ML-DSA (CRYSTALS-Dilithium) Threshold Signing Schemes

A Comprehensive Research Report

State of the Art, Challenges, and Implications for OPNet

February 2026

Prepared for: OPNet Research Division

Based on literature survey of ePrint archive, NIST publications, and major cryptography conferences through February 2026.

Table of Contents

1. Introduction

Digital signatures are among the most fundamental cryptographic primitives, providing authentication, integrity, and non-repudiation guarantees that underpin the security of modern digital infrastructure. In traditional signature schemes, a single private key is held by one entity. However, entrusting the entire signing capability to a single key creates a critical single point of failure: if the key is compromised, lost, or its holder becomes unavailable, the entire system's security collapses.

Threshold signatures address this problem by distributing the signing key among *N* parties such that any subset of at least *t* parties can collaboratively produce a valid signature, while fewer than *t* parties learn nothing about the key. This *(t, N)*-threshold paradigm provides both **fault tolerance** (the system remains operational even if some parties are offline) and **enhanced security** (an adversary must compromise multiple parties to forge signatures). Threshold signatures have found wide deployment in cryptocurrency custody, certificate authorities, distributed ledger technologies, and secure multi-party protocols [1, 2].

In the pre-quantum setting, threshold signatures for ECDSA and Schnorr-like schemes have reached a mature state, with efficient protocols such as FROST [3], GG18/GG20 [4], and CGGMP [5] seeing production deployments in cryptocurrency wallets and key management systems. The algebraic structure of elliptic curve groups—particularly the linearity of Schnorr signatures—makes thresholdization relatively natural: secret key shares can be combined via Lagrange interpolation, and partial signatures aggregate through simple group operations.

The looming threat of quantum computers, however, necessitates a transition to post-quantum cryptographic (PQC) primitives. The U.S. National Institute of Standards and Technology (NIST) standardized **ML-DSA** (Module-Lattice-Based Digital Signature Algorithm), formerly known as CRYSTALS-Dilithium, as FIPS 204 in August 2024 [6]. ML-DSA is expected to become the most widely deployed post-quantum signature scheme. Yet, constructing efficient threshold versions of ML-DSA has proven to be significantly more challenging than for pre-quantum schemes, due to fundamental structural differences in lattice-based cryptography.

This report provides a comprehensive survey of the current state of threshold ML-DSA/Dilithium research. We examine the core technical challenges that make lattice-based threshold signing difficult, survey all major published proposals from 2018 through early 2026, compare their properties and trade-offs, catalog existing implementations, and assess the implications for OPNet's use of ML-DSA in its smart contract infrastructure.

2. Background

2.1 Lattice-Based Cryptography Primer

Lattice-based cryptography derives its security from the presumed hardness of problems defined over mathematical lattices—regular, discrete subgroups of $\blacksquare\blacksquare$. A lattice Λ can be described by a basis $\mathbf{B} \in \blacksquare^{m \times n}$ as the set of all integer linear combinations of the basis vectors. The two principal hard problems are:

• **Learning With Errors (LWE)**: Given a matrix $\mathbf{A} \in \blacksquare_q^{m \times n}$ and a vector $\mathbf{b} = \mathbf{As} + \mathbf{e} \pmod q$ where $\mathbf{s}$ is a secret vector and $\mathbf{e}$ is a "small" error vector sampled from a discrete Gaussian, distinguish ($\mathbf{A}$, $\mathbf{b}$) from uniformly random. The module variant (MLWE) works over polynomial rings $R_q = \blacksquare_q[X]/(X^n+1)$ for efficiency.

• **Short Integer Solution (SIS)**: Given $\mathbf{A} \in \blacksquare_q^{m \times n}$, find a non-zero "short" vector $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{0} \pmod q$. The module variant (MSIS) provides the unforgeability foundation for ML-DSA.

These problems are believed to be hard even for quantum computers. Worst-case to average-case reductions by Ajtai (1996) and Regev (2005) provide strong theoretical foundations: solving average-case instances of LWE/SIS implies solving worst-case lattice problems such as the Shortest Vector Problem (SVP) within polynomial approximation factors [7, 8].

For practical efficiency, ML-DSA and related schemes use **module lattices** over the polynomial ring $R_q = \blacksquare_q[X]/(X^{256}+1)$ with $q = 8380417$. Operations in this ring can be performed in $O(n \log n)$ time using the Number Theoretic Transform (NTT), making module-lattice schemes significantly faster than their unstructured counterparts.

