



Efficient Threshold ML-DSA

USENIX Security 2026

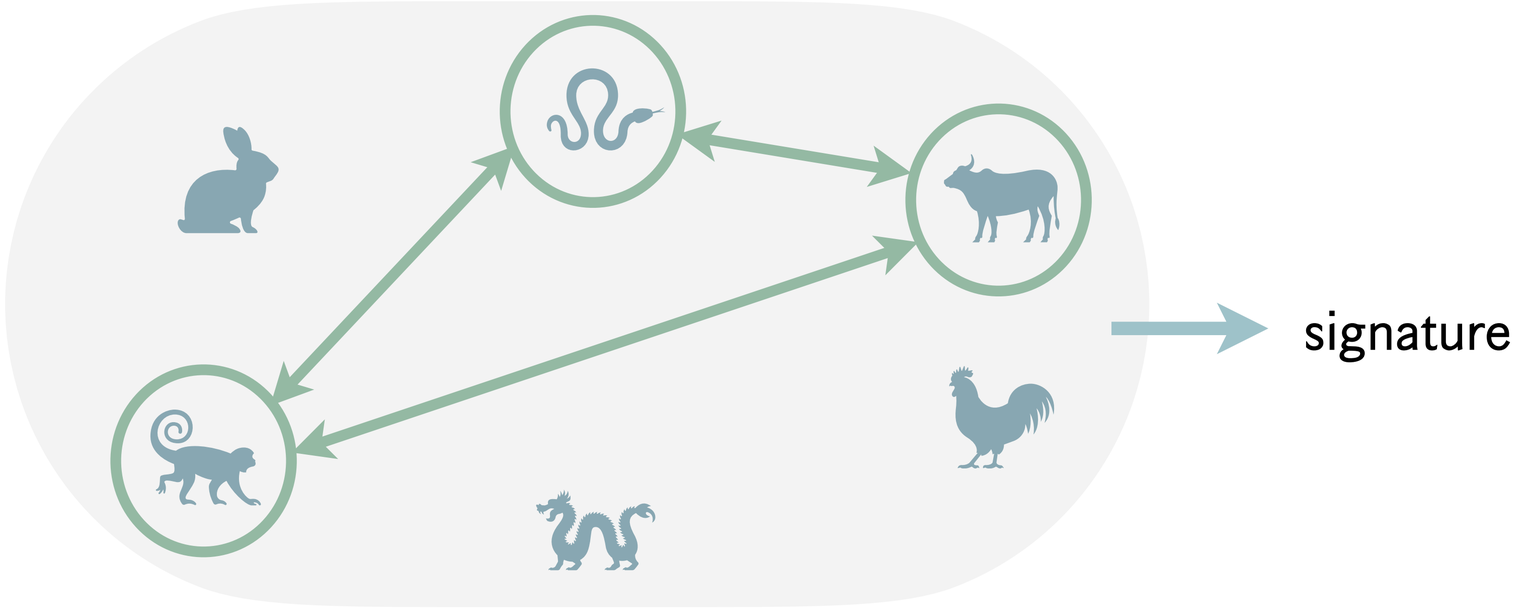
Rafael del Pino, Sofía Celi, Gustavo Delerue, Thomas Espitau, **Guilhem Niot**, Thomas Prest

Threshold Signatures

Storing a secret key centrally is a single point of failure (security and availability).

Solution: Share the key among several parties.





They execute an interactive protocol to compute the signature such that no single party learns the secret key.

Threshold Signatures: The PQ Standards?

It is desirable to design threshold solutions for the new NIST PQ signature standards.

ML-DSA

(Dilithium)

FN-DSA

(Falcon)

SLH-DSA

(SPHINCS+)

Threshold Signatures: The PQ Standards?

It is desirable to design threshold solutions for the new NIST PQ signature standards.



Our Proposal: Mithril, An Efficient Threshold ML-DSA

Mithril: Threshold protocol producing signatures compatible with **ML-DSA** verifiers (note: signatures follow a distribution).

- Signing protocol in 3 rounds.
- Support Distributed Key Generation.
- Communication between 20kB and 1MB for ***a few parties***.

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

- $\mathsf{vk} = \mathbf{A} \cdot \mathsf{sk} + \mathbf{e}$, for sk, $\mathbf{e}$ short

MLWE **assumption:** vk appears uniformly distributed.

