

Efficient Threshold ML-DSA

Sofía Celi^{1,2}, Rafael del Pino³, Thomas Espitau³, Guilhem Niot^{3,4}, Thomas Prest³

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

Abstract

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

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

1 Introduction

As organizations migrate to post-quantum cryptography, they are adopting the Module-Lattice-based Digital Signature Algorithm (ML-DSA), recently finalized by NIST [51]. However, this transition creates a critical security gap: modern systems rely on threshold signature schemes (TSS) to eliminate single points of failure and distribute trust, but no practical threshold construction for ML-DSA exists. This forces a dangerous trade-off between achieving quantum resistance and maintaining the operational resilience provided by mature classical

threshold schemes for RSA, Schnorr, and ECDSA. Systems that migrate to ML-DSA today must forfeit the threshold guarantees they depend on.

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

1.1 Our Contribution: The First Practical Threshold ML-DSA

We introduce the first practical threshold signature scheme for the ML-DSA standard. Our protocol allows an existing ML-DSA key to be securely distributed among multiple parties (a posteriori sharing). It also includes a fully distributed key generation (DKG) protocol that eliminates any trusted dealer. We propose parameters for signing up to 6 parties, which are at most 1 MB of communication per party on average¹ for ML-DSA-44. While a threshold of six may appear small at first glance, it represents a significant step forward and is already more than sufficient for most industrial applications (which requires usually 2-out-of-2 and 2-out-of-3 settings). A few more parties can be supported with higher communication costs as well, potentially of use in some niche applications.

Our design overcomes the fundamental conflict between ML-DSA’s abort-and-retry procedure and multi-party computation. We achieve this by tailoring the core ideas of short

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

secret sharing and per-party rejection from del Pino and Niot [24] to ML-DSA’s unique constraints.

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

- **Unbalanced Hyperball Rejection Sampling.** We replace the rejection sampling technique from ML-DSA with one based on hyperballs. This significantly reduces abort probability when multiple parties must simultaneously satisfy the rejection criteria. Further, we fine-tune this new rejection sampling by using simultaneously different norms.
- **Optimized Share Reconstruction (Section 3.3).** The probability of signing success is directly tied to the size of the secret shares. We designed a tailored reconstruction procedure that minimizes this size by modeling the problem as a variant of the B-matching problem, which we solve efficiently using a maximum flow algorithm.
- **Parallel Protocol Instances.** To ensure high signing success rates at reconstruction time, our protocol runs multiple instances in parallel. This approach guarantees fast and reliable signing without compromising security.

For key generation, we cover two complementary scenarios:

- **A Posteriori Key Sharing (Section 3.4).** For settings where the signing key is fixed a priori (e.g., for backward compatibility), we show how it can be securely shared among multiple users, while crucially preserving the original verification key.
- **Distributed Key Generation.** We design a fully decentralized key generation protocol that removes the need for a trusted dealer.

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

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

1.2 Related Work

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

Threshold Signature Schemes. Classical TSS constructions include schemes based on RSA [35], Schnorr [46, 16], and ECDSA [11, 12], many of which offer advanced properties such as identifiable aborts. However, the design of post-quantum TSS has proven more difficult, with many proposals relying on assumptions outside of the NIST call for standardization. Proposed post-quantum schemes span a variety of hardness assumptions: lattice-based, isogeny-based [18], and hash-based [43]. A recent multivariate-based TSS by Celi et al. [13] achieves compatibility with the MAYO and UOV signature schemes using generic MPC techniques, potentially offering practical efficiency for specific parameter choices.

Lattice-based Fiat-Shamir threshold schemes. Given the perceived impracticality of directly thresholdizing ML-DSA due to its rejection sampling mechanism, early efforts in the lattice setting focused on Fiat-Shamir-style signatures without rejection sampling, targeting variants of the Raccoon signature scheme [19]. These include threshold schemes based on threshold FHE by Agrawal et al. [2] and Gur et al. [37], and standard tools by del Pino et al. [21] Espitau et al. [29], Katsumata et al. [42] and Chairattanaï-Apirom et al. [14]. While these schemes typically yield larger signatures, they offer attractive features such as adaptive security [42], robustness [30], identifiable aborts [20, 22], and Beyond Unforgeability Features (BUFF) [34]. We provide an efficiency and size comparison against two representative schemes in Section 4.2. These design directions avoid the difficulties associated with ML-DSA’s rejection mechanism but sacrifice compatibility with a standardized scheme.

Direct Thresholdization of ML-DSA. Despite its complexity, several works have pursued direct thresholdization of

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

Scheme	# Parties (upper bound)	# Rounds Off + Online	Communication Off + Online (MB)	Computation	Model	Security
This work	6	2+4	$10^{-5} + 0.021$ to 1.05	Lightweight	Game-based	Dishonest majority
Bienstock et al. [7]	Unlimited	$136^* + 79$ $37^* + 23$	$> 0.87^* + 0.70$ $> 1.9^* + 1.5$	Online lightweight*	UC	Honest majority
Trilithium [27]	2	$1^\dagger + 60$	$234^\dagger + 0.40$	Heavy	UC	Trusted party [†]
Generic MPC [15]	Unlimited	High	High	Impractical	UC	Dishonest majority

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

ML-DSA, primarily through generic MPC techniques that address its rejection sampling behavior (Table 1).

The generic MPC-based approach of Cozzo and Smart [15] is secure in theory but suffers from high computational and communication costs, making practical deployment infeasible. The “Trilithium” scheme by Dufka et al. [27] supports two-party signing, but it assumes a *trusted party* to generate correlated randomness: an assumption that falls outside standard adversarial models and still incurs high communication overhead. Moreover, its security analysis lacks a full proof of unforgeability, relying instead on heuristic assumptions about the distribution of rejected ML-DSA transcripts.

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

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

Secret Sharing Techniques Recent works by Chairattana-Apirom et al. [14] and del Pino et al. [20] highlighted the benefits of short secret sharing techniques – consisting of a secret sharing with short partial shares – which enable more compact and efficient protocols [10, 14], and support advanced features such as identifiable aborts at no additional cost [20]. Additionally, these techniques simplify integration with rejection

sampling [24], which is crucial for this work.

2 Preliminaries

2.1 General notations

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

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

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

2.2 Threshold signatures

In all this work, we denote the security parameter by κ . Following the syntax of Katsumata et al. [42], a R -round threshold signature (*without* session identifiers) is as a tuple $\text{TSS} =$

(Setup, Keygen, $(\text{Sign}_i)_{i \in [R]}$, Combine, Verify). $\text{act} \subseteq [N]$ denotes a set of T parties used for signing. Each signer i maintains a state st_i : short-lived session-specific information.

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

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

$\text{Sign}_r(\text{vk}, \text{act}, \text{msg}, (\text{pm}_{r-1}^j)_{j \in \text{act}}, i, \text{sk}_i, \text{st}_i) \rightarrow (\text{pm}_r^i, \text{st}_i)$.
For the r -th round ($r \in \{1, \dots, R\}$), takes as input the verification key vk , the set of signers act , a message msg , the protocol messages exchanged during the previous round $(\text{pm}_{r-1}^j)_{j \in \text{act}}$, 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 $\text{pm}_0^i = \perp$.

