3576915.3623219

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/3-540-47721-7_12

FSZ22. Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 1109–1123, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. https://doi.org/10.1145/3548606.3560655

GHK+26. Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. Olingo: Threshold lattice signatures with DKG and identifiable abort. In *ACM CCS 2026: 33rd Conference on Computer and Communications Security*, The Hague, The Netherlands, November 15–19, 2026. ACM Press. https://doi.org/10.1145/3830454.3832660

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, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 458–487, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-06944-4_16

GKS24. Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 266–300, Oxford, UK, June 12–14, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-62746-0_12

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. https://doi.org/10.1145/1374376.1374407

GRR98. Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In Brian A. Coan and Yehuda Afek, editors, *17th ACM Symposium Annual on Principles of Distributed Computing*, pages 101–111, Puerto Vallarta, Mexico, June 28 – July 2, 1998. Association for Computing Machinery. https://doi.org/10.1145/277697.277716

Gt12. Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. http://gmplib.org/.

HMM+26. Lucjan Hanzlik, Mark Manulis, Marzio Mula, Alan Pulval-Dady, Tjerand Silde, and Daniel Slamanig. BRaccoon: Concurrently secure blind lattice signatures from Raccoon. Cryptology ePrint Archive, Paper 2026/1084, 2026. https://eprint.iacr.org/2026/1084

IOZ14. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 369–386, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-662-44381-1_21

Kat21. Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 580–610, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-84245-1_20

KG20. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-81652-0_2

KKK08. Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of VSS in point-to-point networks. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 499–510, Reykjavik, Iceland, July 7–11, 2008. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-540-70583-3_41

KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-MLWE. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 549–580, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-38554-4_18

KRT24. Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 459–491, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-68394-7_15

LDK+22. 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, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Lin17. Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 613–644, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-63715-0_21

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 71–101, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-15979-4_3

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. https://doi.org/10.1007/s10623-014-9938-4

LSS14. Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-642-55220-5_14

LSS24. Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The LaZer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 3125–3137, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press. https://doi.org/10.1145/3658644.3690330

MBH23. Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for RLWE-based multiparty homomorphic encryption. *J. Cryptol.*, 36(2), mar 2023. https://doi.org/10.1007/s00145-023-09452-8

Mic18. Daniele Micciancio. On the hardness of learning with errors with binary secrets. *Theory of Computing*, 14(13):1–17, 2018. https://doi.org/10.4086/toc.2018.v014a013

MS81. Robert J. McEliece and Dilip V. Sarwate. On sharing secrets and Reed-Solomon codes. *Commun. ACM*, 24(9):583–584, 1981. https://doi.org/10.1145/358746.358762

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/3-540-46766-1_9

PKN+25. Rafaël Del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking TRaccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part VI*, volume 16005 of *Lecture Notes in Computer Science*, pages 423–

456, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-032-01887-8_14

PN25. Rafaël Del Pino and Guilhem Niot. Finally! A compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025: 28th International Conference on Theory and Practice of Public Key Cryptography, Part III*, volume 15676 of *Lecture Notes in Computer Science*, pages 169–199, Røros, Norway, May 12–15, 2025. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-91826-1_6

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-540-85174-5_31

Rab94. Tal Rabin. Robust sharing of secrets when the dealer is honest or cheating. *J. ACM*, 41(6):1089–1109, 1994. https://doi.org/10.1145/195613.195621

RB89. Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 73–85, Seattle, WA, USA, May 15–17, 1989. ACM Press. https://doi.org/10.1145/73007.73014

SAB+22. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. https://doi.org/10.1145/359168.359176

TPCZ23. Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. Efficient lattice-based threshold signatures with functional interchangeability. *IEEE Transactions on Information Forensics and Security*, 18:4173–4187, 2023. https://doi.org/10.1109/TIFS.2023.3293408

TZ23. Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 628–658, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-30589-4_22

ZSS20. Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. FACCT: FAst, compact, and constant-time discrete gaussian sampler over integers. *IEEE Transactions on Computers*, 69(1):126–137, 2020. https://doi.org/10.1109/TC.2019.2940949

ZT25. Chenzhi Zhu and Stefano Tessaro. The algebraic one-more MISIS problem and applications to threshold signatures. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part I*, volume 16000 of *Lecture Notes in Computer Science*, pages 548–581, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-032-01855-7_18

A Related Work

A.1 State-of-the-Art

Gur, Katz, and Silde [GKS24] (from now on, GKS) presented the first t-out-of-n threshold signature scheme based on standard lattice assumptions, while avoiding reliance on threshold fully homomorphic encryption (FHE) or generic secure multi-party computation (MPC). They achieved a two-round signing protocol within a framework combining ideas from threshold FHE [BGG+18] and n-out-of-n threshold signatures [DOTT21], utilizing a non-aborting underlying signature scheme [ASY22]. To do this, they design a linearly homomorphic encryption scheme alongside a novel threshold decryption algorithm, computing the linear operations of the signature scheme in the encrypted domain. However, both signature rounds depend on the message, and the additional use of trapdoor commitments necessitated the inclusion of a commitment opening in the final signature, altering the verification algorithm and effectively doubling the signature size. They also devised a combined distributed key generation protocol for the encryption and signature schemes based on Shamir secret sharing [Sha79], although they do not provide concrete communication costs for this sub-protocol. Their scheme offers identifiable abort using non-interactive zero-knowledge proofs, though full details of how one could instantiate the proofs are not given. They provide a concrete instantiation for only five users which already has very large keys, signatures and communication complexity. Furthermore, they do not provide an implementation of their scheme.

del Pino et al. [DKM+24] introduce TRaccoon, a threshold variant of the Raccoon signature scheme [dPKPR24] which is based on the MLWE-with-hints assumption by Kim et al. [KLSS23]. They achieved a three-round scheme with very small public keys, signature sizes, communication costs, and efficient implementation for up to 1024 users. However, the scheme's drawbacks include the message dependency of all three rounds and the necessity for involved parties to maintain state to prevent secret reuse (due to their masking procedure that safeguards the signing key shares when applying Shamir secret sharing). Their scheme requires a trusted setup (it is not directly compatible with the distributed key generation protocol by [GKS24]) and does not provide identifiable aborts.

Espitau, Katsumata, and Takemure [EKT24] designed a two-round threshold signature scheme founded on a new assumption known as the Algebraic One-More Module Learning With Errors (AOM-MLWE) problem. Their scheme is quite efficient, but the parameters are determined based on their cryptanalysis of this new assumption, proving only selective security of AOM-MLWE to standard lattice problems such as MSIS and MLWE. The first round of their scheme is independent of the message and signing set, and they avoid maintaining a state across different signature procedures. They leave distributed key generation, identifiable aborts, and implementation as future work. A follow up work by Zhu and Tessaro [ZT25] introduced the Algebraic One-More Module Short Integer Solution (AOM-MSIS) assumption, showed a full reduction to standard MSIS and MLWE, as well as a proof for AOM-MLWE against any algebraic adver-

saries, solving an open problem in [EKT24]. They also provide new parameters based on their security proof, which are slightly worse than the initial instantiation based on the incomplete security reduction. We include both parameter sets.

Katsumata, Reichle, and Takemure [KRT24] present the only lattice-based threshold signature scheme with adaptive security, allowing an adversary to adaptively corrupt users instead of stating up-front which users are corrupted. Their framework is generic, stateless, and works for the Raccoon signature scheme, leading to very efficient public keys and signatures. However, their scheme requires five rounds of communication, of which three are message dependent, making it unpractical for real-world usage with many parties. They have not provided communication estimates nor an implementation of their scheme.

Chairattana-Apirom, Tessaro, and Zhu [CATZ24] presented a non-interactive two-round threshold signature based on standard lattice assumptions, using Benaloh-Leichter [BL90] linear secret sharing and linear hash function techniques inspired by Tessaro and Zhu [TZ23]. While their scheme is round-optimal, their public keys, signatures, and communication costs are substantially larger than the other related works. Furthermore, their secret sharing scheme requires each user to (on average) hold secret keys of size 610.7 KB and a distributed key generation protocol would require generic MPC techniques and is not considered practical. The recent work by Zhu and Tessaro [ZT25] improves the construction, leading to slightly better parameters, but the sizes are still larger than other works. Their protocol does not achieve identifiable aborts, however, in contrast to TRaccoon [DKM+24] which requires PRFs and uniform masks, this scheme could potentially be extended using lattice-based zero-knowledge proofs similar to the GKS-framework, given that commitments to all of the secret key shares are provided during the trusted setup.

Boschini et al. [BKL+25] introduced Ringtail, which is similar in spirit to [EKT24], being a stateless two-round non-interactive scheme, but based on the standard MLWE assumption. Their non-interactive communication is only 10 KB, and they provide an implementation with pre-processing connecting servers across 8 countries in 5 continents showing that the end-to-end latency mostly depends on the network costs, emphasizing the importance of non-interactive schemes. However, Ringtail also requires a trusted setup and does not provide identifiable aborts. Additionally, their security proof and parameter setting assume that the number of hash queries and signatures queries from the adversary are the same, while in practice one wants security against an adversary that can make 2^{128} or even 2^{256} hash queries but can only request 2^{60} signatures. The updated analysis by Zhu and Tessaro [ZT25] provides a security proof based on the AOM-MSIS assumption which avoids this gap, leading to worse parameters.

del Pino and Niot [PN25] achieved the first practical threshold instantiation of the CRYSTALS-Dilithium signature scheme [DKL+18] with small verification key, signature size, and communication. The scheme is a three-round scheme with non-interactive signing. However, they use replicated secret sharing which means that the secret key size is exponential in the number of parties and the

scheme does not provide identifiable aborts nor accommodate for a distributed key generation. They provide an implementation for up to 8 parties.

del Pino et al. [PKN+25] provides a technique using the lattice-based succinct zero-knowledge proof system LaBRADOR [BS23] to add identifiable aborts to the Raccoon signature scheme [DKM+24], only impacting the communication of the protocol when signatures fail verification. The extra communication crucially scales linearly with the threshold t.

A.2 Other Related Work

Lattice-Based Secret Sharing Gentry, Halevi, and Lyubashevsky [GHL22] presented a verifiable secret sharing scheme from lattices for up to 1000 parties. Their construction is based on the lattice-based encryption scheme by Peikert, Vaikuntanathan, and Waters [PVW08] and the Bulletproof zero-knowledge proof system by Bunz et al. [BBB+18], however, Bulletproof is based on discrete logarithm assumptions and is not secure against quantum computers. Espitau, Niot, and Prest [ENP24] developed a robust distributed key generation protocol for lattice-signatures, but their scheme requires 2/3 of the parties involved to behave honestly.

Schemes from Generic MPC Bendlin, Krehbiel, and Peikert [BKP13] used MPC to produce trapdoors for the Gentry, Peikert, Vaikuntanathan signature scheme [GPV08], but this is a theoretical construction that would not be useful in practice. Cozzo and Smart [CS19] used generic MPC to implement NIST post-quantum signatures, and concluded that "all schemes, except the ones in the MQ family, have significant issues when one wishes to convert them using relatively generic MPC techniques. The lattice based schemes are hampered by requiring a mix of operations which are suited to both linear secret shared-based schemes and garbled circuits-based MPC techniques". Tang, Pang, and Chen [TPCZ23] implemented a threshold version of CRYSTALS-Dilithium [LDK+22], however, their construction requires 15 rounds of communication for 2-out-of-3 signatures, and scales worse when the number of parties involved increases.

Schemes from Generic FHE Boneh et al. [BGG+18] presented a threshold signature framework based on fully homomorphic encryption, evaluating the complete signature (including rejection sampling and the concrete hash-function used as a random oracle) in the encrypted domain before applying threshold decryption to recover the signature. Agrawal, Stehle, and Yadav [ASY22] improves their construction using rejection-free signatures, however, none of the constructions are practical due to the complex evaluation algorithm, and they did not present a concrete instantiation of the threshold decryption. The latter can be implemented from Mouchet, Bertrand, and Hubaux [MBH23] or using GKS [GKS24].