Signing → Fiat-Shamir transform applied to protocol proving knowledge of sk

1. **Randomness + Commitment:** Sample short $\mathbf{r}$, and commit $\mathbf{w} = \lfloor \mathbf{A} \cdot \mathbf{r} \rceil$.
2. **Challenge:** Derive challenge $c = H(\mathbf{w}, \mathrm{msg})$.
3. **Response:** Compute $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$.

Verification → Check that $\mathbf{w}$ can be recovered from $\mathbf{z}$ and vk, and $\mathbf{z}$ is short.

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

- $\mathrm{vk} = \mathbf{A} \cdot \mathrm{sk} + \mathbf{e}$, for sk, $\mathbf{e}$ short

MLWE **assumption:** vk appears uniformly distributed.

Signing → Fiat-Shamir transform applied to protocol proving knowledge of sk

1. **Randomness + Commitment:** Sample short $\mathbf{r}$, and commit $\mathbf{w} = \lfloor \mathbf{A} \cdot \mathbf{r} \rceil$.
2. **Challenge:** Derive challenge $c = H(\mathbf{w}, \mathrm{msg})$.
3. **Response:** Compute $\mathbf{z} = c \cdot \mathrm{sk} + \mathbf{r}$. Rejection sample:
   - If $\mathbf{z} \notin S$ → **restart**
   - If $\mathbf{z} \in S$ → **output**

Verification → Check that $\mathbf{w}$ can be recovered from $\mathbf{z}$ and vk, and $\mathbf{z}$ is short.

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

- $\mathrm{vk} = \mathbf{A} \cdot \mathrm{sk} + \mathbf{e}$, for sk, $\mathbf{e}$ short

MLWE **assumption:** vk appears uniformly distributed.

Signing → Fiat-Shamir transform applied to protocol proving knowledge of sk

1. **Randomness + Commitment:** Sample short $\mathbf{r}$, and commit $\mathbf{w} = \lfloor \mathbf{A} \cdot \mathbf{r} \rceil$.
2. **Challenge:** Derive challenge $c = H(\mathbf{w}, \mathrm{msg})$.
3. **Response:** Compute $\mathbf{z} = c \cdot \mathrm{sk} + \mathbf{r}$. Rejection sample:
   - If $\mathbf{z} \notin S$ → **restart**
   - If $\mathbf{z} \in S$ → **output**

Verification → Check that $\mathbf{w}$ can be recovered from $\mathbf{z}$ and vk, and $\mathbf{z}$ is short.