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

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

$\text{Verify}(\text{vk}, \text{msg}, \text{sig}) \rightarrow 0$ 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 [21] or adaptively [42], and whether the communication channel is controlled by the adversary [21], or requires consensus properties [30, 20], and for which messages a forgery is accepted.

In this work, we consider the unforgeability notion from del Pino et al. [21] adapted to remove the use of session identifiers. It ensures static security – i.e. corrupted users are chosen before executing the protocol, a communication channel fully controlled by the adversary, and forgeries are accepted for messages for which no signing oracle was called. It can also be seen as a multi-round adaptation of the TS-UF-0 notion from [6]. We provide formal definitions of these properties below.

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

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

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

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

$\text{Game}_{\text{TSS}, \mathcal{A}}^{\text{TS-UF}}(\kappa, N, T)$
1: $Q_{\text{msg}} := \emptyset$ 2: $\text{pp} \leftarrow \text{Setup}(\kappa, N, T)$ 3: $\text{corrupt}, \text{st}_{\mathcal{A}} \leftarrow \mathcal{A}(\text{pp})$ 4: assert { $\text{corrupt} \subseteq [N]$ and $ \text{corrupt} < T$ } 5: $\text{honest} := [N] \setminus \text{corrupt}$ 6: $(\text{vk}, (\text{sk}_i)_{i \in [N]}) \leftarrow \text{Sign.Keygen}(\text{pp})$ 7: for { $i \in \text{honest}$ } do { $\text{st}_i := \emptyset$ } 8: $(\text{msg}, \text{sig}) \leftarrow \mathcal{A}^{\text{H}, (\text{OSign}_r(\cdot))_{r \in [R]}}(\text{st}_{\mathcal{A}}, \text{vk}, (\text{sk}_i)_{i \in \text{corrupt}})$ 9: req { $\text{msg} \notin Q_{\text{msg}}$ } 10: return $\text{Verify}(\text{vk}, \text{msg}, \text{sig})$
$\text{OSign}_r(\text{act}, \text{msg}, i, (\text{pm}_{r-1}^j)_{j \in \text{act}})$
1: req { $i \in [N] \wedge i \in \text{act} \cap \text{honest}$ } 2: $Q_{\text{msg}} := Q_{\text{msg}} \cup \{\text{msg}\}$ 3: $(\text{pm}_r^i, \text{st}_i) \leftarrow \text{ShareSign}_r(\text{vk}, \text{act}, \text{msg},$ 4: $\quad (\text{pm}_{r-1}^j)_{j \in \text{act}}, i, \text{sk}_i, \text{st}_i)$ 5: return pm_r^i

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

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

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

$$\Pr [\text{Game}_{\text{TSS}}^{\text{TS-corr}}(\kappa, N, T, \text{msg}, \text{act}) \neq \perp] \geq p \quad (1)$$

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

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

2.3 Modulus Rounding

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

Game $_{\text{TSS}}^{\text{TS-corr}}(\kappa, N, T, \text{msg}, \text{act})$
1: $\text{pp} \leftarrow \text{Setup}(\kappa, N, T)$
2: $(\text{vk}, (\text{sk}_i)_{i \in [N]}) \leftarrow \text{Sign.Keygen}(\text{pp})$
3: for $\{i \in [N]\}$ do $\{\text{st}_i := \emptyset\}$
4: for $r \in [R]$ do
5: for $i \in \text{act}$ do
6: $(\text{pm}_r^i, \text{st}_i) \leftarrow \text{ShareSign}_r(\text{vk}, \text{act}, \text{msg},$
7: $(\text{pm}_{r-1}^i)_{j \in \text{act}}, i, \text{sk}_i, \text{st}_i)$
8: if $\{\exists i \in \text{act}, \text{pm}_R^i = \text{abort}\}$ then $\{\text{return } \perp\}$
9: sig $:= \text{Combine}(\text{vk}, \text{act}, (\text{pm}_r^i)_{r \in [R], i \in \text{act}})$
10: return $\text{Verify}(\text{vk}, \text{msg}, \text{sig})$

Figure 2: Correctness game for TSS with aborts.

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 $r_1 = (r - r_0)/2^d$.

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

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.

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 $\text{HighBits}(r, 2\gamma_2)$. The goal is to recover $\text{HighBits}(r + z, 2\gamma_2)$. 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 ([44]). *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 $\|\mathbf{z}\|_\infty \leq \alpha/2$. Then:*

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

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 [44] 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 MLWE $_{q,k,\ell,\chi}$ problem is defined as:*

$$|\Pr[\mathcal{A}(\mathbf{A}, [\mathbf{A} \ \mathbf{I}_k] \cdot \mathbf{s}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]| \quad (3)$$

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

2.5 The Rényi divergence

Following Devevey et al. [25], we rely on the *Rényi divergence of infinite order* R_∞ as well as the *smooth Rényi divergence* R_∞^ε to measure the distance between two distributions.

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

We additionally recall a relaxed version of the Rényi divergence from [25, Def. 2.1], called *smooth Rényi divergence*, where one can remove a few problematic points from the support, including those that may lie in $\text{Supp}(\mathcal{P}) \setminus \text{Supp}(\mathcal{Q})$.

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

Lemma 2.2 ([4] and [40]). *For two distributions \mathcal{P}, \mathcal{Q} , the Rényi divergence satisfies the following properties:*

- **Data processing inequality.** $R_\infty(\mathcal{P}^f||\mathcal{Q}^f) \leq R_\infty(\mathcal{P}||\mathcal{Q})$ for any function f , where \mathcal{P}^f (resp. \mathcal{Q}^f) is 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 $R_\infty(\mathcal{P}_{i,|x_{<i}=v_{<i}}||\mathcal{Q}_{i,|x_{<i}=v_{<i}}) \leq r_i$ for any possible $v_{<i}$. Then, $R_\infty(\mathcal{P}||\mathcal{Q}) \leq \prod_i r_i$.
- **Probability preservation.** for any event $E \subseteq \text{Supp}(\mathcal{Q})$, we have $Q(E) \geq P(E)/R_\infty(\mathcal{P}||\mathcal{Q})$.

2.6 Rejection Sampling

ML-DSA is a Fiat-Shamir with Aborts signature scheme. It heavily relies on the rejection sampling technique introduced by Lyubashevsky [49], which we recall with the same notations as Finally! [24].

$\text{Rej}(\mathbf{v}, \chi_z, \chi_r, M) \rightarrow \mathbf{z} \in \mathcal{R}^{k+\ell} \cup \{\perp\}$	$\text{Ideal}(\chi_z, M) \rightarrow \mathbf{z} \in \mathcal{R}^{k+\ell} \cup \{\perp\}$
1: $\mathbf{r} \leftarrow \chi_r$	1: $\mathbf{z} \leftarrow \chi_z$
2: $\mathbf{z} := \mathbf{v} + \mathbf{r}$	2: $b \leftarrow \text{Ber}(\frac{1}{M})$
3: $b \leftarrow \text{Ber}(\min(\frac{\chi_z(\mathbf{z})}{M\chi_r(\mathbf{r})}, 1))$	3: if $\{b = 0\}$ then $\{\mathbf{z} = \perp\}$
4: if $\{b = 0\}$ then $\{\mathbf{z} = \perp\}$	4: return \mathbf{z}
5: return \mathbf{z}	

Figure 3: Rejection algorithm and ideal algorithm.

