Efficient Threshold ML-DSA up to 6 parties

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

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

Abstract

Threshold signatures enable multiple participants to collaboratively produce a digital signature, ensuring both fault tolerance and decentralization. As we transition to the post-quantum era, lattice-based threshold constructions have emerged as promising candidates. However, existing approaches often struggle to scale efficiently and are incompatible with standard schemes — most notably, the NIST-standard ML-DSA.

In this work, we propose the first practical ML-DSA-compatible threshold signature scheme, supporting up to 6 users, with only 3 rounds of communication. It is computationally very efficient, with transcripts computed in at most a few milliseconds, and the communication cost of our protocol ranges from 10.5 kB to 525 kB depending on the threshold used, allowing the full protocol execution in a few hundred milliseconds in a WAN setting, mainly driven by the network latency.

Our work leverages recent results from the signature scheme Finally! (del Pino and Niot, PKC 2025), and performs per-party rejection-based partial signing, before aggregating the response. We introduce several critical optimizations to remain under the parameters of ML-DSA while preserving a high efficiency.

We provide a full implementation and benchmarks of our scheme, demonstrating its practicality and efficiency for real-world deployment.

1 Introduction

Threshold signature schemes (TSS) are an essential cryptographic primitive that allows a group of users to generate signatures collaboratively. In a threshold signature scheme, a private key is distributed among N participants in such a way that any subset of at least T participants (where $T \leqslant N$) can jointly produce a valid signature, while any subset of $T-1$ or fewer participants cannot.

The practical applications of threshold signatures span various domains that require security, decentralization, and fault tolerance. Financial institutions use threshold signatures to safeguard high-value transactions, requiring multiple approvals before transferring funds. Blockchain networks employ these schemes to secure multi-signature wallets and facilitate decentralized consensus mechanisms without single points of failure, typically in the $N = 3, T = 2$ setting. Critical infrastructure systems use them to ensure that sensitive operations require approval from multiple authorized parties, thereby mitigating external attacks and insider threats.

While numerous candidates have been proposed in the classical setting and are competitive in terms of communication cost and efficiency, the situation becomes significantly more complex in the post-quantum setting. Recently, the cryptographic community has focused on lattice-based threshold constructions. As research in this area progresses, *Fiat-Shamir-based signatures* combined with noise flooding have emerged as the baseline for the most promising solutions. Various trade-offs involving the number of users, robustness guarantees, and the capacity to identify corrupted parties during the signing operations have been identified.

As the research landscape evolves, a notable gap persists: the absence of threshold signature schemes that are interoperable with the NIST standard ML-DSA. By interoperability, we mean the resulting threshold

signature should be verifiable using the standard ML-DSA signature scheme. This compatibility is crucial for real-world adoption because it allows threshold signatures to be seamlessly integrated into existing cryptographic infrastructures without necessitating modifications to verification procedures.

1.1 Our Contributions

In this paper, we introduce the *first practical threshold variant of ML-DSA*, based on the Finally! signature scheme [dPN25]. Our scheme supports up to *6 parties* and ensures compatibility with ML-DSA's rejection sampling via a tailored *ad hoc* replicated sharing technique.

Our approach relies on the concept of *short secret-sharing* [dPENP25] schemes for key distribution among participants, and per-party rejection-based signing. Our work further demonstrates the versatility of this class of secret sharing, reinforcing its value and applicability in real-world threshold cryptographic protocols.

We leverage compact distributions based on hyperballs to remain under the parameters of ML-DSA, balancing the degradation of requiring several parties to succeed signing simultaneously. We introduce critical optimizations to support 6 parties including an optimized secret sharing reconstruction (c.f. Alg. 6), an unbalanced hyperball distribution to tackle the hint in ML-DSA (c.f. parameter ν in Table 1), a tailored correctness analysis for ML-DSA's hint mechanism (Theorem 3.1 and Heuristic 1), the introduction of parallel protocol repetitions.

We implement and benchmark our scheme under realistic deployment scenarios in LAN and WAN environments, demonstrating its efficiency and suitability for practical use.

1.2 Related Work

Threshold signature schemes

There exists numerous TSS based on RSA [GHKR08], Schnorr signatures [KG20, CKM23] and ECDSA [CGG+20, CCL+23]. The design of practical schemes in the post-quantum setting however appears more challenging, with most of the literature proposing schemes incompatible with NIST standards or candidates. Proposals leveraged assumptions on lattices, isogenies [DM20], or hash functions [KCLM22]. The recent work [CEN24] proposed a threshold variant of the multivariate standard candidates MAYO and UOV using generic MPC and may achieve reasonable effiency.

Lattice-based Fiat-Shamir threshold schemes

[CS19, DKLS25] proposed to apply generic MPC to thresholdize Dilithium. However, the former is inefficient in computation time and communication, and the latter requires a trusted third party to produce correlated pseudorandomness, deviating from the stronger active threat model. In contrast, TSS based on plain Fiat-Shamir (without rejection sampling) have been proposed recently, where the scheme either relies on threshold FHE [ASY22, GKS24] or on standard tools [DKM+24, EKT24, KRT24, CATZ24] applied on (variants of) the Raccoon scheme [dEK+23], reaching better practicality at the cost of larger signature sizes. Advanced security properties of lattice threshold signatures have also been considered, such as adaptive security [KRT24], robustness [ENP24], identifiable aborts [dPENP25, dPKN+25], and Beyond UnForgeability Features (BUFF) [FMT25].

Improved Secret Sharing

For lattice-based threshold schemes, the secret sharing technique's choice dictates the resulting scheme's characteristics. Recent advances suggest that short secret sharing techniques [dPENP25] enable simpler and more secure constructions [BS23, CATZ24], are naturally compatible with rejection sampling [dPN25], and support identifiable aborts at no additional cost [dPENP25].

1.3 Full version

This work is an excerpt from a more comprehensive paper that also introduces a threshold Raccoon signature scheme featuring an efficient identifiable abort mechanism up to 64 parties, additionally co-authored with Giacomo Borin (IBM Research Europe). For the purpose of this conference, we chose to focus on our threshold variant of ML-DSA. Interested readers are referred to the full version for details on the Raccoon construction.

2 Preliminaries

2.1 General notations

In all this work, we denote by κ the security parameter.

Sets, functions, and distributions. For an integer $N > 0$, we denote $[N] = \{0, \dots, N-1\}$. To denote the *assign* operation, we use $y := f(x)$ when f is deterministic and $y \leftarrow f(x)$ when randomized. We recall that the union $S \cup T$ of two sets writes $S \sqcup T$ when disjoint.

Distributions. We also introduce useful distributions for this work. For a set S that is finite or with a finite measure, we denote $\mathcal{U}(S)$ the *uniform distribution* over S, and shorthand $x \xleftarrow{\$} S$ for $x \leftarrow \mathcal{U}(S)$. Given a real $0 \leqslant p \leqslant 1$, we define the *Bernoulli distribution* $\mathsf{Ber}(p)$ that returns 1 with probability p, and 0 otherwise. The *Binomial distribution* $\mathsf{Bin}(m, p)$ is the sum of m Bernoulli distributions $\mathsf{Ber}(p)$.

Linear algebra. Throughout the work, 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. We extend Gaussian distributions over $\mathcal{R}_{\mathbb{R}}$ using vector coefficient embeddings.

Lattices. We define a lattice as a discrete subgroup of $\mathbb{R}^n$. We recall the definition of the smoothing parameter. The dual of a lattice L, noted $L^\vee$, is the set $L^\vee = \{\mathbf{x} \in \mathsf{Span}_{\mathbb{R}}(L) \mid \forall \mathbf{v} \in L, \langle x, v\rangle \in \mathbb{Z}\}$. For any lattice L, and any real $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(L)$ is the smallest real $s > 0$ such that $\rho_{1/(\sqrt{2\pi}s)}(L^\vee \backslash \{0\}) \leqslant \varepsilon$. We also define a scaled variant $\eta'_\varepsilon(L) := \frac{1}{\sqrt{2\pi}}\eta_\varepsilon(L)$.