Full Threshold Schemes and Multisignatures A line of recent works build full-threshold (n-out-of-n) and multi-signatures from lattice assumptions [DOTT21, ADP24, FH20, FSZ22, BTT22, FHSZ23, Che23], but none of these can easily be adjusted to the arbitrary threshold setting. The t-out-of-n GKS-framework can be seen as combining the full threshold scheme by Damgård et al. [DOTT21] with the FHE framework by Boneh et al. [BGG+18].

B Additional Background

B.1 Shamir Secret Sharing

The t-out-of-n sharing of $\mathbf{S} \in R_q^{m_1 \times m_2}$ proceeds as follows.

1. Generate a uniform polynomial $\mathbf{P} \in R_q^{m_1 \times m_2}$, conditioned on $\mathbf{P}(0) = \mathbf{S}$.
2. For $i \in [n]$, compute the ith share as $[\![\mathbf{S}]\!]_i := \mathbf{P}(i)$.

Given $\mathcal{S} \subseteq [n]$ of size $\geq t$, Lagrange interpolation can be used to reconstruct $\mathbf{S}$ as

$$\mathbf{S} = \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}} [\![\mathbf{S}]\!]_i, \quad \text{where} \quad \lambda_i^{\mathcal{S}} := \prod_{j \in \mathcal{S} \setminus \{i\}} \frac{j}{j-i}. \tag{3}$$

B.2 Gaussians

In this work we use the standard discrete Gaussian distribution of support $\mathbb{Z}^N$, center 0, and standard deviation σ defined by its probability distribution function

$$\mathcal{D}_\sigma(z) := \frac{\rho_\sigma(z)}{\sum_{z \in \mathbb{Z}^N} \rho_\sigma(z)},$$

where $\rho := \exp\left(\frac{-\|z\|_2}{2\sigma^2}\right)$. Moreover, we will always use a *truncated* Gaussian distribution, from which sampling yields N-dimensional integer vectors with bounded infinity and two-norms. Specifically, sampling from this truncated Gaussian will be denoted '$z \leftarrow \mathcal{D}_\sigma$'[18] and, for real numbers $t_\infty > 0$ and $t_2 > 1$, is implemented as follows:

We choose t_∞ and t_2 so that Step 2 occurs with small probability.

B.3 Module-SIS/-LWE Hardness Assumptions

Definition 6 (MSIS [LS15]). *Let k, ℓ be positive integers and $0 < \eta \ll q$. Then, given $\mathbf{A} \leftarrow R_q^{k \times \ell}$, the Module-SIS problem asks an adversary $\mathcal{A}$ to find $\mathbf{z} \in R_q^\ell$ such that $\mathbf{Az} = \mathbf{0}$ and $0 < \|\mathbf{z}\|_2 \leq \eta$. $\mathcal{A}$ is said to have advantage ϵ_{MSIS} in solving $\mathsf{MSIS}_{k,\ell,\eta}$ if*

$$\Pr\left[0 < \|\mathbf{z}\|_2 \leq \eta \wedge \mathbf{Az} = \mathbf{0} \mid \mathbf{A} \leftarrow R_q^{k \times \ell};\ \mathbf{z} \leftarrow \mathcal{A}(\mathbf{A})\right] \geq \epsilon_{\mathsf{MSIS}}.$$

[18] We abuse this notation in places, for example denoting by $\mathcal{D}_{\mathbf{y}}$ the Gaussian from which secrets $\mathbf{s}$ and $\mathbf{e}$ are sampled to construct $\mathbf{y} := \mathbf{As} + \mathbf{e}$ as in the signing public key.

Truncated Gaussian Sampling, denoted '$z \leftarrow \mathcal{D}$'

1. Sample z from the regular (non-truncated) Gaussian of parameter σ and center 0.
2. **if** either $\|z\|_\infty \geq t_\infty \cdot \sigma$ **or** $\|z\|_2 \geq t_2 \cdot \sigma \cdot \sqrt{N}$ **then restart**
3. **return** z.

Fig. 8. Truncated Gaussian sampling.

Definition 7 (MLWE [LS15]). *Let k, ℓ be positive integers, and χ be a probability distribution over R_q. The Module-LWE problem then asks an adversary $\mathcal{A}$ to distinguish between the following two cases:*

1. $(\mathbf{A}, \mathbf{As})$ *for public* $\mathbf{A} \leftarrow R_q^{k \times (\ell+k)}$ *and secret* $\mathbf{s} \leftarrow \chi^{\ell+k}$,
2. $(\mathbf{A}, \mathbf{b}) \leftarrow R_q^{k \times (\ell+k)} \times R_q^k$ *where both are sampled uniformly.*

Then $\mathcal{A}$ is said to have advantage ϵ_{MLWE} in solving $\mathsf{MLWE}_{k,\ell,\chi}$ if

$$\Big| \Pr\left[b = 1 \mid \mathbf{A} \leftarrow R_q^{k \times (\ell+k)}; \mathbf{s} \leftarrow \chi^{\ell+k}; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{As})\right] - \Pr\left[b = 1 \mid \mathbf{A} \leftarrow R_q^{k \times (\ell+k)}; \mathbf{b} \leftarrow R_q^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})\right] \Big| \geq \epsilon_{\mathsf{MLWE}}.$$

B.4 Rényi Divergence

We rely on the standard notion of Rényi divergence in this work. Specifically, we require a lemma regarding Rényi divergence between Gaussians with different centers and the notion of *public sampleability*.

For any two discrete probability distributions ϕ and ϕ' such that $\mathsf{Supp}(\phi) \subseteq \mathsf{Supp}(\phi')$ and an $a \in (1, +\infty)$, the Rényi divergence of order a can be defined as:

$$R_a(\phi \| \phi') := \left(\sum_{x \in \mathsf{Supp}(\phi)} \frac{\phi(x)^a}{\phi'(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

with the special case $R_\infty(\phi || \phi') := \max_{x \in \mathsf{Supp}(\phi)} \frac{\phi(x)}{\phi'(x)}$. The Rényi divergence R_a has the following properties for probability distributions ϕ, ϕ', ϕ'' with $\mathsf{Supp}(\phi) \subseteq \mathsf{Supp}(\phi') \subseteq \mathsf{Supp}(\phi'')$:

- **Log. Positivity:** $R_a(\phi \| \phi') \geq R_a(\phi \| \phi) = 1$.
- **Data Processing Inequality:** $R_a(\phi^f \| \phi'^f) \leq R_a(\phi \| \phi')$ for any function f where ϕ^f denotes the distribution of $f(y)$ where $y \leftarrow \phi$.
- **Multiplicativity:** Assume ϕ and ϕ' are two distributions for a pair of random variables (Y_0, Y_1). For $i \in \{0, 1\}$, let ϕ_i denote the marginal distribution of Y_i under ϕ, and let $\phi_{1|0}(\cdot|y_0)$ denote the conditional distribution of Y_1 given $Y_0 = y_0$. Then

- $R_a(\phi\|\phi') = R_a(\phi_0\|\phi'_0) \cdot R_a(\phi_1\|\phi'_1)$ if Y_0 and Y_1 are independent for a in given interval.
- $R_a(\phi\|\phi') \leq R_\infty(\phi_0\|\phi'_0) \cdot \max_{\mathbf{y}_0 \in X} R_a(\phi_{1|0}(\cdot|y_0)\|\phi'_{1|0}(\cdot|y_0))$

– **Probability Preservation:** Let $E \subseteq \mathsf{Supp}(Q)$ be an arbitrary event. For some a: $\phi'(E) \geq \phi(E)^{\frac{a}{a-1}}/R_a(\phi\|\phi')$. Furthermore; $\phi'(E) \geq \phi(E)/R_\infty(\phi\|\phi')$.
– **Weak Triangle Equality:**

$$R_a(\phi\|\phi'') \leq R_a(\phi\|\phi') \cdot R_\infty(\phi'\|\phi'') \quad \text{and}$$
$$R_a(\phi\|\phi'') \leq R_\infty(\phi\|\phi')^{\frac{a}{a-1}} \cdot R_a(\phi'\|\phi'').$$

Lemma 3 ([LSS14, Lemma 4.2]). *Let ϕ and ϕ' be distributions corresponding to Gaussian $\mathcal{D}^m_{\mathbf{c},s}$ and $\mathcal{D}^m_{\mathbf{c}',s}$ with centers $\mathbf{c}$ and $\mathbf{c}'$, and $s \geq \eta(R^m)$. Then for any $a \in (1, +\infty)$:*

$$R_a(\phi||\phi') \leq \exp\left(a \cdot \pi \cdot \frac{\|\mathbf{c} - \mathbf{c}'\|_2^2}{s^2}\right)$$

Lemma 4 ([BLR+18, Theorem 4.2]). *Let ϕ, ϕ' denote two distributions with $\mathsf{Supp}(\phi) \subseteq \mathsf{Supp}(\phi')$, and $D_0(r)$ and $D_1(r)$ denote two distributions determined by some parameter $r \in \mathsf{Supp}(\phi')$. Let P, P' be two decision problems defined as follows:*

– *Problem P: Distinguish whether input y is sampled from distribution Y_0 or Y_1, where*

$$Y_0 = \{y\colon\ r \leftarrow \phi,\ y \leftarrow D_0(r)\}, \quad Y_1 = \{y\colon\ r \leftarrow \phi, y \leftarrow D_1(r)\}$$

– *Problem P': Distinguish whether input y is sampled from distribution Y'_0 or Y'_1, where*

$$Y'_0 = \{y\colon r \leftarrow\!\$\ \phi',\ y \leftarrow\!\$\ D_0(r)\}, \qquad Y'_1 = \{y\colon r \leftarrow\!\$\ \phi',\ y \leftarrow\!\$\ D_1(r)\}.$$

Assume that $D_0(\cdot)$ and $D_1(\cdot)$ satisfy the following public sampleability property: There exists a sampling algorithm S with runtime T_S such that for all (r, b), given any sample y from $D_b(r)$ over coins S:

– *$S(0, y)$ outputs a fresh sample distributed as $D_0(r)$.*
– *$S(1, y)$ outputs a fresh sample distributed as $D_1(r)$.*

Then, given a T-time distinguisher $\mathcal{A}$ for problem P with advantage ϵ we can construct a distinguisher $\mathcal{A}'$ for problem P' with runtime and distinguishing advantage respectively bounded from above and below, for any $a \in (1, \infty]$, by:

$$\frac{64}{\epsilon^2}\log\left(\frac{8R_a(\phi\|\phi')}{\epsilon^{a/(a-1)+1}}\right) \cdot (T_S + T) \text{ and } \frac{\epsilon}{4 \cdot R_a(\phi\|\phi')} \cdot \left(\frac{\epsilon}{2}\right)^{\frac{a}{a-1}}.$$

B.5 Multi-Proof Extractable NIZK

We recall the non-interactive zero-knowledge proof techniques first introduced by Lyubashevsky, Nguyen, and Plançon [LNP22], often referred to as the LNP framework. We will use these techniques to prove knowledge of short $\mathbf{s}_1$ satisfying equations of the form

$$\mathbf{P}\mathbf{s}_1 = \mathbf{v} \text{ over } R_q, \tag{4}$$

where $\mathbf{P} \in R_q^{n \times m}$ and $\mathbf{v} \in R^m$ are public matrices and vectors (resp.) of elements in R_q. In particular, we recall the multi-proof extractable[19] variant which is obtained after applying the Katsumata transform [Kat21] to the LNP framework [LNP22], as shown in [BLNS23]. We require this variant in order to prove the unforgeability property of our signature scheme wherein efficient extractors cannot be constructed from traditional rewinding arguments.

Our relations fit into this framework natively since parts of the relations to be proven already include the "additional" information required as part of the transform. Informally, Katsumata transform needs additional extractable commitments to the witness as part of the proof so the extractor can extract without rewinding. Since we are using BDLOP commitments as shown in Section 2.4, all we need to have is commitments to our witnesses. In Appendix D we will see that each relation contains commitments to the witnesses explicitly, which help us achieve straight-line extractability using multi-proof extractable NIZKs and BDLOP commitments without any additional cost.