We extend the algorithm Rej to $\text{Rej}(\mathbf{v}, \chi_z, \chi_r, M; \mathbf{r})$ to parse the randomness $\mathbf{r} \leftarrow \chi_r$ as an input. We will also write $(\mathbf{z}|\text{acc})_{\mathbf{v}}$ to denote the distribution of $\mathbf{z} := \mathbf{v} + \mathbf{r}$ conditioned on $b = 1$, and $(\mathbf{z}|\text{rej})_{\mathbf{v}}$ to denote the distribution of $\mathbf{z} := \mathbf{v} + \mathbf{r}$

conditioned on $b = 0$ (note that in that case $\mathbf{z} = \perp$ but the distribution we are interested in is the one of $\mathbf{v} + \mathbf{r}$ so we abuse this notation).

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

Lemma 2.3 (Lemma 4.1, [25]). *Let $M > 1$ and $\varepsilon < 1$ such that $R_{\infty}^{\varepsilon}(\chi_{\mathbf{z}} || \mathbf{v} + \chi_{\mathbf{r}}) \leq M$. Then, $R_{\infty}(\text{Rej} || \text{Ideal}) \leq 1 + \frac{\varepsilon}{M-1}$.*

2.7 Rejection sampling over hyperballs

Devevey et al. [25] showed that hyperball-based rejection sampling achieves the most compact parameters, and outperforms uniforms [49] and polytopes as used by Bambury et al. [5]. Hyperballs achieve the same compactness as using Gaussians, but have the advantage of a simpler rejection condition in the form of a norm two bound check.

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

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

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

To bound the Rényi divergence we rely on the followings.

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

Lemma 2.4 ([25, Lemma 5.1]). *Let $\ell \geq 1$ and $\mathbf{v} \in \mathbb{R}^{n\ell}$. Let $\varepsilon \in [0, 1/2)$ and $\phi \geq 1$ be such that $2\varepsilon = I_{1-1/\phi^2}(\frac{n\ell+1}{2}, \frac{1}{2})$. Let $r, r' > 0$ such that $r'^2 \geq r^2 + \|\mathbf{v}\|_2^2 + 2r\|\mathbf{v}\|_2/\phi$. It holds that:*

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

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

Lemma 2.5 (from [25, Section A.6]). *For $n > 1, \phi > 1$, we have $I_{1-1/\phi^2}(\frac{n+1}{2}, \frac{1}{2}) < \left(1 - \frac{1}{\phi^2}\right)^{n-1} \cdot n \cdot \left(1 - \frac{1}{\phi}\right)$.*

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

3 Threshold ML-DSA

This section describes our variant of ML-DSA (Module-Lattice-Based Digital Signature Algorithm) optimized for small threshold settings. We build upon the foundation laid

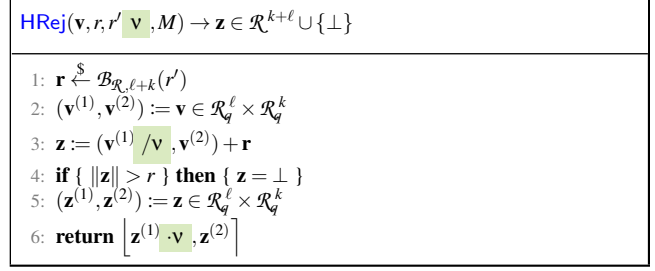


Figure 4: Imbalanced rejection sampling over hyperballs. Major differences with Rej (Fig. 3) are highlighted.

by Finally! [24], which introduced the key innovation of leveraging short replicated secret sharing to distribute rejection sampling in lattice-based signatures, by performing parallel rejection sampling steps in each party. However, Finally! was designed as a custom signature scheme, and adapting its techniques to the ML-DSA standard presents significant challenges due to fundamental differences in their rejection sampling, target norm (Finally! relies on the Euclidean norm, while ML-DSA uses the infinity norm) and rounding procedures.

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

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

As the *hint* technique used in ML-DSA only targets the second portion $\mathbf{z}^{(2)}$ of the signature, it introduces a much stricter verification condition on $\mathbf{z}^{(2)}$ than $\mathbf{z}^{(1)}$. We optimize the success rate of this check by using imbalanced rejection sampling in which $\mathbf{z}^{(2)}$ rejects towards a smaller hyperball than $\mathbf{z}^{(1)}$. The imbalance is parametrized by a factor \mathbf{v} , that expands the randomness used to sample the first ℓ coordinates of \mathbf{r} by a factor $\mathbf{v} > 1$. This results in a larger acceptance region for $\mathbf{z}^{(1)}$ while keeping the same acceptance region for $\mathbf{z}^{(2)}$, thus significantly lowering the rejection rate due to the

Table 2: Parameters used in Threshold ML-DSA.

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

first part of the signature. See [HRej](#) (Fig. 4) for a concrete description.

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

3.1 Construction

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

Key Generation. Our construction relies on the *short secret sharing* notion introduced by del Pino et al. [20] ([RSS](#)) and already leveraged in [Finally!](#) [24], which is particularly fit for lattice-based constructions. These secret sharings are special instances of Linear Secret Sharing Schemes (LSSS) that additionally guarantee shares with *small norm* and *small reconstruction coefficients*. This is particularly suited for rejection-sampling-based schemes, as it enables each party to perform rejection sampling locally on a *short* partially reconstructed secret. This *short replicated secret sharing* scheme consists in sampling $\binom{N}{N-T+1}$ ML-DSA secrets $(sk_I)_I$ from χ_s , that is, one for each distinct set I of $N - T + 1$ users. Since there are at most $T - 1$ corrupted parties, at least one of these sets is fully non-corrupted and its secret remains hidden from the corrupted parties.

The final public key is derived by summing all secrets, multiplying by $[\mathbf{A} \ \mathbf{I}]$, and rounding the result as in ML-DSA,

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

The key generation procedure is detailed in Fig. 5. While Fig. 5 assumes a trusted dealer (as [Finally!](#)) for simplicity, we show that a distributed key generation protocol is possible in the full version of this work.

$\text{RSS}(N, T, \chi) \rightarrow (\mathbf{s}, \text{sk} := (sk_1, \dots, sk_N))$
<ol style="list-style-type: none"> 1: for $I \subset [N]$ such that $I = N - T + 1$ do 2: $\mathbf{s}_I \leftarrow \chi$ 3: $\mathbf{s} := \sum_I \mathbf{s}_I$ 4: for $i \in [N]$ do ▷ Distribute the shares 5: $sk_i := \{I : \mathbf{s}_I \mid I \subset [N] \text{ s.t. } i \in I \wedge I = N - T + 1\}$ 6: return $(\mathbf{s}, \text{sk} := (sk_0, \dots, sk_{N-1}))$
$\text{Keygen}(\kappa, N, T) \rightarrow (\text{vk}, \text{sk})$
<ol style="list-style-type: none"> 1: $\rho \leftarrow \{0, 1\}^{256}$ 2: $\mathbf{A} := \text{ExpandA}(\rho)$ 3: $(\mathbf{s}, (sk_i)_{i \in [N]}) \leftarrow \text{RSS}(N, T, \chi_s)$ 4: $\mathbf{t} := [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{s}$ 5: $(\mathbf{t}_\top, \mathbf{t}_\perp) := \text{Power2Round}(\mathbf{t}, d)$ 6: $tr \in \{0, 1\}^{256} := H(\rho \parallel \mathbf{t}_\top)$ 7: return $(\text{vk} := (\rho, \mathbf{t}_\top), \text{sk} := (tr, sk_i)_{i \in [N]})$