2.2 Threshold signatures

We recall the definition of an R-round threshold signature (*without* session identifiers) from [KRT24], as a tuple $\mathsf{TSS} = (\mathsf{Setup}, \mathsf{Keygen}, (\mathsf{Sign}_i)_{i\in[R]}, \mathsf{Combine}, \mathsf{Verify})$.
We denote $\mathsf{act} \subseteq [N]$ 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 $(\mathsf{sk}_i)_{i\in[N]}$.

$\mathsf{Sign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_{r-1}^j)_{j\in\mathsf{act}}, i, \mathsf{sk}_i, \mathsf{st}_i) \to (\mathsf{pm}_r^i, \mathsf{st}_i)_{\{r=1,\cdots,R\}}$**.** For the r-th round, 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 also that one may chose to delay passing $\mathsf{msg}, \mathsf{act}$ to 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 [DKM+24] or adaptively [KRT24], and whether the communication channel is controlled by the adversary [DKM+24], or requires consensus properties [ENP24, dPENP25], and for which messages a forgery is accepted. In this work, we consider the unforgeability notion from [DKM+24] 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. We provide formal definitions of said properties below. Note that static security implies adaptive security for small thresholds (T, N), with a theoretical loss of at most 5 bits up to $N = 6$ parties, as supported by our Threshold ML-DSA construction. Still, this loss only appears to be due to techniques limitations rather than an actual weakness in our case.

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:*

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

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 $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathsf{TSS},\mathcal{A}}(\kappa, N, T) \leqslant \mathsf{negl}(\kappa)$.

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

$Q_{\mathsf{msg}} \coloneqq \varnothing$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, N, T)$
$\mathsf{corrupt}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}(\mathsf{pp})$
assert$\{\ \mathsf{corrupt} \subseteq [N]$ **and** $|\mathsf{corrupt}| < T\ \}$
$\mathsf{honest} \coloneqq [N] \backslash \mathsf{corrupt}$
$(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Sign.Keygen}(\mathsf{pp})$
for $\{\ i \in \mathsf{honest}\ \}$ **do** $\{\ \mathsf{st}_i \coloneqq \varnothing\ \}$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OSign}_i(\cdot))_{i\in[R]}}(\mathsf{st}_{\mathcal{A}}, \quad \mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
req $\{\ \mathsf{msg} \notin Q_{\mathsf{msg}}\ \}$
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

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

req $\{\ i \in [N] \wedge i \in \mathsf{act} \cap \mathsf{honest}\ \}$
$Q_{\mathsf{msg}} \coloneqq Q_{\mathsf{msg}} \cup \{\mathsf{msg}\}$
$(\mathsf{pm}^i_r, \mathsf{st}_i) \leftarrow \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^j_{r-1})_{j\in\mathsf{act}}, i, \mathsf{sk}_i, \mathsf{st}_i)$
return pm^i_r

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. The game returns 0 if a requirement **req** fails. 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:

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

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

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** $\bot$ $\}$
$\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{act}, (\mathsf{pm}^i_r)_{r\in[R], i\in\mathsf{act}})$
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Figure 2: Correctness game for TSS with aborts.

2.3 Modulus Rounding

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

$\mathsf{Power2Round}_q(r)$: 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. *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 [KLS18] 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:*

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

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

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. We define them as in [DFPS22].

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

We additionally recall a relaxed version of the Rényi divergence from [DFPS22, Def. 2.1], termed the *smooth Rényi divergence*, where one can remove a few problematic points from the support, including those that may lie in $\mathsf{Supp}(\mathcal{P}) \backslash \mathsf{Supp}(\mathcal{Q})$.

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

The Renyi divergence provides interesting cryptographic properties, as noted in [BLL+15, HPRR20].

Lemma 2.2 ([BLL+15, Lemma 2.9] and [HPRR20, Prop. 4]). *For two distributions $\mathcal{P}, \mathcal{Q}$, the Renyi divergence satisfies the following properties:*

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

2.6 Rejection Sampling

ML-DSA is a Fiat-Shamir with Aborts signature scheme. It heavily relies on the rejection sampling technique introduced in [Lyu09], which we recall with the same notations as [dPN25].

$\mathbf{Rej}(\mathbf{v}, \chi_\mathbf{z}, \chi_\mathbf{r}, M) \to \mathbf{z} \in \mathcal{R}^{k+\ell} \cup \{\perp\}$	$\mathbf{Ideal}(\chi_\mathbf{z}, M) \to \mathbf{z} \in \mathcal{R}^{k+\ell} \cup \{\perp\}$
1: $\mathbf{r} \leftarrow \chi_\mathbf{r}$ 2: $\mathbf{z} := \mathbf{v} + \mathbf{r}$ 3: $b \leftarrow \mathsf{Ber}\left(\min\left(\frac{\chi_\mathbf{z}(\mathbf{z})}{M\chi_\mathbf{r}(\mathbf{r})}, 1\right)\right)$ 4: **if** $\{\ b = 0\ \}$ **then** $\{\ \mathbf{z} = \perp\ \}$ 5: **return** $\mathbf{z}$	1: $\mathbf{z} \leftarrow \chi_\mathbf{z}$ 2: $b \leftarrow \mathsf{Ber}\left(\frac{1}{M}\right)$ 3: **if** $\{\ b = 0\ \}$ **then** $\{\ \mathbf{z} = \perp\ \}$ 4: **return** $\mathbf{z}$

Figure 3: Rejection algorithm and ideal algorithm.

We extend the algorithm Rej to $\mathrm{Rej}(\mathbf{v}, \chi_\mathbf{z}, \chi_\mathbf{r}, M; \mathbf{r})$ 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 secret – 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.

Lemma 2.3 (Lemma 4.1, [DFPS22]). *Assume that $M > 1$ and $\varepsilon < 1$ are such that $R^{\varepsilon}_{\infty}(\chi_{\mathbf{z}} || \mathbf{v} + \chi_{\mathbf{r}}) \leqslant M$, then, $R_{\infty}(\mathrm{Rej} || \mathrm{Ideal}) \leqslant 1 + \frac{\varepsilon}{M-1}$.*

2.7 Rejection sampling over hyperballs

Devevey et al. [DFPS22] showed that hyperballs-based rejection sampling achieves the most compact parameters and outperforms uniforms [Lyu09] and polytopes [BBRS24]. Hyperballs are as compact 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:*

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

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

We now recall useful definitions and lemmas to bound the smooth Rényi divergence of hyperball-based rejection sampling.

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 ([DFPS22, 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 $r'^2 \geqslant r^2 + \|\mathbf{v}\|_2^2 + 2r\|\mathbf{v}\|_2/\phi$. Then it holds that:*

$$R^{\varepsilon}_{\infty}\left(\mathcal{U}(\mathcal{B}_{\mathcal{R},\ell}(r)) || \mathcal{U}(\mathcal{B}_{\mathcal{R},\ell}(r', \mathbf{v}))\right) = \left(\frac{r'}{r}\right)^{n\ell} \tag{3}$$

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. (3) is upper-bounded by M.

Lemma 2.5 (from [DFPS22, Section A.6]). *For $n > 1, \phi > 1$, we have $I_{1-1/\phi^2}(\frac{n+1}{2}, \frac{1}{2}) < \left(1 - \frac{1}{\phi^2}\right)^{n-1} \cdot n \cdot \left(1 - \frac{1}{\phi}\right)$.*

2.8 Short secret sharings

Secret sharings are useful primitives for designing threshold schemes. Our construction of Threshold ML-DSA relies on the *short secret sharing* notion introduced by del Pino et al. [dPENP25], 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. These properties benefit the design of lattice-based schemes, which will leverage short terms to sample securely or verify signatures. While these constraints inherently partially leak the secret, their core observation was that these schemes often do not need perfect secrecy, and it suffices that the public key $\mathsf{vk} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ remains pseudo-uniform, even after up to $T-1$ shares are corrupted. It notably supports that corrupted shares reveal an offset of the secret key or hints based on the Hint-MLWE assumption [KLSS23] which reduces to MLWE for some parameter regimes.