2.2 ML-DSA / CRYSTALS-Dilithium Structure

ML-DSA follows the **Fiat-Shamir with Aborts** paradigm, introduced by Lyubashevsky (2009, 2012) [9]. This paradigm adapts the classical Fiat-Shamir transform to the lattice setting, but introduces a crucial rejection sampling step that is central to security—and central to the difficulties in threshold construction. The three algorithms are:

Key Generation (ML-DSA.KeyGen)

1. Sample a uniform matrix $\mathbf{A} \in R_q^{k \times \blacksquare}$.
2. Sample short secret vectors $\mathbf{s}_1 \in R_q^{\blacksquare}$ and $\mathbf{s}_2 \in R_q^{k}$ with coefficients bounded by η.
3. Compute $\mathbf{t} = \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2$.
4. The public key is pk = $(\rho, \mathbf{t}_1)$ where ρ is a seed for $\mathbf{A}$ and $\mathbf{t}_1$ is the high-order bits of $\mathbf{t}$. The secret key contains $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$ where $\mathbf{t}_0$ holds the low-order bits.

Signing (ML-DSA.Sign)

1. Compute $\mu = H(\text{pk} \blacksquare \text{msg})$ and a commitment randomness ρ'.
2. **Repeat**: Sample a masking vector $\mathbf{y}$ with coefficients bounded by γ_1-1. Compute $\mathbf{w} = \mathbf{Ay}$, extract high bits $\mathbf{w}_1$. Compute the challenge $c\blacksquare = H(\mu \blacksquare \mathbf{w}_1)$, derive challenge polynomial $c \in R_q$ with τ non-zero coefficients of ±1. Compute $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{s}_1$.
3. **Rejection check**: If $\blacksquare\mathbf{z}\blacksquare_\infty \geq \gamma_1 - \beta$ or the low-order bits of $\mathbf{A} \cdot \mathbf{z} - c \cdot \mathbf{t}$ are too large, **restart from step 2**.
4. Output signature $\sigma = (c\blacksquare, \mathbf{z}, \mathbf{h})$ where $\mathbf{h}$ encodes hint bits.

The critical insight is the **rejection sampling** in step 3: the distribution of **z** must be statistically independent of the secret key $\mathbf{s}_1$. If the norm of **z** is too large, the relationship $\mathbf{z} = \mathbf{y} + c\cdot\mathbf{s}_1$ could leak information about $\mathbf{s}_1$. The expected number of iterations before acceptance is approximately 4–7 for ML-DSA parameters [6].

Verification (ML-DSA.Verify)

Given (pk, msg, σ), reconstruct **A**, recompute $\mathbf{w}_1'$ from $\mathbf{A}\cdot\mathbf{z} - c\cdot\mathbf{t}$ using the hints **h**, and check that c■ = H(μ ■ $\mathbf{w}_1'$) and that ■**z**■$_\infty < \gamma_1 - \beta$. Verification is deterministic and efficient.

2.3 NIST FIPS 204

FIPS 204, published on August 13, 2024, standardizes ML-DSA at three security levels [6]:

Parameter	ML-DSA-44	ML-DSA-65	ML-DSA-87
Security Level	NIST Level 2 (~128-bit)	NIST Level 3 (~192-bit)	NIST Level 5 (~256-bit)
(k, ■)	(4, 4)	(6, 5)	(8, 7)
Public Key Size	1,312 bytes	1,952 bytes	2,592 bytes
Signature Size	2,420 bytes	3,309 bytes	4,627 bytes
Secret Key Size	2,560 bytes	4,032 bytes	4,896 bytes

The standard mandates deterministic signing for security (hedged mode is optional), specifies exact rejection bounds, and requires that implementations produce bit-for-bit identical outputs for given inputs. This determinism requirement has important implications for threshold constructions, as discussed in Section 3.

3. Challenges of Threshold ML-DSA

Constructing threshold versions of lattice-based signatures is fundamentally harder than for discrete-log-based schemes. We identify four core challenges:

3.1 Rejection Sampling and Abort Probability

The Fiat-Shamir with Aborts paradigm requires that the final signature vector $\mathbf{z} = \mathbf{y} + c\cdot\mathbf{s}$ be statistically independent of the secret $\mathbf{s}$. This is enforced by rejection sampling: if $\mathbf{z}$ falls outside a prescribed range, the entire signing attempt is aborted and restarted from scratch with fresh randomness.

In a standard (single-signer) ML-DSA, this is manageable—the expected number of attempts is ~4.25 for ML-DSA-44 and ~5.1 for ML-DSA-65. However, in a threshold setting where *t* parties must coordinate, the situation becomes dramatically worse:

• **Correlated aborts**: Each party's partial signature involves their secret share and their masking share. The rejection check must be performed on the *combined* signature, but partial information from individual shares could leak secrets. If rejection sampling is performed independently by each party, the combined abort probability grows exponentially: if each party aborts with probability p, the probability that all t parties succeed simultaneously is $(1-p)^t$, which can become vanishingly small.