B.6 Threshold Homomorphic Encryption

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

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

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

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

[19] This is sometimes called 'online' or 'straight-line' extractable.

$\mathsf{Exp}^{\mathrm{IND}}_{\mathcal{E}}(\mathcal{A})$

1. $\mathcal{A}$ outputs a set of indices of corrupted users $\mathcal{C}$ of size at most $t-1$. Let $\mathcal{H} := [n] \backslash \mathcal{C}$.
2. $\mathcal{A}$ (controlling parties in $\mathcal{C}$) runs $\mathsf{KGen}_{\mathcal{E}}$ with the parties in $\mathcal{H}$. The view of the honest parties defines a public key $\mathsf{pk}_{\mathcal{E}}$ as well as keys $\{\mathsf{sk}^{(i)}_{\mathcal{E}}\}_{i\in\mathcal{H}}$.
3. At any time during the experiment, $\mathcal{A}$ may make the "τ-th threshold decryption query" of the form $(\mathcal{U}^{(\tau)}, \{\mathsf{ptx}^{(\tau)}_j, \rho^{(\tau)}_{j,1}\}_{j\in[k^{(\tau)}]}, F^{(\tau)} \in \mathcal{F})$ up to a total of Q times. For each $\tau \in [Q]$, the query is answered as follows:
   - $\forall j \in [k^{(\tau)}]$: sample encryption randomness $\rho^{(\tau)}_{j,2}$ and compute $\mathsf{ctx}^{(\tau)}_j := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathsf{ptx}^{(\tau)}_j; (\rho^{(\tau)}_{j,1}, \rho^{(\tau)}_{j,2}))$.
   - Evaluate $F^{(\tau)}$ on $\{\mathsf{ctx}^{(\tau)}_j\}$ to get ciphertext $\mathsf{ctx}^{*(\tau)}$.
   - Let $\mathcal{H}_{\mathcal{U}} := \mathcal{H} \cap \mathcal{U}^{(\tau)}$. For $i \in \mathcal{H}_{\mathcal{U}}$, compute $\mathsf{ds}^{(\tau)}_i \leftarrow \mathsf{TDec}(\mathcal{U}^{(\tau)}, \mathsf{sk}^{(i)}_{\mathcal{E}}, \mathsf{ctx}^{*(\tau)})$.
   - Return decryption shares $\{\mathsf{ds}^{(\tau)}_i\}_{i\in\mathcal{H}_{\mathcal{U}}}$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs plaintexts ptx_0 and ptx_1, it samples $b \leftarrow\$ \{0,1\}$, compute $\mathsf{ctx}_b := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathsf{ptx}_b)$, and give ctx_b to $\mathcal{A}$.
5. $\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$.

Fig. 9. Experiment $\mathsf{Exp}^{\mathrm{IND}}_{\mathcal{E}}(\mathcal{A})$ parameterized by (t, n).

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

We define security in the threshold setting as follows:

Indistinguishability. $\mathcal{E}$ is (T, ϵ)-threshold indistinguishable if the probability of any adversary $\mathcal{A}$ running in time T succeeding in experiment $\mathsf{Exp}^{\mathrm{IND}}_{\mathcal{E}}(\mathcal{A})$ as depicted in Figure 9 is at most $1/2 + \epsilon$.

We note that our definition of threshold indistinguishability is a slight weakening of the one used in [GKS24]. Specifically, threshold decryption queries allow for the challenger to determine part of the encryption randomness used. This is necessary in order to argue that partial decryption shares appear uniform under the MLWE assumption. We emphasize that this weaker definition is enough to prove the security of our overall threshold signature scheme since in that setting, at least one party will always be honest and thus no adversary is able to fully determine the encryption randomness.

Similar to threshold signatures, we also define security with identifiable aborts for threshold homomorphic encryption.

Security with Identifiable Aborts $\mathcal{E}$ has (T, ϵ)-security with identifiable aborts if the probability of any adversary $\mathcal{A}$ running in time T succeeding in experiment $\mathsf{Exp}_{\mathcal{E}}^{\mathsf{ID\text{-}AB}}(\mathcal{A})$ as depicted in Figure 10 is at most ϵ.

$\mathsf{Exp}_{\mathcal{E}}^{\mathsf{ID\text{-}AB}}(\mathcal{A})$

1. $\mathcal{A}$ outputs a set of corrupted users $\mathcal{C}$ of size at most $t-1$. Let $\mathcal{H} := [n] \setminus \mathcal{C}$.
2. $\mathcal{A}$ (controlling parties in $\mathcal{C}$) runs $\mathsf{KGen}_{\mathcal{E}}$ with the parties in $\mathcal{H}$. The view of the honest parties defines a public key $\mathsf{pk}_{\mathcal{E}}$ as well as keys $\{\mathsf{sk}_{\mathcal{E}}^{(i)}\}_{i\in\mathcal{H}}$.
3. At any time during the experiment, $\mathcal{A}$ may make a "threshold decryption query" of the form $(\mathcal{U}^{(\tau)}, \{\mathsf{ptx}_j^{(\tau)}, \rho_j^{(\tau)}\}, F \in \mathcal{F})$, answered as follows:
   - $\forall j$: compute $\mathsf{ctx}_j^{(\tau)} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathsf{ptx}_j^{(\tau)}; \rho_j^{(\tau)})$.
   - Evaluate F on $\{\mathsf{ctx}_j^{(\tau)}\}$ to get ciphertext $\mathsf{ctx}^{*(\tau)}$.
   - Let $\mathcal{H}_{\mathcal{U}} := \mathcal{H} \cap \mathcal{U}^{(\tau)}$. For $i \in \mathcal{H}_{\mathcal{U}}$, compute $\mathsf{ds}_i^{(\tau)} \leftarrow \mathsf{TDec}(\mathcal{U}^{(\tau)}, \mathsf{sk}_{\mathcal{E}}^{(i)}, \mathsf{ctx}^{*(\tau)})$.
   - Return decryption shares $\{\mathsf{ds}_i^{(\tau)}\}_{i\in\mathcal{H}_{\mathcal{U}}}$ to $\mathcal{A}$.
4. At any time $\mathcal{A}$ can make a subset of corrupt parties $\mathcal{CS}^{(\tau)} \subseteq \mathcal{C}$ arbitrarily deviate from protocol execution.
5. Each honest party i then either continues with the protocol execution or aborts and outputs a set of parties $\mathcal{CS}_i^{(\tau)}$ that they think are cheating.
6. $\mathcal{A}$ succeeds if any of the following is true:
   - $\exists i, i' \in \mathcal{H}$ s.t. $\mathcal{CS}_i^{(\tau)} \neq \mathcal{CS}_{i'}^{(\tau)}$.
   - $\exists i \in \mathcal{H}$ s.t. $\mathcal{CS}_i^{(\tau)} \not\subseteq \mathcal{C}$ or $i' \in \mathcal{CS}_i^{(\tau)}$ but $i' \notin \mathcal{CS}^{(\tau)}$.
   - $\exists i \in \mathcal{H}, \exists i' \in \mathcal{C}$ s.t. $i' \notin \mathcal{CS}_i$ but $i' \in \mathcal{CS}^{(\tau)}$.
   - Combined decryption is $\perp$ or not equal to evaluation of $\{\mathsf{ptx}_j^{(\tau)}\}$ under F but $\mathcal{CS}^{(\tau)} = \emptyset$.

Fig. 10. Experiment $\mathsf{Exp}_{\mathcal{E}}^{\mathsf{ID\text{-}AB}}(\mathcal{A})$ parameterized by (t, n).

Additionally, we use the standard notion of commitments with *hiding* and *binding* properties. We instantiate our commitments using BDLOP commitment scheme defined in Section 2.4.

C Omitted Proofs

C.1 VSS Proofs

Lemma 5 (Privacy). *The* WSS *presented above is private.*

Proof. Let $\mathcal{C}$ denote the set of indices of parties controlled by the (static) adversary. We show that, if the dealer is honest then, at the end of the sharing phase, the view of the adversary is independent of the dealer's input $\mathbf{S}$. In the real world, the view and output of the adversary are

$$\{\{\mathsf{com}_{[\![\mathbf{S}]\!]_i}\}_{i\in[n]}, \pi_{\mathsf{VSS}}\} \quad \text{and} \quad \{([\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i})\}_{i\in\mathcal{C}},$$

respectively. By the zero-knowledge property of π_{VSS}, and the hiding property of the BDLOP commitment, all proofs and commitments with $i \in [n] \setminus \mathcal{C}$ can be simulated. Furthermore, since the sharing polynomials have degree $< t$ and $|\mathcal{C}| < t$, a standard argument implies $\{([\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i})\}_{i \in \mathcal{C}}$ reveals nothing about $\mathbf{S}$.

Lemma 6 (Correctness). *The* WSS *presented above is correct.*

Proof. Assuming an honest dealer, the correctness of π_{VSS} and of the BDLOP commitment scheme ensure that each party outputs some $\mathbf{S}_i \neq \perp$ after the sharing phase. We aim to show that $\mathbf{S}_i = [\![\mathbf{S}]\!]_i$.

Owing to the correctness of the constituent proofs π_1, π_2, and π_3, we have that the committed $\mathbf{S}$ is short, that the inner product of a random dual code word with the set of shares $\mathbf{S}_i$ is zero, and that a subset of size t of the $\mathbf{S}_i$ reconstructs to $\mathbf{S}$. By Lemma 2 we have that with overwhelming probability, the $\mathbf{S}_i$ are correct sharings of $\mathbf{S}$ and by the correctness of the BDLOP, we have $\mathbf{S}_i = [\![\mathbf{S}]\!]_i$.

Lemma 7 (Weak Commitment). *Our* WSS *satisfies the weak commitment property.*

Proof. Observe that if verification of π_{VSS} fails, all honest parties will output $\perp$ (i.e. the view of honest parties defines the value $\perp$). Similarly, all honest parties will output $\perp$ in the reconstruction phase.

We now proceed assuming no party aborts in the sharing phase. This scenario only arises if π_{VSS} verifies correctly. Then by the knowledge soundness of π_{VSS}, there exists a polynomial time extractor Ext (run by the simulator) which extracts a witness to $\mathcal{R}_{\mathsf{VSS}}^{\mathbf{S}^*}$ with overwhelming probability. Such a witness defines a value $\mathbf{S}^* \in R_q^{m_1 \times m_2}$ (the object that has been shared).

Now suppose a set $\mathcal{S}$ of $\geq t$ parties run the reconstruction phase. Without loss of generality assume $|\mathcal{S}| = t$ (e.g. by selecting a subset of size t is necessary). If, for honest $\mathcal{P}_i \in \mathcal{S}$, the check $\mathsf{Open}(\mathsf{pk}_{\mathsf{Com}}, \mathsf{com}_{[\![\mathbf{S}]\!]_j}, \mathsf{d} = ([\![\mathbf{S}]\!]_j, r_{[\![\mathbf{S}]\!]_j}, 1)) = 1$ fails for any $j \in \mathcal{S} \setminus \{i\}$, then $\mathcal{P}_i$ outputs $\perp$. Since this check is performed by all $\mathcal{P}_i \in \mathcal{S}$, if one honest party outputs $\perp$, all honest parties output $\perp$. Assuming that no honest party outputs $\perp$, the soundness of π_{VSS} ensures that all $\mathbf{S}_i$ values broadcast by members of $\mathcal{S}$ in reconstruction are genuine shares. By a similar argument to that used to prove correctness of our WSS, conditions 1 and 2 of Definition 5 are satisfied. In particular, all honest parties in $\mathcal{S}$ output $\mathbf{S}^*$.

C.2 Proof of Theorem 3

Proof. We prove security through a series of experiments.