3 Threshold ML-DSA

This section introduces our variant of ML-DSA (Module-Lattice-Based Digital Signature Algorithm) optimized for small threshold settings. It builds on Finally! [dPN25], a recent three-round threshold scheme offering significantly smaller signatures than earlier work. Yet, Finally! still lags behind ML-DSA's compactness, with 2.7 kB signatures versus ML-DSA's 2.4 kB for few parties. With careful parameter tuning, we show that the techniques in [dPN25] yield a *practical threshold ML-DSA*. Our construction securely distributes ML-DSA signing across parties and further improves [dPN25] by trading communication for higher signature

Table 1: 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 $\pm$ 1's in challenge c
d	Amount of bits dropped from the public key $\mathbf{t}$
η	Secret key range: $\chi_{\mathbf{s}} = \mathcal{U}([-\eta, \eta]^{n(\ell+k)})$
$(\gamma_1, \gamma_2, \beta)$	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) $M = \left(\frac{r'}{r}\right)^{n\ell}$

capacity, enabling smaller parameters while staying ML-DSA-compatible. At a high level, our scheme samples several short secrets and derives a public key as the sum of their corresponding commitments. These secrets are shared among all parties such that any subset of at least T parties can reconstruct them. Signing proceeds via parallel rejection sampling performed independently by each party. While such parallelism could negatively affect the parameters, we mitigate this by employing tighter distributions for the rejection step, which is particularly effective in small-threshold regimes. Hence, our thresholdization exploits the non-tightness in ML-DSA's use of uniform sampling. We operate within 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. To balance a somewhat low success probability per iteration, we run $K > 1$ iterations of the scheme in parallel. That is, during the signing procedure, each party i produces K commitments $\mathbf{w}_{i,j}$, and a response $\mathbf{z}_{i,j}$ for each aggregated commitment $\sum_{i \in \mathsf{act}} \mathbf{w}_{i,j}$.

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 1.

Key Generation. The key generation procedure is detailed in Fig. 4. It builds upon the short replicated secret sharing (RSS) technique introduced in [dPENP25], and involves sampling $\binom{N}{N-T+1}$ ML-DSA secrets from $\chi_{\mathbf{s}}$. Each secret is then given to a distinct set I of $N-T+1$ non-corrupted user. Since there are at most $T-1$ corrupted parties, this ensures that at least one honest party receives each secret. The final public key is derived by summing all secrets and rounding the result as in ML-DSA, yielding a sample from the MLWE distribution. Thanks to at least one honest share per subset, the public key maintains pseudorandomness under the same MLWE assumption as ML-DSA.

Signing. The signing procedure is detailed in Fig. 5. Any set act of T signers collectively knows all the secrets $\mathbf{s}_I$, since for any secret $\mathbf{s}_I$ only $N-(N-T+1) = T-1$ parties do not have said secret. The core idea is to decide on a partition $\bigsqcup_{i \in \mathsf{act}} \mathbf{m}_i = \{I\}_I$ of the secrets among the T parties using a deterministic function RSSRecover, and then run T Fiat-Shamir protocols in parallel to produce a partial signature for each partial secret $\mathbf{s}_i^{\mathsf{part}} = \sum_{I \in \mathbf{m}_i} \mathbf{s}_I$, i.e. partial commitments $\mathbf{w}_i = \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{r}_i$, and signatures $\mathbf{z}_i = c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$.

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

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 4: Key generation procedure of Threshold ML-DSA.

In particular, each party uses a full MLWE sample as commitment $\mathbf{w}_i$ (i.e. $\mathbf{w}_i = \mathbf{A}\mathbf{y} + \mathbf{e}$) 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. To ensure correctness, we verify that the sum of the responses $\mathbf{z} = \sum_{i\in\mathsf{act}} \mathbf{z}_i$ verifies the ML-DSA verification equation $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} = c \cdot \mathbf{t} + \mathbf{w}$: 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 $\mathbf{w} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$.
Following prior works [DKM+24, dPN25], we design a three-round scheme that protects against rushing adversaries on the choice of $\mathbf{w}$:

1. **First round:** parties sample randomness $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$ 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 $\mathbf{w} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$ after proper rejection sampling.

In the complete scheme, 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. We are interested in their sum—retrieved from the public key—: the aggregated commitment $\mathbf{w}$ and the sum of the first part of the responses $\mathbf{z}^{(1)} = \sum_{i\in\mathsf{act}} \mathbf{z}_i^{(1)} \in \mathcal{R}_q^\ell$: $\sum_{i\in\mathsf{act}} \mathbf{z}_i^{(2)} = \mathbf{w} + c \cdot \mathbf{t} - \mathbf{A} \cdot \mathbf{z}^{(1)}$.

Combine and Verify. The combine and verify procedures are detailed in Fig. 6. 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 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 information.

3.2 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

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

1: 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 $\max_{i \in \mathsf{act}} |m_i|$, detailed in Section 3.2.
2: **return** $\mathbf{m}$

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

1: $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
2: $\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
3: $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
4: $\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i) \in \{0,1\}^{2\kappa}$
5: **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: **assert**$\{$ $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$ $\}$
2: **assert**$\{$ $(\mathsf{pm}_1^i, \cdot, \cdot) \in \mathsf{st}_i$ $\}$
3: Pick $(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_1^i = \mathsf{cmt}_i$
4: $\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
5: $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
6: **return** $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

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

1: **assert**$\{$ $(\cdot, \mathsf{pm}_2^i, \cdot) \in \mathsf{st}_i$ $\}$
2: Pick $(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_2^i = \mathbf{w}_i$
3: Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$ and $\mathsf{pm}_2 = (\mathbf{w}_j)_{j \neq i}$
4: **assert**$\{$ $\forall j \in \mathsf{act}, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$ $\}$
5: $\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
6: $\mu \in \{0,1\}^{512} := H(tr || \mathsf{msg})$
7: $\mathbf{w} := \sum_{j \in \mathsf{act}} \mathbf{w}_i$
8: $\mathbf{w}_\top := \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
9: $\tilde{c} \in \{0,1\}^{256} := \mathsf{H}(\mu || \mathbf{w}_\top)$
10: $c := \mathsf{SampleInBall}(\tilde{c})$ ▷ Global challenge
11: $\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
12: $\mathbf{s}_i^{\mathsf{part}} := \sum_{I \in m_i} \mathbf{s}_I$
13: $\mathbf{z}_i \leftarrow \mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r}_i)$ ▷ Individual response
14: **if** $\{$ $\mathbf{z}_i = \perp$ $\}$ **then** $\{$ **abort** $\}$
15: $\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)} := \mathbf{z} \in \mathcal{R}_q^\ell \times \mathcal{R}_q^k$
16: **return** $(\mathsf{pm}_3^i := \mathbf{z}_i^{(1)}, \mathsf{st}_i)$

Figure 5: Signing procedure in Threshold ML-DSA. When an assertion fails, the function returns $\perp$.

as possible in a session for each party. To tackle this, we design a recovery algorithm that computes 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 $i \in \mathsf{act}$.

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

Parse $\mathsf{pm}_2 = (\mathbf{w}_i)_i$ and $\mathsf{pm}_3 = (\mathbf{z}_i^{(1)})_i$
$\mu \in \{0,1\}^{512} := H(tr||\mathsf{msg})$
$\mathbf{w} := \sum_{j\in\mathsf{act}} \mathbf{w}_i$ ▷ Aggregated commitment in $\mathcal{R}_q^k$
$\mathbf{w}_\top := \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
$\tilde{c} := \mathsf{H}(\mu||\mathbf{w}_\top)$ ▷ $\tilde{c} \in \{0,1\}^{256}$
$c := \mathsf{SampleInBall}(\tilde{c})$ ▷ Global challenge
$\mathbf{z}^{(1)} = \sum_{i\in\mathsf{act}} \mathbf{z}_i^{(1)}$
$\boldsymbol{\delta} := \mathbf{w} - \left(\mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top\right)$ ▷ Recovery error in $\mathcal{R}_q^k$
$\mathbf{h} := \mathsf{MakeHint}_q(\boldsymbol{\delta}, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top, 2\gamma_2)$
if $\left(\|\mathbf{z}^{(1)}\|_\infty \geqslant \gamma_1 - \beta\right)$ **or** $\left(\|\boldsymbol{\delta}\|_\infty > \gamma_2\right)$ **or** $\left(\|\mathbf{h}\|_1 > \omega\right)$ **then**
  abort
return $\mathsf{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$

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

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

Figure 6: Combine and verification of Threshold ML-DSA.

