Efficient Threshold ML-DSA

Full Version

Sofía Celi [1,2], Rafael del Pino [3], Thomas Espitau [3], Guilhem Niot [3,4], Thomas Prest [3]

[1] Brave Research [2] University of Bristol [3] PQShield [4] Univ Rennes, CNRS, IRISA

This is the full version of a paper to appear in the proceedings of the 35th USENIX Security Symposium (USENIX Security 2026). It supersedes the preprint [9] containing an earlier version of this work, and includes additional appendices with extended technical details and benchmarks.

Abstract

Threshold signature schemes allow a group of users to jointly generate a digital signature, providing resilience against faults and enhancing decentralization. With the advent of post-quantum cryptography, lattice-based threshold signatures have gained attention as viable PQ-threshold solutions. Nevertheless, existing constructions are limited in terms of their scalability, robustness. Worse, none is compatible with standardized schemes, particularly with the NIST-selected and standardized Module-Lattice-based Digital Signature Algorithm (ML-DSA) algorithm.

In this work, we present the first threshold signature scheme that is *fully compatible* with ML-DSA, supporting secure and efficient signing for a small number of parties, with an average communication per party upper bounded by 1 MB up to 6 parties. Our construction leverages advanced short secret sharing techniques and integrates optimized rejection sampling to achieve a favorable balance between communication efficiency and correctness in distributed environments. We implement our construction in Go and evaluate its performance across local, LAN, and WAN network settings. Our benchmarks demonstrate that our threshold ML-DSA scheme is not only practically deployable but also well-suited for real-world applications, including multi-device cryptocurrency wallets, threshold-based TLS authentication, and for Tor's directory authorities.

1 Introduction

As organizations migrate to post-quantum cryptography, they are adopting the Module-Lattice-based Digital Signature Algorithm (ML-DSA), recently finalized by NIST [55]. However, this transition creates a critical security gap: modern systems rely on threshold signature schemes (TSS) to eliminate single points of failure and distribute trust, but no practical threshold construction for ML-DSA exists. This forces a dangerous trade-off between achieving quantum resistance and maintaining the operational resilience provided by mature classical threshold schemes for RSA, Schnorr, and ECDSA. Systems that migrate to ML-DSA today must forfeit the threshold guarantees they depend on.

The absence of a threshold ML-DSA scheme is not an oversight but a deep technical challenge. The core of ML-DSA's design—the *Fiat-Shamir-with-aborts* paradigm—is fundamentally at odds with multi-party computation. This paradigm relies on a rejection sampling procedure where a signature attempt is "aborted" and retried. Rejected candidates must not be leaked and therefore, in a threshold setting, this requires multiple parties to collaboratively decide whether to abort without revealing the candidate to each other, nor to an adversary. Naïve protocols for this either statistically leak or incur catastrophic performance overhead. Even the two-party case has remained an open problem, as the abort decision depends on the final signature, which cannot be revealed until all parties agree the signature is valid.

1.1 Our Contribution: The First Practical Threshold ML-DSA

We introduce the first practical threshold signature scheme for the ML-DSA standard. Our protocol allows an existing ML-DSA key to be securely distributed among multiple parties (a posteriori sharing). It also includes a fully distributed key generation (DKG) protocol that eliminates any trusted dealer. We propose parameters for signing up to 6 parties, which are at most 1 MB of communication per party on average[1] for ML-DSA-44. While a threshold of six may appear small at first glance, it represents a significant step forward and is already more than sufficient for most industrial applications (which requires usually 2-out-of-2 and 2-out-of-3 settings). A few

[1] "Average communication" takes into account the probabilistic nature of ML-DSA, as several attempts may be needed to output a valid signature.

more parties can be supported with higher communication costs as well, potentially of use in some niche applications.

Our design overcomes the fundamental conflict between ML-DSA's abort-and-retry procedure and multi-party computation. We achieve this by tailoring the core ideas of short secret sharing and per-party rejection from del Pino and Niot [26] to ML-DSA's unique constraints.

Technical Innovations. Our key contribution is a distributed protocol described in Section 3.1. It is a novel combination of (i) short replicated secret sharing, and (ii) optimized per-party rejection sampling, specifically designed for ML-DSA's unique constraints. Our protocol includes a two-stage rejection sampling: a per party rejection, to ensure the good shape of all partial signatures, and that their reveal does not leak any secret; and a global one, at the time of combination, which ensures that the final combined signature fits in the sizes of ML-DSA signatures. Note that while the first one acts as in ML-DSA as is required for security, the second one is only there for size compatibility. Even though this idea seems simple, it requires several new pieces of technology to make it work altogether:

- **Unbalanced Hyperball Rejection Sampling.** We replace the rejection sampling technique from ML-DSA with one based on hyperballs. This significantly reduces abort probability when multiple parties must simultaneously satisfy the rejection criteria. Further, we fine-tune this new rejection sampling by using simultaneously different norms.
- **Optimized Share Reconstruction (Section B).** The probability of signing success is directly tied to the size of the secret shares. We designed a tailored reconstruction procedure that minimizes this size by modeling the problem as a variant of the B-matching problem, which we solve efficiently using a maximum flow algorithm.
- **Parallel Protocol Instances.** To ensure high signing success rates at reconstruction time, our protocol runs multiple instances in parallel. This approach guarantees fast and reliable signing without compromising security.

For key generation, we cover two complementary scenarios:

- **A Posteriori Key Sharing (Section 3.3).** For settings where the signing key is fixed a priori (e.g., for backward compatibility), we show how it can be securely shared among multiple users, while crucially preserving the original verification key.
- **Distributed Key Generation.** We design a fully decentralized key generation protocol that removes the need for a trusted dealer.

Formal Security (Section 3.2) We provide a full formal analysis of our scheme, showing that it preserves the security and correctness guarantees of ML-DSA under the same standard MLWE assumption.

Practical Deployment (Section 4) We implement our scheme in Go and benchmark its performance across local, LAN, and WAN settings. Our experiments show that a signing attempt completes in milliseconds in local deployments, and under one second in wide-area settings across continents. Average communication costs to produce a signature range from 21 kB to 1050 kB per party depending on the threshold (T, N) and up to 6 parties, which is competitive with established classical TSS. Crucially, the output signatures are fully compliant with the standard ML-DSA format, enabling drop-in integration with existing systems.

1.2 Related Work

Recent years have witnessed a surge of interest in constructing TSS that are secure against quantum adversaries. While classical threshold signatures have been deployed and extensively studied, extending them to the post-quantum setting introduces significant challenges, especially for schemes compatible with NIST-designated standards and candidates.

Threshold Signature Schemes. Classical TSS constructions include schemes based on RSA [38], Schnorr [50, 18], and ECDSA [12, 13], many of which offer advanced properties such as identifiable aborts. However, the design of post-quantum TSS has proven more difficult, with many proposals relying on assumptions outside of the NIST call for standardization. Proposed post-quantum schemes span a variety of hardness assumptions: lattice-based, isogeny-based [20], and hash-based [47]. A recent multivariate-based TSS by Celi et al. [14] achieves compatibility with the MAYO and UOV signature schemes using generic MPC techniques, potentially offering practical efficiency for specific parameter choices.

Lattice-based Fiat-Shamir threshold schemes. Given the perceived impracticality of directly thresholdizing ML-DSA due to its rejection sampling mechanism, early efforts in the lattice setting focused on Fiat-Shamir-style signatures without rejection sampling, targeting variants of the Raccoon signature scheme [21]. These include threshold schemes based on threshold FHE by Agrawal et al. [2] and Gur et al. [40], and standard tools by del Pino et al. [23] Espitau et al. [31], Katsumata et al. [46] and Chairattanai-Apirom et al. [15]. While these schemes typically yield larger signatures, they offer attractive features such as adaptive security [46], robustness [32], identifiable aborts [22, 24], and Beyond Unforgeability Features (BUFF) [37]. We provide an efficiency and size comparison against two representative schemes in

Table 1: Comparison of approaches to thresholdize ML-DSA-44, with per party metrics averaged over successful signing attempts.

Scheme	# Parties (upper bound)	# Rounds Off + Online	Communication Off + Online (MB)	Computation	Model	Security
This work	6	2+4	10^{-5} + 0.021 to 1.05	Lightweight	Game-based	Dishonest majority
Bienstock et al. [7]	Unlimited	$136^{*}+79$ $37^{*}+23$	$>0.87^{*}+0.70$ $>1.9^{*}+1.5$	Online lightweight*	UC	Honest majority
Trilithium [29]	2	$1^{\dagger}$+60	$234^{\dagger}+0.40$	Heavy	UC	Trusted party†
Generic MPC [17]	Unlimited	High	High	Impractical	UC	Dishonest majority

Rounds and communication costs are reported *as per-party averages*, measured over successful signing attempts for Threshold ML-DSA-44. These reflect the impact of ML-DSA's inherent probabilistic abort mechanism. For [7], we include the costs for 5 parties, with 1 and 8 parallel protocol executions. *Offline costs for [7] do not include the generation of correlated randomness (e.g., Beaver triples, edaBits, etc.). †Requires 234 MB of correlated randomness to be generated by a trusted party.

Section 4.2. These design directions avoid the difficulties associated with ML-DSA's rejection mechanism but sacrifice compatibility with a standardized scheme.

Direct Thresholdization of ML-DSA. Despite its complexity, several works have pursued direct thresholdization of ML-DSA, primarily through generic MPC techniques that address its rejection sampling behavior (Table 1).

The generic MPC-based approach of Cozzo and Smart [17] is secure in theory but suffers from high computational and communication costs, making practical deployment infeasible. The "Trilithium" scheme by Dufka et al. [29] supports two-party signing, but it assumes a *trusted party* to generate correlated randomness: an assumption that falls outside standard adversarial models and still incurs high communication overhead. Moreover, its security analysis lacks a full proof of unforgeability, relying instead on heuristic assumptions about the distribution of rejected ML-DSA transcripts.

Most recently, and concurrently with this work, Bienstock et al. [7] proposed a threshold ML-DSA variant using MPC under an honest-majority assumption. Their construction achieves either 79 rounds with 705 kB of *online* communication per party or 23 rounds with 1.5 MB, depending on parameter choices (c.f., [7, Table 5]). To which must be added the cost of the offline phase (between 136 and 37 rounds on average), which in particular includes correlated randomness generation, which is known to be expensive computational and/or communication-wise. We provide a more thorough comparison in Table 9.

Our scheme departs from generic MPC paradigms by tailoring the design to ML-DSA design. It supports a small number of parties in the strongest adversarial model (dishonest majority), retains compatibility, and achieves practical efficiency without relying on trusted setup or heuristic security assumptions.

Secret Sharing Techniques Recent works by Chairattana-Apirom et al. [15] and del Pino et al. [22] highlighted the benefits of short secret sharing techniques – consisting of a secret sharing with short partial shares – which enable more compact and efficient protocols [11, 15], and support advanced features such as identifiable aborts at no additional cost [22]. Additionally, these techniques simplify integration with rejection sampling [26], which is crucial for this work.

2 Preliminaries

2.1 General notations

Algorithmic notations. To denote the *assign* operation, we use $y \coloneqq f(x)$ (resp. $y \leftarrow f(x)$) when f is deterministic (resp. randomized). We note **req** (resp. **assert**) the function that takes a proposition x as input and outputs 0 (resp. $\perp$) if x is false, else it does nothing.

Set and distributions. For an integer $N > 0$, we denote $[N] = \{0, \ldots, N-1\}$. We recall that the union $S \cup T$ of two sets writes $S \sqcup T$ when disjoint. For a set S, we denote by $\mathcal{U}(S)$ the *uniform distribution* over S, and write $x \xleftarrow{\$} S$ as shorthand for $x \leftarrow \mathcal{U}(S)$. For a real parameter $0 \leqslant p \leqslant 1$, the *Bernoulli distribution* $\mathsf{Ber}(p)$ outputs 1 with probability p and 0 otherwise. The *Binomial distribution* $\mathsf{Bin}(m, p)$ is defined as the sum of m independent Bernoulli trials sampled from $\mathsf{Ber}(p)$.

Linear algebra. For a fixed power-of-two n, we denote $\mathcal{R} = \mathbb{Z}[x]/(x^n+1)$ the cyclotomic ring and $\mathcal{R}_q = \mathcal{R}/(q\mathcal{R})$. We also let $\mathcal{R}_{\mathbb{R}} = \mathbb{R}[x]/(x^n+1)$ be a polynomial ring over real numbers. Given $\mathbf{x} \in \mathcal{R}_{\mathbb{R}}^{\ell}$, we denote $\|\mathbf{x}\|_2$ the Euclidean norm (resp. $\|\mathbf{x}\|_\infty$ the infinite norm) of the $(n\ell)$-dimensional vector of the coefficients of $\mathbf{x}$. Unless specified otherwise, vectors are *column* vectors.

2.2 Threshold signatures

In all this work, we denote the security parameter by κ. Following the syntax of Katsumata et al. [46], a R-round threshold signature (*without* session identifiers) is as a tuple $\mathsf{TSS} = (\mathsf{Setup}, \mathsf{Keygen}, (\mathsf{Sign}_i)_{i\in[R]}, \mathsf{Combine}, \mathsf{Verify})$. $\mathsf{act} \subseteq [N]$ denotes a set of T parties used for signing. Each signer i maintains a state st_i: short-lived session-specific information.

$\mathsf{Setup}(\kappa, N, T) \to \mathsf{pp}$**.** Takes as input a security parameter κ, the total number of parties N, the reconstruction threshold $T \leqslant N$. Outputs a set of public parameters pp.

$\mathsf{Keygen}(\mathsf{pp}) \to (\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]})$**.** Takes as input the public parameters. Outputs a verification key vk, and partial private keys $(\mathsf{sk}_i)_{i\in[N]}$.

$\mathsf{Sign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_{r-1}^j)_{j\in\mathsf{act}}, i, \mathsf{sk}_i, \mathsf{st}_i) \to (\mathsf{pm}_r^i, \mathsf{st}_i)$**.** For the r-th round ($r \in \{1, \cdots, R\}$), takes as input the verification key vk, the set of signers act, a message msg, the protocol messages exchanged during the previous round $(\mathsf{pm}_{r-1}^j)_{j\in\mathsf{act}}$, as well as the partial private key sk_i and state st_i of user i. Outputs a new message pm_r^i and updated state st_i. We define $\mathsf{pm}_0^i = \perp$.

Note that msg and act may optionally be provided only from a round later than the first.

$\mathsf{Combine}(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r^i)_{r\in[R], i\in\mathsf{act}}) \to \mathsf{sig}$**.** Takes as input the verification key vk, the set of signers act, the message msg, all messages exchanged during the protocol $(\mathsf{pm}_r^i)_{r\in[R], i\in\mathsf{act}}$. Outputs a signature sig.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \to 0$ **or** 1**.** Takes as input a verification key vk, a message msg, and a signature sig. Outputs 1 if sig is valid and 0 otherwise.

A key property of threshold signatures is unforgeability, which ensures that an adversary cannot produce valid signatures without the cooperation of honest parties. Several security models exist, differing in whether the adversary chooses corrupted users statically [23] or adaptively [46], and whether the communication channel is controlled by the adversary [23], or requires consensus properties [32, 22], and for which messages a forgery is accepted.

In this work, we consider the unforgeability notion from del Pino et al. [23] adapted to remove the use of session identifiers. It ensures static security – i.e. corrupted users are chosen before executing the protocol, a communication channel fully controlled by the adversary, and forgeries are accepted for messages for which no signing oracle was called. It can also be seen as a multi-round adaptation of the TS-UF-0 notion from [6]. We provide formal definitions of these properties below.

Note that, for small thresholds (T, N), static security already implies adaptive security via the *complexity leveraging* technique, with at most a 5-bit theoretical loss when $N \leqslant 6$, as is the case for our Threshold ML-DSA construction. This loss appears to stem from limitations of existing proof techniques rather than from any inherent weakness of the scheme itself.

Definition 2.1 (Unforgeability). *For a R-round* TSS*, the advantage of an adversary $\mathcal{A}$ (with oracle access to a random oracle* H*) against the unforgeability of* TSS *is defined as:*

$$\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathsf{TSS},\mathcal{A}}(\kappa, N, T) := \Pr\left[\mathsf{Game}^{\mathsf{TS\text{-}UF}}_{\mathsf{TSS},\mathcal{A}}(\kappa, N, T) = 1\right]$$

where $\mathsf{Game}^{\mathsf{TS\text{-}UF}}$ *is described in Figure 1. We say that* TSS *is unforgeable in the random oracle model, if for all PPT adversary $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathsf{TSS},\mathcal{A}}(\kappa, N, T) \leqslant \mathsf{negl}(\kappa)$.

$\mathsf{Game}^{\mathsf{TS\text{-}UF}}_{\mathsf{TSS},\mathcal{A}}(\kappa, N, T)$

1: $Q_{\mathsf{msg}} := \emptyset$
2: $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, N, T)$
3: $\mathsf{corrupt}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}(\mathsf{pp})$
4: **assert**$\{\ \mathsf{corrupt} \subseteq [N]$ **and** $|\mathsf{corrupt}| < T\ \}$
5: $\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
6: $(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Sign.Keygen}(\mathsf{pp})$
7: **for** $\{\ i \in \mathsf{honest}\ \}$ **do** $\{\ \mathsf{st}_i := \varnothing\ \}$
8: $(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OSign}_i(\cdot))_{i\in[R]}}(\mathsf{st}_{\mathcal{A}}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
9: **req** $\{\ \mathsf{msg} \notin Q_{\mathsf{msg}}\ \}$
10: **return** $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

$\mathsf{OSign}_r(\mathsf{act}, \mathsf{msg}, i, (\mathsf{pm}_{r-1}^j)_{j\in\mathsf{act}})$

1: **req** $\{\ i \in [N] \wedge i \in \mathsf{act} \cap \mathsf{honest}\ \}$
2: $Q_{\mathsf{msg}} := Q_{\mathsf{msg}} \cup \{\mathsf{msg}\}$
3: $(\mathsf{pm}_r^i, \mathsf{st}_i) \leftarrow \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg},$
4: $\quad (\mathsf{pm}_{r-1}^j)_{j\in\mathsf{act}}, i, \mathsf{sk}_i, \mathsf{st}_i)$
5: **return** pm_r^i

Figure 1: Unforgeability game for TSS, where H denotes the random oracle. We consider that the OSign_r oracles return $\perp$ if the corresponding $\mathsf{ShareSign}_r$ fails to output a protocol message. Finally, $\mathcal{A}$ wins if the game TS-UF returns 1, i.e. if it forged a new signature.

We also wish to ensure *correctness*, i.e. that a valid signature is output when signers behave honestly. We however tolerate a probability of abort to support the *Fiat-Shamir with Aborts* paradigm.

Definition 2.2 (Correctness of TSS **with aborts).** *We say that a R-round* TSS *with aborts p-terminates if:*

$$\Pr\left[\mathsf{Game}^{\mathsf{TS\text{-}corr}}_{\mathsf{TSS}}(\kappa, N, T, \mathsf{msg}, \mathsf{act}) \neq \perp\right] \geqslant p \tag{1}$$

$$\Pr\left[\mathsf{Game}^{\mathsf{TS\text{-}corr}}_{\mathsf{TSS}}(\kappa, N, T, \mathsf{msg}, \mathsf{act}) = 0\right] = \mathsf{negl}(\kappa) \tag{2}$$

where $\mathsf{Game}^{\mathsf{TS\text{-}corr}}_{\mathsf{TSS}}$ *is defined in Figure 2.*

$\mathsf{Game}^{\mathsf{TS\text{-}corr}}_{\mathsf{TSS}}(\kappa, N, T, \mathsf{msg}, \mathsf{act})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, N, T)$
$(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Sign.Keygen}(\mathsf{pp})$
for $\{i \in [N]\}$ **do** $\{\mathsf{st}_i := \varnothing\}$
for $r \in [R]$ **do**
  for $i \in \mathsf{act}$ **do**
    $(\mathsf{pm}^i_r, \mathsf{st}_i) \leftarrow \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg},$
      $(\mathsf{pm}^j_{r-1})_{j\in\mathsf{act}}, i, \mathsf{sk}_i, \mathsf{st}_i)$
if $\{\ \exists i \in \mathsf{act}, \mathsf{pm}^i_R = \mathbf{abort}\ \}$ **then** $\{$ **return** $\perp$ $\}$
$\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{act}, (\mathsf{pm}^i_r)_{r\in[R], i\in\mathsf{act}})$
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Figure 2: Correctness game for TSS with aborts.

2.3 Modulus Rounding

We recall helper functions over $\mathbb{Z}_q$, used in ML-DSA. These functions are applied coefficient-wise when applied to polynomials in $\mathcal{R}_q$ to optimize the sizes of keys and signatures.

$\mathsf{Power2Round}_q(r)$: rounds an integer $r \in \mathbb{Z}_q$. It outputs a pair (r_0, r_1), where $r_0 = r \bmod^{\pm} 2^d$ and $r_1 = (r - r_0)/2^d$.

$\mathsf{HighBits}(r, \alpha)$: returns the high-order bits r_1 of r, defined as $\mathsf{HighBits}(r,\alpha) = \frac{r-(r \bmod^{\pm}\alpha)}{\alpha}$ if $r - (r \bmod^{\pm}\alpha) \neq q-1$ and 0 otherwise.

$\mathsf{MakeHint}_q(z, r, \alpha)$: produces a binary hint $h \in \{0,1\}$ indicating that the high bits of z and $z+r$ differ. The hint mechanism ensures the correct reconstruction of high bits during verification.

$\mathsf{UseHint}_q(h, r, \alpha)$: takes a hint bit $h \in \{0,1\}$ and an integer $r \in \mathbb{Z}_q$. Based on the hint, it corrects the computation of $\mathsf{HighBits}(r, 2\gamma_2)$. The goal is to recover $\mathsf{HighBits}(r+z, 2\gamma_2)$. If $h = 1$, a carry occurred, and the high bits need adjustment: the high bits by $+1$ if $r - \alpha \cdot r_1 > 0$, and otherwise by -1.

Lemma 2.1 ([48]). *For $q, \alpha > 0$ with $q > 2\alpha$, $q = 1 \bmod \alpha$ and α is even. Let $\mathbf{r}, \mathbf{z} \in \mathcal{R}_q$ where $\|\mathbf{z}\|_\infty \leqslant \alpha/2$. Then:*

$$\mathsf{UseHint}_q(\mathsf{MakeHint}_q(\mathbf{z}, \mathbf{r}, \alpha), \mathbf{r}, \alpha) = \mathsf{HighBits}_q(\mathbf{r} + \mathbf{z}, \alpha).$$

2.4 Hardness Assumptions

We rely on the MLWE assumption underlying the security of ML-DSA. Note that we do not explicitly leverage the STMSIS [48] assumption, which also underlies ML-DSA, and instead assume the unforgeability of ML-DSA in our proofs.

Definition 2.3 (MLWE)**.** *Let k, ℓ, q be integers, χ be a probability distribution over $\mathcal{R}^{k+\ell}$. The advantage of an algorithm $\mathcal{A}$ in solving the $\mathsf{MLWE}_{q,k,\ell,\chi}$ problem is defined as:*

$$\left|\Pr[\mathcal{A}(\mathbf{A}, \begin{bmatrix}\mathbf{A} & \mathbf{I}_k\end{bmatrix} \cdot \mathbf{s}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]\right| \qquad (3)$$

where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k\times\ell}, \mathbf{u} \xleftarrow{\$} \mathcal{R}_q, \mathbf{s} \leftarrow \chi$. The MLWE assumption states that any efficient adversary $\mathcal{A}$ has a negligible advantage against MLWE.