• **Selective abort attacks**: A malicious party can deliberately cause aborts to extract information about honest parties' shares. Over many signing sessions with controlled aborts, the attacker can statistically reconstruct secret key material. This necessitates either identifiable abort mechanisms (so malicious parties can be detected and excluded) or more expensive zero-knowledge techniques to prove correct behavior.

• **Incompatibility with pre-computation**: Unlike Schnorr-based threshold schemes where nonce shares can be pre-computed and stored, the abort mechanism in ML-DSA means that pre-computed values may be wasted, reducing the effectiveness of offline/online splitting.

3.2 Noise Flooding Requirements

In pre-quantum threshold signatures, proving security of partial signature simulation is straightforward because group elements perfectly mask individual shares. In the lattice setting, shares and partial signatures are "noisy" vectors over integers or polynomial rings, and the noise distributions carry information about secrets.

Noise flooding is a technique where each party adds extra random noise (much larger than the "natural" noise in the system) to their partial output, so that the natural noise is statistically drowned out. Specifically, if natural noise has norm B, noise flooding adds noise of magnitude $2^{\lambda}\cdot B$ where λ is the security parameter. This guarantees that the partial outputs are statistically close to being independent of the secrets [10].

The cost of noise flooding is severe: the signature components become much larger (by a factor of 2^{λ}), drastically increasing signature sizes from kilobytes to potentially megabytes. Early constructions by Boneh et al. (CRYPTO 2018) [11] suffered signature sizes of $\Omega(\lambda^3)$ bits. The work of Agrawal, Stehlé, and Yadav (2021) [12] reduced this to $\tilde{O}(\lambda)$ using ring structure, but the concrete constants remain large.

A major research direction has been to reduce or eliminate the need for noise flooding, either through careful protocol design (e.g., the Raccoon approach), use of fully homomorphic encryption (FHE) to perform rejection sampling in encrypted form, or novel algebraic techniques for distributed key generation with short shares.

3.3 Absence of Simple Linear Secret Sharing

For ECDSA and Schnorr signatures, secret key shares can be distributed using Shamir's secret sharing over a finite field, and reconstructed via Lagrange interpolation. The reconstruction is a linear operation that commutes with the group operations used in signing. This linearity is what makes threshold Schnorr/FROST so elegant: each party computes a partial signature, and these are combined via the same Lagrange coefficients.

In ML-DSA, the secret key $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ consists of short vectors over polynomial rings. While one could still use Shamir sharing over the ring R_q, the critical constraint is that the **shortness** of shares must be preserved. Lagrange interpolation coefficients in R_q are not ±1—they are general polynomials. Multiplying a short share by a large Lagrange coefficient produces a long vector, which violates the norm bounds required by the rejection sampling check and can leak secret information [13].

This has led to two broad approaches: (a) additive secret sharing where each party simply holds one of n shares that sum to the secret (works naturally for n-out-of-n but requires additional techniques for general t-of-n), and (b) novel DKG protocols that produce shares with specifically controlled shortness properties.

3.4 Distributed Key Generation Complexity

Distributed Key Generation (DKG) for lattice-based schemes is significantly more complex than for discrete-log-based schemes. In DL-based DKG (e.g., Pedersen DKG), each party generates a random polynomial and broadcasts commitments using Pedersen commitments, which are perfectly hiding and computationally binding. Verification is efficient.

For lattice-based DKG, several challenges arise:

• **Verifiable Secret Sharing (VSS)**: Lattice-based VSS schemes exist but typically require either large proof sizes (e.g., based on lattice-based commitments) or computational assumptions beyond MLWE/MSIS. Efficient constructions often rely on specific lattice trapdoor structures.
• **Short share maintenance**: The DKG must produce shares that are "short" enough for the signing protocol to work. Standard Shamir sharing over R_q does not guarantee this.
• **Robustness**: Achieving robustness (ability to complete DKG even with malicious parties) typically requires complaint/dispute resolution phases with additional communication rounds.
• **Asynchronous DKG**: Recent work by Yang et al. (2025) [14] addresses asynchronous lattice-based DKG, but with significant overhead compared to synchronous protocols.

The breakthrough by del Pino, Espitau, Niot, and Prest (CRYPTO 2024) [13] introduced Distributed Key Generation with Short Shares (sDKG), which ensures that shares and Lagrange reconstruction coefficients remain short. This was a critical enabler for practical threshold lattice signatures.

4. Existing Research and Proposals

We present a chronological survey of the major research contributions to threshold lattice-based signatures, with emphasis on those directly applicable to ML-DSA/Dilithium.