3. We add an exit flow 1 on the secret nodes.

This is represented in Fig. 7. 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 $\mathsf{act} = \{1, ..., T\}$ and each values (T, N). We obtain a partition for all other possible signing set act by symmetry by permutting the index of parties. The idea is formalized in Alg. 6.

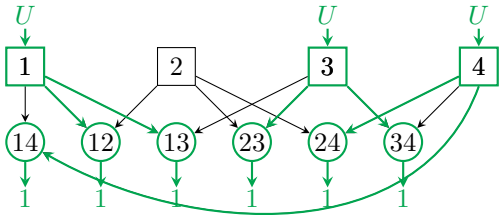

Figure 7: Illustration of the replicated secret sharing with $(N, T) = (4, 3)$: users are on top (rectangles), shares at the bottom (circles). Balanced assignment of shares to active users is performed via a max-flow solving algorithm (in green).

Alg. 6 RSSRecover(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 $(m_i)_{i \in \mathsf{act}}$

3.3 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 RSSRecover and $(\mathbf{u}_1, \mathbf{u}_2) = \sum_{I \in \mathbf{m}_i} \mathbf{s}_I \in \mathcal{R}_q^{\ell+k}$, $\|(\frac{1}{\nu} \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 \leqslant B$ 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 $\|\mathbf{v}\|_2 \leqslant B$,*

$$R_\infty^\varepsilon\left(\chi_{\mathbf{z}} || \chi_{\mathbf{r}} + \mathbf{v}\right) = M$$

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

Theorem 3.1. *Assume* $\Pr[\mathsf{HighBits}(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{w}', 2\gamma_2)] = o(1)$, $\varepsilon = o(1)$. *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 $\mathsf{Game}_0 := \mathsf{Game}^{\mathsf{TS\text{-}corr}}_{\mathsf{TSS}}$. 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 $\|(\frac{1}{\nu} \cdot c \cdot \mathbf{s}_1^{\mathsf{part}}, c \cdot \mathbf{s}_2^{\mathsf{part}})\|_2 \leqslant B$, 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:

$$p_1 \geqslant p_2 - K^2 \cdot (\Pr\left[\mathsf{HighBits}(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{w}', 2\gamma_2)\right] + 2^{-256})$$

where $\mathbf{w}, \mathbf{w}'$ are sampled from $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \sum_{i=1}^{T} \chi_{\mathbf{r}}$, and $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k\times\ell}$.

Game_3. In this game, we sample the challenge c of each session in advance, sample $\mathbf{z}_i \leftarrow \mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$ for each party. If $\mathbf{z}_i = \perp$, we instead sample $\mathbf{z}_i$ from the distribution $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$ 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 "$\forall(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}^{\mathsf{part}}, \|(\frac{1}{\nu} \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 \leqslant B$". 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:

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

Noting that E has the same probability in both games, we obtain,

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

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. Note v_k for $k \in [0, K]$ the probability that in at least one of k sessions we have $\|\mathbf{z}\|_\infty \leqslant \gamma_1 - \beta$ and $\|\boldsymbol{\delta}\|_\infty \leqslant \gamma_2$ and #'s of 1's in $\mathbf{h}$ is less than ω, when $\mathbf{z}$ is sampled from $\sum_{i\in[T]} \chi_{\mathbf{z}}$.
Finally, we can combine the above results to evaluate the final success probability of the protocol: $p_4 = \mathbb{E}_{k \leftarrow \mathsf{Bin}\left(K, (1-\frac{1}{M})^T\right)}[v_k]$ □

For our parameter selection and analysis of our scheme's correctness, we introduce the following heuristic which assumes that the K parallel signing attempts are roughly independent to simplify the evaluation of the success probability of our scheme.

Heuristic 1. *Heuristically, we consider that the rejections in* Combine *are roughly independent so that* $p \approx 1 - \left(1 - \left(1 - \frac{1}{M}\right)^T \cdot v\right)^K$*, where* v *is the probability that one combination succeeds.*

In theory, these attempts are correlated by the dependency on the public key of the rejection of the hint at the end of the combination. However, the parameters of ML-DSA are selected such that this correlation is relatively small. Indeed, in ML-DSA there's a 1-2% chance of rejection due to an invalid hint, while the overall rejection rate is more than 75% [LDK+22, Section 3.4]. This is thus a natural simplification, which will yield very close estimates of the success probability.
We now state the unforgeability of Threshold ML-DSA.

Theorem 3.2 (Unforgeability). *For any parameter* $\phi > 0$*, let* $Q_s = 2/(K \cdot I_{1-1/\phi^2}\left(\frac{n(k+\ell)+1}{2}, \frac{1}{2}\right))$.
Our threshold signature scheme 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 the distributions* $\chi = \chi_{\mathbf{s}}, \chi_{\mathbf{r}}$ *and* $\chi_{\mathbf{z}}$.

We deduce from Theorem 3.2 that breaking the unforgeability of Threshold ML-DSA is as hard as breaking ML-DSA. Indeed, the MLWE assumptions 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 A.1.
Our proof is very similar to the one of [dPN25, Theorem 4], except for two differences: (i) we update the proof to tackle the slightly different rounding of commitments $\mathbf{w}$, and (ii) instead of reducing to STMSIS [KLS18] at the end, we reduce to the unforgeability of ML-DSA. It proceeds with a sequence of games, where the first games in their proof allow them to move to a game the signing oracles return values independent of any secret material. Applying $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$ twice, we then replace the public key with an ML-DSA public key, and as signing oracles do not leak the secret key, we can prove that a forgery in the last game directly provides a forgery for ML-DSA.

Game 2. In this game, we assert that an adversary cannot guess the value of honest commitments $\mathbf{w}_i$ before they are revealed in the second round. This is ensured by their high min-entropy. This allows the game to return a random hash in round 1, and lazily sample $\mathbf{w}_i$ later in the second round while programming the random oracle $\mathsf{H}_{\mathsf{cmt}}$ for consistency.
At a high-level, this game ensures that the adversary is unable to bias the aggregated commitment $\mathbf{w} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$ as honest commitments will be chosen

Game 3-6. In these games, we sample the challenges c in advance during round 2. At a high-level, the pre-image and collision resistance of $\mathsf{H}_{\mathsf{cmt}}$ ensure that the adversary already decided on its commitments $\mathbf{w}_i$ before starting round 2. In particular, with the lazy sampling introduced in the previous game, honest commitments are sampled last. Hence, we can already compute the aggregated commitment $\mathbf{w} = \sum_i \mathbf{w}_i$ in round 2 when sampling the honest commitments $\mathbf{w}_i$, and as it has now a high min-entropy thanks to the honest commitments being sampled last, we can program the random oracle to return the challenge we just sampled without being detected by the adversary.

Game 7. In this game, we compute $\mathbf{z}_i$ first, even in case of rejection and then derive $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}\cdot\mathbf{z}_i - c\cdot\mathbf{t}_i^{\mathsf{part}}$. That is, we first attempt to compute $\mathbf{z}_i = \mathrm{Rej}(c\cdot\mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$, and in case of rejection, we sample it from the distribution of rejection $\mathbf{z}_i$: $\mathbf{z}_i \leftarrow (\mathbf{z}|\mathrm{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$. This is perfectly equivalent to the previous game, and just consists in rewriting distributions.

Game 8. This game ensures that the secret vector $c\cdot\mathbf{s}_i^{\mathsf{part}}$ has a norm smaller than B. This is true with overwhelming probability by assumption.

Game 9. In this game, we replace the commitments $\mathbf{w}_i$ by a uniform value when the corresponding $\mathbf{z}_i$ is rejected and will never be output. We show that this change is indistinguishable for an adversary if the assumptions $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ and $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}$ are hard. This is a direct application of the novel simulation results of [dPN25, Lemma 5].