2.5 The Rényi divergence

Following Devevey et al. [27], we rely on the *Rényi divergence of infinite order* R_∞ as well as the *smooth Rényi divergence* R^ε_∞ to measure the distance between two distributions.

Definition 2.4. *Let $\mathcal{P}, Q$ two distributions such that $\mathcal{P}$ is absolutely continuous with respect to Q. The Rényi divergence of infinite order is $R_\infty(\mathcal{P}||Q) = \mathsf{ess\ sup}\frac{\partial\mathcal{P}}{\partial Q}(x)$.*

We additionally recall a relaxed version of the Rényi divergence from [27, Def. 2.1], called *smooth Rényi divergence*, where one can remove a few problematic points from the support, including those that may lie in $\mathsf{Supp}(\mathcal{P})\backslash\mathsf{Supp}(Q)$.

Definition 2.5. *Let $\varepsilon > 0$. Let $\mathcal{P}, Q$ two probability distributions such that $\int_{\mathsf{Supp}(Q)} \mathcal{P}(x)\partial\mu(x) \geqslant 1-\varepsilon$ for the measure μ. Their ε-smooth Rényi divergence is $R^\varepsilon_\infty(\mathcal{P}||Q) = \inf\{M > 0 \mid \Pr_{x\leftarrow\mathcal{P}}(\mathcal{P}(x) \leqslant MQ(x)) \geqslant 1-\varepsilon\}$.*

Lemma 2.2 ([4] and [44]). *For two distributions $\mathcal{P}, Q$, the Rényi divergence satisfies the following properties:*

- ***Data processing inequality.*** *$R_\infty(\mathcal{P}^f||Q^f) \leqslant R_\infty(\mathcal{P}||Q)$ for any function f, where $\mathcal{P}^f$ (resp. Q^f) is the distribution of $f(y)$ induced by sampling $y \leftarrow \mathcal{P}$ (resp. $y \leftarrow Q$).*
- ***Multiplicativity.*** *Assume that $\mathcal{P}$ and Q are the joint distributions of random variables $(x_i)_i$. Note $\mathcal{P}_{i,|x_{<i}=v}, Q_{i,|x_{<i}=v}$, the conditional distributions of x_i given $x_{<i} = v_{<i}$. Assume $R_\infty(\mathcal{P}_{i,|x_{<i}=v_{<i}}||Q_{i,|x_{<i}=v_{<i}}) \leqslant r_i$ for any possible $v_{<i}$. Then, $R_\infty(\mathcal{P}||Q) \leqslant \prod_i r_i$.*
- ***Probability preservation.*** *for any event $E \subseteq \mathsf{Supp}(Q)$, we have $Q(E) \geqslant \mathcal{P}(E)/R_\infty(\mathcal{P}||Q)$.*

2.6 Rejection Sampling

ML-DSA is a Fiat-Shamir with Aborts signature scheme. It heavily relies on the rejection sampling technique introduced by Lyubashevsky [53], which we recall with the same notations as Finally! [26].

We extend the algorithm Rej to $\mathsf{Rej}(\mathbf{v}, \chi_\mathbf{z}, \chi_\mathbf{r}, M; \mathbf{r})$ to parse the randomness $\mathbf{r} \leftarrow \chi_\mathbf{r}$ as an input. We will also write $(\mathbf{z}|\mathrm{acc})_\mathbf{v}$ to denote the distribution of $\mathbf{z} := \mathbf{v} + \mathbf{r}$ conditioned on $b = 1$, and $(\mathbf{z}|\mathrm{rej})_\mathbf{v}$ to denote the distribution of $\mathbf{z} := \mathbf{v} + \mathbf{r}$ conditioned on $b = 0$ (note that in that case $\mathbf{z} = \perp$ but the distribution we are interested in is the one of $\mathbf{v} + \mathbf{r}$ so we abuse this notation).

Rejection sampling aims to map a distribution $\mathbf{v} + \chi_\mathbf{r}$ – that depends on the sensitive value $\mathbf{v}$ – to a distribution $\chi_\mathbf{z}$ that does not. We recall the following lemma that bounds the Rényi divergence between the real and ideal distributions above.

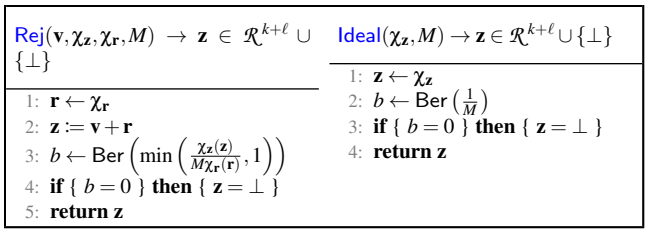

Figure 3: Rejection algorithm and ideal algorithm.

Lemma 2.3 (Lemma 4.1, [27]). *Let $M > 1$ and $\varepsilon < 1$ such that $R^{\varepsilon}_{\infty}(\chi_{\mathbf{z}} || \mathbf{v} + \chi_{\mathbf{r}}) \leqslant M$. Then, $R_{\infty}(\mathsf{Rej} || \mathsf{Ideal}) \leqslant 1 + \frac{\varepsilon}{M-1}$.*

2.7 Rejection sampling over hyperballs

Devevey et al. [27] showed that hyperball-based rejection sampling achieves the most compact parameters, and outperforms uniforms [53] and polytopes as used by Bambury et al. [5]. Hyperballs achieve the same compactness as using Gaussians, but have the advantage of a simpler rejection condition in the form of a norm two bound check.

Definition 2.6 (Continuous hyperball). *We denote:*

$$\mathcal{B}_{\mathcal{R},\ell}(r,\mathbf{c}) = \left\{ \mathbf{x} \in \mathcal{R}_{\mathbb{R}}^{\ell} \mid \|x - \mathbf{c}\|_2 \leqslant r \right\}$$

the continuous hyperball with center $\mathbf{c} \in \mathcal{R}_{\mathbb{R}}^{\ell}$ and radius $r > 0$ in dimension $\ell > 0$. When $\mathbf{c} = 0$, we omit it.

To bound the Rényi divergence we rely on the followings.

Definition 2.7 (Regularized Incomplete Beta Function). *The incomplete beta function is defined over $[0,1] \times \mathbb{R}^+ \times \mathbb{R}^+$ as $B : (x;a,b) \mapsto \int_0^x t^{a-1}(1-t)^{b-1}\partial t$. We also define $I_x(a,b) = B(x;a,b)/B(1;a,b)$.*

Lemma 2.4 ([27, Lemma 5.1]). *Let $\ell \geqslant 1$ and $\mathbf{v} \in \mathbb{R}^{n\ell}$. Let $\varepsilon \in [0,1/2)$ and $\phi \geqslant 1$ be such that $2\varepsilon = I_{1-1/\phi^2}\left(\frac{n\ell+1}{2}, \frac{1}{2}\right)$. Let $r, r' > 0$ such that $r'^2 \geqslant r^2 + \|\mathbf{v}\|_2^2 + 2r\|\mathbf{v}\|_2/\phi$. It holds that:*

$$R^{\varepsilon}_{\infty}\left(\mathcal{U}(\mathcal{B}_{\mathcal{R},\ell}(r)) \,||\, \mathcal{U}(\mathcal{B}_{\mathcal{R},\ell}(r',\mathbf{v}))\right) = \left(\frac{r'}{r}\right)^{n\ell} \tag{4}$$

Let $M > 1$. If $r \geqslant \|\mathbf{v}\|_2 \cdot \frac{\frac{1}{\phi} + \sqrt{\frac{1}{\phi^2} + M^{2/(n\ell)} - 1}}{M^{2/(n\ell)} - 1}$ and $r' = M^{1/(n\ell)} r$, then the value in Eq. (4) is upper-bounded by M.

Lemma 2.5 (from [27, Section A.6]). *For $n > 1, \phi > 1$, we have $I_{1-1/\phi^2}\left(\frac{n+1}{2}, \frac{1}{2}\right) < \left(1 - \frac{1}{\phi^2}\right)^{n-1} \cdot n \cdot \left(1 - \frac{1}{\phi}\right)$.*

We describe in Fig. 3 the imbalanced rejection sampling that we will use for Threshold-ML-DSA.

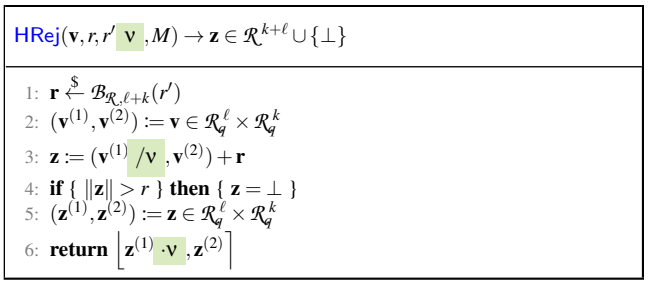

Figure 4: Imbalanced rejection sampling over hyperballs. Major differences with Rej (Fig. 3) are highlighted.

3 Threshold ML-DSA

This section describes our variant of ML-DSA (Module-Lattice-Based Digital Signature Algorithm) optimized for small threshold settings. We build upon the foundation laid by Finally! [26], which introduced the key innovation of leveraging short replicated secret sharing to distribute rejection sampling in lattice-based signatures, by performing parallel rejection sampling steps in each party. However, Finally! was designed as a custom signature scheme, and adapting its techniques to the ML-DSA standard presents significant challenges due to fundamental differences in their rejection sampling, target norm (Finally! relies on the Euclidean norm, while ML-DSA uses the infinity norm) and rounding procedures.

At a high level, the template of Finally! consists in sampling several short secrets $\mathbf{s}_I$ and deriving a public key as the sum of their corresponding commitments $\mathsf{vk} = \sum \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \mathbf{s}_I$. These secrets are shared among all parties such that any subset of at least T parties jointly knows all the $\mathbf{s}_I$'s. Signing proceeds via parallel rejection sampling performed independently by each party. We adopt this template, and tailor it to ML-DSA on two fronts: we design a distributed and a posteriori key generation, and we tailor the parallel rejection sampling to ML-DSA constraints.

While performing rejection sampling over T parties has a negative impact on the overall acceptance probability (explaining the larger sizes of Finally! compared to ML-DSA), we mitigate this by using tighter distributions over *hyperballs*, which is particularly effective in small-threshold regimes. Hence, our thresholdization exploits the non-tightness in ML-DSA's use of uniform sampling. We exploit the slack between rejection sampling over hyperballs (as noted in Section 2.7) and uniform distributions, allowing us to retain compact signature sizes even in the distributed setting.

As the *hint* technique used in ML-DSA only targets the second portion $\mathbf{z}^{(2)}$ of the signature, it introduces a much stricter verification condition on $\mathbf{z}^{(2)}$ than $\mathbf{z}^{(1)}$. We optimize the success rate of this check by using imbalanced rejection sampling in which $\mathbf{z}^{(2)}$ rejects towards a smaller hyperball

Table 2: Parameters used in Threshold ML-DSA.

ML-DSA	
$\mathcal{R}_q$	Polynomial ring $\mathcal{R}_q = \mathbb{Z}[X]/(q, X^n+1)$
(k,ℓ)	Dimension of the public matrix $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$
τ	Number of ± 1's in challenge c
d	Amount of bits dropped from the public key $\mathbf{t}$
η	Secret key range: $\chi_s = \mathcal{U}([-\eta,\eta]^{n(\ell+k)})$
$(\gamma_1,\gamma_2,\beta)$	Ranges for the signature generation and verification ($\beta = \tau\cdot\eta$)
ω	Number of 1's in the hint h
Threshold ML-DSA	
K	Number of parallel protocol repetitions
ν	Expansion factor for the first ℓ coordinates of the randomness
r'	Randomness ball rad. $\chi_{\mathbf{r}} = \left\{ \lfloor(\nu\mathbf{x}_1, \mathbf{x}_2)\rceil \mid (\mathbf{x}_1,\mathbf{x}_2) \xleftarrow{\$} \mathcal{B}_{\mathcal{R},\ell+k}(r') \right\}$
r	Target ball rad. $\chi_{\mathbf{z}} = \left\{ \lfloor(\nu\mathbf{x}_1, \mathbf{x}_2)\rceil \mid (\mathbf{x}_1,\mathbf{x}_2) \xleftarrow{\$} \mathcal{B}_{\mathcal{R},\ell+k}(r) \right\}$
M	Rejection parameter (per party) $M = \left(\frac{r'}{r}\right)^{n(k+\ell)}$

than $\mathbf{z}^{(1)}$. The imbalance is parametrized by a factor ν, that expands the randomness used to sample the first ℓ coordinates of $\mathbf{r}$ by a factor $\nu > 1$. This results in a larger acceptance region for $\mathbf{z}^{(1)}$ while keeping the same acceptance region for $\mathbf{z}^{(2)}$, thus significantly lowering the rejection rate due to the first part of the signature. See HRej (Fig. 4) for a concrete description.

To amplify the success probability p, we run $K > 1$ iterations of the protocol in parallel. That is, during the signing procedure, each party i produces K commitments $(\mathbf{w}_{i,j})_{j\in[K]}$, and a response $\mathbf{z}_{i,j}$ for each aggregated commitment $\mathbf{w}_j = \sum_{i\in\mathsf{act}} \mathbf{w}_{i,j}$. Concretely, taking $K = \lceil 1/p \rceil$ guarantees a correctness probability $\geqslant 1 - e^{-1} \approx 0.632$.

3.1 Construction

As defined in Section 2, threshold signature schemes primarily consist of key generation, signing, combining, and verifying procedures. To formalize these procedures for Threshold ML-DSA, we first outline the parameters used in the single-party ML-DSA scheme, along with the additional hyperball parameters $r_{N,T}$ and $r'_{N,T}$ required for our threshold variant. We collect all of them in Table 2.

Key Generation. Our construction relies on the *short secret sharing* notion introduced by del Pino et al. [22] (RSS) and already leveraged in Finally! [26], which is particularly fit for lattice-based constructions. These secret sharings are special instances of Linear Secret Sharing Schemes (LSSS) that additionally guarantee shares with *small norm and small reconstruction coefficients*. This is particularly suited for rejection-sampling–based schemes, as it enables each party to perform rejection sampling locally on a *short* partially reconstructed secret. This *short replicated secret sharing* scheme consists in sampling $\binom{N}{N-T+1}$ ML-DSA secrets $(\mathsf{sk}_I)_I$ from $\chi_{\mathbf{s}}$, that is, one for each distinct set I of $N-T+1$ users. Since there are at most $T-1$ corrupted parties, at least one of these sets is fully non-corrupted and its secret remains hidden from the corrupted parties.

The final public key is derived by summing all secrets, multiplying by $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}$, and rounding the result as in ML-DSA, yielding a sample from the MLWE distribution. Thanks to at least one secret remaining hidden from corrupted parties, the public key maintains pseudorandomness under the same MLWE assumption as ML-DSA. In addition, any set J of T parties can reconstruct the aggregated secret: by construction, there exists a partition $\mathcal{P}_{j\in J}$ so that $\mathsf{sk} = \sum_{j\in J}(\sum_{I\in\mathcal{P}_j} \mathsf{sk}_I)$. As an example, for $N=3$ and $T=2$, we have $\mathsf{sk} = \mathsf{sk}_{01} + \mathsf{sk}_{02} + \mathsf{sk}_{12}$. If users $0,2$ want to sign, then user 0 will use $\mathsf{sk}_{01} + \mathsf{sk}_{12}$ as partial key, while user 2 will simply use sk_{12}.

The key generation procedure is detailed in Fig. 5. While Fig. 5 assumes a trusted dealer (as Finally!) for simplicity, we show that a distributed key generation protocol is possible in: see Section D.

$\mathsf{RSS}(N,T,\chi) \to (\mathbf{s}, \mathsf{sk} := (\mathsf{sk}_1,\ldots,\mathsf{sk}_N))$

1: **for** $I \subset [N]$ such that $|I| = N-T+1$ **do**
2: $\quad \mathbf{s}_I \leftarrow \chi$
3: $\mathbf{s} := \sum_I \mathbf{s}_I$
4: **for** $i \in [N]$ **do** ▷ Distribute the shares
5: $\quad \mathsf{sk}_i := \{I : \mathbf{s}_I \mid I \subset [N] \text{ s.t. } i \in I \wedge |I| = N-T+1\}$
6: **return** $(\mathbf{s}, \mathsf{sk} := (\mathsf{sk}_0, \cdots, \mathsf{sk}_{N-1}))$

$\mathsf{Keygen}(\kappa, N, T) \to (\mathsf{vk}, \mathsf{sk})$

1: $\rho \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} := \mathsf{ExpandA}(\rho)$
3: $(\mathbf{s}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{RSS}(N,T,\chi_{\mathbf{s}})$
4: $\mathbf{t} := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}$
5: $(\mathbf{t}_\top, \mathbf{t}_\bot) := \mathsf{Power2Round}(\mathbf{t}, d)$
6: $tr \in \{0,1\}^{256} := H(\rho \| \mathbf{t}_\top)$
7: **return** $(\mathsf{vk} := (\rho, \mathbf{t}_\top), \mathsf{sk} := (tr, \mathsf{sk}_i)_{i\in[N]})$

Figure 5: Key generation procedure of Threshold ML-DSA.

Signing. The distributed signing protocol is detailed in Fig. 6. Any set act of T signers collectively knows all the secrets $\mathbf{s}_I$. The core idea is to publicly decide on a partition $\bigsqcup_{i\in\mathsf{act}} \mathbf{m}_i = \{I\}_I$ of the secrets (or rather their indices) among the T parties using a deterministic function (RSSRecover), and then run T Fiat-Shamir with Abort protocols in parallel to produce a partial signature for each partial secret $\mathbf{s}_i^{\mathsf{part}} = \sum_{I\in\mathbf{m}_i} \mathbf{s}_I$: partial commitments $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$ and signatures $\mathbf{z}_i = c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$.

In particular, each party uses a full MLWE sample as commitment $\mathbf{w}_i$ (i.e. $\mathbf{w}_i = \mathbf{A}\cdot\mathbf{y} + \mathbf{e}$) instead of using $\mathbf{A}\cdot\mathbf{y}$ as in ML-DSA. This change allows for direct reveal of $\mathbf{w}_i$ before rounding, even in case of rejection of the partial signatures. To ensure correctness, we perform a second rejection step and

verify that the sum of the responses $\mathbf{z} = \sum_{i \in \mathsf{act}} \mathbf{z}_i$ verifies the ML-DSA verification, roughly that the equation $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} = c \cdot \mathbf{t} + \mathbf{w}$ holds and the signature $\mathbf{z}$ has a small norm. The challenge c can be computed as $c = \mathsf{SampleInBall}(\tilde{c})$, where $\tilde{c} = \mathsf{H}(\mu || \mathsf{HighBits}(\mathbf{w}, 2\gamma_2))$ and $\mathbf{w} = \sum_{i \in \mathsf{act}} \mathbf{w}_i$. $\mathsf{SampleInBall}$ maps $\tilde{c}$ to a polynomial c such that $\|c\|_1 = \tau$ and $\|c\|_\infty = 1$.

Following prior works [23, 26], we design a three-round[2] scheme secure against rushing adversaries (who target $\mathbf{w}$):

1. **First round:** parties sample randomness $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$ and compute the commitment $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$. They publish a hash of the commitment $\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i)$.

2. **Second round:** parties reveal the values of $(\mathbf{w}_i)_{i \in \mathsf{act}}$.

3. **Third round:** Each party computes the response $\mathbf{z}_i = c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$, where c is derived from the aggregated commitment $\mathbf{w} = \sum_{i \in \mathsf{act}} \mathbf{w}_i$, and apply rejection sampling.

In the complete scheme, and as ML-DSA, we omit the second part $\mathbf{z}_i^{(2)} \in \mathcal{R}_q^k$ of the responses since the verifier can reconstruct it from public values. Any public party can then aggregate the partial signatures into $\mathbf{z}^{(1)} = \sum_{i \in \mathsf{act}} \mathbf{z}_i^{(1)} \in \mathcal{R}_q^\ell$ to obtain the signature and compute the corresponding hint $\mathbf{h}$ to output the signature $(c, \mathbf{z}^{(1)}, \mathbf{h})$, after checking that the verification equation holds.

Remark 1 (Balanced partition of the shares for Replicated Secret Sharing)**.** *The sizes of partial secrets $\mathbf{s}_i^{\mathsf{part}}$ in Threshold ML-DSA impact greatly the efficiency of parameters as it directly correlates with the norm 2 of the hyperballs that are used, and it is important to use* as few secrets as possible *in a session for each party in* $\mathsf{RSSRecover}$*. To tackle this, we propose in Section B a graph based algorithm that computes a balanced partition of the shares, i.e. each party signs with at most $\left\lceil \binom{N}{N-T+1} / T \right\rceil$ secrets, instead of $\binom{N}{N-T}$.*

In practice, we can hardcode the optimal partition produced by our algorithm and include it as part of the public description of the Threshold ML-DSA scheme.

Remark 2 (Offline efficiency)**.** *Although not the main focus of this work, it is worth noting that the first round of the signing protocol can be executed in an offline phase, independently of the signing set and message.*

Remark 3 (Signature distribution)**.** *It may be observed that the distribution of signatures produced by Threshold ML-DSA differs from that of ML-DSA, as the response $\mathbf{z}$ is generated using rejection sampling over hyperballs, rather than uniform sampling. However, it retains the same support as ML-DSA signatures, and its unforgeability holds under the same assumptions, c.f. Section 3.2.*

[2]This is per signing attempt.

$\mathsf{RSSRecover}(\mathsf{act}) \to \mathcal{P}(S)^{\mathsf{act}}$

Compute partition $\mathbf{m} = (m_i)_{i \in \mathsf{act}}$ s.t. $\bigsqcup m_i = \{I \subset [N] \mid |I| = N - T + 1\}$ and minimizing $\max_{i \in \mathsf{act}} |m_i|$, using a maximum flow algorithm. See Section B.
return $\mathbf{m}$

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
$\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i) \in \{0,1\}^{2\kappa}$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}_1^i := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ (\mathsf{pm}_1^i, \cdot, \cdot) \in \mathsf{st}_i\ \}$
Pick $(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_1^i = \mathsf{cmt}_i$
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_3(\mathsf{vk}, \mathsf{pm}_2, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ (\cdot, \mathsf{pm}_2^i, \cdot) \in \mathsf{st}_i\ \}$
Pick $(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_2^i = \mathbf{w}_i$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$ and $\mathsf{pm}_2 = (\mathbf{w}_j)_{j \neq i}$
assert$\{\ \forall j \in \mathsf{act}, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)\ \}$
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
$\mu \in \{0,1\}^{512} := H(tr || \mathsf{msg})$
$\mathbf{w} := \sum_{j \in \mathsf{act}} \mathbf{w}_i$
$\mathbf{w}_\top := \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
$\tilde{c} \in \{0,1\}^{2\cdot\kappa} := \mathsf{H}(\mu || \mathbf{w}_\top)$
$c := \mathsf{SampleInBall}(\tilde{c})$ ▷ Global challenge
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I \in m_i} \mathbf{s}_I$
$\mathbf{z}_i \leftarrow \mathsf{HRej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, r, r', \nu, M; \mathbf{r}_i)$ ▷ Individual response
if $\{\ \mathbf{z}_i = \perp\ \}$ **then** $\{$ **abort** $\}$
$\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)} := \mathbf{z} \in \mathcal{R}_q^\ell \times \mathcal{R}_q^k$
return $(\mathsf{pm}_3^i := \mathbf{z}_i^{(1)}, \mathsf{st}_i)$

Figure 6: Signing procedure in Threshold ML-DSA.