Distributing ML-DSA: *Mithril at a high level*

Centralized

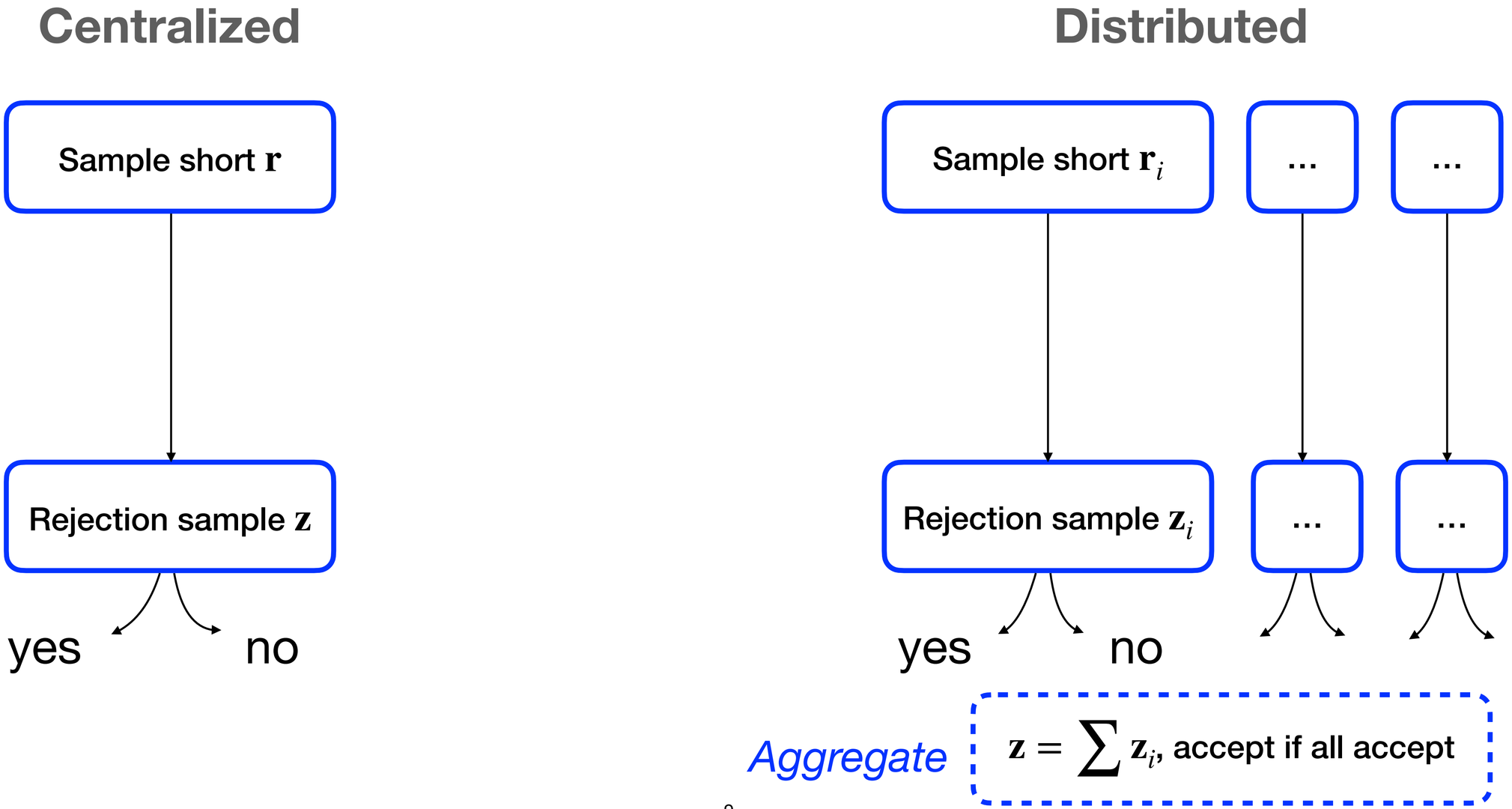
Sample short $\mathbf{r}$

Distributed

Sample short $\mathbf{r}_i$ | ... | ...

Aggregate $\mathbf{r} = \sum_i \mathbf{r}_i$

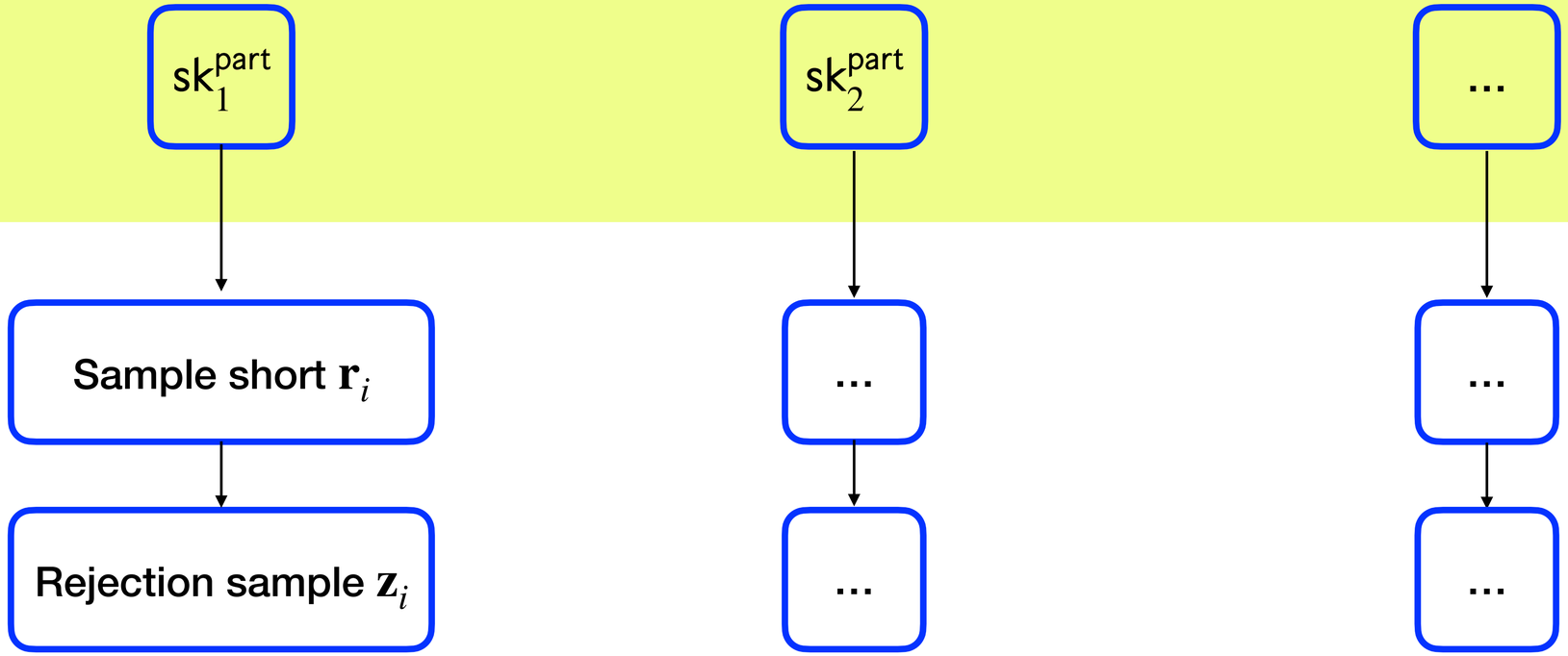
Distributing ML-DSA: *Mithril at a high level*