4.1 Early Foundations (2018–2021)

Boneh, Komlo, et al. (CRYPTO 2018) [11]: Provided the first lattice-based threshold signature scheme from standard assumptions (LWE). This foundational work demonstrated feasibility but relied on heavy noise flooding ($2^{\Omega(\lambda)}$), producing impractical signature sizes. Only selective security was achieved.

Cozzo and Smart (2019) [15]: In "Sharing the LUOV: Threshold Post-Quantum Signatures," the authors explored thresholdizing the LUOV (Lifted Unbalanced Oil and Vinegar) signature scheme, an MQ-based post-quantum candidate. While not directly about Dilithium, this was one of the first systematic studies of threshold post-quantum signatures and highlighted the general difficulties of the PQ threshold setting, including the absence of efficient linear secret-sharing structures found in pre-quantum schemes.

Damgård, Orlandi, Takahashi, and Tibouchi (2021) [16]: "Two-round n-out-of-n and Multi-Signatures and Trapdoor Commitment from Lattices" presented a two-round n-out-of-n threshold signature and multi-signature scheme from lattice assumptions. The key technique was a novel lattice-based trapdoor commitment scheme that enables efficient two-round protocols. This work was limited to the full-threshold (t=n) case and produced signatures significantly larger than standard Dilithium, but established important techniques for the field. Published at PKC 2021.

Agrawal, Stehlé, and Yadav (2021) [12]: "Towards Practical and Round-Optimal Lattice-Based Threshold and Blind Signatures" (ePrint 2021/381) significantly improved the state of the art by reducing noise flooding from $2^{\Omega(\lambda)}$ to $\sqrt{Q_S}$ where Q_S bounds the number of signing queries. Using ring structure, they achieved threshold signatures of ~3 KB using a Dilithium-G variant at 128-bit security, for adversaries limited to 256 signatures. They also improved round optimality and advanced toward adaptive security.

4.2 Two-Party Constructions

The 2-out-of-2 case is of particular practical interest for client-server models (e.g., a phone and a cloud server jointly signing), and has received dedicated attention:

DiLizium (Laud and Snetkov, 2022–2023) [17]: "DiLizium" and its improved version "DiLizium 2.0: Revisiting Two-Party CRYSTALS-Dilithium" proposed protocols specifically for two-party Dilithium signing. In DiLizium 2.0, the two parties (typically a phone and a server) additively share the secret key components and collaboratively perform the signing operation, handling the rejection sampling step through careful protocol design. Security is proven against a malicious server or phone individually. The authors also provided concrete benchmarks showing practical performance for the 2-party setting.

Trilithium (Dufka, Kravtšenko, Laud, and Snetkov, 2025) [18]: Building on DiLizium, Trilithium presents a protocol for distributed key generation and signing compliant with FIPS 204 (ML-DSA). The protocol allows a "server" and "phone" with assistance from a Correlated Randomness Provider (CRP) to produce a standard ML-DSA signature. Crucially, the output is a **valid FIPS 204 signature**—indistinguishable from one produced by a single signer. Security is proved in the Universal Composability (UC) framework against a malicious server or phone. Novel techniques are introduced for arguing security of two-party protocols with private outputs (where each party's output depends on both parties' inputs).

TOPCOAT (ISRI-PQC, 2024–2025) [19]: A two-party lattice-based signature scheme that embodies ML-DSA compression techniques. TOPCOAT does not produce standard ML-DSA signatures but uses similar algebraic structures and achieves compact two-party threshold signing. An open-source Go implementation is available.

4.3 Threshold FHE Approaches

Gur, Katz, and Silde (PQCrypto 2024) [20]: "Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption" (ePrint 2023/1318) presented a breakthrough: a two-round threshold signature scheme based on standard lattice assumptions supporting *arbitrary thresholds* $t \leq n$ (not just $t=n$). The key idea is to use threshold (linearly) homomorphic encryption (THE) to perform the rejection sampling check in encrypted form. Each party encrypts their partial signature under the THE scheme, the encrypted partial signatures are combined, the rejection check is performed on the encrypted result, and only if the check passes is the result decrypted.

Performance estimates at 128-bit security for a 3-out-of-5 threshold yielded signature sizes of ~46.6 KB and public key sizes of ~13.6 KB—dramatically larger than standard ML-DSA but a significant improvement over prior work. The construction achieves ~5× better parameters if only a limited number of signatures are issued per key.

Espitau, Katsumata, and Takemure (CRYPTO 2024) [21]: "Two-Round Threshold Signature from Algebraic One-More Learning with Errors" improved on the THE approach by introducing a novel hardness assumption (Algebraic One-More LWE, AOM-LWE) to achieve more efficient two-round threshold signatures. This work was subsequently strengthened by Zhu and Tessaro [22] who showed that the AOM-MISIS variant of this assumption can be reduced to standard MSIS and MLWE.