Combine and Verify. The combine and verify procedures are detailed in Fig. 7. We use the ML-DSA verification. Combine involves computing the hint $\mathbf{h}$ that allows verifiers to recover $\mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$ from the value $\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top$ used as an approximation of $\mathbf{w}$. Similarly to ML-DSA, we compute the difference $\boldsymbol{\delta} = \mathbf{w} - \left(\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top\right)$ and use the function $\mathsf{MakeHint}$ with $\boldsymbol{\delta}$ to compute the hint. The high bits of $\mathbf{w}$ are correctly recovered as long as $\|\boldsymbol{\delta}\|_\infty \leqslant \gamma_2$; we thus restart the protocol if this is not the case. Verification proceeds just as

in ML-DSA: the procedure also restarts if $\|\mathbf{z}^{(1)}\|_\infty \geqslant \gamma_1 - \beta$ or if $\mathbf{h}$ has more than ω 1's. While this check ensures security in ML-DSA, it is only used for correctness here. Indeed, the hiding property of the rejection step in $\mathsf{ShareSign}_3$ already ensures that $\mathbf{z}$ does not leak any secret information. Thanks to parallel repetitions we can minimize the need for restarts as we ensure that any execution of the protocol will output a signature with probability at least $1/2$.

$\mathsf{Combine}(\mathsf{vk} = (\mathsf{seed}, \mathbf{t}_\top), \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_k)_{k\in\{1,2,3\}})$

1: Parse $\mathsf{pm}_2 = (\mathbf{w}_i)_i$ and $\mathsf{pm}_3 = (\mathbf{z}_i^{(1)})_i$
2: $\mu \in \{0,1\}^{512} := H(tr||\mathsf{msg})$
3: $\mathbf{w} := \sum_{j\in\mathsf{act}} \mathbf{w}_i$ ▷ Aggregated commitment in $\mathcal{R}_q^k$
4: $\mathbf{w}_\top := \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
5: $\tilde{c} := \mathsf{H}(\mu||\mathbf{w}_\top)$ ▷ $\tilde{c} \in \{0,1\}^{2\cdot\kappa}$
6: $c := \mathsf{SampleInBall}(\tilde{c})$ ▷ Global challenge
7: $\mathbf{z}^{(1)} = \sum_{i\in\mathsf{act}} \mathbf{z}_i^{(1)}$
8: $\boldsymbol{\delta} := \mathbf{w} - \left(\mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top\right)$ ▷ Recover error in $\mathcal{R}_q^k$
9: $\mathbf{h} := \mathsf{MakeHint}_q(\boldsymbol{\delta}, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d\cdot c\cdot\mathbf{t}_\top, 2\gamma_2)$
10: **if** $\left(\|\mathbf{z}^{(1)}\|_\infty \geqslant \gamma_1 - \beta\right)$ **or** $(\|\boldsymbol{\delta}\|_\infty > \gamma_2)$ **or** $(\|\mathbf{h}\|_1 > \omega)$ **then**
11: **abort**
12: **return** $\mathsf{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$

$\mathsf{Verify}(\mathsf{vk} := (\rho, \mathbf{t}_\top), \mathsf{msg}, \mathsf{sig})$

1: Parse $\mathsf{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$
2: $\mathbf{A} := \mathsf{ExpandA}(\rho)$
3: $\mu \in \{0,1\}^{512} := H(tr||\mathsf{msg})$
4: $c := \mathsf{SampleInBall}(\tilde{c})$
5: $\mathbf{w}'_\top := \mathsf{UseHint}_q\left(\mathbf{h}, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d\cdot c\cdot\mathbf{t}_\top, 2\gamma_2\right)$
6: **return** $\|\mathbf{z}^{(1)}\|_\infty < \gamma_1 - \beta$ **and** $\tilde{c} = \mathsf{H}(\mu||\mathbf{w}'_\top)$ **and** $\|\mathbf{h}\|_1 \leqslant \omega$

Figure 7: Combine and verification of Threshold ML-DSA.

3.2 Correctness and security

We prove that Threshold ML-DSA (i) is unforgeable, and (ii) p-terminates. For (i), Threshold ML-DSA is unforgeable if ML-DSA is unforgeable and the MLWE problem used in ML-DSA is hard. For (ii), we introduce a bound B such that for any $\mathbf{m}_i$ returned by $\mathsf{RSSRecover}$ and $(\mathbf{u}_1, \mathbf{u}_2) = \sum_{I\in\mathbf{m}_i} \mathbf{s}_I \in \mathcal{R}_q^{\ell+k}$, $\|(\frac{1}{\mathrm{v}}\cdot c\cdot\mathbf{u}_1, c\cdot\mathbf{u}_2)\|_2 \leqslant B$ with overwhelming probability for a uniformly sampled challenge c and over the randomness of the key generation.

Let $\varepsilon > 0$. We assume that (r, r', ε) verify the conditions of Lemma 2.4 for secret vectors of norm B, i.e. $2\varepsilon = I_{1-1/\phi^2}\left(\frac{n(k+\ell)+1}{2}, \frac{1}{2}\right)$ for some ϕ, and $r'^2 \geqslant r^2 + B^2 + 2rB/\phi$, implying Lemma 3.1.

Lemma 3.1. *For the above constraints, we have for any* $\|\mathbf{v}\|_2 \leqslant B$, $R_\infty^\varepsilon(\chi_\mathbf{z} || \chi_\mathbf{r} + \mathbf{v}) = M$

Proof. The equation checks out by applying Lemma 2.4 with the data processing inequality for Rényi divergence. □

Theorem 3.1. *Assume* $\varepsilon = o(1)$ *and* $\Pr[\mathsf{HighBits}(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{w}', 2\gamma_2)] = o(1)$, *where* $\mathbf{w}, \mathbf{w}'$ *are sampled from* $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \sum_{i=1}^T \chi_\mathbf{r}$, *and* $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k\times\ell}$ *– remark that this holds notably under the* $\mathsf{MLWE}_{q,k,\ell,\sum_{i=1}^T \chi_\mathbf{r}}$ *assumption. Threshold ML-DSA p-terminates in the ROM for*

$$p = \mathbb{E}_{k\leftarrow\mathsf{Bin}\left(K,(1-\frac{1}{M})^T\right)}[v_k] + o(1)$$

where v_k, *for* $k \in [0, K]$, *is the probability that in at least one of the* k *combinations we have* $\|\mathbf{z}^{(1)}\|_\infty \leqslant \gamma_1 - \beta$ *and* $\|\boldsymbol{\delta}\|_\infty \leqslant \gamma_2$ *and the number of 1's in* $\mathbf{h}$ *is less than* ω, *when* $\mathbf{z}$ *is sampled from* $\sum_{i\in[T]} \chi_\mathbf{z}$ *and* $\boldsymbol{\delta} = c\cdot\mathbf{z}^{(2)} - c\cdot\mathbf{t}_\perp$.

Proof. First, observe that by design, any signature produced by the scheme is a valid ML-DSA signature. The signature necessarily (i) verifies $\|\mathbf{z}^{(1)}\| < \gamma_1 - \beta$, and (ii) $\mathbf{h}$ has at most ω 1's. Additionally, the scheme ensures that $\boldsymbol{\delta} := \mathbf{w} - \left(\mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d\cdot c\cdot\mathbf{t}_\top\right)$ has an infinity norm at most γ_2. By applying Lemma 2.1, we obtain $\mathsf{UseHint}_q\left(\mathbf{h}, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d\cdot c\cdot\mathbf{t}_\top, 2\gamma_2\right) = \mathbf{w}_\top$ during verification. Thus, the verification bounds are satisfied, and the correct challenge c is successfully recovered.

It remains to ensure that the scheme succeeds with a probability of at least p. We prove this with a series of games starting from $\mathsf{Game}_0 := \mathsf{Game}_{\mathsf{TSS}}^{\mathsf{TS\text{-}corr}}$. We denote $p_i = \Pr[\mathsf{Game}_i() \neq \perp]$.

Game_1. In this game, we assert that for any signing party and parallel session during the second round, we have $\left\|(\frac{1}{\mathrm{v}}\cdot c\cdot\mathbf{s}_1^{\mathsf{part}}, c\cdot\mathbf{s}_2^{\mathsf{part}})\right\|_2 \leqslant B$, otherwise the game returns 0. We need to add at most $T\cdot K$ such assertions. As B is an overwhelming bound, we obtain by union-bound:

$$p_0 \geqslant p_1 - TK\cdot\mathsf{negl}(\kappa) = p_1 - \mathsf{negl}(\kappa)$$

Game_2. In this game, we ensure that the parallel signing sessions use different $\tilde{c}$ and $\mathbf{w}$ with different high bits. Otherwise, the game returns 0. It ensures that different challenges are effectively used in every session. As we have at most K independent sessions, we have from the union bound:

$$p_1 \geqslant p_2 - K^2\cdot(\Pr[\mathsf{HighBits}(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{w}', 2\gamma_2)] + 2^{-2\cdot\kappa})$$

Game_3. In this game, we sample the challenge c of each session in advance, sample $\mathbf{z}_i \leftarrow \mathsf{Rej}(c\cdot\mathbf{s}_i^{\mathsf{part}}, \chi_\mathbf{z}, \chi_\mathbf{r}, M)$ for each party. If $\mathbf{z}_i = \perp$, we instead sample $\mathbf{z}_i$ from the distribution $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$ of rejected $\mathbf{z}$. Then, we program the random oracle after computing $\mathbf{w} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}\cdot\mathbf{z} - c\cdot\mathbf{t}$'s, with $\mathbf{z} = \sum_{i\in\mathsf{act}} \mathbf{z}_i$

in each parallel session. The distribution of $\mathbf{w}$ is identical as in Game_2, and Game_2 ensured that we can program c without affecting the probability of winning as no two $\mathbf{w}$ have the same high bits, and then computing $\mathbf{z}$ first in this way is equivalent: $p_2 = p_3$.

Game_4. In this game, we sample the partial responses $\mathbf{z}_i$ using the Ideal rejection sampling from Fig. 3. Recall the Game_1 ensured that $c \cdot \mathbf{s}_i^{\mathsf{part}}$ has a twisted norm bounded by B. We can thus apply the result Lemma 2.3 on the Rényi divergence of rejection sampling, using the intermediate result on hyperballs of Lemma 2.4, since by assumption r, r', B verify its conditions. The Rényi divergence when replacing a single $\mathbf{z}_i$ is $1 + \frac{\varepsilon}{M-1}$.

Let E the event "$\forall(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}^{\mathsf{part}}, \|(\frac{1}{\mathsf{v}} \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 \leqslant B$". Formally, we consider the random variable $X = ((\mathsf{sk}_i)_{i\in[N]}, (\mathbf{z}_i^j)_{i\in\mathsf{act}, j\in[K]})$ conditioned on E, where the $\mathbf{z}_i^j$ are the responses output by the parties in each session. After conditioning on $(\mathsf{sk}_i)_{i\in[N]}$ we observe that the $\mathbf{z}_i^j$ become independent. We can thus apply the multiplicativity of the Rényi divergence (Lemma 2.2) to bound the Rényi divergence between X in Game_3 and Game_4 by $\left(1 + \frac{\varepsilon}{M-1}\right)^{KT}$.

We then apply the data processing inequality and probability preservation of the Rényi divergence to obtain:

$$\Pr[\mathsf{Game}_4() = \perp \mid E] \cdot \left(1 + \tfrac{\varepsilon}{M-1}\right)^{KT} \geqslant \Pr[\mathsf{Game}_3() = \perp \mid E]$$

As E has the same probability in both games, we have:

$$\Pr[\mathsf{Game}_4() = \perp] \cdot \left(1 + \frac{\varepsilon}{M-1}\right)^{KT} \geqslant \Pr[\mathsf{Game}_3() = \perp]$$

Finally, $p_3 \geqslant 1 + \left(1 + \frac{\varepsilon}{M-1}\right)^{KT} \cdot (p_4 - 1)$.

Conclusion. At this point, acceptance during the protocol occurs independently of the responses $\mathbf{z}_i$ with probability $(1 - \frac{1}{M})$ for each party. Hence, each session succeeds during the protocol with probability $(1 - \frac{1}{M})^T$.

Additionally, we wish for the combination of responses to succeed. We can combine the above results to evaluate the final success probability of the protocol: $p_4 = \mathbb{E}_{k \leftarrow \mathsf{Bin}\left(K, (1-\frac{1}{M})^T\right)}[v_k]$ □

We now state the unforgeability of Threshold ML-DSA.

Theorem 3.2 (Unforgeability). *For any parameter $\phi > 0$, let $Q_s = 2/(K \cdot I_{1-1/\phi^2}\left(\frac{n(k+\ell)+1}{2}, \frac{1}{2}\right))$.*

The TSS *from Figs. 5 to 7 is* TS-UF *secure in the ROM, for up to Q_s calls to the individual signing oracles, under the unforgeability of ML-DSA as well as the hardness of the* $\mathsf{MLWE}_{q,k,\ell,\chi}$ *assumptions for $\chi \in \{\chi_{\mathbf{s}}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}\}$.*

We deduce from Theorem 3.2 that breaking the unforgeability of Threshold ML-DSA is as hard as breaking ML-DSA. Indeed, the MLWE assumption over $\chi_{\mathbf{s}}$ is the one underlying ML-DSA, and the ones over $\chi_{\mathbf{r}}$ and $\chi_{\mathbf{z}}$ are almost statistically verified due to the large width of the hyperballs we use, and in particular are much harder than the assumptions underlying ML-DSA. We provide below an overview of the proof and defer the formal statement and proof to Section G.

Our proof closely follows the approach of [26, Theorem 4], with two main differences: (i) we adapt the argument to account for the slightly different rounding of commitments $\mathbf{w}$, and (ii) instead of reducing to STMSIS [48], we reduce directly to the unforgeability of ML-DSA. The proof proceeds via a sequence of games, gradually transforming the real attack scenario into one where a forgery against the threshold scheme yields a forgery against single-party ML-DSA, under the hardness of the relevant MLWE problems.

- **Game 2.** We show that honest commitments $\mathbf{w}_i$ have high min-entropy before being revealed, so the adversary cannot guess them in advance. This allows us to program the random oracle $\mathsf{H}_{\mathsf{cmt}}$ lazily, returning random hashes in round 1 and sampling $\mathbf{w}_i$ only in round 2 for honest parties. This step is justified in the random oracle model.

- **Games 3–6.** We sample the challenges c in advance in round 2. The pre-image and collision resistance of $\mathsf{H}_{\mathsf{cmt}}$ ensure that the adversary has already chosen its $\mathbf{w}_i$ before round 2. With the lazy sampling from the previous game, honest commitments are sampled last, so the aggregated commitment $\mathbf{w} = \sum_i \mathbf{w}_i$ has high min-entropy. We can thus program the random oracle to return the desired challenge c without the adversary noticing.

- **Game 7.** We compute $\mathbf{z}_i$ first, even in case of rejection, and then derive $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{t}_i^{\mathsf{part}}$. If $\mathbf{z}_i$ is rejected, we sample it from the distribution of rejected $\mathbf{z}$. This is statically equivalent to the previous game.

- **Game 8.** We ensure that the norm of the secret vector $c \cdot \mathbf{s}_i^{\mathsf{part}}$ is at most B, which holds with overwhelming probability.

- **Game 9.** We replace the commitments $\mathbf{w}_i$ by uniform values when the corresponding $\mathbf{z}_i$ is rejected and will never be output. This change is indistinguishable to the adversary under the MLWE assumptions for the relevant distributions, following [26, Lemma 5].

- **Games 10–11.** We remove the last dependencies on the secrets by replacing the sampling of $\mathbf{z}_i$ with rejection sampling from the ideal functionality (recalled in Fig. 3). The Rényi divergence argument (see Section 2.7) ensures that, for up to Q_s queries, the adversary's advantage increases by at most 2 bits.

- **Games 12–13.** We replace the public key of Threshold ML-DSA (composed of $\binom{N}{N-T+1}$ secrets) by an ML-DSA

public key (a single secret). Since the signing oracles are now independent of secret values, this is justified by the MLWE assumption.

At the end of this sequence, any forgery output by the adversary yields a valid forgery against an ML-DSA key. Thus, the unforgeability of the threshold scheme reduces to that of ML-DSA, under the stated assumptions and in the ROM.

3.3 A Posteriori Key Sharing

In this section, we describe how a user with an existing ML-DSA signing key can split it into shares without affecting verification. To share a secret key $\mathbf{s}$, we sample a short secret sharing (as in the key generation of 3.1) with the difference that the shares will be conditioned on their sum being $\mathbf{s}$.

Algorithmic overview of key sharing (Section E.2). For $S = \binom{N}{N-T+1}$ shares, we use the root lattice $\mathcal{L} = \{(s_1,\ldots,s_S) \in \mathbb{Z}^S, \sum s_i = 0\}$ corresponding to the sharings of 0, and the goal of the user is to sample a Gaussian vector in the appropriate coset of $\mathcal{L}$ to share $\mathbf{s}$. While this can be done using the GPV sampler of Gentry et al. [39], using the parameters prescribed in [39] would result in shares with a significant standard deviation, causing ML-DSA verification to reject with very high probability and increasing communication by over $1000\times$. We address this by using Rényi divergence instead of statistical distance to obtain more compact parameters, as the sharing is used only once during key generation. We present an analysis of the smoothing parameter with Rényi divergence in Section E.2 which is of independent interest.

Security analysis (Section E.3). A caveat is that this key sharing reveals non-trivial information about the secret $\mathbf{s}$. We show in Section E.3 how this is captured by the hint-MLWE problem. However, this hint-MLWE instance is non-standard: the secret $\mathbf{s}$ comes from a uniform distribution (instead of Gaussian as in [49]), and the hint standard deviation is below the smoothing parameter of $\mathbb{Z}^n$. We prove that for Gaussian secrets, this still reduces efficiently to standard MLWE (Section E.3). We believe our generalized hint-MLWE problem is of independent interest.

Given this sampling algorithm and hint-MLWE problem, we prove that our threshold signature scheme remains secure with a posteriori key sharing - assuming secrets are Gaussian distributed (for the hint-MLWE reduction to MLWE).

Theorem 3.3 (Unforgeability). *The* TSS *from Figs. 5 to 7 with* a posteriori key sharing *is* TS-UF *secure in the ROM under the same conditions as Theorem 3.2 and if the* $\mathsf{MLWE}_{q,k,\ell,\chi_{hint}}$ *assumption holds for* χ_{hint} *the discrete Gaussian of standard deviation* $\sigma = 2^{-1/2}\left(\frac{1}{\sigma^2_{\text{ML-DSA}}} + \frac{S-1}{S\sigma^2_{share}}\right)^{-1/2}$ *with* $\sigma_{\text{ML-DSA}}, \sigma_{share}$ *the standard deviation of respectively the ML-DSA secret distribution and the secret shares,* $S = \binom{N}{N-T+1}$.

Proof. The proof is identical to the one of Theorem 3.2, expect for the games after Game_{11}.

- **Game 11'**. We replace the shares of the secret obtained with GPV, by the ideal Gaussian distribution. The advantage of the adversary is multiplied at most by the Rényi divergence between the two distributions, c.f. Section E.2.
- **Game 13**. As previously we replace the public key with an ML-DSA public key. Since one of the secret shares remains unknown the information leaked on the secret corresponds to a hint-MLWE, which we reduce to MLWE (Section E.3).

This concludes the proof. □

Discussion on parameters. Using these techniques, we can construct a secret sharing that reduces the unforgeability of Threshold ML-DSA by at most 7 bits (some loss is inevitable, since a short secret sharing always leaks information about the secret key), while increasing the communication cost by at most a factor 10.

While this result is already notable, we are more interested in estimating the security loss incurred by a posteriori key sharing under the constraint that the share distribution matches the standard deviation of the original key generation (and thus preserves the communication cost). However, this parameter choice falls outside our formal hint-MLWE reduction due to the necessary Gaussian standard deviation $\sqrt{2}$ being below the smoothing parameter of $\mathbb{Z}^n$. Note that this does not necessarily indicate a weakness in the key but rather a bias on all distributions. We provide a heuristic analysis as a sanity check: The leakage from $T-1$ corruptions yields a hint $\mathbf{h}$ with conditional entropy $H(\mathbf{s}|\mathbf{h})$. For shares with standard deviation $\sqrt{2}$, the effective entropy (obtained by direct computation) equals that of a secret with standard deviation 0.74. The lattice estimator by Albrecht et al. [3] indicates a security loss of at most 12 bits.

In summary, a posteriori secret sharing preserves all aspects of the original scheme (all algorithms but key generation remain unchanged), enables the reuse of an existing ML-DSA secret key, but incurs a loss between 7 to 12 bits of security.

3.4 Parameter Selection

To maintain backward compatibility with ML-DSA, we keep the same public parameters listed in Table 2, introducing only the new parameters K, r', r, and ν (respectively, the number of parallel repetitions, the respective radii of the randomness $\mathbf{r}'_i$ and rejected randomness $\mathbf{z}_i$, and the expansion factor for the first part of the rejected randomness

$\mathbf{z}_i^{(1)}$). Following the security analysis in Section 3.2, we first choose ϕ such that $\varepsilon = I_{1-1/\phi^2}\left(\frac{n(k+\ell)+1}{2}, \frac{1}{2}\right)/2 \leqslant 1/(KQ_s)$. Then, given an upper bound B on the norm of partial secrets (holding with overwhelming probability), and choosing $r'^2 \geqslant r^2 + B^2 + \frac{2rB}{\phi}$, we ensure that Threshold ML-DSA retains the same level of security as ML-DSA for up to Q_s signing queries. Concretely, we take $Q_s = 2^{50}$. We set $B = 1.3 \cdot \sqrt{\tau} \cdot \sqrt{n \cdot (k+\ell/\nu^2)} \cdot \sqrt{\mathsf{Var}(\mathcal{U}(-\eta,\eta))} \cdot \sqrt{\left\lceil \binom{N}{T-1}/T \right\rceil}$ as a heuristic overwhelming bound on the norm of partial secrets[3]. Moreover, RSSRecover ensures that each party uses at most $\left\lceil \binom{N}{T-1}/T \right\rceil$ secrets in a session for our selected parameters, c.f. Section B. Finally, we select K, r, r' for each possible pair (T, N) with a manual search of the parameter space so as to minimize communication cost, while targeting a total success probability of at least $1/2$ for one protocol execution. We evaluate the success probability following Theorem 3.1, evaluating ν numerically.

To showcase the efficiency of our scheme, we provide the communication cost for each threshold $2 \leq T \leq N \leq 6$ of Threshold ML-DSA-44 in Table 3, for a single signing attempt with success probability $1/2$. $N \leqslant 6$ is chosen to keep communication low although our analysis could extend to larger N, with communication costs growing quickly, roughly as $\exp\left(O\left(T \cdot \sqrt{\binom{N}{T-1}}\right)\right)$.

For some niche applications, a larger N may be of interest despite the communication overhead, especially when $T \ll N$. For instance, for ML-DSA-44, $N = 7$ requires about 6 MB at worst and 200 kB for $T = 3$, while $N = 8$ requires about 70 MB at worst and 300 kB for T=3, per attempt.