Technique 1: Replicated Secret Sharing

For this to work, we need a **short partial secret** per party for each session.

$$\mathsf{sk} = \sum_i \mathsf{sk}_i^{\mathsf{part}}$$



Technique 1: Replicated Secret Sharing

For this to work, we need a **short partial secret** per party for each session.

Use Replicated Secret Sharing as in [dPN25].

ML-DSA* . Keygen() $\to$ sk, vk

- For every possible set I of $N - T + 1$ parties
  - $\mathsf{vk}_I = \mathbf{A} \cdot \mathsf{sk}_I + \mathbf{e}_I$, where $\mathsf{sk}_I, \mathbf{e}_I$ short
  - Distribute $\mathsf{sk}_I, \mathbf{e}_I$ to parties in I
- $\mathsf{vk} = \sum_i \mathsf{vk}_I$

1. When at most $T - 1$ parties are corrupted, at least one of these secrets remains hidden.

2. T parties can collaboratively reconstruct the full secret.

Partition $\sqcup_{i \in SS} m_i = \{I \text{ s.t. } |I| = N - T + 1\}$:

$$\mathsf{sk} = \sum_{i \in SS} \sum_{I \in m_i} \mathsf{sk}_I, \quad \mathbf{e} = \sum_{i \in SS} \sum_{I \in m_i} \mathbf{e}_I$$

Distributing ML-DSA: *Mithril at a high level*

ML-DSA signing

1. **Randomness + Commitment:** Sample short $\mathbf{r}$, and commit $\mathbf{w} = \lfloor \mathbf{A} \cdot \mathbf{r} \rceil$.

2. **Challenge:** Derive challenge $c = H(\mathbf{w}, \text{msg})$.

3. **Response:** Compute $\mathbf{z} = c \cdot \text{sk} + \mathbf{r}$.
Rejection sample:

Our protocol

Mithril . Sign(msg) → sig

Round 1:

- Sample short $\mathbf{r}_i, \mathbf{e}'_i$
- $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- Broadcast $\text{commit}_i = H(\mathbf{w}_i)$

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$ + abort if inconsistent commit_i
- $c = H(\lfloor \mathbf{w} \rceil, \text{msg})$
- $\mathbf{z}_i = c \cdot \sum_{I \in m_i} \text{sk}_I + \mathbf{r}_i$
- If $\mathbf{z}_i$ in S', **broadcast $\mathbf{z}_i$, else abort**

Combine:

- $\text{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If $\neg$Verify(sig), **restart**
- **return** sig

Technique 2: Optimized rejection sampling

When T users sign $\rightarrow$ proba that all parties pass rejection sampling is p^T.

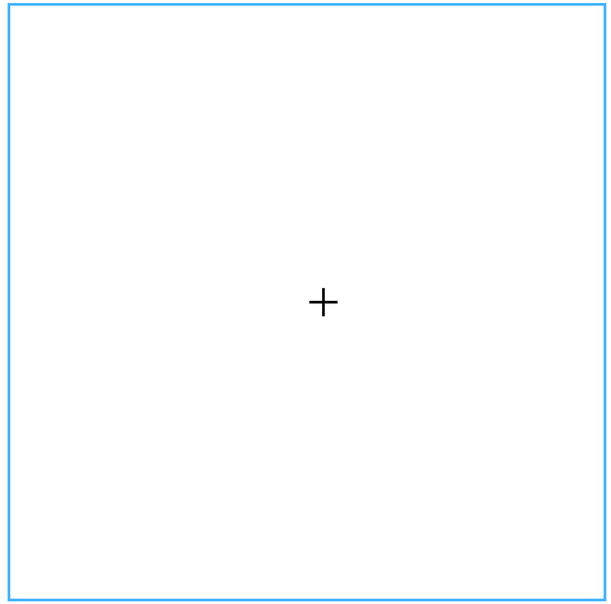
Exponential degradation over centralized setting.