Games 10-11. In these games, we remove the last dependencies on the secrets by replacing the sampling of $\mathbf{z}_i$ with rejection sampling by a sample obtained using the ideal functionality recalled in Fig. 3. We apply the Renyi divergence results of Section 2.7 to show that up to Q_s this increases the advantage of the adversary by at most 2 bits.

Games 12-13. In this games, we replace the public key of Threshold ML-DSA – which is composed of $\binom{N}{N-T+1}$, by an ML-DSA public key – containing a single secret. Since at this point the signing oracles do no longer depend on secret values, we can simply do so by applying twice the assumption $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{s}}}$.

Conclusion. At this point, any forgery output by the game is a valid forgery against an ML-DSA key, which allows to conclude the proof.

3.4 Parameter selection

To maintain backward compatibility with ML-DSA, we keep the same public parameters listed in Table 1, 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 3.3, 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[1]. 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 3.2. Finally, we select K, r, r' for each possible pair (T,N), targeting a total success probability of at least $1/2$ for one protocol execution. We evaluate the success probability following Heuristic 1, evaluating v numerically.

[1] 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.

As an introduction to showcase the effiency of our sheme, we provide the communication cost for each threshold $2 \leqslant T \leqslant N \leqslant 6$ of Threshold ML-DSA-44 in Table 2.

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 2: Communication costs of Threshold ML-DSA-44 for $2 \leqslant T \leqslant N \leqslant 6$, aiming for a success probability 1/2. Full parameters are given in Section 3.5.

3.5 Complete Threshold ML-DSA Parameters and Figures

In the following, we detail the parameters (r, r', K, ν) as set for every value $2 \leqslant T \leqslant N \leqslant 6$ for each security level of ML-DSA. For Threshold ML-DSA-44, they are in Fig. 8, for Threshold ML-DSA-65 in Fig. 9a, and for Threshold ML-DSA-87 in Fig. 9b. Note that we also provide a visual representation for the bandwidth and runtime costs of Threshold ML-DSA-44 in Fig. 10.

(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)	31060	310138	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 8: Parameter sets for Threshold ML-DSA 44, aiming for a success probability 1/2. All sets use the same multiplicative factor $\nu = 3$.

4 Performance Evaluation

We provide implementation details and experimental evaluation of our Threshold ML-DSA scheme.[2]

[2]Our implementations can be found at https://github.com/GuilhemN/threshold-ml-dsa.

(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

(a) Parameters for Threshold ML-DSA 65, with $\nu = 6$.

(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

(b) Parameters for Threshold ML-DSA 87, with $\nu = 7$.

Figure 9: Parameter sets for Threshold ML-DSA 65 and 87, aiming for a success probability 1/2. All thresholds (T, N) within a set use the same multiplicative factor ν.

Implementation 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 [LJL+22, BSW24]. Our implementation of Threshold ML-DSA builds on the code of the CIRCL library [FHK19], that we extended to support our threshold variant, including our secret-sharing functionality.

4.1 Experimental Analysis

We evaluated the performance of Threshold ML-DSA locally, as well as in LAN and WAN settings. All parties ran on a consumer-grade MacBook M3 for local and LAN experiments with 25 GB RAM. We conducted single-threaded local simulations of both schemes 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 3, for security level set to 44 for Threshold ML-DSA. Visualizations of bandwidth and runtime for Threshold ML-DSA-44 appear in Fig. 10.
For our WAN and LAN experiments, we implemented the communication layer of our scheme using the library `libp2p` [TSG21, TRT+22], 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 4. 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.

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

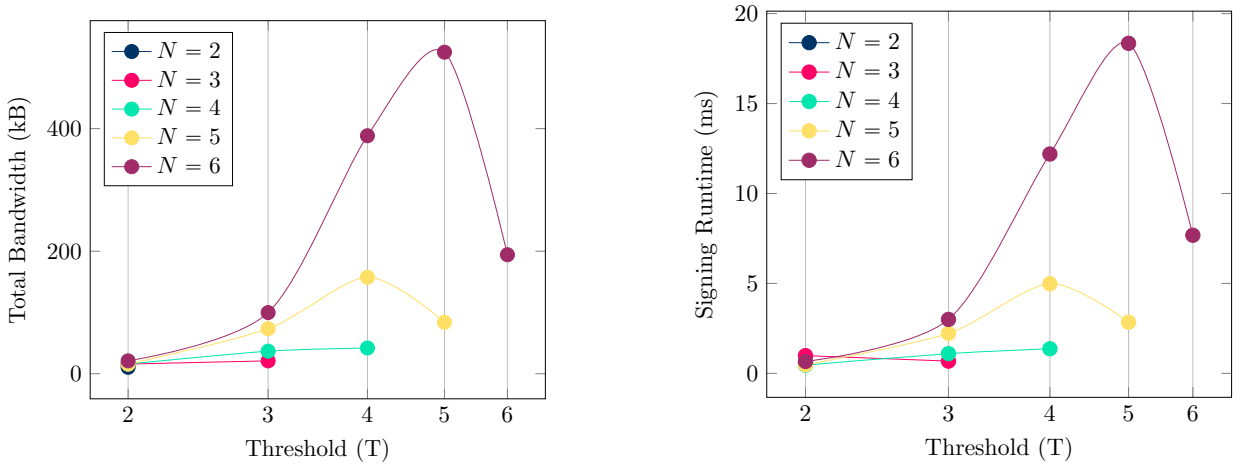

Figure 10: Bandwidth costs (left) and runtime signing costs (right) for Threshold ML-DSA-44. Note that the values for $N = 2$ and $N = 3$ overlap for all T.

(T, N)	**KeyGen** (ms)	**Sign+Combine** (ms)	**Verify** (ms)
(3,3)	0.3669	0.6810	
(2,4)	0.1709	0.4570	
(3,4)	0.2062	1.0961	
(4,4)	0.1655	1.3672	
(3,5)	0.2870	2.2263	0.0306
(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 3: Local simulation for the average (mean) costs 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.

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.
Latencies between machines are reported in Table 5, based on median round-trip measurements from [Clo] (consistent with latency variation over the Internet [HJAHB16]). WAN measurements reflect the slowest communication path (the "critical path" [ESTT22]), consistently between Taipei and Virginia, to capture worst-case delays. In real-world cloud deployments, infrastructure optimizations (e.g., load balancing, any-cast 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 [IBM23].

4.2 Applications and Discussion

Cryptocurrency wallets. One primary application of distributed cryptography (particularly threshold and multi-signature schemes) is *multi-device* cryptocurrency wallets [Eya21]. In these systems, the private

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

Table 4: LAN measurements for Threshold ML-DSA-44. All costs are presented per party.

	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 5: WAN median latencies between different AWS EC2 machines, as noted in [Clo].

key is split across multiple devices, which may include both the user's devices and servers operated by the wallet provider. In a threshold-based multi-device wallet, the private key is shared among N devices, and any T devices can collaborate to produce a signature to authorize actions such as transactions.
Despite their growing importance, research on multi-device wallets' practical and usability requirements is limited. Most studies focus on *single-device* wallets [FGA20, ECBS18, KJGW16]. Recent work [MDM+23] examined user perceptions of multi-device threshold wallets, revealing that users prefer fewer, highly reputable parties to hold key shares and favor a lower threshold T for higher availability; this suggests that users prioritize reliability and reduced communication overhead over the security benefits of a larger N. Nonetheless, studies [YSDW24, AVBB21] show that users perceive multi-device wallets as more secure, making our schemes ideal for practical applications, as they perform best with lower (T, N). Research targeting cryptocurrency wallets has highlighted the need for diverse properties in threshold signature schemes. Some works, like [CATZ24], emphasize partial non-interactivity, where certain signing rounds can be performed independently of the message, improving usability in constrained environments. However, non-interactive schemes often require maintaining a state (e.g., precomputed nonces), which can introduce security risks and increased storage. Stateless schemes [KOR23, FYZ+24] address this by eliminating the need for an intermediate state. Our scheme offer configurability, supporting message-independent rounds without requiring pre-processing, allowing deployments to balance interactivity and statefulness based on the context.