4.4 Raccoon and Threshold-Friendly Designs

A pivotal development has been the design of signature schemes that are *inherently threshold-friendly*, rather than attempting to thresholdize the existing ML-DSA specification:

Threshold Raccoon (del Pino, Katsumata, Maller, Mouhartem, Prest, Saarinen; Eurocrypt 2024) [23]: Raccoon is a new lattice-based signature scheme specifically designed for threshold operation. It achieves signature sizes of only ~13 KB (compared to 2.4 KB for ML-DSA-44), with a 3-round signing protocol, supporting arbitrary (t, N) thresholds. Raccoon was submitted as an additional signature candidate in NIST’s PQC process. The design avoids the rejection sampling problem entirely by using a masking technique where the distribution of the final signature is independent of individual shares by construction, rather than by rejection. This eliminates the abort problem but results in somewhat larger signatures.

Unmasking TRaccoon (del Pino, Katsumata, Niot, Reichle, Takemure; 2024) [24]: Extended Threshold Raccoon with an efficient identifiable abort protocol. The original TRaccoon used masking to hide partial signatures, which prevented identifying malicious parties causing failures. This work introduced techniques to unmask the identity of misbehaving parties while maintaining security, a crucial feature for practical deployment.

Olingo (Gur, Hough, Katz, Sandsbråten, Silde; 2025) [25]: A framework for threshold lattice signatures that is the first to offer *all* desired properties simultaneously: small keys and signatures, low communication and round complexity, non-interactive online signing, distributed key generation (DKG), and identifiable abort. Olingo builds on the Gur-Katz-Silde framework (PQCrypto 2024) but switches the underlying signature scheme to Raccoon, achieving practical efficiency.

4.5 ML-DSA Compatible Schemes (2025–2026)

The most recent and practically significant development is the construction of threshold schemes that produce **standard FIPS 204 ML-DSA signatures**:

"Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts" (del Pino, Espitau, Niot, Prest; CRYPTO 2024) [13]: This landmark paper introduced Distributed Key Generation with Short Shares (sDKG), ensuring that shares and Lagrange reconstruction coefficients remain short. The key innovation is that "signature shares double as valid signatures under the corresponding secret key shares"—realizing the "threshold designer's dream." The two-round signing protocol achieves signature shares of only 4.4 KB (for Raccoon-128). An identifiable abort variant adds only a few extra kilobytes of communication.

"Efficient Threshold ML-DSA" (Celi, del Pino, Espitau, Niot, Prest; 2025) [26]: This paper directly targets the NIST standard, presenting a threshold scheme whose output is a valid ML-DSA signature under the standard verification algorithm. The scheme supports up to 6 parties efficiently, with practical DKG and signing protocols. The Go implementation based on Cloudflare's CIRCL library demonstrates feasibility. LAN/WAN experiments show acceptable latencies for practical deployment scenarios.

"Threshold Signatures Reloaded: ML-DSA and Enhanced Raccoon with Identifiable Aborts" (Borin, Celi, del Pino, Espitau, Niot, Prest; 2025–2026) [27]: The most recent work explores the full design space of lattice-based threshold signatures, providing both a threshold ML-DSA construction (producing FIPS 204 compliant signatures) and an enhanced version of Threshold Raccoon with identifiable aborts. The paper includes detailed parameter selection, concrete security analysis, and comprehensive benchmarks.

"Quorus: Efficient, Scalable Threshold ML-DSA Signatures from MPC" (Bienstock, de Castro, Escudero, Polychroniadou, Takahashi; 2025–2026) [28]: Takes a fundamentally different approach by using generic MPC (secure multi-party computation) techniques to compute ML-DSA signing as a distributed function evaluation. Unlike the algebraic approaches above, Quorus treats ML-DSA signing as a black-box function and uses optimized MPC protocols (based on SPDZ/Overdrive) to evaluate it. This approach naturally supports arbitrary thresholds and produces standard ML-DSA signatures, but incurs higher communication costs. The advantage is that it requires no modification to the signature scheme and directly benefits from MPC optimization research.

4.6 NIST's Position on Threshold PQC

NIST has been actively engaged in the threshold cryptography space through its Multi-Party Threshold Cryptography (MPTC) project. Key developments include:

• **NISTIR 8214 series**: A series of reports (8214, 8214A, 8214B, 8214C) establishing the framework for threshold cryptography standardization.
• **Call for Threshold Schemes (NISTIR 8214C, 2nd public draft, March 2025)** [29]: NIST issued a formal call for multi-party threshold scheme submissions, with the final version expected in 2025 and submissions accepted thereafter. The call explicitly includes post-quantum primitives: Class N covers NIST-specified primitives including ML-DSA, ML-KEM, and SLH-DSA. Class S covers "threshold-friendlier" alternatives and auxiliary tools (FHE, ZKP, gadgets).