Technique 2: Optimized rejection sampling

When T users sign $\rightarrow$ proba that all parties pass rejection sampling is p^T.

Exponential degradation over centralized setting.

Sample r in a centered **hypercube**.



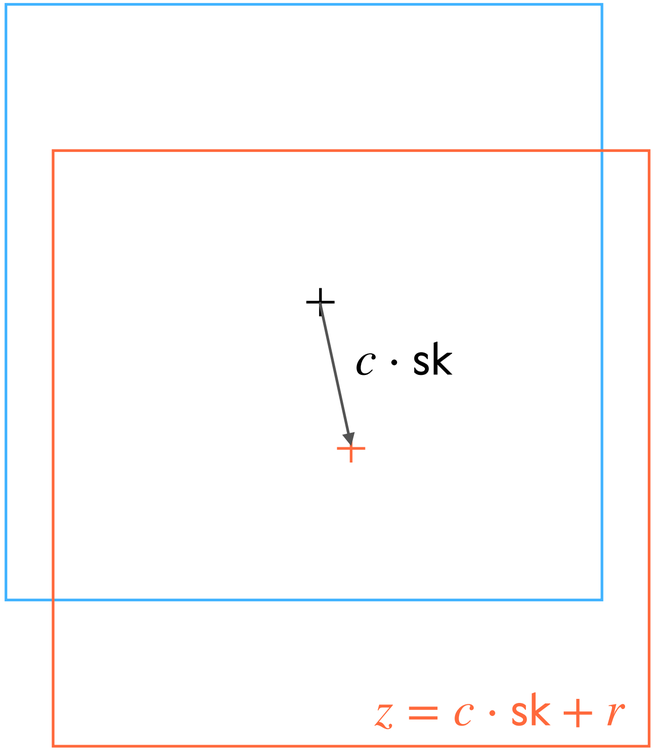
Technique 2: Optimized rejection sampling

When T users sign $\rightarrow$ proba that all parties pass rejection sampling is p^T.

Exponential degradation over centralized setting.

Sample r in a centered **hypercube**.

Then, the distribution of z depends on the secret.



Technique 2: Optimized rejection sampling

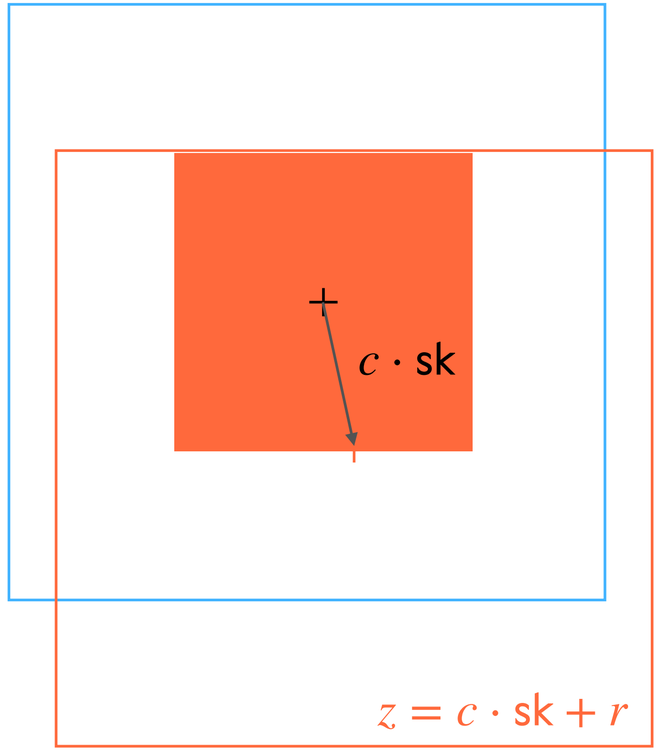
When T users sign $\rightarrow$ proba that all parties pass rejection sampling is p^T.

Exponential degradation over centralized setting.

Sample r in a centered **hypercube**.

Then, the distribution of z depends on the secret.

We reject any z outside of .
The resulting distribution is independent of the secret.