Experiment G_0**.** The first experiment corresponds to the threshold ciphertext indistinguishability experiment. $\mathcal{A}$ first outputs a set of corrupted users $\mathcal{C}$ of size at most $t-1$, views of which are always given to $\mathcal{A}$ throughout the game. The challenger then runs $\mathsf{KGen}_{\mathcal{E}}$ to generate public keys and the secret key shares sk_i and sends keys for corrupted parties to $\mathcal{A}$. $\mathcal{A}$ then makes threshold decryption queries for different sets of users $\mathcal{U}$ and messages $\mathbf{m}$. The challenger encrypts $\mathbf{m}$

and computes partial shares for each participant in $\mathcal{U}$ and gives them to $\mathcal{A}$. At the end, $\mathcal{A}$ outputs two messages $\mathbf{m}_0$ and $\mathbf{m}_1$. The challenger randomly chooses one of the messages and returns the ciphertext to $\mathcal{A}$. If $\mathcal{A}$ guesses correctly, it is said to win the game.

$$\Pr[\mathbf{G}_0] = \mathsf{Adv}^{\mathrm{IND}}_{\mathcal{E}}(\mathcal{A}).$$

Experiment G_1**.** We first change how $\pi_{\mathsf{KGen}_\mathcal{E}}$ is computed for all honest parties. Instead of honestly computing $\pi_{\mathsf{KGen}_\mathcal{E}}$ each honest party calls the simulator $\mathsf{Sim}_{\mathsf{KGen}_\mathcal{E}}$ and sends a simulated proof instead. The rest of the experiment is the same. The only difference between G_0 and G_1 is then how NIZKAoKs are computed for honest parties. By the zero-knowledge property of the underlying proof system, these are indistinguishable. We then have:

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0]| \leq |\mathcal{H}| \cdot \epsilon_{\mathsf{ZK}}.$$

Experiment G_2**.** Now, we extract $\{\mathsf{sk}_{j,k}\}_{k\in[n]}$ for each corrupted party $\mathcal{P}_j$. When honest parties receive $\pi_{\mathsf{KGen}_\mathcal{E}}$ from the corrupted parties, they call $\mathsf{Extract}_{\mathsf{KGen}_\mathcal{E}}$ to obtain $\mathbf{S}_{j,k}, \rho_{j,k}$ for all $j \in \mathcal{C}, k \in [n]$, abort if the extraction fails. Then honest parties continue to compute sk_j. The rest is the same. G_2 is then indistinguishable from G_1 as long as the extraction from $\pi_{\mathsf{KGen}_\mathcal{E}}$ does not fail. Hence:

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1]| \leq |\mathcal{C}| \cdot \epsilon_{\mathsf{extract}}.$$

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

In G_2 the partial decryption share ds_i is computed as $\mathsf{ds}_i := \lambda_i \mathbf{S}_i^\top \mathbf{u} + qE_i$, whereas here

$$\begin{aligned}
\mathsf{ds}'_i &= \mathbf{v} - \mathbf{m} - \sum_{j\in\mathcal{U}, j\neq i} \lambda_j \mathbf{S}_j^\top \mathbf{u} + qE'_i \\
&= \mathbf{v} - \mathbf{m} - (\mathbf{S}_\mathcal{E}^\top \mathbf{u} - \lambda_i \mathbf{S}_i^\top \mathbf{u}) + qE'_i \\
&= \mathbf{v} - \mathbf{S}_\mathcal{E}^\top \mathbf{u} - \mathbf{m} + \lambda_i \mathbf{S}_i^\top \mathbf{u} + qE'_i.
\end{aligned}$$

For Q partial decryption queries and an appropriate choice of σ_{TDec}, we bound the loss between G_2 and G_3 using Lemma 4. Define the randomness r as the additional term in each of i's decryption shares, i.e., $\mathsf{ds}_i = \mathbf{v} - \mathbf{m} - \sum_{j\in\mathcal{U}, j\neq i} \lambda_j \mathbf{S}_j^\top \mathbf{u} + r$. In G_2 and G_3 we have $r = qE_i - (\mathbf{v} - \mathbf{S}_\mathcal{E}^\top \mathbf{u} - \mathbf{m})$ and $r = qE'_i$, respectively; denote by ϕ and ϕ' the two corresponding distributions of r, aggregated over the Q queries. For a challenge bit $b \in \{0, 1\}$, consider $\mathcal{A}$'s view in the indistinguishability game as the following distribution:

$$D_b(r) = \{\mathsf{pk}_\mathcal{E}, \{\mathbf{S}_j\}_{j\in\mathcal{C}}, \{\mathsf{ptx}^{(\tau)}, \rho^{(\tau)}, \mathcal{U}^{(\tau)}, \{\mathsf{ds}_j^{(\tau)}\}_{j\in\mathcal{U}}\}_{\tau\in[Q]}, \mathsf{ptx}_0, \mathsf{ptx}_1, \mathsf{ctx}_b\}$$

Distinguishing $D_0(r)$ from $D_1(r)$ for $r \leftarrow \phi$ is exactly the task of $\mathcal{A}$ in G_2 and corresponds to problem P of Lemma 4, while the same task for $r \leftarrow \phi'$ is the task of $\mathcal{A}$ in G_3 and corresponds to problem P'.

To invoke Lemma 4, we have to show that the public sampleability property is satisfied. We define the sampler $S(b', y)$ that on input $b' \in \{0, 1\}$ and a sample y from distribution $D_b(r)$ generates a fresh sample from $D_{b'}(r)$. $S(b', y)$ is defined as follows: Upon receiving

$$y = \{\mathsf{pk}_{\mathcal{E}}, \{\mathbf{S}_j\}_{j\in\mathcal{C}}, \{\mathsf{ptx}^{(\tau)}, \rho^{(\tau)}, \mathcal{U}^{(\tau)}, \{\mathsf{ds}_j^{(\tau)}\}_{j\in\mathcal{U}}\}_{\tau\in[Q]}, \mathsf{ptx}_0, \mathsf{ptx}_1, \mathsf{ctx}_b\},$$

the sampler replaces the challenge ciphertext ctx_b in y by a fresh encryption $\mathsf{ctx}_{b'} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathsf{ptx}_{b'})$; all remaining components of y are left unchanged. This can be done publicly since $\mathsf{pk}_{\mathcal{E}}$, ptx_0, and ptx_1 are part of y and the challenge ciphertext is generated from fresh randomness, independent of the remaining components; in particular the decryption shares, and thus the dependence on r, are untouched. Thus we conclude that $D_b(r)$ is publicly sampleable.

It remains to bound $R_a(\phi\|\phi')$. Both distributions are Gaussians of parameter $q\sigma_{\mathsf{TDec}}$ whose centers differ by $\mathbf{v}-\mathbf{S}_{\mathcal{E}}^{\top}\mathbf{u}-\mathbf{m}$, so by Lemma 3, the Rényi divergence of order a at the τ-th decryption query satisfies:

$$R_a(\phi^{(\tau)}\|\phi'^{(\tau)}) \leq \exp\left(a \cdot \pi \cdot \frac{\left\|\mathbf{v} - \mathbf{S}_{\mathcal{E}}^{\top}\mathbf{u} - \mathbf{m}\right\|_2^2}{q^2\sigma_{\mathsf{TDec}}^2}\right)$$

Using the multiplicative property for aggregating Q queries, we have:

$$R_a(\phi\|\phi') \leq \exp\left(a \cdot \pi \cdot Q \cdot \frac{\left\|\mathbf{v} - \mathbf{S}_{\mathcal{E}}^{\top}\mathbf{u} - \mathbf{m}\right\|_2^2}{q^2\sigma_{\mathsf{TDec}}^2}\right)$$

Since $(\mathbf{u}, \mathbf{v})$ is correctly generated we have:

$$\begin{aligned}\mathbf{v} - \mathbf{S}^{\top}\mathbf{u} - \mathbf{m} &= \mathbf{B}_{\mathcal{E}}^{\top}\mathbf{r} + q \cdot \mathbf{e}_2 + \mathbf{m} - \mathbf{S}_{\mathcal{E}}^{\top}(\mathbf{A}_{\mathcal{E}}^{\top}\mathbf{r} + q \cdot \mathbf{e}_1) - \mathbf{m} \\ &= \mathbf{S}_{\mathcal{E}}^{\top}\mathbf{A}_{\mathcal{E}}^{\top}\mathbf{r} + q\mathbf{E}_{\mathcal{E}}^{\top}\mathbf{r} + q \cdot \mathbf{e}_2 - \mathbf{S}_{\mathcal{E}}^{\top}\mathbf{A}_{\mathcal{E}}^{\top}\mathbf{r} - q\mathbf{S}_{\mathcal{E}}^{\top} \cdot \mathbf{e}_1 \\ &= q(\mathbf{E}_{\mathcal{E}}^{\top}\mathbf{r} + \mathbf{e}_2 - \mathbf{S}_{\mathcal{E}}^{\top} \cdot \mathbf{e}_1)\end{aligned}$$

which is the noise in the ciphertext. Denoting by B_2 an upper bound on the 2-norm of this noise term, we obtain

$$R_a(\phi\|\phi') \leq \exp\left(a \cdot \pi \cdot Q \cdot \frac{B_2^2}{q^2\sigma_{\mathsf{TDec}}^2}\right).$$

If $\sigma_{\mathsf{TDec}} \geq \sqrt{\frac{Q \cdot B_2^2}{q^2}}$, we have $R_a(\phi\|\phi') \leq \exp(\pi \cdot a)$. By Lemma 4, a distinguisher with advantage $\Pr[\mathbf{G}_2]$ in G_2 yields a distinguisher for G_3 with advantage

$$\Pr[\mathbf{G}_3] \geq \frac{\Pr[\mathbf{G}_2]}{4 \cdot R_a(\phi\|\phi')} \cdot \left(\frac{\Pr[\mathbf{G}_2]}{2}\right)^{\frac{a}{a-1}} \geq \frac{\Pr[\mathbf{G}_2]^{\frac{2a-1}{a-1}}}{2^{\frac{a}{a-1}+2} \cdot \exp(\pi a)},$$

and hence

$$\Pr[\mathbf{G}_2] \leq \left(2^{\frac{a}{a-1}+2} \cdot \exp(\pi a) \cdot \Pr[\mathbf{G}_3]\right)^{\frac{a-1}{2a-1}}.$$

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

All it remains to show is that $\lambda_j \mathbf{S}_j^\top \mathbf{u} + qE_j$ is indistinguishable from uniform. Since λ_j are invertible in $\mathbb{Z}_{\hat{q}}$, if we can argue $\lambda_j \mathbf{S}_j^\top \mathbf{u} + qE_j \approx \mathbf{S}_j^\top \mathbf{u} + \lambda_j^{-1} qE_j = \mathbf{S}_j^\top \mathbf{u} + q'E_j \approx \mathbf{u}^\top \mathbf{S}_j + q'E_j^\top$, for invertible q' and relatively small E_j considering distribution independent of vector orientation, then we can argue that $\lambda_j \mathbf{S}_j^\top \mathbf{u} + qE_j$ is indistinguishable from uniform.

We first argue that $\mathcal{A}$ distinguishing $\mathbf{S}_j^\top \mathbf{u} + \lambda_j^{-1} qE_j$ from uniform ds_j can be used to break MLWE: The MLWE distinguisher interacts with $\mathcal{A}$ and uses its replies in G_3 to answer MLWE challenges. The distinguisher obtains an MLWE challenge $(\mathbf{A}, \mathbf{t}^\top)$, for each partial decryption query and sets the first ciphertext element $\mathbf{u} = \mathbf{a}_i$ (where $\mathbf{a}_i$ is the first row of $\mathbf{A}$) and partial decryption share $\mathsf{ds}_j = \mathbf{t}$. During each query, if $\mathcal{A}$ behaves noticeably different than G_3, the MLWE distinguisher replies that $\mathbf{t}$ is uniform. Note, we can replace $\mathbf{u}$ with $\mathbf{a}_i$ here since $\mathbf{u}$ is indistinguishable from uniform to the adversary by the underlying indistinguishability of the BGV encryption scheme. Crucially, this is ensured by the simulator's input ρ_2 to the encryption randomness.

