"Preview Writeup": In anticipation of a package submission to the NIST Threshold Call

Title: Mithril

Subtitle: Efficient Threshold ML-DSA from Secret Sharing with Short Shares

Version: 1.0 (2026-01-19)[1]

Team name: Mithril Team

Team members: Sofía Celi, Gustavo Delerue, Rafael del Pino, Thomas Espitau, Guilhem Niot, Thomas Prest

Abstract: This document is a preview of the Mithril Team's submission to the NIST Multi-Party Threshold Cryptography (MPTC) Call. This submission will specify a threshold signature protocol for the Module-Lattice-based Digital Signature Algorithm (ML-DSA) or FIPS 204. The proposed scheme resolves the core incompatibility between ML-DSA's rejection sampling and multi-party computation by using replicated secret sharing with short shares. This enables local, per-party rejection sampling, thus avoiding the need for a costly global abort multi-party computation.

The protocol supports both distributed key generation (DKG) and a posteriori sharing of an existing ML-DSA key, preserving the original public key. It is proven secure in the dishonest majority model and is fully compatible with verifiers for the ML-DSA standard. Our evaluation demonstrates practicality for any threshold T up to at least up to $N = 8$ parties (small set). Signing latency is under 20 ms locally, while it stays under 1s in a global WAN setting.

Proposed crypto-systems: Mithril (Threshold ML-DSA): Categories N4.1 and N1.1

Keywords: Threshold Cryptography; NIST Threshold Call; ML-DSA; Short Secret Sharing

[1] Version submitted to NIST-MPTC for publication

Preview writeup. This document is provided to NIST for online publication, to foster public awareness and support public discussion within the scope of the NIST First Call for Multi-Party Threshold Schemes [NIST-IR8214C]. This "preview writeup" represents a good-faith plan for a subsequent "package submission". However, until the deadline for package submission, the team may still modify its own composition and the submission plan, including possible changes to the technical scope, and/or the used techniques or achieved results.

Team members: Sofía Celi [i1,a1,a2], Gustavo Delerue [i2,a3], Rafael del Pino [i3,a3], Thomas Espitau [i4,a3], Guilhem Niot [i5,a3,a4], Thomas Prest [i6,a3]

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0002-3333-7764); i2 (0009-0005-6458-1419); i3 (0009-0001-8638-787X); i4 (0000-0002-7655-9594); i5 (0000-0002-2497-8770); i6 (0000-0003-1445-6212)

Affiliations:

[a1] Brave Research @ Portugal

[a2] University of Bristol @ UK

[a3] PQShield SAS @ France

[a4] Univ Rennes, CNRS, IRISA @ France

Main contacts:

- **Team mailing list:** contact@mithril-th.org
- **Primary technical contact person:** Guilhem Niot, guilhem@gniot.fr

Produced by humans. The team hereby confirms that the content in this preview writeup: (i) was produced by the team members, and (ii) was not produced by generative artificial intelligence (AI), with the possible exception of AI-proposed grammar improvements, minor integrated suggestions, or some well-identified and short localized portions of auxiliary content (e.g., some illustration); and (iii) was proofread by the team members.

1. Introduction

The planned package submission proposes a *practical threshold signature scheme* built from the Module-Lattice-based Digital Signature Algorithm (ML-DSA), a standard recently finalized by NIST. This work is of particular relevance to the Threshold Call, as it addresses a significant gap in the current landscape of threshold signatures: the absence of a practical, distributed construction directly from a standardized post-quantum signature scheme. Our proposed scheme, Mithril, falls within categories **N4.1** (Key Generation for a NIST Standard) and **N1.1** (Signing for a NIST Standard).

The scheme was initially introduced in [CPENP26] and adapts and optimizes techniques from the "Finally!" threshold signature construction [PN25] to the specific setting of ML-DSA. It leverages replicated secret sharing with *short shares*, enabling parties to locally perform the most challenging operations to distribute in ML-DSA (most notably short vector sampling and signature rejection) while introducing optimizations tailored to the algebraic and algorithmic structure of ML-DSA. This design departs from other approaches that rely on more generic techniques to distribute sampling and rejection in ML-DSA [DKLS25; BCEPT25], which typically incur a high number of interaction rounds as well as substantial computational and communication overhead.