Technique 2: Optimized rejection sampling

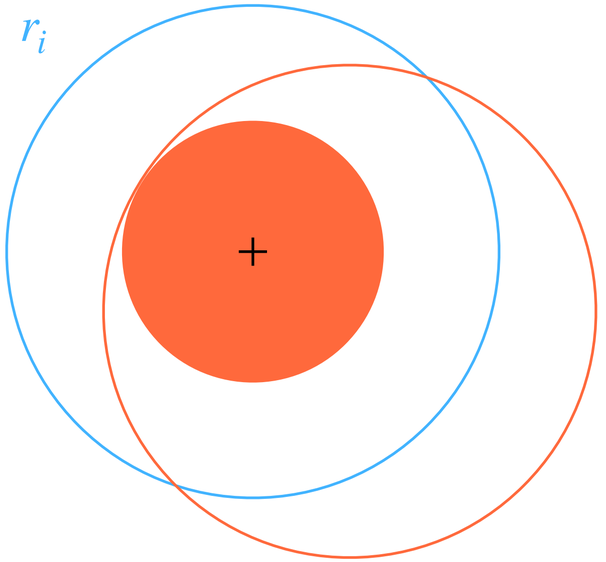
When T users sign $\rightarrow$ proba that all parties pass rejection sampling is p^T.

Exponential degradation over centralized setting.

Sample r in a centered **hyperball**.

Then, the distribution of z depends on the secret.

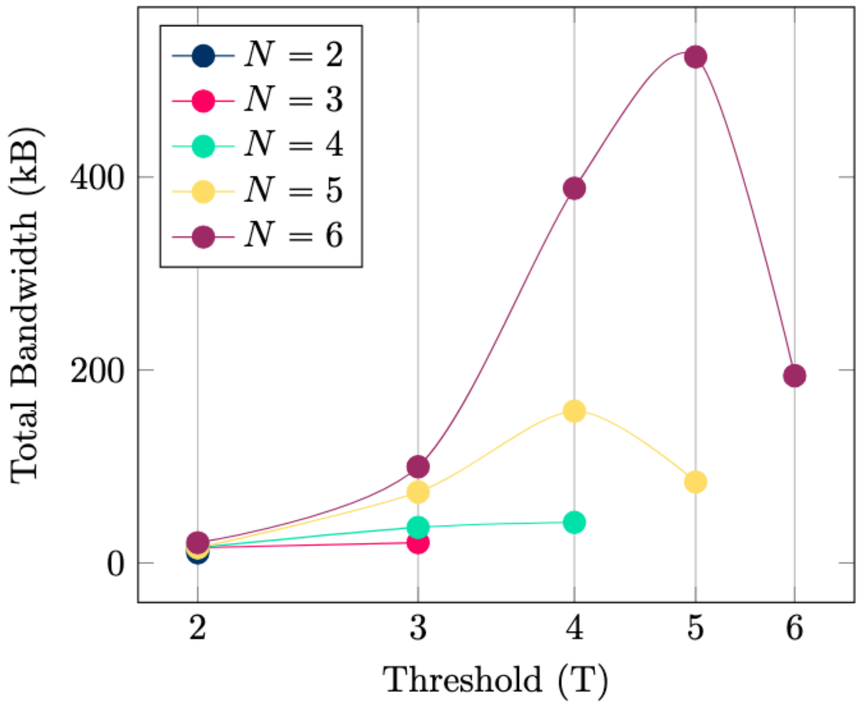
We reject any z outside of  .
The resulting distribution is independent of the secret.