Since λ_j is invertible and q is a prime, λ_j^{-1} exists and $\lambda^{-1}q$ is relatively prime with $\hat{q}$. Hence, if $\mathbf{t}^\top$ is an MLWE instance, then there is no reason for the adversary to behave differently than in G_3 since $\mathbf{a}_i^\top$ is indistinguishable from $\mathbf{u}^\top \mathbf{S}_j + \lambda_j^{-1} qE_i^\top$ for $\lambda_j^{-1}q$ relatively prime with $\hat{q}$ [BGV12]. If $\mathbf{t}$ is uniform, any significant advantage $\mathcal{A}$ has is MLWE distinguisher advantage. Hence we conclude $\mathbf{S}_j^\top \mathbf{u} + \lambda_j^{-1} qE_j$ is indistinguishable from uniform ds_j.

If $\mathbf{S}_j^\top \mathbf{u} + \lambda_j^{-1} qE_j$ is indistinguishable from uniform ds_j then $\lambda_j(\mathbf{S}_j^\top \mathbf{u} + \lambda_j^{-1} qE_j) = \lambda_j \mathbf{S}_j^\top \mathbf{u} + qE_j$ is indistinguishable from $\lambda_j \mathsf{ds}_j$. Since ds_j is uniform in $R_{\hat{q}}$, so is $\lambda_j \mathsf{ds}_j$. Thus we conclude that $\lambda_j \mathbf{S}_j^\top \mathbf{u} + qE_j$ is indistinguishable from uniform ds_j by $\mathsf{MLWE}_{N,q,\mathcal{D}_{\sigma_{\mathsf{LWE}}}}$ assumption.

$$|\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_3]| \leq (|\mathcal{H}_\mathcal{U}| - 1) \cdot \epsilon_{\mathsf{MLWE}}.$$

Experiment G_5**.** Now that partial decryption does not rely on combined secret key, for i we now replace the public key share $\mathbf{B}_i$ for honest parties. Sample a random $\mathbf{B}_i$ and t-out-of-n secret share it into $\{\mathbf{B}_{i,j}\}_{j\in[n]}$. Then derive $\mathbf{S}_{i,j}$, $\mathbf{E}_{i,j}$ from $\mathbf{B}_{i,j}$. The rest of the experiment remains the same.

We again show that $\mathcal{A}$ distinguishing between G_4 and G_5 can be used to break MLWE assumption. We initiate the challenger for $\mathsf{MLWE}_{N,q,\mathcal{D}_{\sigma_{\mathsf{LWE}}}}$ with $\mathbf{A}$ being the uniform element. We then forward the answer of the challenger as $\mathbf{B}_i$. If the $\mathcal{A}$ behaves significantly differently than in G_4 we reply the challenger as the sample being uniform.

$\mathbf{B}_i = \mathbf{A}\mathbf{S}_i + q\mathbf{E}_i$ is an MLWE sample hence if the challenger returns an MLWE sample, $\mathcal{A}$ has no reason to behave differently. If the challenger returns a uniform sample, however, any non-negligible advantage $\mathcal{A}$ has is non-negligible advantage in answering challenger's queries since any behavioral difference is forwarded directly. Since $\mathbf{B}_i$ is t-out-of-n secret shared, each $\mathbf{B}_{i,j}$ is uniform in R_q. This is statistically indistinguishable from the real execution where $\mathbf{S}_i$ and $\mathbf{E}_i$ are t-out-of-n shared therefore $\mathbf{S}_{i,j}$, $\mathbf{E}_{i,j}$ and consequently $\mathbf{B}_i = \mathbf{A}\mathbf{S}_i + q\mathbf{E}_i$ is uniform in R_q. G_5 is then indistinguishable from G_4 by the MLWE assumption and $\mathcal{A}$'s distinguishing advantage can be bounded by ϵ_{MLWE}.

$$|\Pr[\mathbf{G}_5] - \Pr[\mathbf{G}_4]| \leq |\mathcal{H}| \cdot \epsilon_{\mathsf{MLWE}}.$$

Experiment G_6**.** Since public key shares are also independent, we now replace unopened commitments with random elements. Instead of committing to $\{\mathbf{S}_{i,j}\}$, honest parties commit to random matrices instead. By the hiding property of Com,

$$|\Pr[\mathbf{G}_6] - \Pr[\mathbf{G}_5]| \leq |\mathcal{H}| \cdot \epsilon_{\mathsf{MLWE}}.$$

Experiment G_7**.** We now derive $\mathbf{B}_i$ a posteriori. Before step 1 of $\mathsf{KGen}_{\mathcal{E}}$, $\mathcal{B}$ receives a BGV key pair $(\mathbf{A}, \mathbf{B}_{\mathcal{E}})$. Instead of honestly computing H, $\mathcal{P}_i$ samples a random $h_i \leftarrow \{0,1\}^\lambda$ instead. Then derive $\mathbf{B}_i := \mathbf{B}_{\mathcal{E}} - \sum_{j\neq i\in[n]} \mathbf{B}_j$ and programs H such that $h_i = \mathsf{H}(i, \mathbf{B}_i)$. The rest of the experiment is the same as before.

By the MLWE assumption, $\mathbf{B}_{\mathcal{E}}$ is indistinguishable from uniform. Then as long as programming of H does not fail, G_6 and G_7 are indistinguishable.

$$|\Pr[\mathbf{G}_7] - \Pr[\mathbf{G}_6]| \leq \frac{Q_H}{2^\lambda}.$$

We now show that if $\mathcal{A}$ wins G_7, then $\mathcal{A}$ can be used to break the IND-CPA security of BGV scheme. $\mathcal{B}$ interacts with the challenger to IND-CPA security of BGV and obtains $\mathbf{A}, \mathbf{B}$ which then are used during $\mathsf{KGen}_{\mathcal{E}}$. Whenever $\mathcal{A}$ sends the challenge plaintexts $\mathbf{m}_0$ and $\mathbf{m}_1$ $\mathcal{B}$ forwards them to the BGV challenger and obtains the ciphertext ctx^*. $\mathcal{B}$ sends ctx^* along with its simulated proof to $\mathcal{A}$. When $\mathcal{A}$ replies with 0 or 1, $\mathcal{B}$ forwards the response to BGV challenger.

The public key $\mathcal{A}$ generates as part of $\mathsf{KGen}_{\mathcal{E}}$ contains the same $(\mathbf{A}, \mathbf{B})$ as $\mathcal{B}$ received from the BGV oracle as a challenge. Hence ctx^* received by $\mathcal{B}$ and consequently $\mathcal{A}$ is a BGV ciphertext encrypted under $\mathbf{A}, \mathbf{B}$ which is indistinguishable

from ctx derived as part of the protocol execution. Then, a correct answer given by $\mathcal{A}$ is also a correct answer for $\mathcal{B}$ for the encryption oracle.

$$|\Pr[\mathbf{G}_7]| \leq \epsilon_{\text{IND-CPA}}.$$

Collecting the hops from G_3 to G_7, we have

$$\Pr[\mathbf{G}_3] \leq (2|\mathcal{H}| + |\mathcal{H}_{\mathcal{U}}| - 1) \cdot \epsilon_{\mathsf{MLWE}} + \frac{Q_H}{2^\lambda} + \epsilon_{\text{IND-CPA}} = \epsilon',$$

and therefore

$$\mathsf{Adv}^{\text{IND}}_{\mathcal{E}}(\mathcal{A}) \leq |\mathcal{H}| \cdot \epsilon_{\mathsf{ZK}} + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract}} + \left(2^{\frac{a}{a-1}+2} \cdot e^{\pi a} \cdot \epsilon'\right)^{\frac{a-1}{2a-1}},$$

matching the statement of the theorem.

This concludes the proof.

C.3 Proof of Theorem 4

Proof. We first start by observing the fact in the synchronous model with access to an authenticated broadcast, each honest party has access to the same communication transcript. Then, throughout the protocol every check and consequent abort by honest parties will have the same result and therefore will be consistent. Hence any corrupted set $\mathcal{CS}$ output by honest parties will be the same and will not include any honest parties. Similarly, if any corrupted user does not send their message, it will be noticed by all honest parties looking at the transcript. Then by assumption $\mathcal{A}$ cannot succeed in $\mathsf{Exp}^{\mathsf{ID\text{-}AB}}_{\mathcal{S}}$ by first two conditions.

If $\mathcal{A}$ does not cause abort, then protocol proceeds as intended and by the correctness of the scheme there exists no $\{\mathsf{ptx}_j\}$ or $F \in \mathcal{F}$ such that decryption fails. Thus, $\mathcal{A}$ cannot win through the fourth condition either.

It remains to argue that $\mathcal{A}$ cannot win through the third condition which we now show cannot be the case for any of the checks in protocol.

During $\mathsf{KGen}_{\mathcal{E}}$, $\mathcal{A}$ can only succeed if any $j \in \mathcal{C}$ can send h_j, $\pi^{(j)}_{\mathsf{KGen}_{\mathcal{E}}}$, $\{\mathsf{com}_{j,k}\}$ that is inconsistent with the protocol information and not detected by any user in $\mathcal{H}$. This can only be the case if (i) j found $\mathbf{B}^*_j \neq \mathbf{B}_j$ where $\mathsf{H}(j, \mathbf{B}^*_j) = \mathsf{H}(j, \mathbf{B}_j)$ (ii) j found $\mathbf{S}^*_{j,k}, \rho^*_{j,k} \neq \mathbf{S}_{j,k}, \rho_{j,k}$ where $\mathsf{com}_{j,k} = \mathsf{Com}(\mathbf{S}^*_{j,k}; \rho^*_{j,k})$ or (iii) j sent $\pi^{(j)}_{\mathsf{KGen}_{\mathcal{E}}}$ without knowledge of satisfying $\mathbf{S}_j$, $\mathbf{E}_j$ or their t-out-of-n shares. If (i) is the case, then j found a collision for H, which does not happen for all but negligible probability. If (ii) is the case, then j can be used as a subroutine to break the binding property of the commitment of the underlying commitment scheme Com. If (iii) is the case however j can be used as a cheating prover for the underlying NIZKAoK $\Pi_{\mathsf{KGen}_{\mathcal{E}}}$, which once again cannot happen for all but negligible probability by the knowledge soundness assumption.

Since F in question is public, final ctx^* will be consistent for each honest party. For TDec and Comb, $\mathcal{A}$ can only succeed if j can send a ds_j without correctly computing it. If that's the case, j can be used as a cheating prover

for the underlying NIZKAoK Π_{ds}, which once again cannot happen for all but negligible probability by the knowledge soundness assumption.

Since $\mathcal{A}$ cannot make any $j \in \mathcal{C}$ cheat without detection with non-negligible probability, it cannot succeed in the identifiable aborts game. This concludes the proof.