TLS and the PKI While the traditional TLS architecture assumes a direct connection between a client and a server, modern deployments often involve a Content Delivery Network (CDN) to improve performance or security (this is especially notable in TLS 1.3, as CDNs drove its centralized deployment [HHA+20]). For protocols like HTTP, integrating the CDN is relatively straightforward: the origin server (e.g., `example.com`) determines what content should be cached and serves it via a subdomain (e.g., `cdn.example.com`). DNS is then used to route requests to the nearest CDN edge server. However, with HTTPS, the situation is more complex. TLS is designed to establish a secure, authenticated channel between the client and the *origin* server. To support HTTPS in this context, the origin server and the CDN must establish a

(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 Threshold ML-DSA-44 across different topologies. L = London, S = Seoul, T = Taipei, V = Virginia.

trust relationship that allows edge servers to terminate TLS connections on behalf of the origin. In the conventional approach, the CDN generates its key pair and obtains certificates for specific origin subdomains (e.g., `cdn.example.com`). This lets edge servers impersonate the origin server for TLS purposes, offloading all cryptographic operations. While convenient, this model has a serious drawback: compromising an edge server exposes the private key, allowing an attacker to impersonate the origin until the certificate is revoked or expires. CDNs introduced a model known as *Keyless SSL* [SS15] to address this risk. In this model, the edge servers never possess the private key, and, instead, TLS handshakes are proxied: the client and edge server establish the session keys, but all private key operations (such as signing or decryption) are forwarded securely to the origin server [BBF+17]. However, private key material or sensitive operations are still handled across multiple machines, even in this proxied model, introducing potential security risks. An alternative and increasingly attractive approach is to use threshold signing, in which only shares of the private key are distributed among multiple parties (the edge servers or servers controlled by the origin entity). In this setup, no single entity ever holds the whole key.
As noted by [Her23, CEN24, DKR23], deploying threshold signatures in this context demands both practicality and compatibility with standardized signature schemes. Similar considerations apply in post-quantum settings: any candidate scheme must be practical, standardized (or on a clear path to standardization), and admit efficient threshold variants. Our scheme fits perfectly in this setting as a practical variant of the ML-DSA standardized signature scheme. However, the choice of threshold parameters (T, N) directly impacts both the availability and security of the infrastructure. For example, in a 3-of-5 deployment, the system can tolerate the failure or disconnection of up to two edge machines while maintaining signing capability, a key requirement as edge servers may be taken offline or experience spikes. This parameterization also resists partial compromise: an attacker must gain control of at least three independent parties to forge a signature. Threshold signing thus enables deployments that are simultaneously distributed, fault-tolerant, and robust, which aligns with the operational dynamics of CDN infrastructure.

The Tor Network and Onion Services The Tor network serves millions of users each day to provide anonymity to its users from the websites they connect to and conceal what they are connecting to from their Internet service provider and any other intermediary in their path [TTP24]. To maintain security, the network needs to produce a fresh random value every day in such a way that it cannot be predicted or influenced by an attacker. This random number is generated every midnight by trusted directory authorities[3]. The technique currently used for this procedure, a distributed random generator scheme, is based on a commit-and-reveal technique. However, it is vulnerable to various attacks, as noted in the Tor security analysis [Pro24]. To address these concerns, [Hop14] proposed that directory authorities use a threshold signature scheme to sign the value H(time-period) in each consensus document (a document that contains information about all known Tor relays), binding it to a specific time period. In this approach, each authority generates a publicly verifiable signature share during voting. As long as at least $\lceil N/2 \rceil$ authorities are honest, the full signature can be reconstructed by each client. The approach has not yet been integrated into the network, primarily because there is no standardized, practical threshold algorithm. Our results suggest that lattice-based threshold schemes such as Threshold ML-DSA offer a compelling solution to this longstanding need. Our scheme is post-quantum secure and features lightweight signing rounds with minimal computational overhead. While latency is a critical factor in the Tor network (where connections often experience delays up to 5× greater than direct Internet paths), the proposed threshold signing schemes would impact only the consensus document generation, which occurs approximately once per hour[4]. Moreover, communication is restricted to the roughly ten directory authorities[5], which are primarily located in North America and Europe. Our experimental results show that Threshold ML-DSA fits this model well: it is based on well-scrutinized cryptographic assumptions, achieve low signing latencies across WAN topologies, and exhibit excellent performance under realistic network conditions.

[3] See https://spec.torproject.org/rend-spec/shared-random.html.
[4] https://support.torproject.org/glossary/consensus/
[5] In 2025: https://metrics.torproject.org/rs.html#search/flag:authority

References

[ASY22] 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.

[AVBB21] Svetlana Abramova, Artemij Voskobojnikov, Konstantin Beznosov, and Rainer Böhme. Bits under the mattress: Understanding different risk perceptions and security behaviors of crypto-asset users. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[BBF+17] Karthikeyan Bhargavan, Ioana Boureanu, Pierre-Alain Fouque, Cristina Onete, and Benjamin Richard. Content delivery over tls: a cryptographic analysis of keyless ssl. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 1–16, 2017.

[BBRS24] 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.

[BLL+15] 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.

[BS23] 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.

[BSW24] 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.

[CATZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 268–302. Springer, Singapore, December 2024.

[CCL+23] 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.

[CEN24] Sofia Celi, Daniel Escudero, and Guilhem Niot. Share the MAYO: thresholdizing MAYO. Cryptology ePrint Archive, Paper 2024/1960, 2024.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

[CKM23] 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.

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

[CS19] Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 128–153. Springer, Cham, December 2019.

[dEK+23] 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.

[DFPS22] 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.

[DKLS25] 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.

[DKM+24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

[DKR23] Jack Doerner, Yashvanth Kondi, and Leah Namisa Rosenbloom. Sometimes you can't distribute random-oracle-based proofs. NIST presentation, 2023.

[DM20] 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.

[dPENP25] 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.

[dPKN+25] 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.

[dPN25] 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.

[ECBS18] Shayan Eskandari, Jeremy Clark, David Barrera, and Elizabeth Stobert. A first look at the usability of bitcoin key management. *CoRR*, abs/1802.04351, 2018.

[EKT24] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024.

[ENP24] Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 425–458. Springer, Cham, August 2024.

[ESTT22] 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.

[Eya21] Ittay Eyal. On cryptocurrency wallet design. Cryptology ePrint Archive, Report 2021/1522, 2021.

[FGA20] Michael Fröhlich, Felix Gutjahr, and Florian Alt. Don't lose your coin! investigating security practices of cryptocurrency users. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, DIS '20, page 1751–1763, New York, NY, USA, 2020. Association for Computing Machinery.

[FHK19] 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.

[FMT25] Marc Fischlin, Aikaterini Mitrokotsa, and Jenit Tomy. BUFFing threshold signature schemes. PKC, 2025.

[FYZ+24] 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.

[GHKR08] 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.

[GKS24] Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 266–300. Springer, Cham, June 2024.

[Her23] Armando Faz Hernandez. Requirements for threshold tls. NIST presentation, 2023.

[HHA+20] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. Tracking the deployment of tls 1.3 on the web: a story of experimentation and centralization. *SIGCOMM Comput. Commun. Rev.*, 50(3):3–15, July 2020.

[HJAHB16] 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.

[Hop14] Nicholas Hopper. A threshold signature-based proposal for a shared rng. Tor Dev Mailing List, 2014.

[HPRR20] 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.

[IBM23] IBM. Handshake timer, 2023. https://www.ibm.com/docs/en/zos/3.1.0?topic=considerations-handshake-timer. Accessed 06/02/24.

[KCLM22] 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.

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020.

[KJGW16] Katharina Krombholz, Aljosha Judmayer, Matthias Gusenbauer, and Edgar R. Weippl. The other side of the coin: User experiences with bitcoin security and privacy. In Jens Grossklags and Bart Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 555–580. Springer, Berlin, Heidelberg, February 2016.

[KLS18] 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.

[KLSS23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-MLWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.

[KOR23] 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.

[KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[LJL+22] 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.

[Lyu09] 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.

[MDM+23] 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.

[Pro24] The Tor Project. Security analysis. Tor Specifications, 2024.

[SS15] 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.

[TRT+22] 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.

[TSG21] 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.

[TTP24] Inc The Tor Project. Tor metrics, 2024. https://metrics.torproject.org/.