The main practical limitation of the proposed techniques is that they naturally support only a small number of parties. We evaluate performance for up to six parties[2]. This specific setting nonetheless captures many practical deployment scenarios, including multi-device cryptocurrency wallets and threshold-based TLS authentication. Our results demonstrate real-world practicality, with per-party communication below 1 MB and signing latency under 20 ms in local environments, and under one second in a global WAN setting.

2. Specification

2.1. Organization

The specification will detail the Mithril scheme, structured around its core procedures: key generation, a three-round signing protocol, and signature combination/verification. The document will modularize and explain the key technical innovations. These include: (i) the use of short replicated secret sharing for key distribution, (ii) an optimized per-party rejection sampling mechanism based on hyperballs to manage aborts efficiently in a distributed setting, and (iii) an optimized share reconstruction method to ensure balanced workloads among parties.

2.2. System Model

Our protocol considers a pessimistic system model. We consider security *against active adversaries*, and within a network that can be adversarially controlled (in particular, we do not require a

[2]We note that a few additional parties can be supported at the cost of increased communication. In particular, up to 8 parties remains decently practical and we will extend our benchmarks in the final submission.

synchronized broadcast channel). We support a *dishonest majority*: for any signing threshold $T \leq N$, up to $T - 1$ parties may be corrupted.

Our design focuses on deployments with a *small* number of parties N (e.g., up to 8), which already capture many practical use cases. This design choice is primarily driven by the exponential growth in the number of shares required by replicated secret sharing with parameters (T, N), as well as by the decreasing success probability of the signing protocol as T increases, both of which limit the scalability of our approach.

The specification will detail three key management scenarios: (i) a trusted dealer setup for key generation, (ii) a fully decentralized distributed key generation (DKG) protocol that eliminates any single point of trust, (iii) and an *a posteriori* key sharing mechanism that allows an existing ML-DSA key to be securely split among parties while preserving the public key.

2.3. Security

We formally analysed the security of our protocol in [CPENP26]. We consider a game-based security notion of unforgeability where a forgery is valid if no honest party attempted signing the forgery's message. We consider this in the presence of a static adversary that can corrupt up to $T - 1$ parties at the start of the protocol. The security proof is conducted in the Random Oracle Model (ROM) and reduces the unforgeability of our threshold scheme to the hardness of the Module-Learning With Errors (MLWE) assumption, which underlies ML-DSA, and to the unforgeability of the standard single-party ML-DSA scheme. For a posteriori key sharing, the reduction to the MLWE assumption differs slightly, as the adversary can obtain a noisy hint on the original ML-DSA secret, leading to a small loss of security: we quantify it to 7 to 12 bits in [CPENP26].

Note that we expect our scheme to be secure in the more realistic model of adaptive adversaries as well (i.e., adversaries that corrupt parties during the execution of the scheme), but proving this stronger property for lattice-based schemes is notoriously hard. In existing proofs, security typically proceeds by replacing the public key with a uniform sample (relying on the MWLE assumption) for one of the honest replicated shares. Under adaptive corruption, this share may later need to be revealed, even though it is no longer known to the security game, which breaks the standard proof strategy. Nevertheless, at least one replicated share necessarily remains uncorrupted at all times. Intuitively, this suffices to preserve the pseudorandomness of the public key and, consequently, the security of the scheme. Moreover, in the concrete setting of up to six parties, a simple guessing argument shows that the provable security loss is bounded by at most 5 bits.

We do not consider identifiable aborts or robustness in our current protocol: adversaries can cause the protocol to abort without being identified. We leave the design of an efficient abort identification mechanism as future work.