C.4 Proof of Theorem 6

Proof. It is clear that the check $c = c^*$ holds. Assuming the correctness of the linearly homomorphic threshold encryption scheme in Figure 5, we have that

$$\mathbf{z} = c \cdot \mathbf{s} + \sum_{j \in \mathcal{U}} \mathbf{r}_j.$$

Then, by an analogous argument to [GKS24, Lemma C.1], the verification check $\|\mathbf{z}, 2^{\kappa_{\mathbf{w}}} \cdot \mathbf{h}\| \leq B_{2,\mathbf{z}}$ holds if

$$B_{2,\mathbf{z}} = e^{1/4} \cdot (n\nu\sigma_{\mathbf{y}} + t\sigma_{\mathbf{w}})\sqrt{N(k+\ell)} + (\nu \cdot 2^{\kappa_{\mathbf{y}}} + 2^{\kappa_{\mathbf{w}}+1}) \cdot \sqrt{Nk}.$$[20]

It now remains to show the correctness of the linearly homomorphic encryption scheme. More precisely, we must choose parameters so that

$$\mathsf{Comb}(\mathsf{ctx}_{\mathbf{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}}) = c \cdot \mathbf{s} + \sum_{j \in \mathcal{U}} \mathbf{r}_j,$$

with probability 1. Denoting by $\mathsf{ctx}_{\mathbf{z}} = (\mathbf{v}_{\mathbf{z}}, \mathbf{u}_{\mathbf{z}})$ the encryption of $\mathbf{z}$, we have that

$$\mathsf{Comb}(\mathsf{ctx}_{\mathbf{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}}) = (\mathbf{v}_{\mathbf{z}} - \sum_{j \in \mathcal{U}} \mathsf{ds}_{\mathbf{z},j}) \mod \hat{q} \mod q,$$

where $\mathsf{ds}_{\mathbf{z},j}$ is the decryption share of user j performing partial decryption of $\mathsf{ctx}_{\mathbf{z}}$. Thus, for correct combination, it suffices to set parameters such that

$$\|\mathbf{v}_{\mathbf{z}} - \sum_{j \in \mathcal{U}} \mathsf{ds}_{\mathbf{z},j}\|_\infty < \hat{q}/2.$$

Now consider

$$\mathbf{v}_{\mathbf{z}} - \sum_{j \in \mathcal{U}} \mathsf{ds}_{\mathbf{z},j} = (c \cdot \sum_{i \in [n]} \mathbf{v}_{\mathbf{s}_i} + \sum_{j \in \mathcal{U}} \mathbf{v}_{\mathbf{r}_j}) - \mathbf{S}_{\mathcal{E}}^{\top}(c \cdot \sum_{i \in [n]} \mathbf{u}_{\mathbf{s}_i} + \sum_{j \in \mathcal{U}} \mathbf{u}_{\mathbf{r}_j}) - q \cdot \sum_{j \in \mathcal{U}} E_j,$$

[20] Note that while $\mathbf{s}_i$ and $\mathbf{r}_i$ are sampled as Gaussians, the sum is not guaranteed to be a Gaussian since we only check for bounds in our NIZKAoKs.

where $\mathbf{S}_{\mathcal{E}} := \sum_{j\in\mathcal{U}} \lambda_j \mathbf{S}_{\mathcal{E}}^{(j)}$ and λ_j is the j'th Lagrange coefficient. Then,

$$\begin{aligned}
\mathbf{v}_{\mathbf{z}} - \sum_{j\in\mathcal{U}} \mathsf{ds}_{\mathbf{z},j} &= c \cdot \sum_{i\in[n]} (\mathbf{B}_{\mathcal{E}}^{\top} \mathbf{r}_{\mathbf{s}_i} + q \cdot \mathbf{e}_{\mathbf{s}_i,2} + \mathbf{s}_i) \\
&\quad - c \cdot \sum_{i\in[n]} \mathbf{S}_{\mathcal{E}}^{\top} (\mathbf{A}_{\mathcal{E}}{}^{\top} \mathbf{r}_{\mathbf{s}_i} + q \cdot \mathbf{e}_{\mathbf{s}_i,1}) \\
&\quad + \sum_{j\in\mathcal{U}} (\mathbf{v}_{\mathbf{r}_j} - \mathbf{S}_{\mathcal{E}}^{\top} \mathbf{u}_{\mathbf{r}_j}) - q \cdot \sum_{j\in\mathcal{U}} E_j \\
&= (c \cdot \sum_{i\in[n]} \mathbf{s}_i + \sum_{j\in\mathcal{U}} \mathbf{r}_j) \\
&\quad + c \cdot \sum_{i\in[n]} (\mathbf{E}_{\mathcal{E}}^{\top} \mathbf{r}_{\mathbf{s}_i} + q \cdot \mathbf{e}_{\mathbf{s}_i,2} - q \cdot \mathbf{S}_{\mathcal{E}}^{\top} \mathbf{e}_{\mathbf{s}_i,1}) \\
&\quad + \sum_{j\in\mathcal{U}} (\mathbf{E}_{\mathcal{E}}^{\top} \mathbf{r}_{\mathbf{r}_j} + q \cdot \mathbf{e}_{\mathbf{r}_j,2} - q \cdot \mathbf{S}_{\mathcal{E}}^{\top} \mathbf{e}_{\mathbf{r}_j,1}) - q \cdot \sum_{j\in\mathcal{U}} E_j.
\end{aligned}$$

Thus

$$\begin{aligned}
\|\mathbf{v}_{\mathbf{z}} - \sum_{j\in\mathcal{U}} \mathsf{ds}_{\mathbf{z},j}\|_{\infty} \leq\ & \nu \cdot n \cdot t_2 \cdot \sigma_{\mathbf{y}} \cdot \sqrt{N\ell} \\
& + t \cdot t_2 \cdot \sigma_{\mathbf{w}} \cdot \sqrt{N\ell} \\
& + \nu \cdot n \cdot \hat{k} \cdot t_2^2 \cdot \sigma_{\mathbf{B}_{\mathcal{E}}} \cdot N \cdot \sigma_{\mathsf{ctx}} \\
& + n \cdot \nu \cdot q \cdot t_2 \cdot \sigma_{\mathsf{ctx}} \cdot \sqrt{N} \\
& + q \cdot \nu \cdot n \cdot \hat{\ell} \cdot t_2^2 \cdot \sigma_{\mathbf{B}_{\mathcal{E}}} \cdot N \cdot \sigma_{\mathsf{ctx}} \\
& + t \cdot t_2^2 \cdot \sigma_{\mathbf{B}_{\mathcal{E}}} \cdot N \cdot \sigma_{\mathsf{ctx}} \cdot \hat{k} \\
& + q \cdot t \cdot t_2 \cdot \sigma_{\mathsf{ctx}} \cdot \sqrt{N} \\
& + t \cdot q \cdot \hat{\ell} \cdot t_2^2 \cdot \sigma_{\mathbf{B}_{\mathcal{E}}} \cdot N \cdot \sigma_{\mathsf{ctx}} \\
& + q \cdot t \cdot t_2 \cdot \sigma_{\mathsf{TDec}}. \qquad (5)
\end{aligned}$$

C.5 Proof of Theorem 7

Proof. We prove this through a series of hybrids written out in full detail. The simulator $\mathcal{B}$ is communicating with a challenger $\mathcal{D}$, and we show that a forgery by $\mathcal{A}$ can be used by $\mathcal{B}$ to answer $\mathcal{D}$.

$\boldsymbol{G}_0$*:* The first game corresponds to the real world. Specifically, $\mathcal{B}$ follows the protocol according to the description, and $\mathcal{A}$ interacts with $\mathcal{B}$ arbitrarily. The oracles H_0,H_1, and H_2 are simulated using tables $\mathcal{HT}_0$, $\mathcal{HT}_1$, and $\mathcal{HT}_2$. Since we assume a rushing adversary, $\mathcal{B}$ sends its messages first. When $\mathcal{A}$ outputs a forgery $(\Psi^* = (c^*, \mathbf{z}^*, \mathbf{h}^*), \mu^*)$, $\mathcal{B}$ aborts if $\mu^* \in \mathcal{M}$. If not, it computes

$\mathbf{w}^* := \lfloor \mathbf{A}_{\mathcal{S}} \mathbf{z}^* - 2^{\kappa_{\mathbf{y}}} c^* \mathbf{y}' \rceil_{q_{\mathbf{w}}} + \mathbf{h}^*$. If the challenge $c^* \neq \mathsf{H}_2(\mathsf{pk}_{\mathcal{S}}, \mathbf{w}^*, \mu^*)$, $\mathcal{B}$ aborts, otherwise halts with output (Ψ^*, μ^*) and $\mathcal{A}$ succeeds.

If the real world is indistinguishable from the programmable random oracle model, then the random oracle simulation is perfect, $\mathcal{B}$'s behavior is exactly the same as in the unforgeability experiment, and we get that

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

$\mathbf{G}_1$: In this game $\mathcal{B}$ changes how non-interactive zero-knowledge proofs for $\mathsf{KGen}_{\mathcal{S}}$ 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 $i \in \mathcal{H}$, $\mathcal{B}$ uses the corresponding honest-verifier zero-knowledge simulator $\mathsf{Sim}_{\mathbf{s}}$ for the relation $\mathcal{R}_{\mathbf{s}}$ where the proof $\pi_{\mathbf{s},i} := \mathsf{Sim}(\mathbf{A}_{\mathcal{S}}, \mathbf{y}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s},i}, B_{\mathbf{s}})$. $\mathcal{B}$ follows the remaining parts of the protocol honestly. $\mathbf{G}_0$ and $\mathbf{G}_1$ are indistinguishable by the ZK property of the NIZK:

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

$\mathbf{G}_2$: This is the same as $\mathbf{G}_1$ except $\mathcal{B}$ uses the extractability properties of the NIZK to learn the signing key shares in $\mathcal{C}$. During key generation, after receiving $\pi_{\mathbf{s},j}$ for $j \in \mathcal{C}$, $\mathcal{B}$ calls the extractor $\mathbf{s}_j := \mathsf{Extract}_{\mathbf{s}}(\pi_{\mathbf{s},j}; \mathbf{A}_{\mathcal{S}}, \mathbf{y}_j, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s},j}, B_{\mathbf{s}})$. If any of the extractions fails, $\mathcal{B}$ aborts.

By assumption, the extractor $\mathsf{Extract}_{\mathbf{s}}$ for $\mathcal{R}_{\mathbf{s}}$ is efficiently computable. If $\mathcal{B}$ does not abort due to a failed extraction, $\mathbf{G}_1$ and $\mathbf{G}_2$ are indistinguishable for $\mathcal{A}$. $\mathcal{A}$'s advantage in $\mathbf{G}_2$ is bounded by the sum of independent extraction failures:

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

$\mathbf{G}_3$: In this game $\mathcal{B}$ changes how non-interactive zero-knowledge proofs for $\mathsf{Sign}_{\mathcal{S}}$ are answered for honest parties. $\mathcal{B}$ executes the protocol the same as $\mathbf{G}_2$, but instead of honestly generating $\pi_{\mathbf{r},i}$ for $i \in \mathcal{H}_{\mathcal{U}}$, $\mathcal{B}$ uses the corresponding honest-verifier zero-knowledge simulator, $\mathsf{Sim}_{\mathbf{r}}$ for relation $\mathcal{R}_{\mathbf{r}}$ where the proof of boundedness $\pi_{\mathbf{r},i} := \mathsf{Sim}(\mathbf{A}_{\mathcal{S}}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{r},i}, B_{\mathbf{r}})$. $\mathcal{B}$ follows the remaining parts of the protocol honestly. $\mathbf{G}_2$ and $\mathbf{G}_3$ are indistinguishable by the ZK property of the NIZK:

$$\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2] \leq |\mathcal{H}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{ZK},\mathbf{r}}.$$

$\mathbf{G}_4$: $\mathbf{G}_4$ is the same as $\mathbf{G}_3$ except $\mathcal{B}$ uses the extractability properties of the $\mathsf{Sign}_{\mathcal{S}}$ NIZK to learn the signature randomness encrypted by parties in $\mathcal{C}_{\mathcal{U}}$. During signing after receiving $\pi_{\mathbf{r},i}$, $\mathcal{B}$ computes $\mathbf{r}_i := \mathsf{Extract}_{\mathbf{r}}(\pi_{\mathbf{r},i}; \mathbf{A}_{\mathcal{S}}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{r},i}, B_{\mathbf{r}})$. If any of the extractions fails, $\mathcal{B}$ aborts.