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

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

In particular, each party uses a full MLWE sample as commitment \mathbf{w}_i (i.e. $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{y} + \mathbf{e}$) instead of using $\mathbf{A} \cdot \mathbf{y}$ as in ML-DSA. This change allows for direct reveal of \mathbf{w}_i before rounding, even in case of rejection of the partial signatures. To ensure correctness, we perform a second rejection step and verify that the sum of the responses $\mathbf{z} = \sum_{i \in \text{act}} \mathbf{z}_i$ verifies the ML-DSA verification, roughly that the equation $[\mathbf{A} \ \mathbf{I}] \cdot \mathbf{z} = c \cdot \mathbf{t} + \mathbf{w}$ holds and the signature \mathbf{z} has a small norm. The challenge c can be computed as $c = \text{SampleInBall}(\tilde{c})$, where $\tilde{c} = H(\mu \parallel \text{HighBits}(\mathbf{w}, 2\gamma_2))$ and $\mathbf{w} = \sum_{i \in \text{act}} \mathbf{w}_i$. [SampleInBall](#)

maps \tilde{c} to a polynomial c such that $\|c\|_1 = \tau$ and $\|c\|_\infty = 1$.

Following prior works [21, 24], we design a three-round² scheme secure against rushing adversaries (who target \mathbf{w}):

1. **First round:** parties sample randomness $\mathbf{r}_i \leftarrow \chi_r$ and compute the commitment $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$. They publish a hash of the commitment $H_{\text{cmt}}(\text{vk}, i, \mathbf{w}_i)$.
2. **Second round:** parties reveal the values of $(\mathbf{w}_i)_{i \in \text{act}}$.
3. **Third round:** Each party computes the response $\mathbf{z}_i = c \cdot \mathbf{s}_i^{\text{part}} + \mathbf{r}_i$, where c is derived from the aggregated commitment $\mathbf{w} = \sum_{i \in \text{act}} \mathbf{w}_i$, and apply rejection sampling.

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

Remark 1 (Balanced partition of the shares for Replicated Secret Sharing). *The sizes of partial secrets $\mathbf{s}_i^{\text{part}}$ in Threshold ML-DSA impact greatly the efficiency of parameters as it directly correlates with the norm 2 of the hyperballs that are used, and it is important to use as few secrets as possible in a session for each party in `RSSRecover`. To tackle this, we propose in Section 3.3 a graph based algorithm that computes a balanced partition of the shares, i.e. each party signs with at most $\lceil \binom{N}{N-T+1} / T \rceil$ secrets, instead of $\binom{N}{N-T}$.*

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

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

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

Combine and Verify. The combine and verify procedures are detailed in Fig. 7. We use the ML-DSA verification. Combine involves computing the hint \mathbf{h} that allows verifiers to recover `HighBits`($\mathbf{w}, 2\gamma_2$) from the value $\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_T$ used as an approximation of \mathbf{w} . Similarly to ML-DSA, we compute the difference $\boldsymbol{\delta} = \mathbf{w} - (\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_T)$ and use the function

²This is per signing attempt.

<code>RSSRecover</code> (act) $\rightarrow \mathcal{P}(S)^{\text{act}}$
<ol style="list-style-type: none"> 1: Compute partition $\mathbf{m} = (m_i)_{i \in \text{act}}$ s.t. $\bigsqcup m_i = \{I \subset [N] \mid I = N - T + 1\}$ and minimizing $\max_{i \in \text{act}} m_i$, using a maximum flow algorithm. See Section 3.3. 2: return \mathbf{m}
<code>ShareSign</code> ₁ (vk, i, sk _i , st _i)
<ol style="list-style-type: none"> 1: $\mathbf{r}_i \leftarrow \chi_r$ 2: $\mathbf{w}_i := [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$ 3: $\text{cmt}_i = H_{\text{cmt}}(\text{vk}, i, \mathbf{w}_i) \in \{0, 1\}^{2\kappa}$ 4: $\text{st}_i := \text{st}_i \cup \{(\text{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$ 5: return ($\text{pm}_1^i := \text{cmt}_i, \text{st}_i$)
<code>ShareSign</code> ₂ (vk, act, msg, pm ₁ , i, sk _i , st _i)
<ol style="list-style-type: none"> 1: assert{ act $\subseteq [N] \wedge i \in \text{act}$ } 2: assert{ ($\text{pm}_1^i, \cdot, \cdot$) $\in \text{st}_i$ } 3: Pick ($\text{cmt}_i, \mathbf{w}_i, \mathbf{r}_i$) from st_i with $\text{pm}_1^i = \text{cmt}_i$ 4: $\text{st}_i := \text{st}_i \setminus \{(\text{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$ 5: $\text{st}_i := \text{st}_i \cup \{(\text{act}, \text{msg}, \text{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$ 6: return ($\text{pm}_2^i := \mathbf{w}_i, \text{st}_i$)
<code>ShareSign</code> ₃ (vk, pm ₂ , i, sk _i , st _i)
<ol style="list-style-type: none"> 1: assert{ ($\cdot, \text{pm}_2^i, \cdot$) $\in \text{st}_i$ } 2: Pick (act, msg, pm₁, $\mathbf{w}_i, \mathbf{r}_i$) from st_i with $\text{pm}_2^i = \mathbf{w}_i$ 3: Parse $\text{pm}_1 = (\text{cmt}_j)_j$ and $\text{pm}_2 = (\mathbf{w}_j)_{j \neq i}$ 4: assert{ $\forall j \in \text{act}, \text{cmt}_j = H_{\text{cmt}}(\text{vk}, j, \mathbf{w}_j)$ } 5: $\text{st}_i := \text{st}_i \setminus \{(\text{act}, \text{msg}, \text{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$ 6: $\mu \in \{0, 1\}^{512} := H(\text{tr} \parallel \text{msg})$ 7: $\mathbf{w} := \sum_{j \in \text{act}} \mathbf{w}_j$ 8: $\mathbf{w}_T := \text{HighBits}(\mathbf{w}, 2\gamma_2)$ 9: $\tilde{c} \in \{0, 1\}^{2\kappa} := H(\mu \parallel \mathbf{w}_T)$ 10: $c := \text{SampleInBall}(\tilde{c})$ ▷ Global challenge 11: $\mathbf{m} := \text{RSSRecover}(\text{act})$ 12: $\mathbf{s}_i^{\text{part}} := \sum_{I \in m_i} \mathbf{s}_I$ 13: $\mathbf{z}_i \leftarrow \text{HRej}(c \cdot \mathbf{s}_i^{\text{part}}, r, r', v, M; \mathbf{r}_i)$ ▷ 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 ($\text{pm}_3^i := \mathbf{z}_i^{(1)}, \text{st}_i$)

Figure 6: Signing procedure in Threshold ML-DSA.

`MakeHint` with $\boldsymbol{\delta}$ to compute the hint. The high bits of \mathbf{w} are correctly recovered as long as $\|\boldsymbol{\delta}\|_\infty \leq \gamma_2$; we thus restart the protocol if this is not the case. Verification proceeds just as in ML-DSA: the procedure also restarts if $\|\mathbf{z}^{(1)}\|_\infty \geq \gamma_1 - \beta$ 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 `ShareSign`₃ already ensures that \mathbf{z} does not leak any secret information. Thanks to parallel repetitions we can minimize the need for restarts

as we ensure that any execution of the protocol will output a signature with probability at least $1/2$.