N\T	2	3	4	5	6
2	10.5 kB				
3	15.8 kB	21.0 kB			
4	15.8 kB	36.8 kB	42.0 kB		
5	15.8 kB	73.5 kB	157.4 kB	84.0 kB	
6	21.0 kB	99.8 kB	388.4 kB	524.8 kB	194.2 kB

Table 3: Communication costs per party of Threshold ML-DSA-44 for $2 \leq T \leq N \leq 6$, aiming for a success probability $1/2$. Full parameters are given in Section A .

Performance scaling rationale. Figure 8 shows that costs grow sharply once T or N increases. This behavior is driven by two compounding effects. (i) Let p be the probability a party accepts one local rejection step (for fixed r, r'). All T parties must accept together, so one joint attempt succeeds with probability about p^T; the expected number of sequential retries therefore scales like p^{-T}. Using $K > 1$ parallel repetitions only trades extra bandwidth for lower latency—it cannot remove the inherent p^T factor. (ii) A partial secret may aggregate up to $\lceil \binom{N}{T-1}/T \rceil$ base secrets, inflating its norm (captured in B) and forcing larger radii r, r'. This aggregation overhead vanishes at $T = N$, where each party holds exactly one base secret, explaining the mild improvement in that case. Plugging this asymptotic behavior in the analysis of rejection sampling from [27] leads to a rough asymptotic estimate of our scheme's efficiency as $\exp\left(O\left(T \cdot \sqrt{\binom{N}{T-1}}\right)\right)$.

[3] We verify experimentally that $\|\mathbf{s}c\|$ behaves like a Gaussian distribution and take a bound $1.3 \cdot \|\mathbf{s}\| \cdot \|c\|$ corresponding to 13 standard deviations. If this heuristic were wrong, we would only incur a small loss in the number of queries since the bound on $\|\mathbf{s}c\|$ only affects the Rényi divergence of rejection sampling.

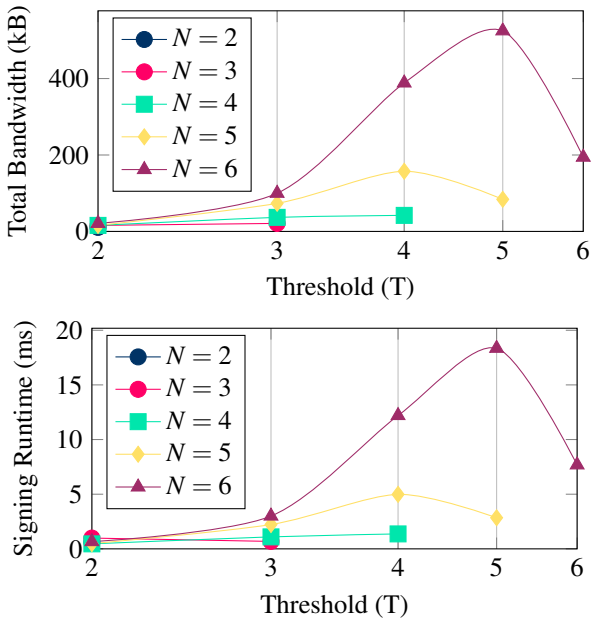

Figure 8: Bandwidth costs (top) and runtime signing costs (bottom) per party per signing attempt for Threshold ML-DSA-44. Note that values for $N = 2$ and $N = 3$ overlap for all T.

We provide the full parameters of Threshold ML-DSA in Section A, including the parameters (r, r', K, ν) for each security level of ML-DSA as well as the communication costs for each threshold $2 \leq T \leq N \leq 6$. The parameters selected ensure a success probability at least $1/2$ for one protocol execution. For Threshold ML-DSA-44, we include a visual representation of the bandwidth and runtime costs in Fig. 8.

4 Performance Evaluation

We provide implementation details and experimental evaluation of our Threshold ML-DSA scheme. We fully implemented our scheme to ease integration with other systems and libraries. We chose `Golang` for its popularity in cryptographic implementations and low incidence of security issues [52, 8]. Our implementation of Threshold ML-DSA builds on the code of the CIRCL library [35], that we extended to support our

threshold variant, including our secret-sharing functionality. We implemented Threshold ML-DSA for all the parameters sets (44, 65, 87) defined in [55].

4.1 Experimental Analysis

We evaluated the performance of Threshold ML-DSA locally (for 44, 65, 87), as well as in LAN and WAN settings (only 44). All parties ran on a consumer-grade MacBook M3 for local and LAN experiments with 24 GB RAM. We conducted single-threaded local simulations of our scheme to benchmark key generation, signing (across parallel rounds), and verification. Each party's computation was emulated in a single process, capturing fine-grained timing data for each protocol phase. The runtime (i.e., computational cost) for various (T, N) values is shown in Table 4, for the parameter set 44 of Threshold ML-DSA, as well as visualizations of its bandwidth and runtime in Fig. 8. We provide the results for the parameter sets 65 and 87 in Section C.

(T, N)	**KeyGen** (ms)	**Sign+Combine** (ms)	**Verify** (ms)
(3,3)	0.3669	0.6810	0.0306†
(2,4)	0.1709	0.4570	
(3,4)	0.2062	1.0961	
(4,4)	0.1655	1.3672	
(3,5)	0.2870	2.2263	
(4,5)	0.2940	4.9832	
(5,5)	0.1956	2.8453	
(4,6)	0.5016	12.1949	
(6,6)	0.2181	7.6784	

Table 4: Local simulation for the average (mean) costs per party of a single signing attempt of Threshold ML-DSA-44. All costs are computed on a MacOS M3 machine and grouped by N. Green and light green indicate the most and second-most optimal cases per group (N) and per scheme, where applicable. †Verification is done using the standard ML-DSA verification algorithm.

For our WAN and LAN experiments, we implemented the communication layer of our scheme using the library `libp2p` [62, 61], a modular peer-to-peer networking stack. Each party acts as an independent `libp2p` peer, with its identity embedded as its network address, and communicates via a custom protocol over a direct TCP stream. Parties listen for incoming connections on a specified TCP port, and others connect using multiaddress. After establishing a connection, they exchange messages over a persistent bidirectional `libp2p` stream. This setup ensures execution over a realistic asynchronous, decentralized network, with `libp2p` handling connection management, peer identity, and stream multiplexing. NAT traversal and relaying were avoided for simplicity, assuming direct TCP connectivity.

LAN experiment We present our results in Table 5. Protocol instances run on separate machines over the same Wi-Fi network. Latency was estimated using an RTT over a TCP connection, yielding 124.166 µs.

(T, N)	**(2,6)**	**(4,6)**	**(6,6)**
Signing (ms)	0.548	18.216	8.683
TCP packets	≈15	≈269	≈137

Table 5: LAN measurements for Threshold ML-DSA-44. All costs are presented per party per signing attempt.

WAN experiment We present our results in Table 6 for Threshold ML-DSA, using a mesh network topology. The protocol ran on Amazon EC2 `"t2.small"` instances (Ubuntu 24.04.02, 2.0 GiB RAM), except for a MacOS M3 machine in Taipei, Taiwan, acting as "leader" and initiating connections. Other machines were distributed across Virginia (USA), London (UK), and Seoul (South Korea). Messages were sent via public IPs and in parallel according to best practices in distributed systems.

(T, N)	**Locations**	**Signing (ms)**
(2,6)	T – S	27.34
(2,6)	T – V	620.43
(4,6)	T – V – L – L	750.65
(6,6)	T – V – L – L – S – S	659.55

Table 6: WAN signing latency (in ms) for one signing attempt of Threshold ML-DSA-44 across different topologies. L = London, S = Seoul, T = Taipei, V = Virginia.

We report in Table 12the median latencies (in ms) between the AWS EC2 instances used in our experiments. WAN measurements reflect the slowest communication path (the "critical path" [30]), consistently between Taipei and Virginia, to capture worst-case delays. In real-world cloud deployments, infrastructure optimizations (e.g., load balancing, anycast routing) may reduce delays and overhead.

The measured signing latencies for Threshold ML-DSA are within milliseconds, even in WAN setting over multiple continents, demonstrating our schemes' practicality in real-world distributed systems. While our scheme require three rounds for a signing operation, these rounds are not executed sequentially in a blocking manner. Instead, messages are sent and processed in parallel across all parties. As a result, the observed latency reflects the most extended communication delay in each round's critical path—usually between the physically furthest machines—rather than the *sum* of point-to-point delays. Our costs are well under a second, far below a typical TLS handshake timeout of 10-20s [45].

4.2 Comparison with Prior Work

In this section, we compare our Threshold ML-DSA scheme with recent works on lattice-based threshold signatures. We include threshold variants of Raccoon: T-Raccoon [23, 46] and Ringtail [10]. We also compare with the recent work on ML-DSA [7], which distributes ML-DSA signing with MPC techniques in the honest-majority setting.

Raccoon threshold variants. Raccoon [25] is a lattice-based signature scheme that has proven particularly amenable to threshold adaptations. By removing the rejection-sampling step present in ML-DSA, it enables highly efficient threshold constructions, at the cost of larger signatures (approximately 10 kB). A notable advantage of these variants is that they scale well to larger thresholds, since the per-party signing bandwidth remains constant across different values of (T, N).

We provide a high-level comparison with T-Raccoon and Ringtail in Table 7, and report local efficiency measurements in Table 8 for NIST level I. Our Threshold ML-DSA scheme has higher round complexity than both T-Raccoon and Ringtail, and exhibits comparable or, in some cases, higher—communication costs for supported threshold sizes. On the other hand, our scheme achieves faster signing times, although this is likely attributable to implementation differences: our implementation builds on the ML-DSA code from CIRCL [35], whereas the Raccoon variants rely on the more general ring-based infrastructure provided by Lattigo [1].

Scheme	$N_{\max}$	**Rounds**	**Comm.**	$\lvert\mathsf{vk}\rvert$	$\lvert\mathsf{sig}\rvert$
Ours	6	6	21–1005	1.3	2.4
T-Raccoon	1024	3	40	3.8	12.4
Ringtail	1024	2	613	4.5	13.4

Table 7: Comparison with T-Raccoon and Ringtail at NIST security level I. All sizes are reported in kB and we report per-party communication sizes. For fairness, we include the *total* communication cost of Ringtail, although its *online* communication reduces to 11 kB. Results for our scheme are averaged over successful signing attempts.

ML-DSA threshold variant. We also compare against the concurrent work of [7], which proposes a threshold variant of ML-DSA in the *honest-majority* setting using MPC techniques. Their construction supports a large number of parties, but this comes at the cost of a high round complexity and potentially significant overhead for generating the required correlated randomness. The authors report benchmarks obtained using SCL [19] on a single thread of a machine with 32 GB of RAM and an Intel i7 processor running at 2.5 GHz. However, no open-source implementation is available, and the prototype appears to rely on extensive low-level optimizations and is implemented in `C++`. In contrast, our scheme is not heavily optimized, is implemented in `Golang`, and includes a fully open-source implementation. Consequently, in Table 9, we report only the performance numbers given in [7], as we are unable to reproduce their benchmarks.

Scheme	(T, N)	**Runtime Signing** (ms)	**Bandwidth Signing** (kB)
Ours	(2, 2)	1.128	21.0
Ringtail	(2, 2)	58.287	633.0
T-Raccoon	(2, 2)	1.787	35.0
Ours	(4, 4)	2.734	84.0
Ringtail	(4, 4)	63.615	633.0
T-Raccoon	(4, 4)	2.106	35.0

Table 8: Comparison of Threshold ML-DSA, T-Raccoon and Ringtail in the local setup (MacOS M3 machine), at NIST level I. Averaged over successful signing for Threshold ML-DSA.

		Runtime (ms)			Comm. (kB)		
Corruptible parties $t^\dagger$		1	2	3	1	2	3
Th. ML-DSA-44	*Ours*	1.1	1.4	2.7	21.0	42.0	84.0
	[7]	13.2	18.7	23.4	416.0	705.0	743.8
Th. ML-DSA-65	*Ours*	1.4	3.6	12.8	45.8	137.0	396.0
	[7]	24.0	31.1	40.3	683.4	1162.0	1218.9
Th. ML-DSA-87	*Ours*	1.7	3.8	9.3	62.2	124.4	290.4
	[7]	24.3	31.2	40.8	701.0	1197.4	1255.1

Table 9: Comparison of Threshold ML-DSA with the scheme of [7] across all NIST levels, reporting online computation (signing + combine) and communication per party averaged over successful signing. For [7], we report their communication-optimised numbers. † As the schemes operate in different security models (dishonest-majority vs. honest-majority), we compare them in the full threshold setting $N = T$, and t denotes the number of corruptible parties in each scheme: for our scheme, it corresponds to $T - 1$, while for [7], it corresponds to $(T - 1)/2$.

4.3 Applications and Discussion

Our threshold ML-DSA implementation addresses critical security gaps in systems currently using classical threshold signatures that must transition to post-quantum cryptography. The performance characteristics make it particularly suitable for replacing existing ECDSA/RSA threshold deployments.

Post-quantum PKI and TLS infrastructure. Modern TLS deployments using Content Delivery Networks (CDNs) face critical security challenges when distributing private key material across edge servers. While traditional approaches like

Keyless SSL [58] proxy private key operations to origin servers, threshold signatures represent an increasingly attractive alternative that avoids single points of failure [41, 14, 28]. As quantum computers threaten classical cryptography, systems must transition to post-quantum alternatives while maintaining the operational benefits of distributed signing.

Our Threshold ML-DSA scheme addresses this transition by providing ML-DSA-compatible threshold signatures that preserve standardized signature formats—ensuring seamless integration with existing infrastructure. A typical 3-of-5 deployment can tolerate two server failures while maintaining signing capability, with our measurements showing subsecond signing latencies across continents, well within TLS timeout requirements of 10-20s [45].

Cryptocurrency wallet security. Multi-device cryptocurrency wallets [34] represent another application where ML-DSA compatibility is essential for quantum-resistant security. Recent work [54] shows that users prefer fewer, highly reputable parties and lower threshold values for higher availability. Our scheme's performance characteristics align well with these preferences (typically 2-of-3 or 3-of-5) while providing a desirable stateless design [51, 36] that avoids storage and synchronization issues.

Importantly, our scheme produces standard ML-DSA signatures, ensuring compatibility with blockchain protocols as they transition to post-quantum cryptography without requiring protocol changes.

Critical infrastructure and Tor network. The Tor network [59] serves millions of users daily and requires fresh random values generated by trusted directory authorities[4]. Current distributed random generation techniques are vulnerable to attacks [57], and proposals for threshold-signed consensus documents [43] have stalled due to the absence of practical, standardized threshold schemes.

Our Threshold ML-DSA implementation directly addresses this gap, providing post-quantum security for the long-term integrity of the Tor network. The scheme operates efficiently in the small-threshold setting (we note that the number of directory authorities is approximately 10 [60]; although this number is subject to change over time, it has historically remained small enough for our scheme to be applicable). Moreover, our experimental results demonstrate sub-second signing latencies across continents, comfortably within the requirements for hourly consensus document generation[5].

[4] See https://spec.torproject.org/rend-spec/shared-random.html.

[5] https://support.torproject.org/glossary/consensus/

5 Acknowledgments

Rafael del Pino, Thomas Espitau, Guilhem Niot and Thomas Prest are supported by the French National Research Agency (ANR), through the project RELATE (reference: ANR-25-CE39-4214-01).

We also thank the anonymous reviewers of USENIX Security 2026 for their valuable feedback.

Ethical Considerations

Stakeholder Analysis We identify the following stakeholders impacted by this research:

Research Community: Our work advances the field by providing the first practical ML-DSA-compatible threshold scheme with formal security guarantees and a complete open-source implementation. This research opens new directions in threshold post-quantum cryptography and may inspire innovations in lattice-based distributed systems and standardization efforts.

System Designers: System designers gain access to practical threshold post-quantum signatures, addressing a critical need as distributed systems transition to quantum-resistant cryptography. Our scheme enables secure and efficient distributed signing protocols while maintaining the security properties of classical distributed systems.

End Users: Individuals benefit from improved security and availability of distributed systems. Preserving these properties in the quantum era is essential, though deploying new cryptographic schemes introduces risks of implementation vulnerabilities.

Adversaries: Nation-state actors and cybercriminals will likely analyze our techniques to develop attacks or countermeasures, as occurs with all published cryptographic research.

Standards Bodies: NIST and other standardization organizations may incorporate our findings into future threshold cryptography standards, influencing security infrastructure development.

Ethical Principles Analysis **Beneficence:** This research benefits *researchers* by advancing threshold post-quantum cryptography with formal guarantees; helps *system designers* maintain distributed security during quantum transition; improves security and availability for *end users*; and provides *standards bodies* valuable input for standardization. Although *adversaries* may analyze our techniques, the security improvements for legitimate users far outweigh potential misuse.

Respect for Persons: We support independent verification by the *research community* through transparent publication and open-source release, while respecting *system designers'* autonomy to decide deployment timing.

Justice: Publishing this work ensures equitable access to our results. Future standardization consideration could enhance this access, enabling broader adoption of secure threshold post-quantum signatures.

Respect for Law and Public Interest: This work complies with research standards and regulations while serving public interest through legitimate security improvements and contributing to standards development that protects users.

Risk Assessment and Mitigation **Potential Harms:**

- *Implementation vulnerabilities:* New cryptographic schemes may contain subtle bugs that lead to security breaches.
- *Cryptanalytic advances:* Future mathematical breakthroughs could compromise our scheme's security.
- *Adversarial misuse:* Attackers may exploit implementation details or side-channel vulnerabilities.

Mitigations Implemented:

- Formal security proofs based on the same assumptions as ML-DSA, ensuring our scheme matches the security of this heavily scrutinized NIST standard.
- Conservative parameter selection based on statistical analysis, reducing the risk of parameter misselection.
- Open-source implementation that enables community audit and verification.

Unmitigated Risks: Residual risks remain that are common to all cryptographic research: potential future cryptanalytic advances and implementation-specific vulnerabilities in production systems.

Decision Rationale We decided to pursue and publish this research after weighing the benefits against the risks. The cryptographic community faces an urgent challenge: existing threshold signatures will become vulnerable to quantum computers, yet these schemes are valuable tools for enhancing user privacy and system security. Organizations seeking to deploy distributed systems—from cryptocurrency platforms to privacy networks like Tor—need quantum-resistant alternatives.

We recognize that publishing novel cryptographic constructions carries risks. Adversaries will scrutinize our techniques, and implementation bugs could lead to vulnerabilities. However, withholding this research would leave critical systems exposed during the quantum transition—a far greater harm. Our formal security proofs, conservative parameters, and open-source implementation provide protection against these risks while enabling community audit and improvement.

Open Science

The implementation artifacts used in this work are available to the community for review and reproducibility. Source code and benchmarking scripts are provided in our public repository at doi.org/10.5281/zenodo.17963721.

Our repository includes: (i) a complete Threshold ML-DSA implementation based on CIRCL [35], (ii) `libp2p` integration for LAN/WAN experiments, (iii) local benchmarking tools, and (iv) parameter selection scripts. All experimental results can be reproduced using the provided artifact.

Note that our implementations are academic proof-of-concept prototypes suitable for research reproduction but requiring additional security review before production use.

References

[1] Lattigo v6. Online: https://github.com/tuneinsight/lattigo, August 2024. EPFL-LDS, Tune Insight SA.

[2] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.

[3] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015.

[4] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 3–24. Springer, Berlin, Heidelberg, November / December 2015.

[5] Henry Bambury, Hugo Beguinet, Thomas Ricosset, and Éric Sageloli. Polytopes in the fiat-shamir with aborts paradigm. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 339–372. Springer, Cham, August 2024.

[6] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022.

[7] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. Efficient, scalable threshold ml-dsa signatures: An mpc approach, 2025.

[8] Jenny Blessing, Michael A. Specter, and Daniel J. Weitzner. Cryptography in the wild: An empirical analysis of vulnerabilities in cryptographic libraries. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '24, page 605–620, New York, NY, USA, 2024. Association for Computing Machinery.

[9] 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.

[10] 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. IEEE Computer Society Press, May 2025.

[11] Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 371–404. Springer, Singapore, December 2023.

[12] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

[13] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security. *Theor. Comput. Sci.*, 939:78–104, 2023.

[14] Sofia Celi, Daniel Escudero, and Guilhem Niot. Share the MAYO: thresholdizing MAYO. Cryptology ePrint Archive, Paper 2024/1960, 2024.

[15] 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, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 268–302. Springer, Singapore, December 2024.

[16] Cloudping. Aws latency monitoring. Cloudping website.

[17] Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 128–153. Springer, Cham, December 2019.

[18] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023.

[19] Anders Dalskov. SCL (Secure Computation Library) — utility library for prototyping MPC applications. https://github.com/anderspkd/secure-computation-library. Accessed: 2025-12-11.

[20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 187–212. Springer, Cham, May 2020.

[21] Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani Saarinen. Raccoon. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[22] Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Simple and efficient lattice threshold signatures with identifiable aborts. Cryptology ePrint Archive, Paper 2025/871, 2025.

[23] 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, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

[24] Rafael del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking TRaccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. Cryptology ePrint Archive, Paper 2025/849, 2025.

[25] 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, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 409–444. Springer, Cham, August 2024.

[26] Rafael del Pino and Guilhem Niot. Finally! a compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography – PKC 2025*, pages 169–199, Cham, 2025. Springer Nature Switzerland.

[27] Julien Devevey, Omar Fawzi, Alain Passelègue, and Damien Stehlé. On rejection sampling in Lyubashevsky's signature scheme. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 34–64. Springer, Cham, December 2022.

[28] Jack Doerner, Yashvanth Kondi, and Leah Namisa Rosenbloom. Sometimes you can't distribute random-oracle-based proofs. NIST presentation, 2023.

[29] Antonín Dufka, Semjon Kravtšenko, Peeter Laud, and Nikita Snetkov. Trilithium: Efficient and universally composable distributed ML-DSA signing. Cryptology ePrint Archive, Paper 2025/675, 2025.

[30] Brian Eaton, Jeff Stewart, Jon Tedesco, and N. Cihan Tas. Distributed latency profiling through critical path tracing: Cpt can provide actionable and precise latency analysis. *Queue*, 20(1):40–79, March 2022.

[31] 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, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024.

[32] 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, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 425–458. Springer, Cham, August 2024.

[33] Thomas Espitau, Alexandre Wallet, and Yang Yu. On gaussian sampling, smoothing parameter and application to signatures. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 65–97. Springer, Singapore, December 2023.

[34] Ittay Eyal. On cryptocurrency wallet design. Cryptology ePrint Archive, Report 2021/1522, 2021.

[35] Armando Faz-Hernandez and Kris Kwiatkowski. *Introducing CIRCL: An Advanced Cryptographic Library*. Cloudflare, June 2019. Available at https://github.com/cloudflare/circl. v1.6.0 Accessed Jan, 2025.

[36] Qi Feng, Kang Yang, Kaiyi Zhang, Xiao Wang, Yu Yu, Xiang Xie, and Debiao He. Stateless deterministic multi-party EdDSA signatures with low communication. Cryptology ePrint Archive, Report 2024/358, 2024.

[37] Marc Fischlin, Aikaterini Mitrokotsa, and Jenit Tomy. BUFFing threshold signature schemes. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 137–168. Springer, Cham, May 2025.

[38] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Threshold RSA for dynamic and ad-hoc groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 88–107. Springer, Berlin, Heidelberg, April 2008.

[39] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork,

editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[40] 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. Springer, Cham, June 2024.

[41] Armando Faz Hernandez. Requirements for threshold tls. NIST presentation, 2023.

[42] Toke Høiland-Jørgensen, Bengt Ahlgren, Per Hurtig, and Anna Brunstrom. Measuring latency variation in the internet. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, page 473–480, New York, NY, USA, 2016. Association for Computing Machinery.