• **Workshops**: NTCW 2019, MPTS 2020, MPTS 2023, and WPEC 2024 have included sessions on threshold post-quantum signatures, demonstrating sustained institutional interest.
• **Timeline**: NIST has indicated that threshold scheme standardization will follow the main PQC standardization, with initial standards expected no earlier than 2027–2028.

5. Comparison of Approaches

The following table compares the major threshold lattice-based signature proposals across key dimensions. We distinguish between schemes that produce standard ML-DSA signatures and those that use custom formats.

Scheme	Year	Threshold	Rounds	Sig Size	ML-DSA Compat.	ID Abort	DKG	Impl.
Boneh et al.	2018	t-of-n	1	$\Omega(\lambda^3)$ bits	No	No	No	No
Damgård et al.	2021	n-of-n	2	~100 KB	No	No	No	No
Agrawal et al.	2021	t-of-n	1	~3 KB*	No	No	No	No
DiLizium 2.0	2023	2-of-2	3–4	~2.4 KB	Partial	N/A	Yes	PoC
Gur-Katz-Silde	2024	t-of-n	2	~46.6 KB	No	No	No	No
T-Raccoon	2024	t-of-n	3	~13 KB	No	No**	Yes	Go
del Pino et al. (sDKG)	2024	t-of-n	2	~4.4 KB	No†	Yes	Yes (sDKG)	No
Trilithium	2025	2-of-2+CRP	2–3	~2.4 KB	**Yes**	N/A	Yes	PoC
Efficient T-ML-DSA	2025	t-of-n (≤6)	2–3	~2.4 KB	**Yes**	Yes	Yes	**Go (CIRCL)**
T-Sigs Reloaded	2025–26	t-of-n	2–3	~2.4 KB	**Yes**	Yes	Yes	**Go**
Quorus (MPC)	2025–26	t-of-n	Many	~2.4 KB	**Yes**	Via MPC	Via MPC	Research
Olingo	2025	t-of-n	2 (online: 1)	~13 KB	No††	Yes	Yes	Research

* Limited to 256 signatures per key. ** Identifiable abort added in subsequent work (Unmasking TRaccoon). † Uses Raccoon-based custom scheme, not FIPS 204. †† Uses Raccoon signatures.

The comparison reveals a clear trajectory: from impractical theoretical constructions in 2018–2021, through specialized two-party and FHE-based approaches in 2022–2024, to the current state where **ML-DSA compatible threshold signatures are achievable with practical performance for moderate party counts**. The most mature constructions—"Efficient Threshold ML-DSA" and "Threshold Signatures Reloaded"—produce standard FIPS 204 signatures with full DKG, identifiable abort, and open-source implementations.

Key trade-offs include:

• **ML-DSA compatibility vs. efficiency**: Raccoon-based schemes are inherently more threshold-friendly but produce larger, non-standard signatures. ML-DSA compatible schemes require more complex protocols but produce standard signatures verifiable by any FIPS 204 implementation.
• **Party count scalability**: Current ML-DSA compatible schemes are limited to ~6 parties. Raccoon-based schemes and MPC approaches scale better to larger groups.
• **Round complexity**: Two-round protocols (with offline pre-processing) offer the best latency. MPC-based approaches typically require more rounds but offer greater flexibility.
• **Trust model**: Most constructions assume honest majority or at least one honest party. The CRP (Correlated Randomness Provider) model used by Trilithium introduces an additional trust assumption.

6. Practical Implementations

The landscape of open-source implementations for threshold lattice signatures has expanded significantly in 2025–2026. We catalog the known implementations:

Project	Language	Scheme	Maturity	Stars
Threshold-ML-DSA/ Threshold-ML-DSA	Go	Efficient Threshold ML-DSA (up to 6)	Academic PoC	5
GuilhemN/threshold-ml-dsa-and-raccoon	Go	T-ML-DSA + T-Raccoon w/ ID Abort	Academic PoC	5
cyberbono3/ dilithium-threshold	Rust	Basic threshold Dilithium	Experimental	4
ISRI-PQC/topcoat	Go	TOPCOAT (2-party)	Academic PoC	1

All implementations are academic proofs-of-concept. None have undergone production security audits. The Go implementations based on Cloudflare's CIRCL library are the most mature, benefiting from CIRCL's existing ML-DSA implementation and test infrastructure.