<p>Combine($\text{vk} = (\text{seed}, \mathbf{t}_\top), \text{act}, \text{msg}, (\text{pm}_k)_{k \in \{1,2,3\}}$)</p> <ol style="list-style-type: none"> 1: Parse $\text{pm}_2 = (\mathbf{w}_i)_i$ and $\text{pm}_3 = (\mathbf{z}_i^{(1)})_i$ 2: $\mu \in \{0, 1\}^{512} := H(\text{tr} \text{msg})$ 3: $\mathbf{w} := \sum_{j \in \text{act}} \mathbf{w}_j \quad \triangleright$ Aggregated commitment in \mathcal{R}_q^k 4: $\mathbf{w}_\top := \text{HighBits}(\mathbf{w}, 2\gamma_2)$ 5: $\tilde{c} := H(\mu \mathbf{w}_\top) \quad \triangleright \tilde{c} \in \{0, 1\}^{2 \cdot \kappa}$ 6: $c := \text{SampleInBall}(\tilde{c}) \quad \triangleright$ Global challenge 7: $\mathbf{z}^{(1)} = \sum_{i \in \text{act}} \mathbf{z}_i^{(1)}$ 8: $\boldsymbol{\delta} := \mathbf{w} - (\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top) \quad \triangleright$ Recover error in \mathcal{R}_q^k 9: $\mathbf{h} := \text{MakeHint}_q(\boldsymbol{\delta}, \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top, 2\gamma_2)$ 10: if $(\ \mathbf{z}^{(1)}\ _\infty \geq \gamma_1 - \beta)$ or $(\ \boldsymbol{\delta}\ _\infty > \gamma_2)$ or $(\ \mathbf{h}\ _1 > \omega)$ then 11: abort 12: return $\text{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$ <p>Verify($\text{vk} := (\rho, \mathbf{t}_\top), \text{msg}, \text{sig}$)</p> <ol style="list-style-type: none"> 1: Parse $\text{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$ 2: $\mathbf{A} := \text{ExpandA}(\rho)$ 3: $\mu \in \{0, 1\}^{512} := H(\text{tr} \text{msg})$ 4: $c := \text{SampleInBall}(\tilde{c})$ 5: $\mathbf{w}'_\top := \text{UseHint}_q(\mathbf{h}, \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top, 2\gamma_2)$ 6: return $\ \mathbf{z}^{(1)}\ _\infty < \gamma_1 - \beta$ and $\tilde{c} = H(\mu \mathbf{w}'_\top)$ and $\ \mathbf{h}\ _1 \leq \omega$

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

3.2 Correctness and security

We prove that Threshold ML-DSA (i) is unforgeable, and (ii) p -terminates. For (i), Threshold ML-DSA is unforgeable if ML-DSA is unforgeable and the MLWE problem used in ML-DSA is hard. For (ii), we introduce a bound B such that for any \mathbf{m}_i returned by `RSSRecover` and $(\mathbf{u}_1, \mathbf{u}_2) = \sum_{l \in \mathbf{m}_i} \mathbf{s}_l \in \mathcal{R}_q^{\ell+k}$, $\|(\frac{1}{v} \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 \leq B$ 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 $r'^2 \geq r^2 + B^2 + 2rB/\phi$, implying Lemma 3.1.

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

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

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

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

where v_k , for $k \in [0, K]$, is the probability that in at least one of the k combinations we have $\|\mathbf{z}^{(1)}\|_\infty \leq \gamma_1 - \beta$ and $\|\boldsymbol{\delta}\|_\infty \leq \gamma_2$ 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 $\|\mathbf{z}^{(1)}\|_\infty < \gamma_1 - \beta$, and (ii) \mathbf{h} has at most ω 1's. Additionally, the scheme ensures that $\boldsymbol{\delta} := \mathbf{w} - (\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top)$ has an infinity norm at most γ_2 . By applying Lemma 2.1, we obtain $\text{UseHint}_q(\mathbf{h}, \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{t}_\top, 2\gamma_2) = \mathbf{w}_\top$ 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 $\text{Game}_0 := \text{Game}_{\text{TS}}^{\text{corr}}$. We denote $p_i = \Pr[\text{Game}_i() \neq \perp]$.

Game₁. In this game, we assert that for any signing party and parallel session during the second round, we have $\|(\frac{1}{v} \cdot c \cdot \mathbf{s}_1^{\text{part}}, c \cdot \mathbf{s}_2^{\text{part}})\|_2 \leq B$, 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:

$$p_0 \geq p_1 - TK \cdot \text{negl}(\kappa) = p_1 - \text{negl}(\kappa)$$

Game₂. 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:

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

Game₃. In this game, we sample the challenge c of each session in advance, sample $\mathbf{z}_i \leftarrow \text{Rej}(c \cdot \mathbf{s}_i^{\text{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$ for each party. If $\mathbf{z}_i = \perp$, we instead sample \mathbf{z}_i from the distribution $(\mathbf{z}|\text{rej})_{\mathbf{v}=c \cdot \mathbf{s}_i^{\text{part}}}$ of rejected \mathbf{z} . Then, we program the random oracle after computing $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{z} - c \cdot \mathbf{t}$'s, with $\mathbf{z} = \sum_{i \in \text{act}} \mathbf{z}_i$ 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: $p_2 = p_3$.

Game₄. In this game, we sample the partial responses \mathbf{z}_i using the **Ideal** rejection sampling from Fig. 3. Recall the Game₁ ensured that $c \cdot \mathbf{s}_i^{\text{part}}$ 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 “ $\forall(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}^{\text{part}}, \|(\frac{1}{\sqrt{v}} \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 \leq B$ ”. Formally, we consider the random variable $X = ((\text{sk}_i)_{i \in [M]}, (\mathbf{z}_i^j)_{i \in \text{act}, j \in [K]})$ conditioned on E , where the \mathbf{z}_i^j are the responses output by the parties in each session. After conditioning on $(\text{sk}_i)_{i \in [M]}$ 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 Game₃ and Game₄ by $(1 + \frac{\varepsilon}{M-1})^{KT}$.

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

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

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

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

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

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

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

We now state the unforgeability of Threshold ML-DSA.

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

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

We deduce from Theorem 3.2 that breaking the unforgeability of Threshold ML-DSA is as hard as breaking ML-DSA. Indeed, the MLWE assumption over χ_s is the one underlying ML-DSA, and the ones over χ_r and χ_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 the full version.

Our proof closely follows the approach of [24, Theorem 4], with two main differences: (i) we adapt the argument to account for the slightly different rounding of commitments \mathbf{w} , and (ii) instead of reducing to STMSIS [44], we reduce directly to the unforgeability of ML-DSA. The proof proceeds via a sequence of games, gradually transforming the real attack scenario into one where a forgery against the threshold scheme yields a forgery against single-party ML-DSA, under the hardness of the relevant MLWE problems.