3. Open-Source Implementation

3.1. Code structure

We currently provide a Go proof-of-concept implementation of Mithril for all security levels (44, 65, 87), and plan to develop a C reference implementation covering all ML-DSA parameter sets for the submission. The existing Go implementation relies on floating-point arithmetic for sampling within hyperballs: this will be replaced by fixed-point arithmetic in the C implementation to ensure portability and constant-time behavior. We further plan to vectorize performance-critical components using AVX2 instructions.

The API of our threshold protocol has no external dependency, and is fully self contained.

3.2. Code progress and availability

Our Go proof-of-concept implementation is available at https://github.com/Threshold-ML-DSA/Threshold-ML-DSA, along with initial benchmark scripts for local, LAN and WAN settings. Performance evaluations presented in the accompanying paper [CPENP26] were conducted using this implementation.

The reference C implementation for the submission is currently under development. It will build upon the ML-DSA PQClean implementation [KSSW22].

3.3. Implementation of the networking model

For both LAN and WAN experiments, the networking layer is implemented using direct TCP streams managed by the libp2p [25] framework, which provides peer identity management and connection handling. In the WAN setting, parties are deployed as independent peers arranged in a mesh network topology, communicating over public IP addresses on geographically distributed hosts.

3.4. Testing

We plan to release test vectors alongside the reference implementation to facilitate verification of correctness, which will include the full protocol transcript, as well as the final signature in case of success. These will be performed by deriving all intermediate values from fixed random seeds, allowing reproducibility of key generation, and signing processes in a single machine.

For malicious behavior testing, we will include unit tests that simulate corrupted parties deviating from the protocol, ensuring that the protocol correctly implements the necessary security checks. We will also test the protocol's security against a network compromise with alterations of honest messages.

4. Experimental Performance Evaluation

4.1. Performance

We provide a preliminary performance evaluation below, evaluating the performance of our proof-of-concept Go implementation of Mithril (parameter set 44 of ML-DSA).

The computational and communication costs of one (local) signing attempt of our scheme for $T \leq N \leq 6$ is shown in Fig. 1, for the parameter set 44 of ML-DSA, having a success probability $1/2$. These experiments were ran on a consumer-grade MacBook M3 with 24GB RAM.

For WAN settings (evaluated using AWS EC2 machines), we observed that our protocol is mostly network-bound, with signing latencies under one second for all tested (T, N) configurations (e.g., 751 ms for 4-of-6), as seen in Table 1. This suggests that our scheme is practical even in globally distributed settings, with latencies well within the timeouts of applications like TLS.

We will include results for all ML-DSA parameter sets up to $N = 8$ parties in the final submission, as well as results for the DKG and *a posteriori* key sharing protocols.

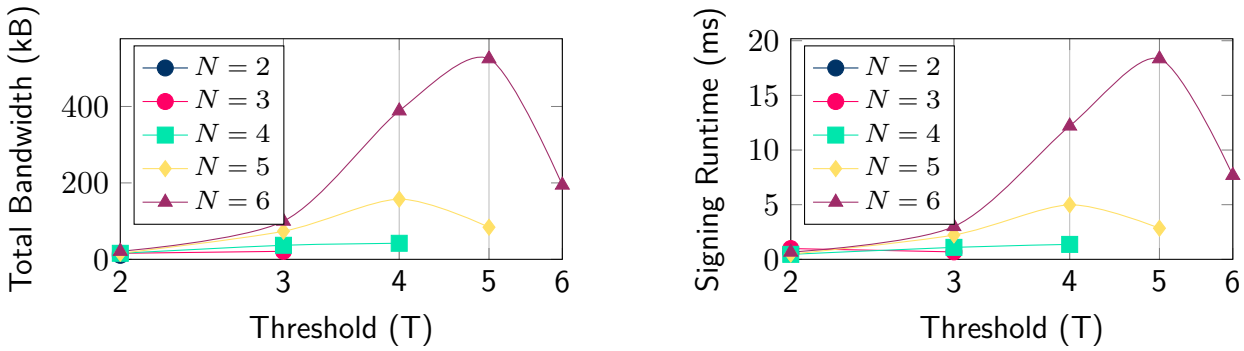

Figure 1: Bandwidth costs (left) and local runtime signing costs (right) per party for one signing attempt of Mithril 44 (success probability $1/2$). Note that the values for $N = 2$ and $N = 3$ overlap for all T.