[43] Nicholas Hopper. A threshold signature-based proposal for a shared rng. Tor Dev Mailing List, 2014.

[44] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 53–71. Springer, Cham, 2020.

[45] IBM. Handshake timer, 2023. https://www.ibm.com/docs/en/zos/3.1.0?topic=considerations-handshake-timer. Accessed 06/02/24.

[46] 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, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.

[47] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. Aggregating and thresholdizing hash-based signatures using STARKs. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22*, pages 393–407. ACM Press, May / June 2022.

[48] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Cham, April / May 2018.

[49] 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, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.

[50] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020.

[51] Yashvanth Kondi, Claudio Orlandi, and Lawrence Roy. Two-round stateless deterministic two-party Schnorr signatures from pseudorandom correlation functions. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 646–677. Springer, Cham, August 2023.

[52] Wenqing Li, Shijie Jia, Limin Liu, Fangyu Zheng, Yuan Ma, and Jingqiang Lin. Cryptogo: Automatic detection of go cryptographic api misuses. In *Proceedings of the 38th Annual Computer Security Applications Conference*, ACSAC '22, page 318–331, New York, NY, USA, 2022. Association for Computing Machinery.

[53] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.

[54] Easwar Vivek Mangipudi, Udit Desai, Mohsen Minaei, Mainack Mondal, and Aniket Kate. Uncovering impact of mental models towards adoption of multi-device crypto-wallets. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 3153–3167, New York, NY, USA, 2023. Association for Computing Machinery.

[55] National Institute of Standards and Technology. FIPS 204: Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standard (FIPS) Publication NIST FIPS 204, U.S. Department of Commerce, National Institute of Standards and Technology, August 2024.

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

[57] The Tor Project. Security analysis. Tor Specifications, 2024.

[58] Douglas Stebila and Nick Sullivan. An Analysis of TLS Handshake Proxying . In *2015 IEEE Trustcom/BigDataSE/IPA*, pages 279–286, Los Alamitos, CA, USA, August 2015. IEEE Computer Society.

[59] Inc The Tor Project. Tor metrics, 2024. https://metrics.torproject.org/.

[60] Tor Project. Directory authority expectations. https://community.torproject.org/policies/dir-auth/dir_auth_expectations/, 2023. Accessed: 2025-08-25.

[61] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of ipfs: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 739–752, New York, NY, USA, 2022. Association for Computing Machinery.

[62] Dennis Trautwein, Moritz Schubotz, and Bela Gipp. Introducing peer copy - a fully decentralized peer-to-peer file transfer tool. In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–2, 2021.

A Complete Threshold ML-DSA Parameters

This section provides the complete parameter sets for Threshold ML-DSA, compatible with the different parameter sets of ML-DSA, ML-DSA 44 in Fig. 9, ML-DSA 65 in Fig. 10, and ML-DSA 87 in Fig. 11. These parameter sets are designed to support threshold signatures with up to 6 parties, ensuring efficient signing and verification processes.

(T,N)	r	r'	K	**Comm. per party**
(2, 2)	252778	252833	2	10.5 kB
(2, 3)	310060	310138	3	15.8 kB
(3, 3)	246490	246546	4	21.0 kB
(2, 4)	305919	305997	3	15.8 kB
(3, 4)	279235	279314	7	36.8 kB
(4, 4)	243463	243519	8	42.0 kB
(2, 5)	285363	285459	3	15.8 kB
(3, 5)	282800	282912	14	73.5 kB
(4, 5)	259427	259526	30	157.4 kB
(5, 5)	239924	239981	16	84.0 kB
(2, 6)	300265	300362	4	21.0 kB
(3, 6)	277014	277139	19	99.8 kB
(4, 6)	268705	268831	74	388.4 kB
(5, 6)	250590	250686	100	524.8 kB
(6, 6)	219245	219301	37	194.2 kB

Figure 9: Parameter sets for Threshold ML-DSA 44, aiming for a success probability $1/2$. All sets use the same multiplicative factor $\nu = 3$.

B Balanced partition of the shares for Replicated Secret Sharing

Recall that we aim at computing a balanced partition of the shares among the party during the signing session, that is for a partition $(m_i)_{i\in\mathsf{act}}$ of the secrets, it minimizes $\max_{i\in\mathsf{act}} |m_i|$. As a general solution when fixing $U = \max_{i\in\mathsf{act}} |m_i|$, we can efficiently look for a solution using a max-flow modelization of the problem. We consider a bipartite graph consisting of (i) the users on one side, (ii) the secrets on the other side, and we add an edge between an user and a secret when said user owns that secret. We can solve the above balanced partition problem as follows:

1. We direct the edges from the users to the secrets.
2. We add a flow U entering the user nodes $i \in \mathsf{act}$.
3. We add an exit flow 1 on the secret nodes.

(T,N)	r	r'	K	**Comm. per party**
(2, 2)	501495	501613	3	22.9 kB
(2, 3)	540212	540378	5	38.1 kB
(3, 3)	510387	510504	9	68.5 kB
(2, 4)	540212	540378	6	45.7 kB
(3, 4)	506761	506928	20	152.3 kB
(4, 4)	433594	433711	26	198.0 kB
(2, 5)	552371	552575	8	61.0 kB
(3, 5)	552909	553145	62	472.2 kB
(4, 5)	474331	474535	205	1561.3 kB
(5, 5)	425914	426032	78	594.1 kB
(2, 6)	571208	571412	8	61.0 kB
(3, 6)	536793	537058	95	723.5 kB
(4, 6)	488704	488969	804	6123.3 kB
(5, 6)	461324	461529	1200	9139.2 kB
(6, 6)	414896	415013	250	1904.0 kB

Figure 10: Parameter sets for Threshold ML-DSA 65, aiming for a success probability $1/2$. All thresholds (T,N) within a set use the same multiplicative factor $\nu = 6$.

This is represented in Fig. 12. If the max-problem solution covers all the secrets, then we obtain a partition of maximal weight K. We can find the optimal weight K by testing values in increasing order.

For our concrete parameters, we consider $N \leqslant 6$, for which we can easily verify that the optimal assignations verify $K = \left\lceil \binom{N}{T-1}/T \right\rceil$.

Concrete implementation. For simplicity and efficiency, we do not wish to implement a max-flow solver in the Threshold ML-DSA code. Instead, we first observe that when $T = N$, then the partition is easily obtained as each signing party possesses exactly one secret. For the other cases $2 \leqslant T < N \leqslant 6$, we explicit an optimal solution for $\mathsf{act} = \{1,\ldots,T\}$ and each values (T,N). We obtain a partition for all other possible signing set act by symmetry by permuting the index of parties. The idea is formalized in Algorithm 6.

C Complete Threshold ML-DSA Benchmarks

This section provides the complete benchmarks for parameter sets for Threshold ML-DSA, compatible with the different parameter sets of ML-DSA, ML-DSA 65 in Table 10, and ML-DSA 87 in Table 11.

D Distributed Key Generation

The key generation procedure described in Fig. 5 relies on a trusted dealer that samples all secrets $\mathbf{s}_S$ for the RSS scheme.

(T, N)	r	r'	K	**Comm. per party**
(2, 2)	503119	503192	3	31.1 kB
(2, 3)	631601	631703	4	41.5 kB
(3, 3)	483107	483180	6	62.2 kB
(2, 4)	632903	633006	4	41.5 kB
(3, 4)	551752	551854	11	114.1 kB
(4, 4)	487958	488031	14	145.2 kB
(2, 5)	607694	607820	5	51.9 kB
(3, 5)	577400	577546	26	269.6 kB
(4, 5)	518384	518510	70	725.8 kB
(5, 5)	468214	468287	35	362.9 kB
(2, 6)	665106	665232	5	51.9 kB
(3, 6)	577541	577704	39	404.4 kB
(4, 6)	517689	517853	208	2156.6 kB
(5, 6)	479692	479819	295	3058.6 kB
(6, 6)	424124	424197	87	902.0 kB

Figure 11: Parameter sets for Threshold ML-DSA 87, aiming for a success probability $1/2$. All thresholds (T,N) within a set use the same multiplicative factor $\nu = 7$.

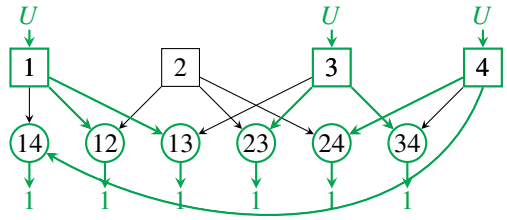

Figure 12: Illustration of the replicated secret sharing with $(N,T) = (4,3)$: users are on top (rectangles), shares at the bottom (circles). Balanced assignment of shares to active users is performed via a max-flow solving algorithm (in green).

This appendix describes how to distribute the key generation process across multiple parties without compromising security.

D.1 Overview

We describe the distributed key generation protocol at a high level. The main requirements for a secure DKG for threshold ML-DSA are: (1) secrets used by honest parties must be sampled from the proper distribution to ensure they remain short enough for safe use in the rejection procedure (c.f. Game 8 of Theorem 3.2), and (2) the final public key must be pseudorandom to maintain unforgeability (c.f. Games 12-13 of Theorem 3.2).

Our distributed key generation protocol eliminates the need for a trusted dealer by having parties collaboratively generate the secrets $\mathbf{s}_S$ through a multi-round commitment-reveal scheme. The key insight is that each subset S of $N-T+1$ parties can collectively generate their corresponding secret $\mathbf{s}_S$ using a shared random oracle, ensuring that no single party learns secrets for groups they don't belong to while maintaining the secret distributions as the centralized approach for honest parties.

The protocol consists of four communication rounds plus a final aggregation step:

Alg. 6 RSSRecover$(\mathsf{act}) \to \mathcal{P}(S)^{\mathsf{act}}$

if $T = N$ **then**
  return $(m_i := (\{i\}))_{i \in \mathsf{act}}$
if $T = 2 \wedge N = 3$ **then**
  $\mathsf{sh} := \{0 : \{01, 02\}, 1 : \{12\}\}$
else if $T = 2 \wedge N = 4$ **then**
  $\mathsf{sh} := \{0 : \{013, 023\}, 1 : \{012, 123\}\}$
else if $T = 3 \wedge N = 4$ **then**
  $\mathsf{sh} := \{0 : \{01, 03\}, 1 : \{12, 13\}, 2 : \{23, 02\}\}$
else if $T = 2 \wedge N = 5$ **then**
  $\mathsf{sh} := \{0 : \{0134, 0234, 0124\}, 1 : \{1234, 0123\}\}$
else if $T = 3 \wedge N = 5$ **then**
  $\mathsf{sh} := \{0 : \{034, 013, 014, 023\}, 1 : \{012, 123, 124, 134\}, 2 : \{234, 024\}\}$
else if $T = 4 \wedge N = 5$ **then**
  $\mathsf{sh} := \{0 : \{01, 03, 04\}, 1 : \{12, 13, 14\}, 2 : \{23, 02, 24\}, 3 : \{34\}\}$
else if $T = 2 \wedge N = 6$ **then**
  $\mathsf{sh} := \{0 : \{02345, 01235, 01245\}, 1 : \{12345, 01234, 01345\}\}$
else if $T = 3 \wedge N = 6$ **then**
  $\mathsf{sh} := \{0 : \{0134, 0124, 0135, 0345, 0125\}, 1 : \{0145, 1345, 1235, 1234, 1245\},$
  $2 : \{0235, 0245, 0234, 0123, 2345\}\}$
else if $T = 4 \wedge N = 6$ **then**
  $\mathsf{sh} := \{0 : \{014, 023, 015, 012, 045\}, 1 : \{135, 134, 125, 145, 124\}, 2 : \{245, 024, 235,$
  $234, 025\}, 3 : \{034, 013, 123, 345, 035\}\}$
else if $T = 5 \wedge N = 6$ **then**
  $\mathsf{sh} := \{0 : \{01, 02, 05\}, 1 : \{12, 13, 15\}, 2 : \{23, 24, 25\}, 3 : \{03, 34, 35\}, 4 : \{45, 04, 14\}\}$
$\phi := \{i : i\}_{i \in [N]}$ ▷ Define a permutation of the user indexes
$i_1 := 0, i_2 := T$
for $j \in [N]$ **do**
  if $j \in \mathsf{act}$ **then**
    $\phi[i_1] = j$
    $i_1 = i_1 + 1$
  else
    $\phi[i_2] = j$
    $i_2 = i_2 + 1$
for $i \in [T]$ **do** ▷ Translate the ideal sharing for act
  $m_{\phi(i)} = \{\phi(u) \mid u \in \mathsf{sh}[i]\}$
return $(m_i)_{i \in \mathsf{act}}$

(T, N)	KeyGen (ms)	Sign+Combine (ms)	Verify (ms)
(3,3)	0.3903	0.2589	
(2,4)	0.3022	1.2584	
(3,4)	0.3721	4.0694	
(4,4)	0.3009	6.4066	
(3,5)	0.5490	14.0362	0.0441
(4,5)	0.5589	45.2436	
(5,5)	0.3623	19.4478	
(4,6)	0.0413	89.8355	
(6,6)	0.0438	41.1715	

Table 10: Local simulation for the average (mean) costs per party of a single signing attempt of Threshold ML-DSA-65. All costs are computed on a MacOS M3 machine and grouped by N. Green and light green indicate the most and second-most optimal cases per group (N) and per scheme, where applicable.

(T, N)	KeyGen (ms)	Sign+Combine (ms)	Verify (ms)
(3,3)	0.3276	1.7690	
(2,4)	0.3830	1.2769	
(3,4)	0.4697	3.8392	
(4,4)	0.3890	4.6442	
(3,5)	0.6257	8.7362	0.0610
(4,5)	0.6235	22.9182	
(5,5)	0.4227	11.4147	
(4,6)	1.0001	61.3065	
(6,6)	0.4749	30.0254	

Table 11: Local simulation for the average (mean) costs per party of a single signing attempt of Threshold ML-DSA-87. All costs are computed on a MacOS M3 machine and grouped by N. Green and light green indicate the most and second-most optimal cases per group (N) and per scheme, where applicable.

1. **Shared secret establishment:** Group leaders establish shared secrets K_S for their respective groups using secure channels, and all parties commit to random strings r_i.
2. **Commitment reveal:** All parties reveal their committed strings to establish global randomness $R = r_1 \| \cdots \| r_N$.
3. **Secret derivation and PK commitment:** Group leaders derive their secrets $\mathbf{s}_S = H_{\text{keygen}}(S, K_S, R)$, compute corresponding partial public keys $\mathbf{t}_S$, and commit to them.
4. **Public key reveal:** Group leaders reveal their partial public keys for verification and aggregation.
5. **Aggregation:** All parties compute the final public key $\mathsf{vk} = (\rho, \mathbf{t}_\top)$ where $\mathbf{t}_\top$ is derived from the sum of all partial public keys.

A key advantage of this approach is that explicit share distribution is unnecessary: each party can directly compute which secrets $\mathbf{s}_S$ they need based on the deterministic group membership structure and the shared secrets they received in round 1.

D.2 Protocol Description

Let $\mathcal{S} = \{S \subset [N] : |S| = N - T + 1\}$ denote the collection of all subsets of size $N - T + 1$. For each $S \in \mathcal{S}$, the parties in S will collectively generate the corresponding secret $\mathbf{s}_S$.

We provide pseudocode in Fig. 13.

D.3 Security Analysis

The security of the distributed key generation protocol relies on the following properties:

Correctness. The protocol produces the same distribution of secret keys as the centralized version for honest parties. Each secret $\mathbf{s}_S$ is generated as $H_{\text{keygen}}(S, K_S, R)$ where K_S is known only to parties in S, and R contains contributions from all parties. As R is sampled with a commit-and-reveal approach, it ensures that $\mathbf{s}_S$ is properly sampled in $\chi_\mathbf{s}$ by the random oracle, even if K_S is maliciously chosen.

Public-key pseudorandomness. The security of Threshold ML-DSA hinges on the pseudorandomness of the final public key vk. At least one group of parties is composed only of honest members, ensuring that their corresponding secret remains hidden for malicious parties. Thus, the corresponding partial keys appears pseudo-random to them. The commit-and-reveal approach for aggregating the final key combined with transcript verification consequently ensures that the final public key vk is also pseudorandom and that adversaries cannot bias it to compromise unforgeability.

D.4 Implementation Considerations

Communication Rounds. The protocol requires exactly 4 communication rounds plus a final local aggregation step: (1) shared secret distribution and commitment broadcast, (2) commitment reveal, (3) transcript signing and partial public key commitment broadcast, (4) transcript verification and partial public key reveal, and (5) local aggregation to compute the final public key. The transcript signing and verification mechanism ensures that all parties agree on the same protocol execution before contributing to the final public key.

Transcript Integrity. Each party signs the full protocol transcript (rounds 1-3) using their long-term signing key in round 4. During aggregation, we verify that everyone signed the same transcript. This prevents attacks where malicious parties would impersonate an honest party and replace their contribution in the protocol to bias the final key.

DKGRound1$(N, T, i) \to (\mathsf{pm}_1^i, \mathsf{st}_i)$

$\mathsf{st}_i \leftarrow \emptyset$
for $S \in \mathcal{S}$ such that $i = \min(S)$ **do**
  Sample $K_S \leftarrow \{0,1\}^{256}$
  for $j \in S \setminus \{i\}$ **do**
    Send K_S securely to party j
  $\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(S, K_S)\}$
Sample $r_i \leftarrow \{0,1\}^{256}$
Compute $c_i = H_{\text{commit}}(i, r_i)$
$\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(r_i, c_i)\}$
return $(\mathsf{pm}_1^i := c_i, \mathsf{st}_i)$

DKGRound2$(\mathsf{pm}_1, i, \mathsf{st}_i) \to (\mathsf{pm}_2^i, \mathsf{st}_i)$

assert{ $(r_i, c_i) \in \mathsf{st}_i$ for some r_i, c_i }
Parse $\mathsf{pm}_1 = (c_j)_{j \in [N]}$
Pick (r_i, c_i) from st_i with $c_i = \mathsf{pm}_1^i$
$\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(\mathsf{pm}_1)\}$
return $(\mathsf{pm}_2^i := r_i, \mathsf{st}_i)$

DKGRound3$(\mathsf{pm}_2, i, \mathsf{st}_i) \to (\mathsf{pm}_3^i, \mathsf{st}_i)$

assert{ $(\mathsf{pm}_1) \in \mathsf{st}_i$ for some pm_1 }
Parse $\mathsf{pm}_1 = (c_j)_{j \in [N]}$ and $\mathsf{pm}_2 = (r_j)_{j \in [N]}$
Assert $\forall j \in [N], c_j = H_{\text{commit}}(j, r_j)$
$R = r_1 \| r_2 \| \cdots \| r_N$
$\rho \leftarrow H_{\text{seed}}(R)$
$\mathbf{A} := \mathsf{ExpandA}(\rho)$
$\mathsf{pm}_3^i \leftarrow \emptyset$
for $S \in \mathcal{S}$ such that $i = \min(S)$ **do** ▷ Party i is leader of group S
  Pick (S, K_S) from st_i
  Compute $\mathbf{s}_S = H_{\text{keygen}}(S, K_S, R)$ where $H_{\text{keygen}} : \mathcal{S} \times \{0,1\}^{256} \times \{0,1\}^{256 \cdot N} \to \chi_{\mathbf{s}}$
  Compute $\mathbf{t}_S := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_S$
  Compute $c_S^{\text{pk}} = H_{\text{commit}}(S, \mathbf{t}_S)$
  $\mathsf{pm}_3^i \leftarrow \mathsf{pm}_3^i \cup \{(S, c_S^{\text{pk}})\}$
  $\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(S, \mathbf{s}_S, \mathbf{t}_S, c_S^{\text{pk}})\}$
return $(\mathsf{pm}_3^i, \mathsf{st}_i)$

DKGRound4$(\mathsf{pm}_3, i, \mathsf{st}_i) \to (\mathsf{pm}_4^i, \mathsf{st}_i)$

Parse $\mathsf{pm}_3 = \bigcup_{j \in [N]} \mathsf{pm}_3^j$
Extract $\{(S, c_S^{\text{pk}}) : S \in \mathcal{S}, \min(S) \in [N]\}$ from pm_3
transcript $\leftarrow \mathsf{pm}_1 \| \mathsf{pm}_2 \| \mathsf{pm}_3$ from st_i
$\mathsf{pm}_4^i \leftarrow \emptyset$
for $(S, c_S^{\text{pk}}) \in \mathsf{pm}_3$ **do**
  Let $j = \min(S)$
  if $i = j$ **then**
    Pick $(S, \mathbf{s}_S, \mathbf{t}_S, c_S^{\text{pk}})$ from st_i
    $\mathsf{pm}_4^i \leftarrow \mathsf{pm}_4^i \cup \{(S, \mathbf{t}_S)\}$
  else
    if $i \in S$ **then**
      Pick (S, K_S) from st_i
      Pick (pm_1) from st_i to get R
      Compute $\mathbf{s}'_S = H_{\text{keygen}}(S, K_S, R)$
      Compute $\mathbf{t}'_S = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}'_S$
      Assert $c_S^{\text{pk}} = H_{\text{commit}}(S, \mathbf{t}'_S)$
$\partial_i \leftarrow \{(S, \mathbf{t}_S) : (S, \mathbf{t}_S) \in \mathsf{pm}_4^i\}$ ▷ Party i's partial output
$\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i^{\text{long}}, \text{transcript} \| \partial_i)$
$\mathsf{pm}_4^i \leftarrow \mathsf{pm}_4^i \cup \{\sigma_i\}$
$\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(\mathsf{pm}_3)\}$
return $(\mathsf{pm}_4^i, \mathsf{st}_i)$

DKGAggregate$(\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4) \to \mathsf{vk}$

assert{ $\forall i, \mathsf{pm}_4^i \neq \perp$ }
Extract $\{\sigma_j : j \in [N]\}$ from pm_4
transcript $\leftarrow \mathsf{pm}_1 \| \mathsf{pm}_2 \| \mathsf{pm}_3$
Parse $\mathsf{pm}_4 = \bigcup_{j \in [N]} \mathsf{pm}_4^j$ where each pm_4^j contains $(S, \mathbf{t}_S)$ and σ_j
for $j \in [N]$ **do**
  $\partial_j \leftarrow \{(S, \mathbf{t}_S) : (S, \mathbf{t}_S) \in \mathsf{pm}_4^j, j = \min(S)\}$
  if $\mathsf{Verify}(\mathsf{vk}_j^{\text{long}}, \text{transcript} \| \partial_j, \sigma_j) = 0$ **then**
    return $\perp$
$\mathbf{t} \leftarrow 0$
for $(S, \mathbf{t}_S) \in \mathsf{pm}_4$ such that $(S, \mathbf{t}_S)$ is not a signature **do**
  Find $(S, c_S^{\text{pk}}) \in \mathsf{pm}_3$
  Assert $c_S^{\text{pk}} = H_{\text{commit}}(S, \mathbf{t}_S)$
  $\mathbf{t} \leftarrow \mathbf{t} + \mathbf{t}_S$
Pick (pm_1) from st_i to get $R = r_1 \| r_2 \| \cdots \| r_N$
$\rho \leftarrow H_{\text{seed}}(R)$
$(\mathbf{t}_\top, \mathbf{t}_\perp) := \mathsf{Power2Round}(\mathbf{t}, d)$
$tr \in \{0,1\}^{256} := H(\rho \| \mathbf{t}_\top)$
return $(\mathsf{vk} := (\rho, \mathbf{t}_\top))$

Figure 13: Distributed key generation for Threshold ML-DSA.