- **Game 2.** We show that honest commitments \mathbf{w}_i have high min-entropy before being revealed, so the adversary cannot guess them in advance. This allows us to program the random oracle H_{cmt} lazily, returning random hashes in round 1 and sampling \mathbf{w}_i only in round 2 for honest parties. This step is justified in the random oracle model.
- **Games 3–6.** We sample the challenges c in advance in round 2. The pre-image and collision resistance of H_{cmt} ensure that the adversary has already chosen its \mathbf{w}_i before round 2. With the lazy sampling from the previous game, honest commitments are sampled last, so the aggregated commitment $\mathbf{w} = \sum_i \mathbf{w}_i$ has high min-entropy. We can thus program the random oracle to return the desired challenge c without the adversary noticing.
- **Game 7.** We compute \mathbf{z}_i first, even in case of rejection, and then derive $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{z}_i - c \cdot \mathbf{t}_i^{\text{part}}$. If \mathbf{z}_i is rejected, we sample it from the distribution of rejected \mathbf{z} . This is statically equivalent to the previous game.
- **Game 8.** We ensure that the norm of the secret vector $c \cdot \mathbf{s}_i^{\text{part}}$ is at most B , which holds with overwhelming probability.
- **Game 9.** We replace the commitments \mathbf{w}_i by uniform values when the corresponding \mathbf{z}_i is rejected and will never be output. This change is indistinguishable to the adversary under the MLWE assumptions for the relevant distributions, following [24, Lemma 5].
- **Games 10–11.** We remove the last dependencies on the secrets by replacing the sampling of \mathbf{z}_i with rejection sampling from the ideal functionality (recalled in Fig. 3). The Rényi divergence argument (see Section 2.7) ensures that, for up to Q_s queries, the adversary’s advantage increases by at most 2 bits.
- **Games 12–13.** We replace the public key of Threshold ML-DSA (composed of $\binom{N}{N-T+1}$ secrets) by an ML-DSA public key (a single secret). Since the signing oracles are now independent of secret values, this is justified by the MLWE assumption.

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

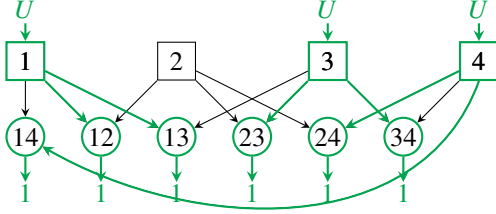


Figure 8: Illustration of the replicated secret sharing with $(N, T) = (4, 3)$: users are on top (rectangles), shares at the bottom (circles). Balanced assignment of shares to active users is performed via a max-flow solving algorithm (in green).

3.3 Balanced partition of the shares for Replicated Secret Sharing

Recall that we aim at computing a balanced partition of the shares among the party during the signing session, that is for a partition $(m_i)_{i \in \text{act}}$ of the secrets, it minimizes $\max_{i \in \text{act}} |m_i|$. As a general solution when fixing $U = \max_{i \in \text{act}} |m_i|$, 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 $i \in \text{act}$.
3. We add an exit flow 1 on the secret nodes.

This is represented in Fig. 8. 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 $N \leq 6$, for which we can easily verify that the optimal assignments 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 $2 \leq T < N \leq 6$, we explicit an optimal solution for $\text{act} = \{1, \dots, T\}$ and each values (T, N) . We obtain a partition for all other possible signing set act by symmetry by permuting the index of parties. The idea is formalized in Algorithm 6.

3.4 A Posteriori Key Sharing

In this section, we describe how a user with an existing ML-DSA signing key can split it into shares without affecting

Alg. 6 $\text{RSSRecover}(\text{act}) \rightarrow \mathcal{P}(S)^{\text{act}}$

```

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

```

verification. To share a secret key \mathbf{s} , we sample a short secret sharing (as in the key generation of 3.1) with the difference that the shares will be conditioned on their sum being \mathbf{s} .

Algorithmic overview of key sharing. For $S = \binom{N}{N-T+1}$ shares, we use the root lattice $\mathcal{L} = \{(s_1, \dots, s_S) \in \mathbb{Z}^S, \sum s_i = 0\}$ corresponding to the sharings of 0, and the goal of the user is to sample a Gaussian

vector in the appropriate coset of \mathcal{L} to share \mathbf{s} . While this can be done using the GPV sampler of Gentry et al. [36], using the parameters prescribed in [36] would result in shares with a significant standard deviation, causing ML-DSA verification to reject with very high probability and increasing communication by over $1000\times$. We address this by using Rényi divergence instead of statistical distance to obtain more compact parameters, as the sharing is used only once during key generation. We present an analysis of the smoothing parameter with Rényi divergence in the full version, which is of independent interest.

Security analysis. A caveat is that this key sharing reveals non-trivial information about the secret \mathbf{s} . We show in the full version how this is captured by the hint-MLWE problem. However, this hint-MLWE instance is non-standard: the secret \mathbf{s} comes from a uniform distribution (instead of Gaussian as in [45]), and the hint standard deviation is below the smoothing parameter of \mathbb{Z}^n . We prove that for Gaussian secrets, this still reduces efficiently to standard MLWE. We believe our generalized hint-MLWE problem is of independent interest.

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

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

Proof. The proof is identical to the one of Theorem 3.2, except for the games after Game $_{11}$.

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

This concludes the proof. \square

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

the secret key), while increasing the communication cost by at most a factor 10.

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

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

3.5 Parameter Selection

To maintain backward compatibility with ML-DSA, we keep the same public parameters listed in Table 2, introducing only the new parameters $K, r', r,$ and v (respectively, the number of parallel repetitions, the respective radii of the randomness \mathbf{r}'_i and rejected randomness \mathbf{z}_i , and the expansion factor for the first part of the rejected randomness $\mathbf{z}_i^{(1)}$). Following the security analysis in Section 3.2, we first choose ϕ such that $\varepsilon = I_{1-1/\phi^2} \left(\frac{n(k+\ell)+1}{2}, \frac{1}{2} \right) / 2 \leq 1/(KQ_s)$. Then, given an upper bound B on the norm of partial secrets (holding with overwhelming probability), and choosing $r'^2 \geq r^2 + B^2 + \frac{2rB}{\phi}$, 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 $B = 1.3 \cdot \sqrt{\tau} \cdot \sqrt{n \cdot (k + \ell/v^2)} \cdot \sqrt{\text{Var}(\mathcal{U}(-\eta, \eta))} \cdot \sqrt{\left[\binom{N}{T-1} / T \right]}$ as a heuristic overwhelming bound on the norm of partial secrets³. Moreover, `RSSRecover` ensures that each party uses at most $\left[\binom{N}{T-1} / T \right]$ secrets in a session for our selected parameters, c.f. Section 3.3. Finally, we select K, r, r' for each possible pair (T, N) with a manual search of the parameter space so as to minimize communication cost, while targeting a total success probability of at least $1/2$ for one protocol execution. We evaluate the success probability following The-

³We verify experimentally that $\|\text{sc}\|$ behaves like a Gaussian distribution and take a bound $1.3 \cdot \|\mathbf{s}\| \cdot \|\mathbf{c}\|$ 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 $\|\text{sc}\|$ only affects the Rényi divergence of rejection sampling.

orem 3.1, evaluating ν numerically.