7. Implications for OPNet

OPNet uses ML-DSA for contract-level signature verification. The btc-runtime provides verifyMLDSASignature() as the recommended signature verification primitive, replacing the deprecated ECDSA-based verifyECDSASignature(). With OPNet's transition to post-quantum cryptography via BIP 360 (P2MR addresses, targeting Q4 2026–Q1 2027), the question of threshold ML-DSA becomes directly relevant for OPNet's wallet and custody infrastructure.

7.1 Feasibility Assessment

Based on the current state of research, we assess the feasibility of threshold ML-DSA wallets for OPNet:

• **2-of-2 (client-server model)**: FEASIBLE TODAY. Trilithium (2025) produces standard FIPS 204 signatures with practical performance. Suitable for phone+server custody models. The CRP (Correlated Randomness Provider) requirement adds a trust assumption but can be implemented as a pre-processing phase.
• **t-of-n (small groups, n ≤ 6)**: FEASIBLE WITH CAVEATS. “Efficient Threshold ML-DSA” (2025) demonstrates practical t-of-n threshold signing for up to 6 parties with standard ML-DSA output. Go implementation exists. However, no production security audit has been performed, and the protocol has not been deployed in adversarial conditions.
• **t-of-n (large groups, n > 6)**: RESEARCH STAGE. Current ML-DSA compatible constructions do not efficiently scale beyond ~6 parties. Raccoon-based schemes (Olingo) scale better but produce non-standard signatures that would require OPNet to support a second signature format.
• **MPC-based approach**: THEORETICALLY POSSIBLE. Quorus demonstrates that generic MPC can compute ML-DSA signing, producing standard signatures for any threshold. Communication costs are high but may be acceptable for low-frequency custody operations.

7.2 Alternative Approaches

Given the current maturity level of threshold ML-DSA, OPNet has several alternative paths for multi-party custody and authorization:

• **On-chain multisig (opMultisig)**: An M-of-N multisig contract where each party independently signs with their own ML-DSA key, and the contract verifies all signatures on-chain. This is already built and deployed on OPNet regtest. It does NOT split the key (each signer has their own full key), but achieves the same practical outcome: M parties must authorize a transaction. Gas cost scales linearly with M (each ML-DSA verification costs gas). This is the pragmatic choice for immediate deployment.
• **P2MR script-path multisig**: With BIP 360's P2MR (Pay-to-Merkle-Root), multi-party authorization can be encoded directly in the Bitcoin script tree. A Merkle tree of M-of-N script conditions (each leaf containing a set of ML-DSA public key checks) provides native Bitcoin-layer multisig without smart contract overhead. This is the most Bitcoin-native approach but requires BIP 360 activation.
• **Hybrid approach**: Use FROST threshold Schnorr for the Bitcoin UTXO layer (P2TR key-path spend) and on-chain multisig for the OPNet contract layer. This provides threshold custody for BTC while using proven technology, at the cost of different security models for the two layers.
• **Social recovery**: A single ML-DSA key with a time-locked recovery mechanism via smart contract. If the primary key is compromised or lost, a set of guardians can authorize a key rotation after a delay. Simpler than threshold signing but provides weaker real-time security guarantees.

7.3 Timeline Estimate

Based on the current research trajectory and NIST's standardization timeline:

Milestone	Estimated Date	Notes
2-of-2 threshold ML-DSA (audited library)	Q3-Q4 2026	Trilithium/TOPCOAT implementations mature enough for audit
t-of-n threshold ML-DSA (audited, n≤6)	Q1-Q2 2027	CIRCL-based implementations reach audit readiness
NIST threshold PQC standard draft	2027-2028	Per NIST MPTC timeline
Production-grade threshold ML-DSA for OPNet	2027-2028	After audit + BIP 360 activation + OPNet VM integration
NIST threshold PQC final standard	2028-2029	Full standardization

Recommendation for OPNet: Deploy on-chain multisig (opMultisig) immediately for multi-party authorization needs. Monitor the "Efficient Threshold ML-DSA" and "Threshold Signatures Reloaded" implementations for maturity. Plan integration of true threshold ML-DSA signing for 2027-2028 timeframe, aligning with BIP 360 activation and NIST standardization progress.

8. Conclusion and Recommendations

Threshold ML-DSA signing has progressed from theoretical impossibility to practical feasibility in a remarkably short period. The key developments of 2024–2026—particularly the sDKG construction, the Raccoon threshold-friendly design, and the recent ML-DSA compatible protocols—have brought the field to a point where deployment is on the horizon.

The core challenges—rejection sampling coordination, noise flooding overhead, shortness preservation in secret sharing, and DKG complexity—have all been addressed to varying degrees by recent work. The most promising path forward is the algebraic approach of del Pino, Espitau, Niot, and Prest, which achieves standard ML-DSA signature output with practical performance for small-to-moderate party counts.