Leader Selection. Each group S has a deterministic leader (the party with smallest index in S). The leader has the role to sample the shared secret of the group, as well as compute the corresponding partial public key, and distribute them.

Shared Secret Distribution. Leaders distribute shared secrets using secure point-to-point channels. This can be implemented using standard authenticated encryption or by leveraging existing secure communication infrastructure.

This distributed approach maintains the same security guarantees as the trusted dealer version while eliminating single

points of failure in key generation, making it suitable for fully decentralized applications.

E A posteriori secret sharing: Details and proofs

E.1 Lattice preliminaries

We denote $\rho_{\sqrt{\Sigma},\mathbf{c}}$ the mass function over $\mathbb{R}^n$ of the Gaussian distribution centered in $\mathbf{c}$: $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp(-(\mathbf{x}-\mathbf{c})^t \cdot \Sigma^{-1} \cdot (\mathbf{x}-\mathbf{c}))$ for $\Sigma \in \mathbb{R}^{n\times n}$ positive definite. For $\sigma \in \mathbb{R}^+$, we note $\rho_\sigma = \rho_{\sqrt{\sigma^2 I},\mathbf{0}}$.

A lattice is a discrete subgroup of $\mathbb{R}^n$. The dual of a lattice L, noted $L^\vee$, is the set $L^\vee = \{\mathbf{x} \in \mathsf{Span}_{\mathbb{R}}(L) \mid \forall \mathbf{v} \in L, \langle x, v\rangle \in \mathbb{Z}\}$. Discretizing the Gaussian mass function $\rho_{\sqrt{\Sigma},\mathbf{c}}$ over L and normalizing to 1, we obtain the discrete Gaussian distribution $D_{L,\sqrt{\Sigma},\mathbf{c}}$. For any lattice L, and any real $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(L)$ is the smallest real $s > 0$ such that $\frac{1}{s^n}\rho_{1/(\sqrt{2\pi}s)}(L^\vee \setminus \{0\}) \leqslant \varepsilon$.

E.2 Sampling the key shares

We first describe in more details how to sample a Gaussian vector from the lattice $\mathcal{L} = \{(\mathbf{s}_1,\ldots,\mathbf{s}_S), \sum \mathbf{s}_i = 0\}$ while minimizing its standard deviation.

Let $A_S = \left\{\mathbf{x} \in \mathbb{Z}^S, \sum x_i = 0\right\}$ be the root lattice of dimension $S-1$, it is clear that $\mathcal{L} = A_S^m$ where $m = 256(k+\ell)$ is the size of the vector $\mathbf{s}$. We are thus interested in computing the smoothing parameter $\eta_\varepsilon(A_n)$. As this is prior work we do not need to use the definition of the smoothing parameter but only some properties that it verifies:

Lemma E.1 (Adapted from Peikert [56], Theorem 3.1). *Let Σ_1, Σ_2 be definite positive such that $\Sigma_b \geqslant \eta_\varepsilon(\mathbb{Z}^n)$, $b \in \{1,2\}$, and $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}^n$. Then for any $\mathbf{x} \in \mathbb{Z}^n$:*

$$\left|\frac{\Pr[X+Y=\mathbf{x}]}{D_{\mathbb{Z}^n,\sqrt{\Sigma_1+\Sigma_2},\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})} - 1\right| \leqslant 3\varepsilon$$

Where $X \leftarrow D_{\mathbb{Z}^n,\sqrt{\Sigma_1},\mathbf{c}_1}$, and $Y \leftarrow D_{\mathbb{Z}^n,\sqrt{\Sigma_2},\mathbf{c}_2}$.

Proof. From [56] Theorem 3.1, we have that $\Pr[X+Y=\mathbf{x}] \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] D_{\mathbb{Z}^n,\sqrt{\Sigma_1+\Sigma_2},\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})$, and for $\varepsilon \leqslant 1/3$ we have $1-3\varepsilon \leqslant \frac{1-\varepsilon}{1+\varepsilon} \leqslant \frac{1+\varepsilon}{1-\varepsilon} \leqslant 1+3\varepsilon$. □

Lemma E.2 (Adapted from [33] Theorem 5). *For any $S > 0$, let $q = \lceil (S-1)/9 \rceil$, $s \geqslant \sqrt{\frac{9}{8}}\eta_\varepsilon(\mathbb{Z}^S)$, and vector $\mathbf{u} \in Span(A_S)$. The algorithm $\mathrm{SampleD}_s$ introduced in [33] is such that for any $\mathbf{v} \in A_S$:*

$$\left|\frac{SampleD_s(\mathbf{v})}{D_{A_S,s,\mathbf{u}}(\mathbf{v})} - 1\right| \leqslant 1 + 2(q+1)\varepsilon$$

Lemma E.3 ([56] Lemma 2.5). *For any $S > 0$, $\varepsilon > 0$, we have:*

$$\eta_\varepsilon(\mathbb{Z}^S) \leqslant \sqrt{\frac{\log(2S(1+1/\varepsilon))}{\pi}}$$

We will also use some additional properties of the Rényi divergence:

Lemma E.4 ([33] Lemma 3). *Let χ and χ' be two distributions with the same support S. If $\exists\delta$ such that $\forall v \in S$:*

$$\left|\frac{\chi(v)}{\chi'(v)} - 1\right| \leqslant \delta$$

Then $\forall \alpha > 1$:

$$R_\alpha(\chi||\chi') \leqslant 1 + \frac{\alpha\delta^2}{2}$$

and

$$R_\infty(\chi||\chi') \leqslant 1 + \delta$$

We now observe that $\mathcal{L} = \left\{(\mathbf{s}_1,\ldots,\mathbf{s}_S) \in \mathcal{R}^{k+\ell}, \sum \mathbf{s}_i = 0\right\} = A_S^m$ since each coordinate can be considered independently. From all these properties we get:

Lemma E.5. *For any $S > 0$, let $q = \lceil (S-1)/9 \rceil$, $s \geqslant \sqrt{\frac{9}{8}}\sqrt{\frac{\log(2S(1+1/\varepsilon))}{\pi}}$, $\mathcal{L} = \left\{(\mathbf{s}_1,\ldots,\mathbf{s}_S)\mathcal{R}^{k+\ell}, \sum \mathbf{s}_i = 0\right\}$. There exists an efficient algorithm $\mathrm{SampleD}_s$ such that, $\forall \mathbf{u} \in Span(\mathcal{L})$, $\forall \alpha \in (1,\infty)$:*

$$R_\alpha(\mathrm{SampleD}_s||D_{\mathcal{L},s,\mathbf{u}}) \leqslant 1 + 4\alpha(q+1)^2\varepsilon^2 \cdot m/2 + o(\varepsilon^2 \cdot m)$$

with $m = 256(k+\ell)$. In particular if we take $\varepsilon = O(\sqrt{m})$ then we get:

$$R_\alpha(\mathrm{SampleD}_s||D_{\mathcal{L},s,\mathbf{u}}) = O(1)$$

Proof. Using the lemmata from this section we directly obtain a similar statement for the lattice A_S:

$$R_\alpha(\mathrm{SampleD'}_s||D_{A_S,s,\mathbf{u}}) \leqslant 1 + 4\alpha(q+1)^2\varepsilon^2/2$$

Where $\mathrm{SampleD'}_s$ is a sampler for A_S. Using the fact that $\mathcal{L} = A_S^m$ and the multiplicativity of Lemma 2.2, we get for $\mathrm{SampleD}_s$ the algorithm that executes m instances of $\mathrm{SampleD'}_s$, and $\mathbf{u} = (\mathbf{u}_1,\ldots,\mathbf{u}_m)$:

$$\begin{aligned} R_\alpha(\mathrm{SampleD}_s||D_{\mathcal{L},s,\mathbf{u}}) &\leqslant \prod R_\alpha(\mathrm{SampleD'}_s||D_{A_S,s,\mathbf{u}_i}) \\ &\leqslant (1 + 4\alpha(q+1)^2\varepsilon^2/2)^m \\ &\leqslant 1 + 4\alpha(q+1)^2\varepsilon^2 \cdot m/2 + o(\varepsilon^2 \cdot m) \end{aligned}$$

□

We now have a sampler that can sample over any coset of $\mathcal{L}$ (securely only once since we use Rényi divergence) for any standard deviation $s \simeq \sqrt{\frac{9}{8}}\sqrt{\frac{\log(2S\sqrt{m})}{\pi}}$ which is much better than what we would get with a statistical distance argument.

This sampler can, however, not be used to directly sample from the key share distribution that we desire because the vectors $(\mathbf{s}_1,\dots,\mathbf{s}_S)$ such that $\sum \mathbf{s}_i = \mathbf{s}$ are not in the span of $\mathcal{L}$. To remedy this we will construct a vector $\mathbf{v} = (\mathbf{v}_1,\dots,\mathbf{v}_S)$ such that $\sum \mathbf{v}_i = \mathbf{s}$, and decompose it into $\mathbf{v} = \mathbf{v}^\top + \mathbf{v}^\perp$, where $\mathbf{v}^\top \in \mathrm{Span}(\mathcal{L})$, and $\mathbf{v}^\perp \in \mathrm{Span}(\mathcal{L})^\perp$. We can then sample $\mathbf{x}$ from $D_{\mathcal{L},s,-\mathbf{v}^\top}$ and output $\mathbf{z} = \mathbf{x} + \mathbf{v}$, which verifies $\sum \mathbf{z}_i = \mathbf{s}$ and $\forall i \in [S], \|\mathbf{z}_i\| \leqslant \left\|\mathbf{z}_i + \mathbf{v}_i^\top\right\| + \left\|\mathbf{v}_i^\perp\right\|$. Since the first component comes from $S_{s,\mathcal{L},0}$ it is distributed exactly as in our original key generation, to keep compactness we thus need to construct $\mathbf{v}$ such that $\forall i \in [S]$ the norm of $\mathbf{v}_i^\perp$ is negligible compared to the first component.

We can observe that the span of the set S of all vectors $\mathbf{v}$ such that $\sum \mathbf{v}_i = \mathbf{s}$ is an hyperplane parallel to $\mathrm{Span}(\mathcal{L})$ and that the vector $(\mathbf{s}/S,\dots,\mathbf{s}/S)$ is in S and orthogonal to $\mathrm{Span}(\mathcal{L})$. From this we get that for any $\mathbf{v} \in S$ (e.g. $(\mathbf{s},0,\dots,0)$) the norm of $\mathbf{v}_i^\perp$ is $\|\mathbf{s}\|/S$, which quickly becomes negligible as $T = \binom{N}{N-T+1}$ grows.

E.3 Security Analysis

We first introduce a generalization of the hint-MLWE problem from [49, 32], where the hint matrix $\mathbf{B}$ can be non integral, and the hints are Gaussian samples centered around the secret part $\mathbf{Bs}$. We prove that it enjoys a similar reduction to MLWE for Gaussian distributed secrets. In fact the problem and proof of [49, 32] can be viewed as specific instantiations of the one presented here.

Definition E.1 (hint-MLWE). *Let k,ℓ,q,m be integers with $m \geqslant (k+\ell)$, χ be a probability distribution over $\mathcal{R}^{k+\ell}$, $\Sigma \in \mathbb{Z}^{mn\times mn}$ be definite positive, and $\mathbf{B} \in \mathcal{R}^{m\times(k+\ell)}$. The advantage of an algorithm $\mathcal{A}$ in solving the hint-$\mathsf{MLWE}_{q,k,\ell,\chi,\sqrt{\Sigma},\mathbf{B}}$ problem is defined as:*

$$\left|\Pr[\mathcal{A}(\mathbf{A},\begin{bmatrix}\mathbf{A} & \mathbf{I}_k\end{bmatrix}\cdot\mathbf{s},\mathbf{z}) = 1] - \Pr[\mathcal{A}(\mathbf{A},\mathbf{u},\mathbf{z}) = 1]\right|$$

where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k\times\ell}, \mathbf{u} \xleftarrow{\$} \mathcal{R}_q^k, \mathbf{s} \leftarrow \chi$, $\mathbf{z} \leftarrow D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{Bs}}$. The hint-MLWE assumption states that any efficient adversary $\mathcal{A}$ has a negligible advantage against hint-MLWE.

In particular if we consider $\mathbf{B}$ limited to integer coefficients, we obtain the hint-MLWE instance of [32].

Lemma E.6. *Let $\sigma > 0$, for any $0 < l < m$, let $\Sigma \in \mathbb{Z}^{nl\times nl}$ be definite positive, let $B \in \mathcal{R}^{m\times l}$ be a matrix of rank nl . The two following distributions are identical:*

$$\left\{(\mathbf{s},\mathbf{z}) \middle| \mathbf{s} \leftarrow D_{\mathcal{R}^l,\sigma,\mathbf{0}}, \mathbf{z} \leftarrow D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{B}\cdot\mathbf{s}}\right\}$$
$$\left\{(\hat{\mathbf{s}},\mathbf{z}) \middle| \mathbf{s} \leftarrow D_{\mathcal{R}^l,\sigma,0}, \mathbf{z} \leftarrow D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{B}\cdot\mathbf{s}}, \hat{\mathbf{s}} \leftarrow D_{\mathcal{R}^l,\sqrt{\widehat{\Sigma}},\hat{\mathbf{c}}}\right\}$$

Where $\widehat{\Sigma}^{-1} = 1/\sigma^2 + \mathbf{B}^T\Sigma^{-1}\mathbf{B}$, and $\hat{\mathbf{c}} = \widehat{\Sigma}B^T\Sigma^{-1}\mathbf{z}$.

Proof. As in [49] we compute the probability of $(\mathbf{u},\mathbf{v}) \in \mathcal{R}^k \times \mathcal{R}^m$:

$$\begin{aligned}\Pr[\mathbf{s}=\mathbf{u},\mathbf{z}=\mathbf{v}] &= \Pr[\mathbf{z}=\mathbf{v}|\mathbf{s}=\mathbf{u}]\cdot\Pr[\mathbf{s}=\mathbf{u}]\\ &= D_{\mathcal{R}^l,\sigma}(\mathbf{u})\cdot D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{Bu}}(\mathbf{v})\\ &\propto D_{\mathcal{R}^l,\sigma}(\mathbf{u})\cdot D_{R^l,\sqrt{\Sigma'},\mathbf{c}'}(\mathbf{u})\\ &\propto D_{R^l,\sqrt{\widehat{\Sigma}},\hat{\mathbf{c}}}(\mathbf{u})\end{aligned}$$

Where $\Sigma'^{-1} = \mathbf{B}^T\Sigma^{-1}\mathbf{B}$ and $\mathbf{c}' = \Sigma'^{-1}\mathbf{B}\Sigma^{-1}\mathbf{v}$. For the transition from the second to third line one can observe by expanding that $(\mathbf{u}-\mathbf{c}')^T\Sigma'^{-1}(\mathbf{u}-\mathbf{c}') = (\mathbf{v}-\mathbf{Bu})^T\Sigma(\mathbf{v}-\mathbf{Bu}) + C(\mathbf{v})$ where C is a term that does not depend on $\mathbf{u}$. The fourth line comes from the fact that the product of two Gaussian PDFs of respective center and covariance (μ_1,Σ_1) and (μ_2,Σ_2) is a Gaussian PDF of parameter $(\mu = \Sigma(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2), \Sigma = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}$ up to constant multiplicative factor.

From this we have that the equality on conditional probabilities $\Pr[\hat{\mathbf{s}}=\mathbf{u}|\mathbf{z}=\mathbf{v}] = \Pr[\mathbf{s}=\mathbf{u}|\mathbf{z}=\mathbf{v}]$ and the result. □

Theorem E.1. *Let k,ℓ,q be integers. Let $\Sigma \in \mathbb{Z}^{n(k+\ell)\times n(k+\ell)}$ be definite positive, let $\mathbf{B} \in \mathcal{R}^{m\times k+\ell}$ be of rank $(k+\ell)n$. Let $\sigma_1 > 0$, $0 < \delta \leqslant 1$, and $\sigma > 0$ be such that $\sigma^{-2} = (1+\delta)\left(\frac{1}{\sigma_1^2} + \frac{\lambda_{\max}(\mathbf{B}^T\mathbf{B})}{\lambda_{\min}(\Sigma)}\right)$ (with $\lambda_{\min}$ and $\lambda_{\max}$ the smallest and largest eigenvalue). For $0 < \varepsilon < 1/3$, if $\delta\sigma > \eta_\varepsilon(\mathbb{Z}^{n(k+\ell)})$ then there exists an efficient reduction from $\mathsf{MLWE}_{q,k,\ell,D_\sigma}$ to hint$-\mathsf{MLWE}_{q,k,\ell,D_\sigma,\Sigma,\mathbf{B}}$ that reduces the advantage by at most $m\cdot 2\varepsilon$.*

Proof. Let $(\mathbf{A},\mathbf{b}) \in \mathcal{R}_q^{(k+\ell)\times k} \times \mathcal{R}_q^k$ be an MLWE instance. We first sample:

$$\begin{aligned}\mathbf{s}_0 &\leftarrow D_{\mathcal{R}^{k+\ell},\sigma,0}\\ \mathbf{z}_0 &\leftarrow D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{Bs}_0}\\ \hat{\mathbf{s}}_0 &\leftarrow D_{\mathcal{R}^{k+\ell},\sqrt{\widehat{\Sigma}-\sigma^2},\hat{\mathbf{c}}}\end{aligned}$$

Where $\widehat{\Sigma}^{-1} = 1/\sigma_1^2 + \mathbf{B}^T\Sigma^{-1}\mathbf{B}$, and $\hat{\mathbf{c}} = \widehat{\Sigma}B^T\Sigma^{-1}\mathbf{s}_0$. We then output $(\mathbf{A},\mathbf{b}+[\mathbf{I}|\mathbf{A}]\cdot\hat{\mathbf{s}}_0,\mathbf{z}_0)$ as a hint-MLWE instance.

First note that since $\sigma^{-2} = \frac{1}{\sigma_1^2} + \frac{\lambda_{\max}(\mathbf{B}^T\mathbf{B})}{\lambda_{\min}(\Sigma)}$ the covariance $\widehat{\Sigma} - \sigma^2\mathbf{I}$ is positive. Now if $\mathbf{b}$ is uniform in $\mathcal{R}_q^k$ then $\mathbf{b}+\hat{\mathbf{s}}_0$ is also uniform. If $\mathbf{b} = \mathbf{As}$ for $\mathbf{s} \leftarrow D_{\mathcal{R}^{k+\ell},\sigma,0}$ then $\mathbf{s}+\hat{\mathbf{s}}_0$ is close to $D_{\mathcal{R}^{k+\ell},\sqrt{\widehat{\Sigma}},\hat{\mathbf{c}}}$ by Lemma E.1 and by Lemma E.6 the distributions:

$$\left\{(\mathbf{A},[\mathbf{A}|\mathbf{I}]\mathbf{s},\mathbf{z}) \middle| \mathbf{s} \leftarrow D_{\mathcal{R}^k,\sigma_1,0}, \mathbf{z} \leftarrow D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{B}\cdot\mathbf{s}}\right\}$$
$$\left\{(\mathbf{A},[\mathbf{A}|\mathbf{I}]\hat{\mathbf{s}},\mathbf{z}) \middle| \mathbf{s} \leftarrow D_{\mathcal{R}^k,\sigma_1,0}, \mathbf{z} \leftarrow D_{\mathcal{R}^m,\sqrt{\Sigma},\mathbf{B}\cdot\mathbf{s}}, \hat{\mathbf{s}} \leftarrow D_{\mathcal{R}^k,\sqrt{\widehat{\Sigma}},\hat{\mathbf{c}}}\right\}$$

are identical. Since the second is the distribution output by the adversary and the first is the hint-MLWE distribution we have the desired result. □

From this reduction we can prove that the hint-MLWE corresponding to our posteriori secret sharing is secure. If $T-1$ parties are corrupted then all but one of the S shares are revealed, corresponding to a hint-MLWE instance with $\mathbf{B} = [1/S,\dots,1/S]^T \in R^{S-1}$, and $\Sigma = \sigma^2_{\text{share}}(\mathbf{I}_{(S-1)(k+\ell)n\times(S-1)(k+\ell)n} - \frac{1}{T}\mathbf{J}_{(S-1)(k+\ell)n\times(S-1)(k+\ell)n})$ with $\mathbf{J}$ the all one matrix, and σ_{share} the standard deviation of the shares. From this we have $\lambda_{\max(\mathbf{B})} = (S-1)/S^2$ and $\lambda_{\min}(\Sigma) = s^2/S$, thus we will reduce to an MLWE instance with parameter

$$\sigma = (1+\delta)^{-1/2}\left(\frac{1}{\sigma^2_{\text{ML-DSA}}} + \frac{S-1}{S\sigma^2_{\text{share}}}\right)^{-1/2}$$

For $\delta > 0$ such that $\delta\sigma \geqslant \eta_\varepsilon(\mathbb{Z}^{n(k+\ell)}) = \sqrt{\frac{\log(n(k+\ell)(1+1/\varepsilon))}{\pi}}$.

Not that the statement of Theorem E.1 has to use the statistical distance as the Rényi divergence does not work with decision problems. However in the proof of unforgeability we can replace this statistical distance with the Rényi divergence by using it before the reduction. We recall the games of Theorem 3.3 in more detail.

- **Game 11'**. We replace the shares of the secret obtained by using the sampler of Section E.2. For $\sigma_{\text{share}} \geqslant \sqrt{\frac{9}{8}}\sqrt{\frac{\log(2S\sqrt{m})}{\pi}}$, the advantage of the adversary is multiplied at most by the Rényi divergence between the two distributions.

$$\mathsf{Adv}^{\mathsf{Game}_1 1}_{\mathcal{A}} \leqslant (1+2\alpha(q+1)^2)\mathsf{Adv}^{\mathsf{Game}_1 1'}_{\mathcal{A}}$$

- **Game 13**. We change the Key generation to use $\mathbf{s} \leftarrow D_{\mathcal{R}^k,\sqrt{\hat{\Sigma}},\hat{\mathbf{c}}}$, by Lemma E.6 this game is identical to the previous one.

- **Game 14**. We replace $\mathbf{s}$ with $\mathbf{s} = \mathbf{s}_A + \mathbf{s}_B$ where $\mathbf{s}_A \leftarrow D_{\mathcal{R}^k,\sqrt{\hat{\Sigma}-\sigma^2},\mathbf{c}}$ and $\mathbf{s}_B \leftarrow D_{\mathcal{R}^k,\sigma,0}$. Since $\sigma \geqslant \delta\sigma \geqslant \eta_\varepsilon(\mathbb{Z}^{(\ell+k)n})$ for $\varepsilon^{-1} = (\ell+k)n$. We have:

$$\mathsf{Adv}^{\mathsf{Game}_1 3}_{\mathcal{A}} \leqslant 4\mathsf{Adv}^{\mathsf{Game}_1 4}_{\mathcal{A}}$$

- **Game 15**. We replace the public key with uniform. Using Theorem E.1:

$$\left|\mathsf{Adv}^{\mathsf{Game}_1 4}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_1 5}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,D_\sigma}}_{\mathcal{B}}$$

F WAN latencies between AWS EC2 instances

Latencies between the machines used in our experiments from 4 are reported in Table 12, based on median round-trip measurements from [16] (consistent with latency variation over the Internet [42]). WAN measurements reflect the slowest communication path (the "critical path" [30]), consistently between Taipei and Virginia, to capture worst-case delays. In real-world cloud deployments, infrastructure optimizations (e.g., load balancing, anycast routing) may reduce delays and overhead.