As a comparison, other proposals for distributing ML-DSA [BCEPT25; DKLS25] incur a much higher number of rounds before a signature is obtained (at least 24 and 60 on average respectively) while our scheme requires only 6 rounds on average, which directly translates in a higher signing latency in a network-bound setting. This is irrespective of their weaker security model (honest majority for [BCEPT25], and honest third-party for [DKLS25]), and higher computational costs to prepare correlated randomness.

4.2. Platform

Our protocol is designed to be efficient on modest hardware. It can be run on the baseline platform (single computer, with 16 cores and 64 GB of RAM [PubComs2PD, item F2b.3.1]) with ease.

Note that we provide a single-threaded implementation but our protocol can be easily parallelized, as each party performs many independent computations. It can leverage specialized instructions, or multiple cores. This would further reduce the signing latency in a real-world setting.

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

Table 1: WAN median latencies (in ms) between different AWS EC2 machines.

5. Licensing, Patent Claims, and Funding

The core implementation will be made available under a permissive open-source license. The Go proof-of-concept implementation depends on the CIRCL library, which is licensed under the BSD 3-Clause License. The benchmarks rely on the libp2p library, which is licensed under MIT.

We disclose a patent application that covers the blocks underlying Mithril, as well as the threshold signature it builds from "Finally!". PQShield owns two pending patents which protect methods underlying Mithril:

- *A Threshold Cryptographic Method* – PAN PCT/GB2025/051022.
- *Threshold Signature Schemes* – PAN 2509575.3.

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).

References

[25] *libp2p: A modular peer-to-peer networking framework*. https://libp2p.io/. Accessed: 2025-12-22. libp2p community / Protocol Labs, 2025.

[BCEPT25] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. *Efficient, Scalable Threshold ML-DSA Signatures: An MPC Approach*. Cryptology ePrint Archive, Paper 2025/1163. 2025. URL: https://eprint.iacr.org/2025/1163.

[CPENP26] Sofia Celi, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. "Efficient Threshold ML-DSA". In: *35th USENIX Security Symposium (USENIX Security 2026)*. Baltimore, United States, August 2026. URL: https://inria.hal.science/hal-05442192v1/document. Also at ia.cr/2026/013.

[DKLS25] Antonín Dufka, Semjon Kravtšenko, Peeter Laud, and Nikita Snetkov. *Trilithium: Efficient and Universally Composable Distributed ML-DSA Signing*. Cryptology ePrint Archive, Paper 2025/675. 2025. URL: https://eprint.iacr.org/2025/675.

[KSSW22] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Improving Software Quality in Cryptography Standardization Projects". In: *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*. Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 19–30. DOI: 10.1109/EuroSPW55150.2022.00010. Also at ia.cr/2022/337.

[PN25] Rafaël Del Pino and Guilhem Niot. "Finally! A Compact Lattice-Based Threshold Signature". In: *PKC 2025, Part III*. Ed. by Tibor Jager and Jiaxin Pan. Vol. 15676. LNCS. Springer, Cham, May 2025, pp. 169–199. DOI: 10.1007/978-3-031-91826-1_6. Also at ia.cr/2025/872.

[NIST-IR8214C] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. (National Institute of Standards and Technology) NIST Internal Report (NISTIR) 8214C. 2026. DOI: 10.6028/NIST.IR.8214C.

[PubComs2PD] NIST-MPTC. *Compilation of Public Comments on NISTIR 8214C 2pd*. National Institute of Standards and Technology (NIST), Multi-Party Threshold Cryptography. June 2025. URL: https://csrc.nist.gov/files/pubs/ir/8214/c/2pd/docs/nistir-8214c-2pd-public-feedback.pdf.