By assumption, the extractor $\mathsf{Extract}_{\mathbf{r}}$ for $\mathcal{R}_{\mathbf{r}}$ is efficiently computable. If $\mathcal{B}$ does not abort due to a failed extraction, $\mathbf{G}_3$ and $\mathbf{G}_4$ are indistinguishable for $\mathcal{A}$. $\mathcal{A}$'s advantage in $\mathbf{G}_4$ is bounded by the sum of independent extraction failures:

$$\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_3] \leq |\mathcal{C}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{extract},\mathbf{r}}.$$

G_5: $\mathbf{G}_5$ is exactly the same as $\mathbf{G}_4$ except $\mathcal{B}$ computes and sends a random hash commitment during signing for each honest party then programs the oracle a posteriori. For each $i \in \mathcal{H}_\mathcal{U}$, after computing $\mathbf{r}_i, \mathbf{e}'_i, \mathbf{w}_i, \mathsf{ctx}_{\mathbf{r},i}$, $\mathcal{B}$ samples random $h_{\mathbf{w},i} \in \{0,1\}^{\ell_1}$ and broadcasts it. After receiving $h_{\mathbf{w},j}$ for $j \in \mathcal{C}_\mathcal{U}$, $\mathcal{B}$ programs the oracle so that $h_{\mathbf{w},i} = \mathsf{H}_1(i, \mathbf{w}_i)$ and aborts if the programming fails at any point. The rest remains the same. Since each $h_{\mathbf{w},i}$ is sampled from the range of H_1, $\mathbf{G}_5$ is then indistinguishable from $\mathbf{G}_4$ as long as programming does not fail. Hence:

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

G_6: In this hybrid $\mathcal{B}$ changes how ciphertexts are computed. $\mathcal{B}$ 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 $\mathsf{ctx}_{\mathbf{r},i'} := \mathsf{Enc}(\mathsf{pk}_\mathcal{E}, 0)$ during signing. The rest of the game is as it is in $\mathbf{G}_5$. Since $\mathbf{r}_j$ and $\mathbf{s}_j$ for $j \in \mathcal{C}_\mathcal{U}$ are extracted in $\mathbf{G}_4$ and $\mathbf{G}_2$, respectively, $\mathcal{B}$ can compute $\mathbf{z} = c\sum_{i\in\mathcal{U}} \mathbf{s}_i + \sum_{i\in\mathcal{U}} \mathbf{r}_i$, which is the combined signature hence the threshold decryption can be simulated. We can now reduce to the indistinguishability of $\mathcal{E}$:

$$|\Pr[\mathbf{G}_6] - \Pr[\mathbf{G}_5]| \leq \epsilon_{\mathrm{IND}}.$$

G_7: In this hybrid $\mathcal{B}$ changes how $h_{\mathbf{y},i'}$ is computed. $\mathcal{B}$ sends a random $h_{\mathbf{y},i'} \in \{0,1\}^{\ell_0}$ as the commitment to $\mathbf{y}_{i'}$ and then programs H_0 such that $\mathsf{H}_0(i', \mathbf{y}_{i'}) = h_{\mathbf{y},i'}$. If programming fails, $\mathcal{B}$ aborts. Then the only difference in $\mathcal{A}$'s view is how $h_{\mathbf{y},i'}$ is computed. By the oracle assumption, the distributions are indistinguishable between these two games hence from the view of the adversary $\mathcal{A}$, $\mathbf{G}_6$ and $\mathbf{G}_7$ are indistinguishable as long as $\mathcal{B}$ does not abort during programming. Thus we can bound the advantage of $\mathcal{A}$ in distinguishing $\mathbf{G}_6$ and $\mathbf{G}_7$ by the failure probability of programming of H_0:

$$|\Pr[\mathbf{G}_7] - \Pr[\mathbf{G}_6]| \leq \frac{|\mathcal{H}|(Q_H + Q_S)}{2^{\ell_0}}.$$

G_8: Now, $\mathcal{B}$ removes its reliance on the secret key $\mathbf{s}$ for signing. We use a similar approach to [HMM+26] and use an external oracle for a variant of Raccoon.

During key generation, $\mathcal{B}$ initializes the signing challenger $\mathcal{D}$ for Raccoon* with the parameters of the combined signature, queries the oracle to obtain public keys, and then sets $\mathsf{crs} = \mathbf{A}_\mathcal{S}$. In particular, $\mathcal{D}$ holds the same key as the threshold signature scheme, so that the public key $\mathbf{y}'$ received from $\mathcal{D}$ is the aggregate one. Instead of honestly computing $\mathbf{y}_{i'}$, $\mathcal{B}$ therefore derives $\mathbf{y}_{i'} = \mathbf{y}' - \sum_{j\in[n]\setminus\{i'\}} \mathbf{y}_j$ and uniformly samples $\kappa_\mathbf{y}$ lower bits of $\mathbf{y}_{i'}$. The rest of key generation continues as in $\mathbf{G}_7$.

For $i \in \mathcal{H}_\mathcal{U} \setminus \{i'\}$, signing starts as in $\mathbf{G}_7$, so that $\mathcal{B}$ knows $\mathbf{r}_i$ and $\mathbf{w}_i$ for every $i \in \mathcal{H}_\mathcal{U} \setminus \{i'\}$, while for i' it broadcasts a uniformly random $h_{\mathbf{w},i'}$ as in $\mathbf{G}_5$. Upon receiving the commitments $\{h_{\mathbf{w},j}\}_{j\in\mathcal{C}_\mathcal{U}}$ from the adversary, $\mathcal{B}$ recovers the corresponding $\mathbf{w}_j$ from the entries of the random oracle table $\mathcal{HT}_1$, aborting if

no matching entry exists, and computes the aggregate contribution of all parties other than i':

$$\mathbf{w}_{-i'} := \sum_{j \in \mathcal{U} \setminus \{i'\}} \mathbf{w}_j.$$

When the signing query for μ is made, $\mathcal{B}$ forwards μ *together with* $\mathbf{w}_{-i'}$ to $\mathcal{D}$ and obtains $(c', \mathbf{z}')$. Internally, $\mathcal{D}$ samples a commitment which $\mathcal{B}$ takes to be the contribution of the honest party i' in the second round of the protocol, i.e., $\mathbf{w}_{i'}$, and returns the challenge $c' = \mathsf{H}_2(\mathsf{pk}_{\mathcal{S}}, \lfloor \mathbf{w}_{i'} + \mathbf{w}_{-i'} \rceil_{q_{\mathbf{w}}}, \mu)$ computed with respect to the aggregate commitment $\mathbf{w} := \mathbf{w}_{i'} + \mathbf{w}_{-i'}$ of the protocol, together with the corresponding signature share $\mathbf{z}'$ of party i'. Accordingly, $\mathcal{B}$ programs H_1 so that $h_{\mathbf{w},i'} = \mathsf{H}_1(i', \mathbf{w}_{i'})$, aborting if the programming fails, uniformly samples the $\kappa_{\mathbf{w}}$ lower bits of $\mathbf{w}_{i'}$, and takes $c := c'$ as the challenge of the final signature.

It remains for $\mathcal{B}$ to complete the signature and to simulate the remaining rounds of the protocol, which it can do entirely on its own. Recall that the secret signing key shares $\mathbf{s}_j$ of the corrupted parties are already known from the distributed key generation, having been extracted in $\mathbf{G}_2$, and that after the second round the signature randomness $\mathbf{r}_j$ of the corrupted parties is likewise known, having been extracted in $\mathbf{G}_4$; the corresponding values of the remaining honest parties are chosen by $\mathcal{B}$ itself. Since the share $\mathbf{z}'$ returned by $\mathcal{D}$ accounts for the contribution of party i', $\mathcal{B}$ obtains the signature of the protocol as

$$\mathbf{z} := \mathbf{z}' + c \cdot \sum_{j \in \mathcal{U} \setminus \{i'\}} \mathbf{s}_j + \sum_{j \in \mathcal{U} \setminus \{i'\}} \mathbf{r}_j,$$

and then computes the hint as $\mathbf{h} := \lfloor \mathbf{w} \rceil_{q_{\mathbf{w}}} - \lfloor \mathbf{A}_{\mathcal{S}} \mathbf{z} - 2^{\kappa_{\mathbf{y}}} c \mathbf{y}' \rceil_{q_{\mathbf{w}}}$, so that the final signature is $\Psi := (c, \mathbf{z}, \mathbf{h})$. Knowing $\mathbf{z}$, $\mathcal{B}$ can simulate the decryption share $\mathsf{ds}_{i'}$ of the honest party in the last round so that combining it with the shares of the remaining parties yields $\mathbf{z}$. The rest of the signing proceeds as in $\mathbf{G}_7$.

Since $\mathbf{A}_{\mathcal{S}}, \mathbf{y}'$ received from $\mathcal{D}$ is a valid MLWE instance and the signing oracle is simulated using the noise leakage from the $\mathsf{H\text{-}MLWE}$ hint $\mathbf{h}$, the distribution of the high bits of $\mathbf{y}_{i'}$ in $\mathbf{G}_8$ is the same as in $\mathbf{G}_7$: by linearity of operations, the derived $\mathbf{y}_{i'} = \mathbf{A}_{\mathcal{S}} \mathbf{s}'_{i'} + \mathbf{e}'_{i'}$ for unknown $\mathbf{s}'_{i'}$ and $\mathbf{e}'_{i'}$, and the distribution is indistinguishable under the $\mathsf{H\text{-}MLWE}$ assumption.

Passing $\mathbf{w}_{-i'}$ to $\mathcal{D}$ is what makes the simulation consistent with the first two rounds of signing: the commitment $\mathbf{w}_{i'}$ chosen by $\mathcal{D}$ is precisely the value opened by the honest party in the second round, and c' is computed with respect to the same aggregate $\mathbf{w} = \mathbf{w}_{i'} + \mathbf{w}_{-i'}$ that the parties derive in the protocol. The hash commitment of the first round, the opening of the second, and the share $\mathbf{z}'$ used to complete the signature therefore agree by construction. This additional input, together with the fact that $\mathcal{B}$ adjusts the signature itself so that no randomness or error terms occur in the oracle computation, is the only way in which our oracle deviates from the one in [HMM+26]; as $\mathbf{w}_{-i'}$ is used only to derive the challenge, the reduction from Raccoon* to plain Raccoon goes through as in [HMM+26]. While bounded, the values chosen by $\mathcal{A}$ need not follow exact Gaussian distributions, but since $\mathcal{B}$ performs the adjustment using

the extracted values, the signature distributions in $\mathbf{G}_7$ and $\mathbf{G}_8$ coincide. The remaining difference in $\mathcal{A}$'s view is $\mathbf{w}_{i'}$, which is indistinguishable from uniform by the way it is computed, so an adversary distinguishing $\mathbf{G}_7$ from $\mathbf{G}_8$ yields a distinguisher for H-MLWE. We then have:

$$|\Pr[\mathbf{G}_8] - \Pr[\mathbf{G}_7]| \leq \epsilon_{\mathsf{H\text{-}MLWE}}.$$

The signatures in $\mathbf{G}_8$ are independent of secret key material defined as part of the protocol and solely depend on the signature received from the challenger $\mathcal{D}$. A forgery against the $\mathcal{S}$ scheme is then a forgery against Raccoon*, and consequently against the original Raccoon signature scheme. If $\mathcal{A}$ outputs a forgery $(c^*, \mathbf{z}^*, \mathbf{h}^*), \mu^*$, $\mathcal{B}$ 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 $\mathbf{G}_8$, it can be used to break the uf-cma security of the underlying scheme, the advantage of $\mathcal{A}$ can then be bounded as:

$$|\Pr[\mathbf{G}_8]| \leq \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{B}).$$

This concludes the proof.

C.6 Proof of Theorem 8

Proof. Since we are in the synchronous model with authenticated broadcast, all of our arguments made in Theorem 4 are still valid for the first two conditions.

If $\mathcal{A}$ does not cause abort, then protocol proceeds as intended and by the correctness of the scheme there exists no message μ_j with signature Ψ_j^* generated through protocol where verification fails. Thus, $\mathcal{A}$ cannot win through the fourth condition either.

Once again, it remains to show that $\mathcal{A}$ cannot win through the third condition which we now show cannot be the case for any of the checks in protocol.