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

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

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

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

Performance scaling rationale. Figure 9 shows that costs grow sharply once T or N increases. This behavior is driven by two compounding effects. (i) Let p be the probability a party accepts one local rejection step (for fixed r, r'). All T parties must accept together, so one joint attempt succeeds with probability about p^T ; the expected number of sequential retries therefore scales like p^{-T} . Using $K > 1$ parallel repetitions only trades extra bandwidth for lower latency—it cannot remove the inherent p^T factor. (ii) A partial secret may aggregate up to $\lceil \binom{N}{T-1} / T \rceil$ base secrets, inflating its norm (captured in B) and forcing larger radii r, r' . This aggregation overhead vanishes at $T = N$, where each party holds exactly one base secret, explaining the mild improvement in that case. Plugging this asymptotic behavior in the analysis of rejection sampling from [25] leads to a rough asymptotic estimate of our scheme’s efficiency as $\exp\left(O\left(T \cdot \sqrt{\binom{N}{T-1}}\right)\right)$.

We provide the full parameters of Threshold ML-DSA in the full version, including the parameters (r, r', K, ν) for each security level of ML-DSA as well as the communication costs for each threshold $2 \leq T \leq N \leq 6$. The parameters selected ensure a success probability at least $1/2$ for one protocol execution. For Threshold ML-DSA-44, we include a visual representation of the bandwidth and runtime costs in Fig. 9.

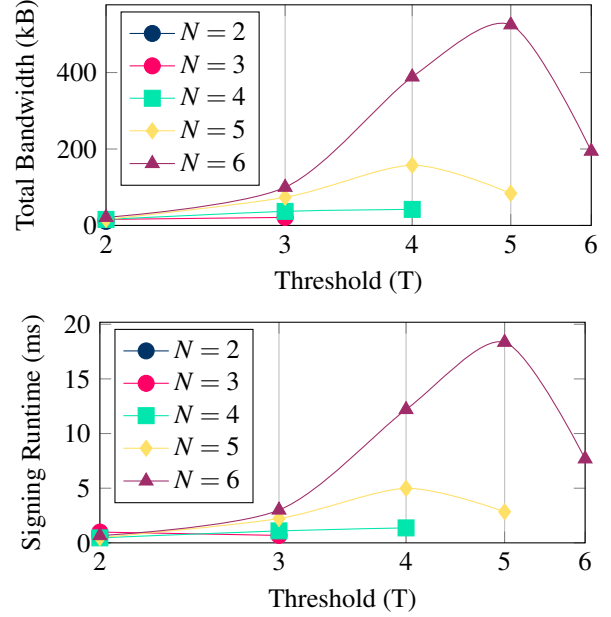


Figure 9: Bandwidth costs (top) and runtime signing costs (bottom) per party per signing attempt for Threshold ML-DSA-44. Note that values for $N = 2$ and $N = 3$ overlap for all T .

4 Performance Evaluation

We provide implementation details and experimental evaluation of our Threshold ML-DSA scheme. We fully implemented our scheme to ease integration with other systems and libraries. We chose `Go` for its popularity in cryptographic implementations and low incidence of security issues [48, 8]. Our implementation of Threshold ML-DSA builds on the code of the CIRCL library [32], that we extended to support our threshold variant, including our secret-sharing functionality. We implemented Threshold ML-DSA for all the parameters sets (44, 65, 87) defined in [51].

4.1 Experimental Analysis

We evaluated the performance of Threshold ML-DSA locally (for 44, 65, 87), as well as in LAN and WAN settings (only 44). All parties ran on a consumer-grade MacBook M3 for local and LAN experiments with 24 GB RAM. We conducted single-threaded local simulations of our scheme to benchmark key generation, signing (across parallel rounds), and verification. Each party’s computation was emulated in a single process, capturing fine-grained timing data for each protocol phase. The runtime (i.e., computational cost) for various (T, N) values is shown in Table 4, for the parameter set 44 of Threshold ML-DSA, as well as visualizations of its bandwidth and runtime in Fig. 9. We provide the results for the parameter sets 65 and 87 in the full version.

For our WAN and LAN experiments, we implemented

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

the communication layer of our scheme using the library `libp2p` [57, 56], a modular peer-to-peer networking stack. Each party acts as an independent `libp2p` peer, with its identity embedded as its network address, and communicates via a custom protocol over a direct TCP stream. Parties listen for incoming connections on a specified TCP port, and others connect using multiaddress. After establishing a connection, they exchange messages over a persistent bidirectional `libp2p` stream. This setup ensures execution over a realistic asynchronous, decentralized network, with `libp2p` handling connection management, peer identity, and stream multiplexing. NAT traversal and relaying were avoided for simplicity, assuming direct TCP connectivity.

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

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

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

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

via public IPs and in parallel according to best practices in distributed systems.

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

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

We report in our full version the median latencies (in ms) between the AWS EC2 instances used in our experiments. WAN measurements reflect the slowest communication path (the "critical path" [28]), consistently between Taipei and Virginia, to capture worst-case delays. In real-world cloud deployments, infrastructure optimizations (e.g., load balancing, anycast routing) may reduce delays and overhead.

The measured signing latencies for Threshold ML-DSA are within milliseconds, even in WAN setting over multiple continents, demonstrating our schemes' practicality in real-world distributed systems. While our scheme require three rounds for a signing operation, these rounds are not executed sequentially in a blocking manner. Instead, messages are sent and processed in parallel across all parties. As a result, the observed latency reflects the most extended communication delay in each round's critical path—usually between the physically furthest machines—rather than the *sum* of point-to-point delays. Our costs are well under a second, far below a typical TLS handshake timeout of 10-20s [41].

4.2 Comparison with Prior Work

In this section, we compare our Threshold ML-DSA scheme with recent works on lattice-based threshold signatures. We include threshold variants of Raccoon: T-Raccoon [21, 42] and Ringtail [9]. We also compare with the recent work on ML-DSA [7], which distributes ML-DSA signing with MPC techniques in the honest-majority setting.

Raccoon threshold variants. Raccoon [23] is a lattice-based signature scheme that has proven particularly amenable to threshold adaptations. By removing the rejection-sampling step present in ML-DSA, it enables highly efficient threshold constructions, at the cost of larger signatures (approximately 10 kB). A notable advantage of these variants is that they scale well to larger thresholds, since the per-party signing bandwidth remains constant across different values of (T, N) .

We provide a high-level comparison with T-Raccoon and Ringtail in Table 7, and report local efficiency measurements in Table 8 for NIST level I. Our Threshold ML-DSA

scheme has higher round complexity than both T-Raccoon and Ringtail, and exhibits comparable or, in some cases, higher—communication costs for supported threshold sizes. On the other hand, our scheme achieves faster signing times, although this is likely attributable to implementation differences: our implementation builds on the ML-DSA code from CIRCL [32], whereas the Raccoon variants rely on the more general ring-based infrastructure provided by Lattigo [1].

Scheme	N_{\max}	Rounds	Comm.	vk	sig
<i>Ours</i>	6	6	21–1005	1.3	2.4
T-Raccoon	1024	3	40	3.8	12.4
Ringtail	1024	2	613	4.5	13.4

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

ML-DSA threshold variant. We also compare against the concurrent work of [7], which proposes a threshold variant of ML-DSA in the *honest-majority* setting using MPC techniques. Their construction supports a large number of parties, but this comes at the cost of a high round complexity and potentially significant overhead for generating the required correlated randomness. The authors report benchmarks obtained using SCL [17] on a single thread of a machine with 32 GB of RAM and an Intel i7 processor running at 2.5 GHz. However, no open-source implementation is available, and the prototype appears to rely on extensive low-level optimizations and is implemented in C++. In contrast, our scheme is not heavily optimized, is implemented in GoLang, and includes a fully open-source implementation. Consequently, in Table 9, we report only the performance numbers given in [7], as we are unable to reproduce their benchmarks.