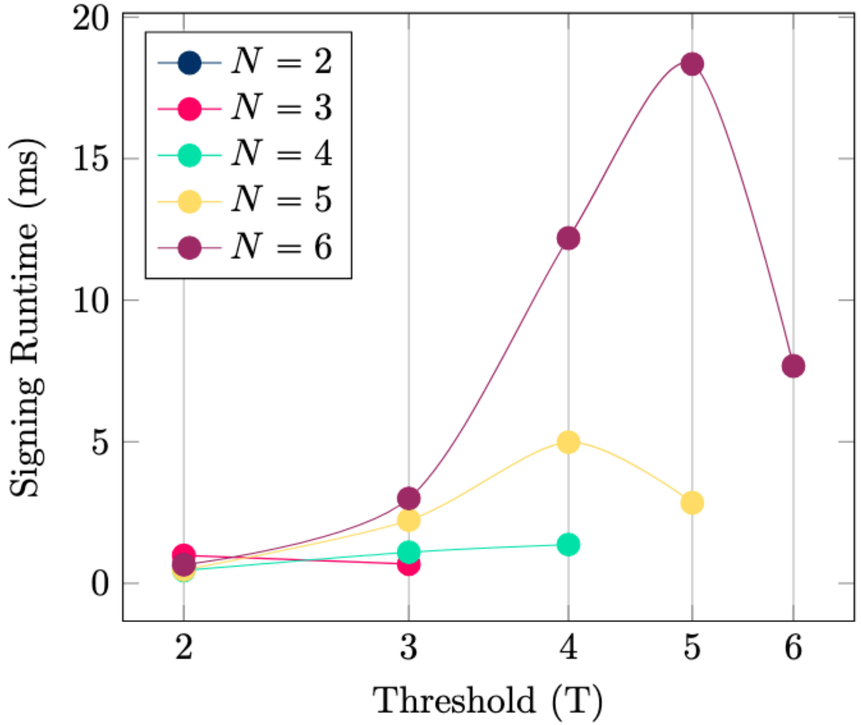


Performance: Bandwidth and local latency

Parameters aim for a success probability 1/2 for each attempt (vs ~1/4 in original ML-DSA).

Efficient up to 6 parties.





Bandwidth and latency of threshold signing for ML-DSA 44 (on a local network)

Parties are executed in parallel, and we average over successful attempts.