Key findings:

• **2-of-2 threshold ML-DSA is production-ready** pending security audits (Trilithium).
• **t-of-n threshold ML-DSA (n≤6) is feasible** with Go implementations available.
• **Large-group threshold (n>6) requires Raccoon** or MPC-based approaches.
• **NIST standardization is 2–3 years away** (2027–2029).
• **On-chain multisig is the pragmatic immediate solution** for OPNet.
• **True TSS wallets for OPNet should target 2027–2028** deployment.

The post-quantum transition creates a unique window: as the entire cryptographic ecosystem migrates from ECDSA/Schnorr to ML-DSA, OPNet’s early adoption of ML-DSA positions it to integrate threshold signing capabilities as they mature, rather than retrofitting them later. The combination of BIP 360’s P2MR addresses and threshold ML-DSA signing will provide OPNet with a quantum-safe, distributed custody solution that is native to Bitcoin and requires no bridges or external trust assumptions.

References

[1] Y. Desmedt, “Society and Group Oriented Cryptography: A New Concept,” CRYPTO 1987.

[2] V. Shoup, “Practical Threshold Signatures,” EUROCRYPT 2000.

[3] C. Komlo and I. Goldberg, “FROST: Flexible Round-Optimized Schnorr Threshold Signatures,” SAC 2020.

[4] R. Gennaro and S. Goldfeder, “One Round Threshold ECDSA with Identifiable Abort,” ePrint 2020/540.

[5] R. Canetti et al., “UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts,” ACM CCS 2020.

[6] NIST, “FIPS 204: Module-Lattice-Based Digital Signature Standard,” August 2024.

[7] M. Ajtai, “Generating Hard Instances of Lattice Problems,” STOC 1996.

[8] O. Regev, “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography,” STOC 2005.

[9] V. Lyubashevsky, “Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures,” ASIACRYPT 2009.

[10] J. Devevey, B. Libert, K. Nguyen, T. Peters, and M. Yung, “Non-Interactive CCA2-Secure Threshold Cryptosystems,” PKC 2023.

[11] D. Boneh et al., “Threshold Cryptosystems From Threshold Fully Homomorphic Encryption,” CRYPTO 2018.

[12] S. Agrawal, D. Stehlé, and A. Yadav, “Towards Practical and Round-Optimal Lattice-Based Threshold and Blind Signatures,” ePrint 2021/381.

[13] R. del Pino, T. Espitau, S. Niot, and T. Prest, “Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts,” CRYPTO 2024.

[14] B. Yang et al., “Asynchronous Lattice-Based Distributed Key Generation,” 2025.

[15] D. Cozzo and N. Smart, “Sharing the LUOV: Threshold Post-Quantum Signatures,” IMA CC 2019.

[16] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, “Two-Round n-out-of-n and Multi-Signatures and Trapdoor Commitment from Lattices,” PKC 2021.

[17] P. Laud and N. Snetkov, “DiLizium 2.0: Revisiting Two-Party CRYSTALS-Dilithium,” ePrint 2023.

[18] J. Dufka, O. Kravtšenko, P. Laud, and N. Snetkov, “Trilithium: Two-party Threshold Signing Compliant with FIPS 204,” 2025.

[19] ISRI-PQC, “TOPCOAT: Two-Party Lattice-Based Threshold Signatures,” 2024–2025.

[20] K. Gur, J. Katz, and T. Silde, “Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption,” PQCrypto 2024.

[21] T. Espitau, S. Katsumata, and K. Takemure, “Two-Round Threshold Signature from Algebraic One-More LWE,” CRYPTO 2024.

[22] J. Zhu and S. Tessaro, “Reduction of AOM-MISIS to Standard MSIS and MLWE,” 2024–2025.

[23] R. del Pino et al., “Threshold Raccoon,” EUROCRYPT 2024.

[24] R. del Pino et al., “Unmasking TRaccoon: Threshold Raccoon with Identifiable Aborts,” 2024.

[25] K. Gur et al., “Olingo: A Framework for Threshold Lattice Signatures,” 2025.

[26] S. Celi, R. del Pino, T. Espitau, S. Niot, and T. Prest, “Efficient Threshold ML-DSA,” 2025.

[27] F. Borin et al., “Threshold Signatures Reloaded: ML-DSA and Enhanced Raccoon with Identifiable Aborts,” 2025–2026.

[28] S. Bienstock et al., “Quorus: Efficient, Scalable Threshold ML-DSA Signatures from MPC,” 2025–2026.

[29] NIST, “NISTIR 8214C: Call for Multi-Party Threshold Scheme Submissions,” 2nd Public Draft, March 2025.