4.3 Applications and Discussion

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

Post-quantum PKI and TLS infrastructure. Modern TLS deployments using Content Delivery Networks (CDNs) face critical security challenges when distributing private key material across edge servers. While traditional approaches like Keyless SSL [53] proxy private key operations to origin

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

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

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

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

servers, threshold signatures represent an increasingly attractive alternative that avoids single points of failure [38, 13, 26]. As quantum computers threaten classical cryptography, systems must transition to post-quantum alternatives while maintaining the operational benefits of distributed signing.

Our Threshold ML-DSA scheme addresses this transition by providing ML-DSA-compatible threshold signatures that preserve standardized signature formats—ensuring seamless integration with existing infrastructure. A typical 3-of-5 deployment can tolerate two server failures while maintaining signing capability, with our measurements showing sub-second signing latencies across continents, well within TLS timeout requirements of 10-20s [41].

Cryptocurrency wallet security. Multi-device cryptocurrency wallets [31] represent another application where ML-DSA compatibility is essential for quantum-resistant security. Recent work [50] shows that users prefer fewer, highly reputable parties and lower threshold values for higher availabil-

ity. Our scheme’s performance characteristics align well with these preferences (typically 2-of-3 or 3-of-5) while providing a desirable stateless design [47, 33] that avoids storage and synchronization issues.

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

Critical infrastructure and Tor network. The Tor network [54] serves millions of users daily and requires fresh random values generated by trusted directory authorities⁴. Current distributed random generation techniques are vulnerable to attacks [52], and proposals for threshold-signed consensus documents [39] have stalled due to the absence of practical, standardized threshold schemes.

Our Threshold ML-DSA implementation directly addresses this gap, providing post-quantum security for the long-term integrity of the Tor network. The scheme operates efficiently in the small-threshold setting (we note that the number of directory authorities is approximately 10 [55]; although this number is subject to change over time, it has historically remained small enough for our scheme to be applicable). Moreover, our experimental results demonstrate sub-second signing latencies across continents, comfortably within the requirements for hourly consensus document generation⁵.

5 Acknowledgments

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

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

Ethical Considerations

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

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

System Designers: System designers gain access to practical threshold post-quantum signatures, addressing a critical need as distributed systems transition to quantum-resistant

cryptography. Our scheme enables secure and efficient distributed signing protocols while maintaining the security properties of classical distributed systems.

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

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

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

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

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

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

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

Risk Assessment and Mitigation

Potential Harms:

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

Mitigations Implemented:

- Formal security proofs based on the same assumptions as ML-DSA, ensuring our scheme matches the security of this heavily scrutinized NIST standard.

⁴See <https://spec.torproject.org/rend-spec/shared-random.html>.

⁵<https://support.torproject.org/glossary/consensus/>

- Conservative parameter selection based on statistical analysis, reducing the risk of parameter misselection.
- Open-source implementation that enables community audit and verification.

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

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

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

Open Science

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

Our repository includes: (i) a complete Threshold ML-DSA implementation based on CIRCL [32], (ii) `libp2p` integration for LAN/WAN experiments, (iii) local benchmarking tools, and (iv) parameter selection scripts. All experimental results can be reproduced using the provided artifact.

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

References

- [1] Lattigo v6. Online: <https://github.com/tuneinsight/lattigo>, August 2024. EPFL-LDS, Tune Insight SA.
- [2] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPICs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.
- [3] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Cryptology ePrint Archive*, Report 2015/046, 2015.
- [4] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 3–24. Springer, Berlin, Heidelberg, November / December 2015.
- [5] Henry Bambury, Hugo Beguinet, Thomas Ricosset, and Éric Sageloli. Polytopes in the fiat-shamir with aborts paradigm. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 339–372. Springer, Cham, August 2024.
- [6] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022.
- [7] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. Efficient, scalable threshold ml-dsa signatures: An mpc approach, 2025.
- [8] Jenny Blessing, Michael A. Specter, and Daniel J. Weitzner. Cryptography in the wild: An empirical analysis of vulnerabilities in cryptographic libraries. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '24, page 605–620, New York, NY, USA, 2024. Association for Computing Machinery.
- [9] Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. Ringtail: Practical two-round threshold signatures from learning with errors. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 149–164. IEEE Computer Society Press, May 2025.
- [10] 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.
- [11] 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.

- [12] Guilhem Castagnos, Dario Catalano, Fabien Laguillamie, 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.
- [13] Sofia Celi, Daniel Escudero, and Guilhem Niot. Share the MAYO: thresholdizing MAYO. Cryptology ePrint Archive, Paper 2024/1960, 2024.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] Anders Dalskov. SCL (Secure Computation Library) — utility library for prototyping MPC applications. <https://github.com/anderspkd/secure-computation-library>. Accessed: 2025-12-11.
- [18] 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.
- [19] Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Méliissa 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>.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Méliissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 409–444. Springer, Cham, August 2024.
- [24] 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.
- [25] 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.
- [26] Jack Doerner, Yashvanth Kondi, and Leah Namisa Rosenbloom. Sometimes you can’t distribute random-oracle-based proofs. NIST presentation, 2023.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerge: 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.

- [31] Ittay Eyal. On cryptocurrency wallet design. Cryptology ePrint Archive, Report 2021/1522, 2021.
- [32] 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.
- [33] 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.
- [34] Marc Fischlin, Aikaterini Mitrokotsa, and Jenit Tomy. BUFFing threshold signature schemes. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 137–168. Springer, Cham, May 2025.
- [35] 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.
- [36] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
- [37] 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.
- [38] Armando Faz Hernandez. Requirements for threshold tls. NIST presentation, 2023.
- [39] Nicholas Hopper. A threshold signature-based proposal for a shared rng. Tor Dev Mailing List, 2014.
- [40] 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.
- [41] IBM. Handshake timer, 2023. <https://www.ibm.com/docs/en/zos/3.1.0?topic=considerations-handshake-timer>. Accessed 06/02/24.
- [42] Shuichi Katsumata, Michael Reichle, and Kaoru Take-mure. 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.
- [43] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. Aggregating and thresholding 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.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] 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.
- [48] 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.
- [49] 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.
- [50] 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.

- [51] National Institute of Standards and Technology. FIPS 204: Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standard (FIPS) Publication NIST FIPS 204, U.S. Department of Commerce, National Institute of Standards and Technology, August 2024.
- [52] The Tor Project. Security analysis. Tor Specifications, 2024.
- [53] Douglas Stebila and Nick Sullivan. An Analysis of TLS Handshake Proxying . In *2015 IEEE Trust-com/BigDataSE/IPA*, pages 279–286, Los Alamitos, CA, USA, August 2015. IEEE Computer Society.
- [54] Inc The Tor Project. Tor metrics, 2024. <https://metrics.torproject.org/>.
- [55] Tor Project. Directory authority expectations. https://community.torproject.org/policies/dir-auth/dir_auth_expectations/, 2023. Accessed: 2025-08-25.
- [56] 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.
- [57] 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.