During $\mathsf{KGen}_{\mathcal{S}}$, $\mathcal{A}$ cannot cheat during $\mathsf{KGen}_{\mathcal{E}}$ without detection as long as underlying encryption scheme has security against identifiable aborts, which is by assumption true. Then, $\mathcal{A}$ can only succeed if for any $j \in \mathcal{C}$ $h_{\mathbf{y},j}$ or $\pi_{\mathbf{s},j}$ is invalid and is not detected by $\mathcal{H}$. This can only happen if (i) j found a $\mathbf{y}_j^*$ such that $\mathbf{y}_j^* \neq \mathbf{y}_j$ with $\mathsf{H}_0(j, \mathbf{y}_j^*) = \mathsf{H}_0(j, \mathbf{y}_j)$ so that $h_{\mathbf{y},j}$ is not the correct hash commitment to $\mathbf{y}_j$ or (ii) j sent $\pi_{\mathbf{s},j}$ without knowledge of satisfying $\mathbf{s}_j$. If (i) is the case, then j found a collision for H_0, which does not happen for all but negligible probability. If (ii) is the case however, then j can be used as a cheating prover for the underlying NIZKAoK $\Pi_{\mathbf{s}}$, which once again cannot happen for all but negligible probability by the knowledge soundness assumption.

For $\mathsf{Sign}_{\mathcal{S}}$, if j sends an incorrect message without any of the honest users aborting (i) j found a $\mathbf{w}_j^*$ such that $\mathbf{w}_j^* \neq \mathbf{w}_j$ with $\mathsf{H}_1(j, \mathbf{w}_j^*) = \mathsf{H}_1(j, \mathbf{w}_j)$ so that $h_{\mathbf{w},j}$ is not the correct hash commitment to $\mathbf{w}_j$ or (ii) j sent $\pi_{\mathbf{r},j}$ without knowledge of satisfying $\mathbf{r}_j$, or (iii) j sends a decryption share ds_j such that combined signature $\mathbf{z} = \perp$. If (i) is the case, then j found a collision for H_1, which does not happen for all but negligible probability. If (ii) is the case however, then j can be used as a cheating prover for the underlying NIZKAoK $\Pi_{\mathbf{r}}$, which

once again cannot happen for all but negligible probability by the knowledge soundness assumption. And finally, if (iii) is the case, j found an incorrect share that does not combine correctly. By the correctness of $\mathcal{E}$, this can only happen if the sent ds_j is incorrectly computed which then would violate the knowledge soundness.

Since $\mathcal{A}$ cannot make any $j \in \mathcal{C}$ cheat without detection with non-negligible probability, it cannot succeed in the identifiable aborts game. This concludes the proof.

D Zero-Knowledge Proofs

VSS Proof. In the VSS we wish to prove that the Hadamard product of the n-tuple of committed shares with a (pseudo) randomly generated element $c^{\perp}$ of the dual code is zero.

$$\mathcal{R}_{\mathsf{VSS}} := \left\{ (\mathsf{X},\mathsf{W}) \;\middle|\; \begin{array}{c} \mathsf{X} := \left(\mathsf{pk}_{\mathsf{Com}}, \left\{\mathsf{com}_{[\![\mathbf{S}]\!]_i}\right\}_{i\in[n]}, \mathbf{C}^{\perp}\right),\ \mathsf{W} := \left(\left\{[\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i}\right\}_{i\in[n]}\right) : \\ \text{for all } i \in [n]:\ \mathsf{BDLOP.Open}\left(\mathsf{pk}_{\mathsf{Com}}, \mathsf{com}_{[\![\mathbf{S}]\!]_i}, \mathsf{d}_i := ([\![\mathbf{S}]\!]_i, r_{[\![\mathbf{S}]\!]_i}, 1) = 1\right) \\ \wedge \sum_{i=1}^{n} \Phi(\mathbf{S}_i) \circ \Phi(\mathbf{C}^{\perp}) = 0^{N\cdot(m_1\times m_2)} \end{array} \right\} \tag{6}$$

Now that we have a way to verify sharing of an element, we define the relations to be proven for the $\mathcal{E}$ and $\mathcal{S}$ schemes. These relations are similar to ones described in Section 5 but more explicit with the help of detailed relations above. First we start with the proof for language $\mathcal{R}_{\mathsf{KGen}_{\mathcal{E}}}$ which shows key shares are generated correctly.

$$\mathcal{R}_{\mathsf{KGen}_{\mathcal{E}}} := \left\{ (\mathsf{X},\mathsf{W}) \;\middle|\; \begin{array}{c} \mathsf{X} := (\mathbf{B}_i, \{\mathbf{B}_{i,j}\}, \{\mathsf{com}_{i,j}\}_{i\in[n]}), \\ \mathsf{W} := (\mathbf{S}_i, \boldsymbol{\rho}_i, \mathbf{E}_i, \rho'_i, \{\mathbf{S}_{i,j}\}_{j\in[n]}, \{\boldsymbol{\rho}_{i,j}\}_{j\in[n]}, \{\mathbf{E}_{i,j}\}_{j\in[n]}) : \\ \|\mathbf{S}_i\|_{\infty}, \|\mathbf{E}_i\|_{\infty} \le t_{\infty}\cdot\sigma_{\mathsf{LWE}} \ \wedge\ \mathbf{B}_i = \mathbf{A}_{\mathcal{E}}\mathbf{S}_i + p\mathbf{E}_i \\ \wedge\ \forall j\ \mathbf{B}_{i,j} = \mathbf{A}_{\mathcal{E}}\mathbf{S}_{i,j} + p\mathbf{E}_{i,j} \ \wedge\ \mathbf{B}_j = \mathsf{WSS.Rec}_{t,n}(\{\mathbf{B}_{i,j}\}) \\ \wedge\ \mathbf{S}_i = \mathsf{WSS.Rec}_{t,n}(\{\mathbf{S}_{i,j}\}) \ \wedge\ \mathbf{E}_i = \mathsf{WSS.Rec}_{t,n}(\{\mathbf{E}_{i,j}\}) \\ \forall i\ \mathsf{BDLOP.Open}(\mathsf{com}_{i,j}, \mathbf{S}_{i,j}, \boldsymbol{\rho}_{i,j}) = 1 \end{array} \right\}. \tag{7}$$

Next, we show the relation for correct partial decryption.

$$\mathcal{R}_{\mathsf{ds}} := \left\{ (\mathsf{X},\mathsf{W}) \;\middle|\; \begin{array}{c} \mathsf{X} := (\mathsf{ds}_i, \mathsf{ctx}, \mathsf{com}_{\mathsf{sk},i}, \mathsf{com}_{E,i}),\ \mathsf{W} := (\mathsf{sk}_i, \boldsymbol{\rho}_{\mathsf{sk},i}, E_i, \boldsymbol{\rho}_{E,i}) : \\ \|E_i\|_{\infty} \le t_{\infty}\cdot\sigma_{\mathsf{TDec}} \ \wedge\ \mathsf{ds}_i = \lambda_i \mathsf{sk}_i \mathbf{u} + pE_i \ \wedge \\ \mathsf{BDLOP.Open}(\mathsf{com}_{\mathsf{sk},i}, \mathsf{sk}_i, \boldsymbol{\rho}_{\mathsf{sk},i}) = 1 \\ \wedge\ \mathsf{BDLOP.Open}(\mathsf{com}_{E,i}, E_i, \boldsymbol{\rho}_{E,i}) = 1 \end{array} \right\}. \tag{8}$$

The commitments are implicit in the proof system used similar to relations for our WSS. Finally we define the two relations in $\mathcal{S}$ itself which contain two similar proofs of encryption showing knowledge of a relevant plaintext. Since the commitment to the secret key shares are also included as part of the distributed key generation, it is trivial to prove the above statements using the proof of linear relations as before, once the commitments to E_i are available. For which we instantiate the proof $\pi_{\mathsf{ds},i}$ as a proof of linear relation to show d is correctly

computed alongside a commitment to E_i.

$$\mathcal{R}_{\mathbf{s}} := \left\{ (\mathsf{X}, \mathsf{W}) \middle| \begin{array}{c} \mathsf{X} := (\mathbf{A}_{\mathcal{S}}, \mathbf{y}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s},i}),\ \mathsf{W} := (\mathbf{s}_i, \mathbf{e}_i)\ : \\ \|\mathbf{s}_i\|_\infty, \|\mathbf{e}_i\|_\infty \leq t_\infty \cdot \sigma_{\mathbf{s}}\ \wedge\ \mathbf{y}_i = \mathbf{A}_{\mathcal{S}}\mathbf{s}_i + \mathbf{e}_i \\ \wedge\ \mathsf{ctx}_{\mathbf{s},i} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i) \end{array} \right\}. \quad (9)$$

$$\mathcal{R}_{\mathbf{r}} := \left\{ (\mathsf{X}, \mathsf{W}) \middle| \begin{array}{c} \mathsf{X} := (\mathbf{A}_{\mathcal{S}}, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{r},i}),\ \mathsf{W} := (\mathbf{w}_i, \mathbf{r}_i, \mathbf{e}'_i)\ : \\ \|\mathbf{r}_i\|_\infty, \|\mathbf{e}'_i\|_\infty \leq \sigma_{\mathbf{r}}\ \wedge\ \mathsf{ctx}_{\mathbf{r},i} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{r}_i) \\ \wedge\ \mathbf{w}_i = \mathbf{A}_{\mathcal{S}}\mathbf{r}_i + \mathbf{e}'_i\ \wedge\ \mathsf{com}_i = \mathsf{BDLOP.Com}(\mathbf{w}_i, \boldsymbol{\rho}_i) \end{array} \right\}. \quad (10)$$

Both of these languages need a proof of plaintext knowledge which we define below:

$$\mathcal{R}_{\mathsf{Enc}} := \left\{ (\mathsf{X}, \mathsf{W}) \middle| \begin{array}{c} \mathsf{X} := (\mathsf{ctx} = (\mathbf{u}, \mathbf{v}), \mathsf{com}_{\mathbf{r}}, \mathsf{com}_{\mathbf{e}_1}, \mathsf{com}_{\mathbf{e}_2}, \mathsf{com}_{\mathbf{m}}), \\ \mathsf{W} := (\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{m}, \boldsymbol{\rho}_{\mathbf{r}}, \boldsymbol{\rho}_{\mathbf{e}_1}, \boldsymbol{\rho}_{\mathbf{e}_2}, \boldsymbol{\rho}_{\mathbf{m}}) : \\ \|\mathbf{r}\|_\infty, \|\mathbf{e}_1\|_\infty, \|\mathbf{e}_2\|_\infty \leq B_{\mathsf{Enc}}\ \wedge \mathbf{u} = \mathbf{A}_{\mathcal{E}}^\top \mathbf{r} + p\mathbf{e}_1 \\ \wedge\ \mathbf{v} = \mathbf{B}_{\mathcal{E}}^\top \mathbf{r} + p\mathbf{e}_2 + \mathbf{m}\ \wedge\ \|\mathbf{m}\|_\infty \leq p \\ \wedge\ \mathsf{BDLOP.Open}(\mathsf{com}_{\mathbf{r}}, \mathbf{r}, \boldsymbol{\rho}_{\mathbf{r}}) = 1 \\ \wedge\ \mathsf{BDLOP.Open}(\mathsf{com}_{\mathbf{e}_1}, \mathbf{e}_1, \boldsymbol{\rho}_{\mathbf{e}_1}) = 1 \\ \wedge\ \mathsf{BDLOP.Open}(\mathsf{com}_{\mathbf{e}_2}, \mathbf{e}_2, \boldsymbol{\rho}_{\mathbf{e}_2}) = 1 \\ \wedge\ \mathsf{BDLOP.Open}(\mathsf{com}_{\mathbf{m}}, \mathbf{m}, \boldsymbol{\rho}_{\mathbf{m}}) = 1 \end{array} \right\}. \quad (11)$$

Commitments are once again implicit to the proof system.