	Taipei	**London**	**Virginia**	**Seoul**
Taipei		177.82	232.46	14.22
London	177.82		52.07	163.54
Virginia	232.46	52.07		130.24
Seoul	14.22	163.54	130.24	

Table 12: WAN median latencies (in ms) between different AWS EC2 machines, as noted in [16].

G Omitted proof of unforgeability of Threshold ML-DSA

In this section, we fully formalize the unforgeability of Threshold ML-DSA. We provide a complete formal statement and its proof, adapting the proof from [26]. We limit ourselves to the case $K = 1$, as the general case can be easily derived by considering $Q'_s = K \cdot Q_s$, the number of calls to signing queries.

Theorem G.1 (Unforgeability of Threshold ML-DSA). *Formally, let $\mathcal{A}$ be an adversary against the* TS-UF *security of our threshold scheme making Q_s calls to signing oracles, and at most $Q_{\mathsf{H}_{\mathsf{cmt}}}, Q_{\mathsf{H}}$ queries respectively to the random oracles $\mathsf{H}_{\mathsf{cmt}}, \mathsf{H}$. There exist adversaries $\mathcal{B}_{\mathbf{s}}$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$, $\mathcal{B}_{\mathbf{r}}$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ game, $\mathcal{B}_{\mathbf{z}}$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}$ game, and $\mathcal{B}_{\mathsf{UF}}$ against the unforgeability of ML-DSA running in time $\mathcal{T}_{\mathcal{B}_{\mathbf{s}}} \approx \mathcal{T}_{\mathcal{B}_{\mathbf{r}}} \approx \mathcal{T}_{\mathcal{B}_{\mathbf{z}}} \approx \mathcal{T}_{\mathcal{B}_{\mathsf{UF}}} \approx \mathcal{T}_{\mathcal{A}}$ such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathcal{A}} \leqslant{}& \frac{Q_s Q_{\mathsf{H}_{\mathsf{cmt}}}}{q^{nk}} + Q_{\mathsf{H}}Q_s^2 \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn} + Q_{\mathsf{H}}Q_s \cdot 2^{-2\cdot\kappa} \\ &+ \frac{T\cdot Q_s Q_{\mathsf{H}_{\mathsf{cmt}}} + (Q_{\mathsf{H}_{\mathsf{cmt}}}+Q_s)^2}{2^{2\kappa}} + Q_{\mathsf{H}}^2 \cdot 2^{-512} + \mathsf{negl}(\kappa) \\ &+ \left(1+\frac{2\varepsilon}{1-\varepsilon}\right)^{Q_s} \cdot \left(2\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{s}}} + \mathsf{Adv}^{\mathsf{UF}}_{\mathcal{B}_{\mathsf{UF}}}\right) \\ &+ Q_s \cdot \frac{3M-2}{M-1} \cdot \left(\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{r}}} + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{z}}} + \frac{2\varepsilon}{1-\varepsilon}\right)\end{aligned}$$

For completeness, we include the full formal proof of the unforgeability of our Threshold ML-DSA scheme. This is a straightforward adaptation of the proof and hybrids of [26, Theorem 4].

We first recall results from [26] to tightly simulate rejected transcripts in the Fiat-Shamir with Aborts paradigm, and to evaluate rejection probabilities as a function of ε.

Lemma G.1 (Adaptation of [26, Lemma 4]). *Let any $M > 1$, $B > 0, \varepsilon < 1$, $\mathbf{v} \in \mathcal{R}^{k+\ell}$, distributions $\chi_{\mathbf{r}}$, $\chi_{\mathbf{z}}$ over $\mathcal{R}^{k+\ell}$ such that $R^\varepsilon_\infty(\chi_{\mathbf{z}}||\chi_{\mathbf{r}}+\mathbf{v}) \leqslant M$. We have:*

$$\frac{1-\varepsilon}{M} \leqslant \Pr[\mathsf{Rej}(\mathbf{v},\chi_{\mathbf{r}},\chi_{\mathbf{s}},M) \neq \perp] \leqslant \frac{1}{M}$$

$$\delta \leqslant \frac{2\varepsilon}{1-\varepsilon}$$

where $\delta = \Delta(\mathbf{z}|\mathrm{acc}, \chi_{\mathbf{z}})$.

Lemma G.2 ([26, Lemma 5]). *Given a secret* $\mathbf{s}$*, and challenge* c*, note* $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}}$ *the distribution of rejected vectors* $\mathbf{z}_i$ *conditioned on* $(\mathbf{s}, c)$*. Let* $\mathcal{A}$ *be an adversary against the* $\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}}}$ *problem. There exist adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *respectively against the* $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ *and* $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}$ *problems, running in time* $\mathcal{T}_{\mathcal{B}_1} \approx \mathcal{T}_{\mathcal{B}_2} \approx \mathcal{T}_{\mathcal{A}}$ *such that:*

$$\mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}}}}_{\mathcal{A}} \leqslant \frac{1}{p}\cdot\left(\mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}}_{\mathcal{B}_2} + \delta\right)$$

where $p = \Pr\left[\mathsf{Rej}(c\mathbf{s}, \chi_{\mathbf{r}}, \chi_{\mathbf{s}}, M; \mathbf{r} \leftarrow \chi_{\mathbf{r}}) = \perp\right]$ *is the probability of rejection of* $\mathbf{z}$ *conditioned on* $(\mathbf{s}, c)$ *and* δ *is the statistical distance between* $(\mathbf{z}|\mathrm{acc})_{\mathbf{v}=c\cdot\mathbf{s}}$ *and the target distribution* $\chi_{\mathbf{z}}$*.*

Applying Lemma G.1, if $R^{\varepsilon}_{\infty}(\chi_{\mathbf{z}}||\chi_{\mathbf{r}} + c\cdot\mathbf{s}) \leqslant M$*, then* $p \geqslant 1 - \frac{1}{M}$*.*

Proof. We proceed with a series of hybrids starting from the TS-UF game. Throughough this proof, we denote the advantage of an adversary $\mathcal{A}$ against a search game G by $\mathsf{Adv}^{G}_{\mathcal{A}} := \Pr[G(\mathcal{A}) \to 1]$ – in particular, we omit passing κ as input.

Game_1**.** This is the TS-UF game from Fig. 1.

Game_2**.** In this hybrid we defer the computation of the honest $\mathbf{w}_i$ to the second round of the protocol, and program the random oracle to be consistent. We also introduce a flag f_{fail} to explicitly mark additional cases where the adversary loses. Notably, f_{fail} is set to $\top$ when the random oracle is programmed on a previously queried value. This is formalized in Fig. 14.

The view of the adversary $\mathcal{A}$ differs if they called the random oracle of one of the $\mathbf{w}_i$ before round 2 was executed, i.e.

$$\left|\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}\right| \leqslant \Pr[E]$$

where E is the event "$\mathsf{H}_{\mathsf{cmt}}$ was queried on a honest $\mathbf{w}_i$ before round 2 in Game_1".

By union-bound, we can reduce this to a single call to OSign_1.

$$\left|\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}\right| \leqslant \sum_{s=1}^{Q_s} \Pr[E'_s] \tag{5}$$

where E'_s is the event "The $\mathbf{w}_i$ produced by the s-th call to OSign_1 is queried on $\mathsf{H}_{\mathsf{cmt}}$ before round 2 in Game_1".

We denote the above individual probabilities p_s for $s \in [Q_s]$. Formally, we introduce in Fig. 15 an intermediary game Int^s_1 in which $\mathcal{A}$ wins with probability exactly p_s. We also introduce a tweak of Int^s_1, named Int^s_2, where we replace the s-th commitment $\mathbf{w}_i$ by a uniform value in $\mathcal{R}^k_q$.

First, the difference in advantage between Int^s_1 and Int^s_2 reduces to $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$. Formally, we can define an adversary $\mathcal{E}^s$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ problem such that:

$$\left|\mathsf{Adv}^{\mathsf{Int}^s_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Int}^s_2}_{\mathcal{A}}\right| = \left|p_s - \mathsf{Adv}^{\mathsf{Int}^s_2}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{E}^s} \tag{6}$$

Also, we note that in Int^s_2, the probability that one call of $\mathcal{A}$ to the random oracle queries the s-th $\mathbf{w}_i$ is bounded by the min-entropy of $\mathbf{w}_i \xleftarrow{\$} \mathcal{R}^k_q$, i.e. by q^{-nk}. By union bound, $\mathsf{Adv}^{\mathsf{Int}^s_2}_{\mathcal{A}} \leqslant Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot q^{-nk}$.

By combining the above inequality with Eq. (5) and Eq. (6), we conclude:

$$\left|\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}\right| \leqslant Q_s \cdot Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot q^{-nk} + Q_s \cdot \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{B}_1}$$

where $\mathcal{B}_1$ is the adversary $\mathcal{E}^s$ maximizing $\mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{E}^s}$.

Remark 4. *As an useful remark for following hybrids, we note that a given* $\mathbf{w}_i$ *can't be used twice by the same party after* Game_2 *as then the random oracle programming would fail, and the adversary loses.*

Game_3**.** In this hybrid, we sample a challenge c and its seed $\tilde{c}$ in advance during round 2. When the last honest hash cmt_i obtains a corresponding $\mathbf{w}_i$ programmed, we program the values $\mathsf{H}(\mu||\mathbf{w}_1) = \tilde{c}$ and $\mathsf{SampleInBall}(\tilde{c}) = c$. This is formalized in Fig. 16.

The proof for this hybrid is analogous to the previous one. First, these changes do not bias H and $\mathsf{SampleInBall}$ as the sampled challenges c and seeds $\tilde{c}$ are used at most once for programming H and $\mathsf{SampleInBall}$, and are not provided to the adversary before programming. The view of the adversary hence differs only if it queried the random oracle (i) H on some $\mu||\mathbf{w}_1$, where $\mathbf{w}$ are the high bits of $\widehat{\mathbf{w}} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$ before it is programmed in round 2, or (ii) $\mathsf{SampleInBall}$ on $\tilde{c}$ before it is programmed.

First, we can easily bound the probability that $\mathsf{SampleInBall}$ is queried on some $\tilde{c}$ before it is programmed by $Q_{\mathsf{H}} Q_s \cdot 2^{-2\cdot\kappa}$.

Then, we are interested in the probability that H is queried on some $\mu||\mathbf{w}_1$ before it is programmed. For each $s \in Q_s$, we reexpress as an advantage the probability p_s that the s-th $\mathbf{w}_i$ produced during round 2 is used to compute a $\mathbf{w}_1$ that was previously queried on the random oracle H. Formally, we define the game Int^s_3 in Fig. 17, that verifies $p_s = \mathsf{Adv}^{\mathsf{Int}^s_3}_{\mathcal{A}}$. By union bound, we have $\left|\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}\right| \leqslant \sum_{s\in[Q_s]} p_s$.

We additionally define the game Int^s_4 in Fig. 17, where we replace the s-th $\mathbf{w}_i$ by a uniform value in $\mathcal{R}^k_q$. The advantage difference between Int^s_3 and Int^s_4 again reduces to $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$: $\left|\mathsf{Adv}^{\mathsf{Int}^s_3}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Int}^s_4}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{E}^s}$ for an adversary $\mathcal{E}^s$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ problem.

Then, we can see that the probability that $\mathcal{A}$ wins in Int^s_4 can be bounded using the min-entropy of $\mathsf{HighBits}_q(\mathcal{U}(\mathcal{R}^k_q), 2\gamma_2)$.

Game_2

Q_{msg} $\mathsf{UnopenedCmt}[\cdot] := \emptyset$
$f_{\mathsf{fail}} := \bot$
$\mathsf{pp}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathsf{Setup}(\kappa, N, T)$
$\mathsf{corrupt} \leftarrow \mathcal{A}(\mathsf{pp})$
assert$\{\ \mathsf{corrupt} \subseteq [N]\ \}$
assert$\{\ |\mathsf{corrupt}| < T\ \}$
$\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
$(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Sign.Keygen}(\mathsf{pp})$
for $i \in \mathsf{honest}$ **do**
  $\mathsf{st}_i := \varnothing$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H},(\mathsf{OSign}_i(\cdot))_{i\in[K]}}(\mathsf{st}_{\mathcal{A}}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
if $f_{\mathsf{fail}} = \top$ **then**
  return 0
req $\{\ \mathsf{msg} \notin Q_{\mathsf{msg}}\ \}$
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$
return 1

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$
$\mathsf{UnopenedCmt}[(i, \mathsf{cmt}_i)] = \top$
return $(\mathsf{pm}_1^i := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top\ \}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) := \mathsf{pm}_1^i$
▷ Program random oracle
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

Figure 14: Second hybrid of the unforgeability proof of Threshold ML-DSA. If the random oracle is programmed on a previously queried index, consider that the adversary loses, i.e. f_{fail} is set to $\top$. Difference with the previous hybrid are highlighted.

Indeed, the s-th $\mathbf{w}_i$ is sampled uniformly from $\mathcal{R}_q^k$ and ensures that each $\mathbf{w}$ that needs to be programmed follows the distribution $\mathsf{HighBits}_q(\mathcal{U}(\mathcal{R}_q^k), 2\gamma_2)$. As there are at most Q_s $\mathbf{w}$ to check, we obtain the bound $\mathsf{Adv}^{\mathsf{Int}^s_4}_{\mathcal{A}} \leqslant Q_{\mathsf{H}} Q_s \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn}$.

Putting together the different inequalities obtained, we finally deduce the existence of an adversary $\mathcal{B}_2$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ such that:

$$\left|\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}\right| \leqslant Q_{\mathsf{H}} Q_s^2 \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn} + Q_{\mathsf{H}} Q_s \cdot 2^{-2\cdot\kappa} + Q_s \cdot \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{B}_2}$$

Game_4. In this hybrid, we ensure that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions and that the adversary cannot find preimages of hashes cmt_i after passing them to the second signing oracle. We introduce a table Cmt that records every cmt sampled in $\mathsf{H}_{\mathsf{cmt}}$ or in OSign_1, as well as those passed to the second signing oracle. Whenever a new hash cmt is sampled, we ensure that it was not previously added to Cmt, or the challenger sets the flag f_{fail} to $\top$ in order to abort the game. This is formalized in Fig. 18.

The view of the adversary differs in Game_4 if it was previously able to (i) find a pre-image of a cmt_i after passing it to OSign_2, or (ii) if a given cmt was sampled twice.

We can bound the probability of (i) due to the pre-image resistance of the hash function $\mathsf{H}_{\mathsf{cmt}}$, and with an union bound over all the commits passed by the adversary to OSign_2 which gives a bound $Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot T \cdot Q_s \cdot 2^{-2\kappa}$.

As for (ii), we rely on the collision resistance of $\mathsf{H}_{\mathsf{cmt}}$. Since the commitments are sampled uniformly from $\{0,1\}^{2\kappa}$, we can bound the probability of (ii) by $\frac{(Q_{\mathsf{H}_{\mathsf{cmt}}}+Q_s)^2}{2^{2\kappa}}$.

We conclude that,

$$\left|\mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_4}_{\mathcal{A}}\right| \leqslant \frac{Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot T \cdot Q_s + (Q_{\mathsf{H}_{\mathsf{cmt}}} + Q_s)^2}{2^{2\kappa}}$$

Game_5. In this hybrid, we ensure that H on $tr||\mathsf{msg}$ does never return twice the same $\tilde{c}$ for two different messages. This is formalized in Fig. 19.

As for the previous hybrid, this is ensured by the collision resistance of H due to the large space of $\tilde{c}$.

We conclude that,

$$\left|\mathsf{Adv}^{\mathsf{Game}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_5}_{\mathcal{A}}\right| \leqslant Q_{\mathsf{H}}^2 \cdot 2^{-512}$$

Game_6. In this hybrid, we ensure that the challenge used in round 3 is the same as the one sampled in round 2, and we precompute the responses $\mathbf{z}_i$ in advance. This is formalized in Fig. 20.

We want to show that responses are identically distributed in Game_6. This is the case if programmed responses use the same c in both round 2 and round 3, and if they are used at most once.

Now, we can first observe that if the hash checks in round 3 pass, then all the $\mathsf{cmt}_i := \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i)$ are correctly defined. Recall that Game_4 ensured that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions, and that pre-images cannot be found after round 2 is called. Hence, if these checks pass, then it means that in round 2, all the

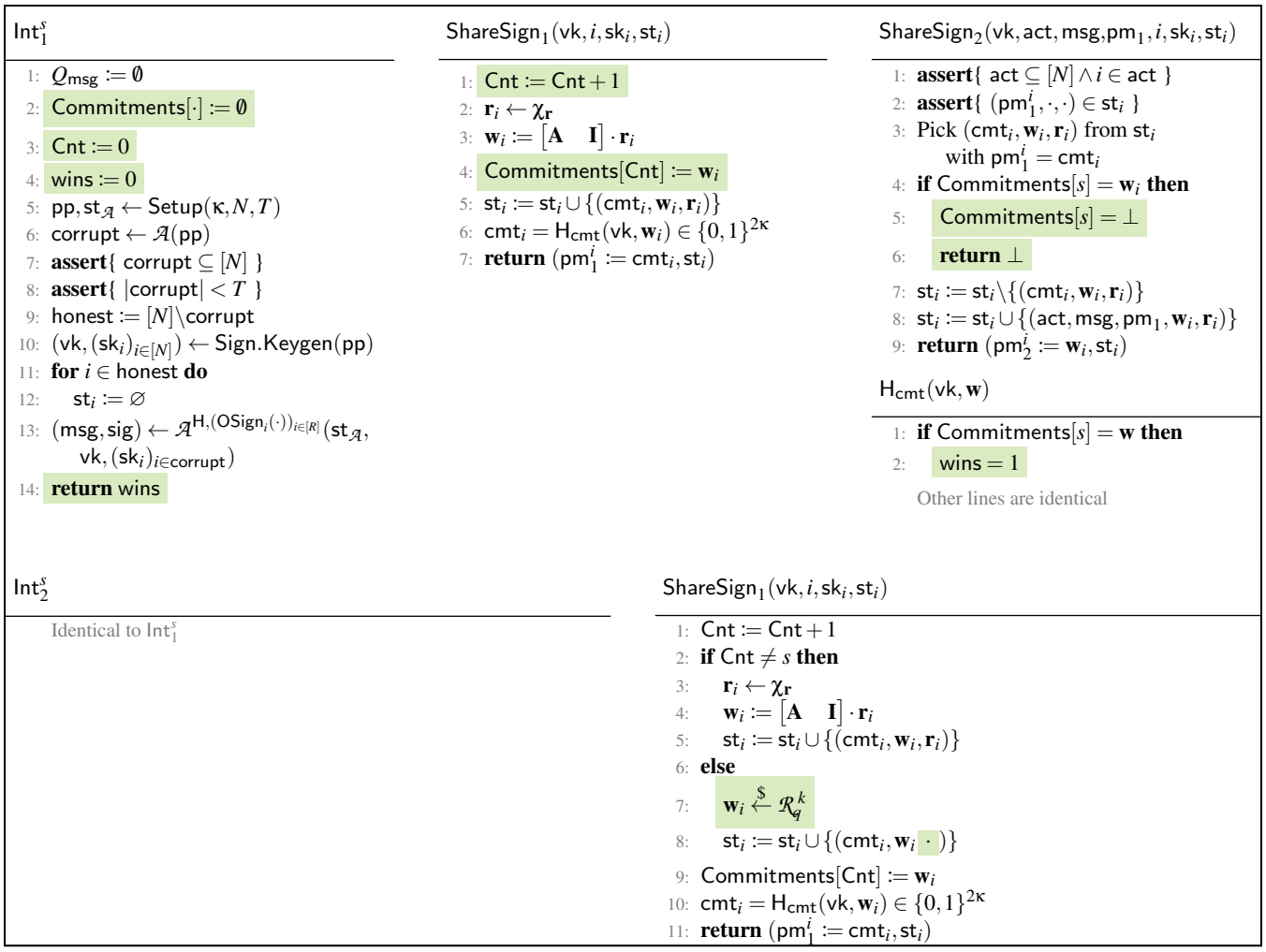

Figure 15: Intermediary games for the second hybrid of the unforgeability proof of Threshold ML-DSA. Difference with Game_1 are highlighted.

hashes cmt_i either: (i) already had pre-images, or (ii) were produced by an honest party in OSign_1 and were waiting for programming. Eventually, all the missing cmt_i from (ii) must thus have been programmed in OSign_2 and as the challenger programs SampleInBall as soon as all the $\mathbf{w}_i$ are defined, then it ensures that the same c is used in round 2 and in round 3.

Additionally, responses are used at most once by a given party due to the collision resistance of H_{cmt} and Remark 4 which ensures that party i will accept cmt_i at most once.

All in all, we conclude that the adversary's view is identically distributed in Game_5 and Game_6 so,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_6} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_5}$$

Game_7. In this hybrid, we no longer sample $\mathbf{r}_i$ out of the rejection sampling. In case the rejection sampling aborts, we sample a fresh $\mathbf{z}_i$ from the aborting distribution $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$. Then, we compute $\mathbf{w}_i$ as $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - \mathbf{t}_i^{\mathsf{part}}$, where $\mathbf{t}_i^{\mathsf{part}} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_i^{\mathsf{part}}$. This is formalized in Fig. 21.

We can see that the view of the adversary is identically distributed. Indeed, in Game_6 we have $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i = c \cdot \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_i^{\mathsf{part}}}_{\mathbf{t}_i^{\mathsf{part}}} + \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i}_{\mathbf{w}_i}$ for $\mathbf{z}_i = \mathbf{r}_i + c \cdot \mathbf{s}_i^{\mathsf{part}}$ (even in case it is rejected). We can thus equivalently write $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{t}_i^{\mathsf{part}}$ and sample $\mathbf{z}_i$ first. The sampling of rejected $\mathbf{z}_i$ from $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$ finally ensures the same distribution of $\mathbf{z}_i$ in the above equality in case of rejections.

So,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_7} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_6}$$

Game_8. In this hybrid, we ensure that the values $c \cdot \mathbf{s}_i^{\mathsf{part}}$ have a twisted norm at most B. That is, whenever this is not the case we set the flag f_{fail} to $\top$ to abort the game. This is

Game_3

$Q_{\mathsf{msg}}, \mathsf{UnopenedCmt}[\cdot] := \emptyset$

$\mathsf{ToProgram}, \mathsf{Programmed}[\cdot] := \emptyset$

Other lines are identical to Game_2

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$

assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top\ \}$

$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$

if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**

  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\kappa}; c \leftarrow \mathcal{C}$

  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$

Pick (a_1, a_2, a_3, c)

from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$

$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$

$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$

$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) := \mathsf{pm}_1^i$

for $(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', c') \in \mathsf{ToProgram}$ **do**

  if $(i \in \mathsf{act}') \wedge (\mathsf{cmt}'_i = \mathsf{pm}_1^i) \wedge (\forall j \in \mathsf{act}' \setminus \{i\}, \exists \mathbf{w}_j, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j))$ **then**

    ▷ Find all the new programmable challenges

    $\mathsf{ToProgram} := \mathsf{ToProgram} \setminus \{(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \tilde{c}', c')\}$

    $\mathbf{w} := \sum_{j\in\mathsf{act}'} \mathbf{w}_j$; $(\mathbf{w}_\top, \mathbf{w}_\bot) = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$

    $\mathsf{H}(\mu || \mathbf{w}_\top) := \tilde{c}'$; $\mathsf{SampleInBall}(\tilde{c}') := c'$ ▷ Programm random oracle

$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$

return $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

Figure 16: Third hybrid of the unforgeability proof of Threshold ML-DSA. If the random oracle is programmed on a previously queried index, consider that the adversary loses, i.e. f_{fail} is set to $\top$. Difference with the previous hybrid are highlighted.

formalized in Fig. 22.