[YSDW24] Yaman Yu, Tanusree Sharma, Sauvik Das, and Yang Wang. "don't put all your eggs in one basket": How cryptocurrency users choose and secure their wallets. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

A Additional material for Threshold ML-DSA

A.1 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 [dPN25]. 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 A.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:*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathcal{A}} \leqslant\ & \frac{Q_s Q_{\mathsf{H}_{\mathsf{cmt}}}}{q^{nk}} + Q_{\mathsf{H}} Q_s^2 \cdot \left(\frac{2\gamma_2 + 1}{q}\right)^{kn} + Q_{\mathsf{H}} Q_s \cdot 2^{-256} \\
& + \frac{T \cdot Q_s Q_{\mathsf{H}_{\mathsf{cmt}}} + (Q_{\mathsf{H}_{\mathsf{cmt}}} + Q_s)^2}{2^{2\kappa}} + Q_{\mathsf{H}}^2 \cdot 2^{-512} + \mathsf{negl}(\kappa) \\
& + \left(1 + \frac{2\varepsilon}{1-\varepsilon}\right)^{Q_s} \cdot \left(2\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{s}}} + \mathsf{Adv}^{\mathsf{UF}}_{\mathcal{B}_{\mathsf{UF}}}\right) \\
& + Q_s \cdot \frac{3M-2}{M-1} \cdot \left(\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{r}}} + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{z}}} + \frac{2\varepsilon}{1-\varepsilon}\right)
\end{aligned}$$

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 [dPN25, Theorem 4].
We first recall results from [dPN25] to tighly simulate rejected transcripts in the Fiat-Shamir with Aborts paradigm, and to evaluate rejection probabilities as a function of ε.

Lemma A.1 (Adaptation of [dPN25, 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:*

$$\begin{aligned}
\frac{1-\varepsilon}{M} \leqslant \Pr\left[\mathrm{Rej}(\mathbf{v}, \chi_{\mathbf{r}}, \chi_{\mathbf{s}}, M) \neq \perp\right] \leqslant \frac{1}{M} \\
\delta \leqslant \frac{2\varepsilon}{1-\varepsilon}
\end{aligned}$$

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

Lemma A.2 ([dPN25, 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 $p = \Pr\left[\mathrm{Rej}(c\mathbf{s}, \chi_{\mathbf{r}}, \chi_{\mathbf{s}}, M; \mathbf{r} \leftarrow \chi_{\mathbf{r}}) = \perp\right]$ 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 A.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 $\mathsf{Adv}^{G}_{\mathcal{A}} := \Pr\left[G(\mathcal{A}) \to 1\right]$ – 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. 11.

Game_2

Q_{msg}, $\mathsf{UnopenedCmt}[\cdot] := \varnothing$
$f_{\mathsf{fail}} := \bot$
$\mathsf{pp}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathsf{Setup}(\kappa, N, T)$
$\mathsf{corrupt} \leftarrow \mathcal{A}(\mathsf{pp})$
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**
  $\mathsf{st}_i := \varnothing$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H},(\mathsf{OSign}_i(\cdot))_{i\in[R]}}(\mathsf{st}_{\mathcal{A}}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
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 11: 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.

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.

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

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.

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}\right| \leqslant \sum_{s=1}^{Q_s} \Pr[E'_s] \tag{4}$$

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. 12 an intermediary game Int_1^s in which $\mathcal{A}$ wins with probability exactly p_s. We also introduce a tweak of Int_1^s, named Int_2^s, where we replace the s-th commitment $\mathbf{w}_i$ by a uniform value in $\mathcal{R}_q^k$.
First, the difference in advantage between Int_1^s and Int_2^s 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:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_1^s} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_2^s}\right| = \left|p_s - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_2^s}\right| \leqslant \mathsf{Adv}_{\mathcal{E}^s}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}} \tag{5}$$

Also, we note that in Int_2^s, 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 $\mathbf{w}_i \xleftarrow{\$} \mathcal{R}_q^k$, i.e. by q^{-nk}. By union bound, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_2^s} \leqslant Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot q^{-nk}$.

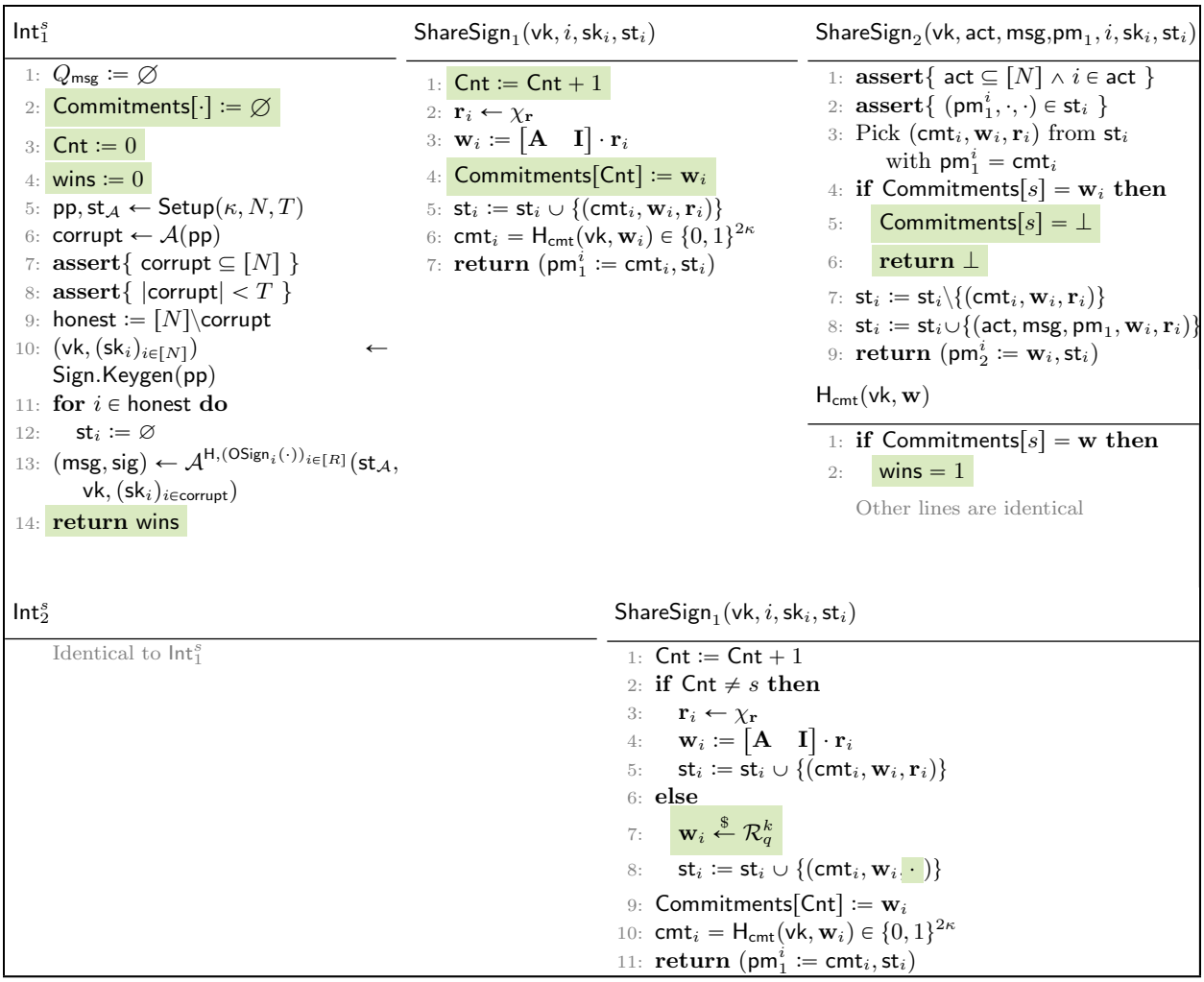

Figure 12: Intermediary games for the second hybrid of the unforgeability proof of Threshold ML-DSA. Difference with Game_1 are highlighted.