By assumption on B, for any choice of act this is true with overwhelming probability over the randomness of c and of the secrets.

Since the adversary chooses act, it may not be independent of the secrets and we cannot directly conclude. However, we can simply guess act since there are $\binom{N}{T} = \mathsf{poly}(\kappa)$ of them by assumption.

By union-bound over all the calls to a signing oracle, we conclude:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_7} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_8}\right| \leqslant Q_s \cdot \binom{N}{T} \cdot \mathsf{negl}(\kappa)$$

Game_9. In this hybrid, we replace the aborting $\mathbf{w}_i$ by a uniform value in $\mathcal{R}_q^k$. This is formalized in Fig. 23.

There are up to Q_s $\mathbf{w}_i$ to replace. We define a family of games G^p for $p \in [Q_s + 1]$, replacing one by one the $\mathbf{w}_i$ by a uniform vector, which verifies:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_8} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_9}\right| \leqslant \sum_{s\in[Q_s]} \left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right|$$

Let $p \in [Q_s + 1]$. We want to bound $\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right|$.

The core idea is to condition on the value of $v := (\mathsf{act}, i, \mathbf{s}_i^{\mathsf{part}}, c)$ used to define the distribution $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right| \leqslant \sum_{\mathbf{s},c} \left|\mathsf{Adv}_{\mathcal{A}}^{F'^v} - \mathsf{Adv}_{\mathcal{A}}^{F^v}\right| \cdot \Pr[v \text{ used by } G^s] \tag{7}$$

where F'^v and F^v are respectively the games G^{p+1} and G^p where we impose the set of signers act, signer i, and challenge c for the p-th $\mathbf{w}_i$ to replace, and sample the secrets $\mathbf{s}_I$ to be consistent with the value $\mathbf{s}_i^{\mathsf{part}}$.

We can then define adversaries $\mathcal{E}^v$ against the $\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}}$ such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{F'^v} - \mathsf{Adv}_{\mathcal{A}}^{F^v}\right| = \mathsf{Adv}_{\mathcal{E}^v}^{\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}}} \tag{8}$$

Game_8 ensured that the twisted norm of $\mathbf{v} = c \cdot \mathbf{s}_i^{\mathsf{part}}$ is bounded by B which ensures that $R_\infty^\varepsilon(\chi_{\mathbf{z}} || \chi_{\mathbf{r}} + \mathbf{v}) \leqslant M$ by Lemma 2.4. We can thus apply Lemma G.2: there exists adversaries $\mathcal{E}'^v$ and $\mathcal{E}''^v$ such that

$$\mathsf{Adv}_{\mathcal{E}^v}^{\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathsf{rej})\mathbf{v}}} \leqslant \frac{1}{1-\frac{1}{M}} \cdot (\mathsf{Adv}_{\mathcal{E}'^v}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{E}''^v}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}} + \delta^v) \tag{9}$$

where $\mathbf{v} = c \cdot \mathbf{s}_i^{\mathsf{part}}$, δ^v is the statistical distance between $\chi_{\mathbf{z}}$ and $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$.

We can also apply Lemma G.1 to bound $\delta^v \leqslant \frac{2\varepsilon}{1-\varepsilon}$.

By combining Equations 7, 8, and 9, we obtain:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right| \leqslant \frac{M}{M-1} \max_{\mathsf{act},i,\|c\mathbf{s}\|\leqslant B} \left(\mathsf{Adv}_{\mathcal{E}'^v}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{E}''^v}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}}\right) + \frac{M}{M-1} \cdot \frac{2\varepsilon}{1-\varepsilon} + \mathsf{negl}(\kappa)$$

Finally, by summing all the above inequalities, we obtain

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_8} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_9}\right| \leqslant Q_s \frac{M}{M-1} \left(\mathsf{Adv}_{\mathcal{B}_3}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{B}_4}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}}\right) + Q_s \frac{M}{M-1} \cdot \frac{2\varepsilon}{1-\varepsilon} + \mathsf{negl}(\kappa)$$

where $\mathcal{B}_3$ is an adversary against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ problem, $\mathcal{B}_4$ is an adversary against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}$ problem.

Game_{10}. In this hybrid, we replace the real rejection sampling algorithm $\mathsf{Rej}(c\cdot\mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r}_i)$ by an ideal functionality $\mathsf{Ideal}(\chi_{\mathbf{z}}, M)$ which is independent of the secret. This is formalized in Fig. 24.

Int^s_3

$Q_{\mathsf{msg}}, \mathsf{UnopenedCmt}[\cdot], := \emptyset$
$\mathsf{ToProgram} := \emptyset$
$\mathsf{Cnt} := 0$
$\mathsf{wins} := 0$
$f_{\mathsf{fail}} := \bot$
$\mathsf{pp}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathsf{Setup}(\kappa, N, T)$
$\mathsf{corrupt} \leftarrow \mathcal{A}(\mathsf{pp})$
assert$\{\ \mathsf{corrupt} \subseteq [N]\ \}$
assert$\{\ |\mathsf{corrupt}| < T\ \}$
$\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
$(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Sign.Keygen}(\mathsf{pp})$
for $i \in \mathsf{honest}$ **do**
  $\mathsf{st}_i := \varnothing$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OSign}_i(\cdot))_{i\in[R]}}(\mathsf{st}_{\mathcal{A}}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
return wins

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{Cnt} := \mathsf{Cnt} + 1$
assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] = \top\}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot)\}$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$; $\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) := \mathsf{pm}^i_1$
for $(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \cdot) \in \mathsf{ToProgram}$ **do**
  if $(i \in \mathsf{act}') \wedge (\mathsf{cmt}'_i = \mathsf{pm}^i_1) \wedge (\forall j \in \mathsf{act}' \backslash \{i\}, \exists \mathbf{w}_j, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j))$ **then**
    $\mathsf{ToProgram} := \mathsf{ToProgram} \backslash \{(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \tilde{c}', c')\}$
    $\mathbf{w} := \sum_{j\in\mathsf{act}'} \mathbf{w}_j$; $(\mathbf{w}_\top, \mathbf{w}_\bot) = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
    if $\mathsf{Cnt} = s \wedge \exists (\mu || \mathbf{w}_\top) \in \mathsf{H}$ **then** ▷ H already queried on $\mathbf{w}_\top$
      $\mathsf{wins} = 1$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}^i_2 := \mathbf{w}_i, \mathsf{st}_i)$

Int^s_4

Identical to Int^s_3

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{Cnt} := \mathsf{Cnt} + 1$
assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] = \top\}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot)\}$
if $\mathsf{Cnt} = s$ **then** $\mathbf{w}_i \xleftarrow{\$} \mathcal{R}^k_q$
else $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$; $\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
Rest is identical

Figure 17: Intermediary games for the third hybrid of the unforgeability proof of Threshold ML-DSA. Difference with Game_2 are highlighted.

We rely on the Rényi divergence to prove that this only slightly increase the probability that $\mathcal{A}$ wins the game. For conciseness, we will abuse Rényi divergence notations to take a random variable as input, implicitly corresponding to the divergence between the marginal distributions from Game_9 and Game_{10}.

We wish to apply a Rényi divergence argument, and its multiplicativity, to the responses $(\mathbf{z}_j)_{j\in[Q_s]}$ produced in signing oracles. However, these responses all depend on the secrets and are not independent, which prevent a simple application of multiplicativity.

To solve that, we will instead apply the Rényi properties from Lemma 2.2 to the tuple $X = ((\mathsf{sk}_i)_{i\in[N]}, \mathbf{z}_1, ..., \mathbf{z}_{Q_s})$.

By the data processing inequality of the Rényi divergence, since we can see Game_9 and Game_{10} as functions of X, their Rényi divergence is bounded by the Rényi divergence of X.

We then apply the multiplicativity of Rényi divergence Lemma 2.2:

$$R_\infty(\mathsf{Game}_9 || \mathsf{Game}_{10}) \leqslant \underbrace{R_\infty((\mathsf{sk}_i)_{i\in[N]})}_{=1} \cdot \prod_j r_j$$

Game$_4$

1: $Q_{\mathsf{msg}}, \mathsf{UnopenedCmt}[\cdot]$ $\mathsf{Cmt} := \emptyset$

Rest is identical to Game$_3$

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

1: $\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$
2: $\mathsf{UnopenedCmt}[(i, \mathsf{cmt}_i)] = \top$
3: **if** $\mathsf{cmt}_i \in \mathsf{Cmt}$ **then**
4: $\quad f_{\mathsf{fail}} := \top$
5: $\quad$ **return** $\bot$
6: $\mathsf{Cmt} := \mathsf{Cmt} \cup \{\mathsf{cmt}_i\}$
7: **return** $(\mathsf{pm}_1^i := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

1: $\mathsf{Cmt} := \mathsf{Cmt} \cup \{\mathsf{cmt}_j := \mathsf{pm}_1^j\}_{j \in \mathsf{act}}$

The rest is identical

$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w})$

1: **if** $Q_{\mathsf{H}_{\mathsf{cmt}}}[(\mathsf{vk}, \mathbf{w})] = \bot$ **then**
2: $\quad \mathsf{cmt} \xleftarrow{\$} \{0,1\}^{2\kappa}$
3: $\quad Q_{\mathsf{H}_{\mathsf{cmt}}}[(\mathsf{vk}, \mathbf{w})] = \mathsf{cmt}$
4: $\quad$ **if** $\mathsf{cmt}_i \in \mathsf{Cmt}$ **then**
5: $\quad\quad f_{\mathsf{fail}} := \top$
6: $\quad\quad$ **return** $\bot$
7: $\quad \mathsf{Cmt} := \mathsf{Cmt} \cup \{\mathsf{cmt}_i\}$
8: **return** $Q_{\mathsf{H}_{\mathsf{cmt}}}[(\mathsf{vk}, \mathbf{w})] = \bot$

Figure 18: Fourth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

where $r_j = \max_{(\mathsf{sk}_i)_{i \in [N]}} R_\infty(\mathbf{z}_j \mid (\mathsf{sk}_i)_{i \in [N]})$.

We can apply again the multiplicativity of the Rényi divergence:

$$R_\infty(\mathbf{z}_j \mid (\mathsf{sk}_i)_{i \in [N]}) \leqslant \max_{\|\mathbf{v}\| \leqslant B} R_\infty(\mathbf{z}_j \mid c_j \cdot \mathbf{s}_j^{\mathsf{part}} = \mathbf{v})$$

as $\mathbf{z}_j$ only differs between the games when $\|c \cdot \mathbf{s}_j^{\mathsf{part}}\| \leqslant B$.

We finally apply Lemma G.1 to deduce that,

$$R_\infty(\mathsf{Game}_9 || \mathsf{Game}_{10}) \leqslant \left(1 + \frac{\varepsilon}{M-1}\right)^{Q_s}$$

Applying the probability preservation of the Rényi divergence, we get:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_9} \leqslant \left(1 + \frac{\varepsilon}{M-1}\right)^{Q_s} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{10}}$$

Game$_5$

Identical to Game$_4$

$\mathsf{H}(\mathsf{str})$

1: **if** $\mathsf{str} = tr || \mathsf{msg}$ and $Q_{\mathsf{H}}[tr||\mathsf{msg}] = \bot$ **then**
2: $\quad \tilde{c} \leftarrow \{0,1\}^{512}$
3: $\quad Q_{\mathsf{H}}[tr||\mathsf{msg}] = \tilde{c}$
4: $\quad$ **if** $\exists \mathsf{msg}', Q_{\mathsf{H}}[tr||\mathsf{msg}'] = \tilde{c}$ or $(\cdot, \tilde{c}, \cdot) \in \mathsf{ToProgram}$ **then**
5: $\quad\quad f_{\mathsf{fail}} := \top$
6: $\quad\quad$ **return** $\bot$

Rest is identical

Figure 19: Fifth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Game$_{11}$. In this hybrid, we remove the abort condition on the norm of $c \cdot \mathbf{s}_i^{\mathsf{part}}$.

This can only increase the winning probability of the adversary, hence

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{10}} \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{11}}$$

Game$_{12}$. In this hybrid, we replace the public key by the rounding of a uniform value, i.e. $\mathsf{Power2Round}(\mathbf{t}, d)$ where $\mathbf{t} \xleftarrow{\$} \mathcal{R}_q^k$. To preserve consistency, we fix a secret index I such that $I_0 \subseteq \mathsf{honest}$, i.e. $\mathbf{s}_{I_0}$ is not given to the adversary, and we replace $\mathbf{t}_{I_0}$ with $\mathbf{t}_{I_0} := \mathbf{t} - \sum_{I \neq I_0} \mathbf{t}_I$. This is formalized in Fig. 25.

As secrets are no longer used by honest parties in the scheme at this stage, we can replace $\mathbf{t}_{I_0}$ by a uniform value by the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$ assumption. Equivalently, we can then sample $\mathbf{t}$ uniformly at random and define $(\mathbf{t}_\top, \mathbf{t}_\bot) := \mathsf{Power2Round}(\mathbf{t}, d)$ and $\mathbf{t}_{I_0} := \mathbf{t} - \sum_{I \neq I_0} \mathbf{t}_I$.

Formally, there exists an adversary $\mathcal{B}_5$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$ problem such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{11}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{12}}\right| \leqslant \mathsf{Adv}_{\mathcal{B}_5}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}}$$

Game$_{13}$. In this hybrid, we first sample the high bits of the public key $\mathbf{t}_\top$ from an ML-DSA public key, then we sample the full key $\mathbf{t}$ from the distribution $\mathcal{R}_q^k$ conditioned on the value of $\mathbf{t}_\top$. This is formalized in Fig. 26.

We can equivalently sample $(\mathbf{t}, \mathbf{t}_\top)$ in Game_{12} by first sampling $(\mathbf{t}_\top, \cdot) = \mathsf{Power2Round}(\mathcal{U}(\mathcal{R}_q^k, d))$ and then sampling $\mathbf{t}$ conditioned on $\mathbf{t}_\top$. Then, we can see that Game_{13} just replaces $\mathcal{U}(\mathcal{R}_q^k)$ in this distribution by an MLWE sample with secrets sampled in $\chi_{\mathbf{s}}$, which is again indistinguishable under the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$ assumption.

Game$_6$

$Q_{\mathsf{msg}}, \mathsf{UnopenedCmt}[\cdot], \mathsf{Cmt}\ \mathsf{Responses}[\cdot] := \emptyset$

Rest is identical to Game$_4$

ShareSign$_2$ $(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top\ \}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\kappa}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$
$\mathbf{z}_i \leftarrow \mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r}_i)$
$\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
Rest is identical

ShareSign$_3$ $(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{pm}_2, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ (\mathsf{act}, \mathsf{msg}, \cdot, \mathsf{pm}_2^i, \cdot) \in \mathsf{st}_i\ \}$
Pick $(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i \cdot)$ from st_i with $\mathsf{pm}_2^i = \mathbf{w}_i$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$, $\mathsf{pm}_2 = (\mathbf{w}_j)_{j\neq i}$
for $j \in \mathsf{act}$ **do**
  if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$ **then**
    abort ▷ Check hash commit.
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i \cdot)\}$
if $\mathbf{z}_i^1, \mathbf{z}_i^2 := \mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)]$ **then**
  return $(\mathbf{z}_i^1, \mathsf{st}_i)$
else abort

Figure 20: Sixth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Formally, there exists an adversary $\mathcal{B}_6$ against the $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$ problem such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{12}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{13}}\right| \leqslant \mathsf{Adv}_{\mathcal{B}_6}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}}$$

Conclusion. Finally, we reduce Game$_{13}$ to the unforgeability of ML-DSA. We design an adversary $\mathcal{B}_7$ against the unforgeability of ML-DSA, and simulate the view of the adversary in Game$_{13}$.

Game$_7$

Identical to Game$_6$

ShareSign$_2$ $(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top\ \}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\kappa}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$
$\mathbf{z}_i \leftarrow \mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
if $\mathbf{z}_i = \bot$ **then** ▷ if reject
  $\mathbf{z}_i \leftarrow (\mathbf{z}|\mathsf{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$
$\mathbf{t}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{t}_I$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{t}_i^{\mathsf{part}}$
if $\mathbf{z}_i \neq \bot$ **then**
  $\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
Rest is identical

Figure 21: Seventh hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

$\mathcal{B}_7$ takes as input an ML-DSA public key $\mathsf{pk} = (\rho, \mathbf{t}_\perp)$, and random oracles ExpandA, H and SampleInBall.

It does the following changes:

- It programs $\mathsf{ExpandA}(\rho) = \mathbf{A}$.
- It chooses $\mathbf{t}_\top$ and ρ provided by the unforgeability challenger in the threshold Keygen.
- Queries to the random oracles H and SampleInBall are lazily passed to the random oracles provided by the unforgeability challenger instead of honestly sampling their values. Note that in case Game$_{12}$ programs them otherwise, this behavior is not affected.

$\mathcal{B}_7$ forwards any forgery it receives from $\mathcal{A}$ to the ML-DSA unforgeability challenger.

The view of the adversary $\mathcal{A}$ is identically distributed since the ML-DSA random oracles sample images from the same distributions as Game$_{13}$, and the ML-DSA public key is sampled as $(\rho, \mathbf{t}_\top)$ in Game$_{13}$.

Game$_8$

Identical to Game$_7$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top\ \}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\kappa}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$; $(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}_i^{\mathsf{part}}$
if $\|(\mathbf{v}\cdot c\cdot\mathbf{u}_1, c\cdot\mathbf{u}_2)\|_2 > B$ **then**
  $f_{\mathsf{fail}} := \top$
  return $\bot$
$\mathbf{z}_i \leftarrow \mathsf{Rej}(c\cdot\mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
if $\mathbf{z}_i = \bot$ **then** ▷ if reject
  $\mathbf{z}_i \leftarrow (\mathbf{z}|\mathsf{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$
$\mathbf{t}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{t}_I$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}\cdot\mathbf{z}_i - c\cdot\mathbf{t}_i^{\mathsf{part}}$
if $\mathbf{z}_i \neq \bot$ **then**
  $\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$

Rest is identical

Figure 22: Eighth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Then, we can see that a valid forgery in Game_{13} will also be a valid forgery for ML-DSA. Indeed, the verification procedure only depends on ρ and $\mathbf{t}_\top$ which are chosen identical for the ML-DSA signature and the threshold signature. Also, the challenge random oracles coincide on messages for which forgeries are valid (i.e. messages for which no signing oracle was called) as a $\tilde{c}$ programmed by Game_{13} will never be used by the ML-DSA oracles thanks to the condition added in Game_5. Hence,

$$\mathsf{Adv}^{\mathsf{UF}}_{\mathcal{B}_7} = \mathsf{Adv}^{\mathsf{Game}_{13}}_{\mathcal{A}}$$

We finally collect all the bounds to obtain the theorem statement, noting that all the intermediary adversaries have an execution time roughly equal to $\mathcal{T}(\mathcal{A})$. □

Game$_9$

Identical to Game$_8$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert$\{\ \mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}\ \}$
assert$\{\ \mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top\ \}$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\kappa}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$; $(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}_i^{\mathsf{part}}$
if $\|(\mathbf{v}\cdot c\cdot\mathbf{u}_1, c\cdot\mathbf{u}_2)\|_2 > B$ **then**
  $f_{\mathsf{fail}} := \top$
  return $\bot$
$\mathbf{z}_i \leftarrow \mathsf{Rej}(c\cdot\mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
if $\mathbf{z}_i = \bot$ **then**
  $\mathbf{w}_i \xleftarrow{\$} \mathcal{R}_q^k$
else
  $\mathbf{t}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{t}_I$
  $\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}\cdot\mathbf{z}_i - c\cdot\mathbf{t}_i^{\mathsf{part}}$
  $\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$

Rest is identical

Figure 23: Nineth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Game$_{10}$

Identical to Game$_6$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert{ $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$ }
assert{ $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$ }
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j \in \mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\kappa}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j \in \mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j \in \mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I \in m_i} \mathbf{s}_I$; $(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}_i^{\mathsf{part}}$
if $\|(\nu \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 > B$ **then**
  $f_{\mathsf{fail}} := \top$
  return $\bot$
$\mathbf{z}_i \leftarrow \mathsf{Ideal}(\chi_{\mathbf{z}}, M)$
if $\mathbf{z}_i = \bot$ **then**
  $\mathbf{w}_i \xleftarrow{\$} \mathcal{R}_q^k$
else
  $\mathbf{t}_i^{\mathsf{part}} := \sum_{I \in m_i} \mathbf{t}_I$
  $\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$
  $\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j \in \mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$

Rest is identical

Figure 24: Tenth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Game$_{12}$

Identical to Game$_8$

$\mathsf{Keygen}(\kappa, N, T) \to (\mathsf{vk}, \mathsf{sk})$

$\rho \leftarrow \{0,1\}^{256}$
$\mathbf{A} := \mathsf{ExpandA}(\rho)$
for $I \subset [N]$ s.t. $|I| = N - T + 1$ and $I \neq I_0$ **do**
  $\mathbf{s}_I \leftarrow \chi_{\mathbf{s}}$
  $\mathbf{t}_I := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_I$
  ▷ Compute partial public keys
  for $i \in I$ **do** ▷ Distribute the keys
    $\mathsf{sk}_i = \mathsf{sk}_i \cup \{\mathbf{s}_I\}$
$\mathbf{t} \xleftarrow{\$} \mathcal{R}_q^k$
$\mathbf{t}_{I_0} := \mathbf{t} - \sum_{I \neq I_0} \mathbf{t}_I$
$(\mathbf{t}_\top, \mathbf{t}_\bot) := \mathsf{Power2Round}(\mathbf{t}, d)$
$\mathsf{vk} := (\rho, \mathbf{t}_\top)$ ▷ Verification key
$\mathsf{sk} := (tr, \mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{sk})$

Figure 25: Twelfth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Game$_{13}$

Identical to Game$_8$

$\mathsf{Keygen}(\kappa, N, T) \to (\mathsf{vk}, \mathsf{sk})$

$\rho \leftarrow \{0,1\}^{256}$
$\mathbf{A} := \mathsf{ExpandA}(\rho)$
for $I \subset [N]$ s.t. $|I| = N - T + 1$ and $I \neq I_0$ **do**
  $\mathbf{s}_I \leftarrow \chi_{\mathbf{s}}$
  $\mathbf{t}_I := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_I$
  ▷ Compute partial public keys
  for $i \in I$ **do** ▷ Distribute the keys
    $\mathsf{sk}_i = \mathsf{sk}_i \cup \{\mathbf{s}_I\}$
$(\mathbf{s}, \mathbf{e}) \leftarrow \chi_{\mathbf{s}}$
$\widehat{\mathbf{t}} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
$(\mathbf{t}_\top, \cdot) := \mathsf{Power2Round}(\widehat{\mathbf{t}}, d)$
Sample $\mathbf{t}$ from $\mathcal{U}(\mathcal{R}_q^k)$ s.t. $(\mathbf{t}_\top, \cdot) = \mathsf{Power2Round}(\mathbf{t}, d)$
$\mathbf{t}_{I_0} := \mathbf{t} - \sum_{I \neq I_0} \mathbf{t}_I$
$\mathsf{vk} := (\rho, \mathbf{t}_\top)$ ▷ Verification key
$\mathsf{sk} := (tr, \mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{sk})$

Figure 26: Thirteenth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.