By combining the above inequality with Eq. (4) and Eq. (5), 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 1. *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 $\mathsf{H}(\mu||\mathbf{w}_\top) = \tilde{c}$ and $\mathsf{SampleInBall}(\tilde{c}) = c$. This is formalized in Fig. 13.

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 $\mu||\mathbf{w}_\top$, where $\mathbf{w}$ are the high bits of $\hat{\mathbf{w}} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$ before it is programmed in round 2, or (ii) $\mathsf{SampleInBall}$ on $\tilde{c}$ before it is programmed.

Game_3

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

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

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**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{256}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
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 $(i \in \mathsf{act}') \wedge (\mathsf{cmt}'_i = \mathsf{pm}_1^i) \wedge (\forall j \in \mathsf{act}'\backslash\{i\}, \exists \mathbf{w}_j, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j))$ **then**
    ▷ Find all the new programmable challenges
    $\mathsf{ToProgram} := \mathsf{ToProgram}\backslash\{(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \tilde{c}', c')\}$
    $\mathbf{w} := \sum_{j\in\mathsf{act}'} \mathbf{w}_j$; $(\mathbf{w}_\top, \mathbf{w}_\bot) = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
    $\mathsf{H}(\mu||\mathbf{w}_\top) := \tilde{c}'$; $\mathsf{SampleInBall}(\tilde{c}') := c'$ ▷ 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 13: 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.

First, we can easily bound the probability that $\mathsf{SampleInBall}$ is queried on some $\tilde{c}$ before it is programmed by $Q_{\mathsf{H}} Q_s \cdot 2^{-256}$.
Then, we are interested in the probability that H is queried on some $\mu||\mathbf{w}_\top$ 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 $\mathbf{w}_\top$ that was previously queried on the random oracle H. Formally, we define the game Int_3^s in Fig. 14, that verifies $p_s = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_3^s}$. By union bound, we have $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3}\right| \leqslant \sum_{s\in[Q_s]} p_s$.
We additionally define the game Int_4^s in Fig. 14, where we replace the s-th $\mathbf{w}_i$ by a uniform value in $\mathcal{R}_q^k$. The advantage difference between Int_3^s and Int_4^s again reduces to $\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$: $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_3^s} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_4^s}\right| \leqslant \mathsf{Adv}_{\mathcal{E}^s}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}$ 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_4^s can be bounded using the min-entropy of $\mathsf{HighBits}_q(\mathcal{U}(\mathcal{R}_q^k), 2\gamma_2)$. 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 $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_4^s} \leqslant Q_{\mathsf{H}} Q_s \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn}$.
Putting together the different inequalities obtained, we finally deduce the existence of an adversary $\mathcal{B}_2$ against the

Int^s_3

$Q_{\mathsf{msg}}, \mathsf{UnopenedCmt}[\cdot], := \varnothing$
$\mathsf{ToProgram} := \varnothing$
$\mathsf{Cnt} := 0$
$\mathsf{wins} := 0$
$f_{\mathsf{fail}} := \bot$
$\mathsf{pp}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathsf{Setup}(\kappa, N, T)$
$\mathsf{corrupt} \leftarrow \mathcal{A}(\mathsf{pp})$
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 $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin$ 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 $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin$ 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 14: Intermediary games for the third hybrid of the unforgeability proof of Threshold ML-DSA. Difference with Game_2 are highlighted .

$\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}$ such that:

$$\left|\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}\right| \leqslant Q_{\mathsf{H}} Q_s^2 \cdot \left(\frac{2\gamma_2 + 1}{q}\right)^{kn} + Q_{\mathsf{H}} Q_s \cdot 2^{-256} + Q_s \cdot \mathsf{Adv}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{B}_2}$$

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. 15.

Game_4

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

Rest is identical to Game_3

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

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

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

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

The rest is identical

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

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

Figure 15: Fourth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

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,

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

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. 16.
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,

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

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. 17.
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.

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

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 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 $\mathsf{SampleInBall}$ 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 1 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,

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

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 $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$. 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. 18.
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 $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$ finally ensures the same distribution of $\mathbf{z}_i$ in the above equality in case of rejections.
So,

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

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 formalized in Fig. 19.
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:

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

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. 20.
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

Game_6

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

Rest is identical to Game_4

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

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**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{256}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$
$\mathbf{z}_i \leftarrow \mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r}_i)$
$\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$

Rest is identical

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

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

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 $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right| \leqslant \sum_{\mathbf{s},c} \left|\mathsf{Adv}_{\mathcal{A}}^{F'^v} - \mathsf{Adv}_{\mathcal{A}}^{F^v}\right| \cdot \Pr\left[v \text{ used by } G^s\right] \tag{6}$$

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 $\mathbf{s}_i^{\mathsf{part}}$.

Game_7

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**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{256}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$
$\mathbf{z}_i \leftarrow \mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
if $\mathbf{z}_i = \bot$ **then** ▷ if reject
  $\mathbf{z}_i \leftarrow (\mathbf{z}|\mathrm{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$
$\mathbf{t}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{t}_I$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{t}_i^{\mathsf{part}}$
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 18: Seventh hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

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

$$\left|\mathsf{Adv}_{\mathcal{A}}^{F'^v} - \mathsf{Adv}_{\mathcal{A}}^{F^v}\right| = \mathsf{Adv}_{\mathcal{E}^v}^{\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}}} \tag{7}$$

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 A.2: there exists adversaries $\mathcal{E}'^v$ and $\mathcal{E}''^v$ such that

$$\mathsf{Adv}_{\mathcal{E}^v}^{\mathsf{MLWE}_{q,k,\ell,(\mathbf{z}|\mathrm{rej})_{\mathbf{v}}}} \leqslant \frac{1}{1-\frac{1}{M}} \cdot (\mathsf{Adv}_{\mathcal{E}'^v}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{E}''^v}^{\mathsf{MLWE}_{q,k,\ell,\chi_{\mathbf{z}}}} + \delta^v) \tag{8}$$

where $\mathbf{v} = c \cdot \mathbf{s}_i^{\mathsf{part}}$, δ^v is the statistical distance between $\chi_{\mathbf{z}}$ and $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$.
We can also apply Lemma A.1 to bound $\delta^v \leqslant \frac{2\varepsilon}{1-\varepsilon}$.
By combining Equations 6, 7, and 8, we obtain:

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

Finally, by summing all the above inequalities, we obtain

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

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**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{256}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
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 $\bot$
$\mathbf{z}_i \leftarrow \mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
if $\mathbf{z}_i = \bot$ **then** ▷ if reject
  $\mathbf{z}_i \leftarrow (\mathbf{z}|\mathrm{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$
$\mathbf{t}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{t}_I$
$\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{t}_i^{\mathsf{part}}$
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 19: Eighth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

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 $\mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r}_i)$ by an ideal functionality $\mathrm{Ideal}(\chi_{\mathbf{z}}, M)$ which is independent of the secret. This is formalized in Fig. 21.
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 Renyi 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:

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

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 Renyi 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 $\|c \cdot \mathbf{s}_j^{\mathsf{part}}\| \leqslant B$.

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**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{256}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
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 $\bot$
$\mathbf{z}_i \leftarrow \mathrm{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
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 20: Nineth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

We finally apply Lemma A.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_{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. 22.
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_{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**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{256}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
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 $\bot$
$\mathbf{z}_i \leftarrow \mathrm{Ideal}(\chi_{\mathbf{z}}, M)$
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 21: Tenth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

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. 23.
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.
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}.
$\mathcal{B}_7$ takes as input an ML-DSA public key $\mathsf{pk} = (\rho, \mathbf{t}_\bot)$, and random oracles ExpandA, H and SampleInBall.
It does the folllowing 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}.
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.

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 22: 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 $\mathcal{U}(\mathcal{R}_q^k)$ conditioned on $(\mathbf{t}_\top, \cdot) = \mathsf{Power2Round}(\mathbf{t}, d)$
$\mathbf{t}_{I_0} := \mathbf{t} - \sum_{I \neq I_0} \mathbf{t}_I$
$\mathsf{vk} := (\rho, \mathbf{t}_\top)$ ▷ Verification key
$\mathsf{sk} := (tr, \mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{sk})$

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

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 $\mathcal{T}(\mathcal{A})$. □