Performance: WAN latency

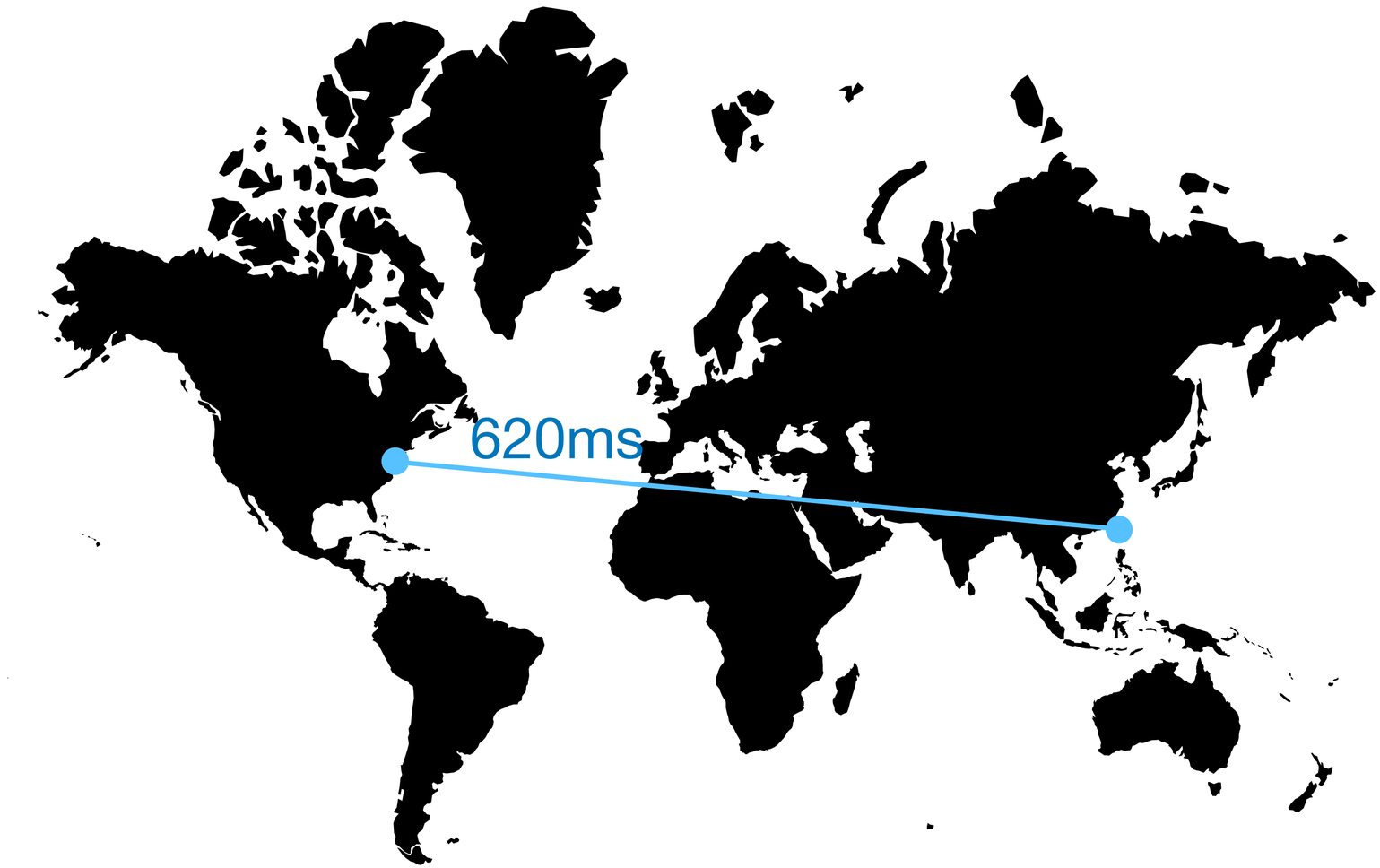
T=2, N=6



Latency of threshold signing for ML-DSA 44.
Parties are executed in parallel, and we average over successful attempts.

Performance: WAN latency

T=2, N=6



Latency of threshold signing for ML-DSA 44.
Parties are executed in parallel, and we average over successful attempts.

Performance: WAN latency

T=4, N=6



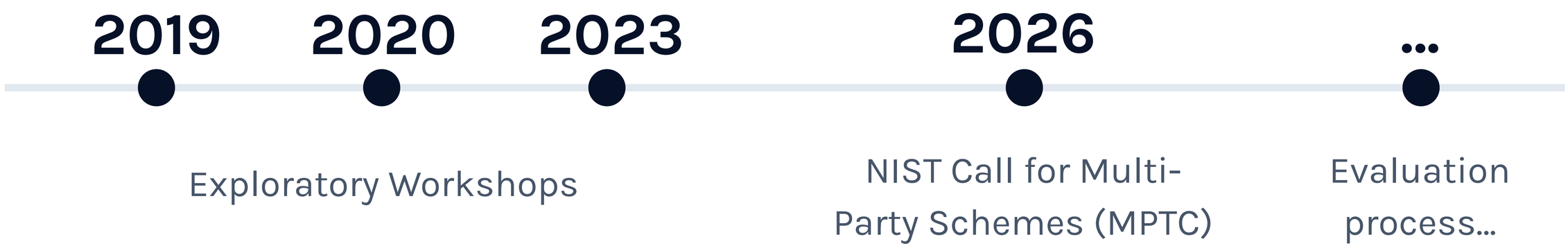
Latency of threshold signing for ML-DSA 44.
Parties are executed in parallel, and we average over successful attempts.

Comparison to other approaches

Paradigm	# parties supported	Comm./party	Computation
MPC (Quorus)	∞	**> 1 MB (> 20 rounds)**	Medium
FHE[1]	∞	25 kB (2 rounds)	**High + Proprietary**
This talk (Mithril)	**< 10**	20 kB - 1MB (3 rounds)	Low

[1]. https://eprint.iacr.org/2026/638

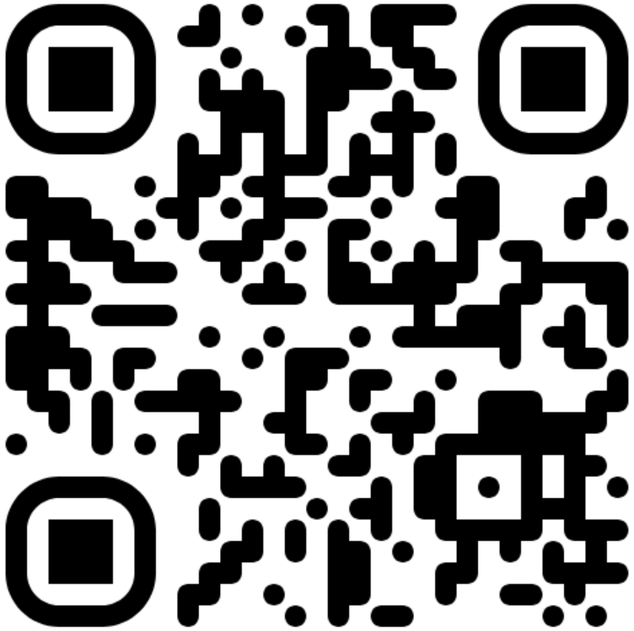
Multi-Party Cryptography and NIST



Stated goal: Gather high-quality reference material, and inform the development of future NIST recommendations and processes.

We will submit Mithril as a proposal to NIST MPTC.

Going further



Full version on ePrint.

Evaluation

Other ML-DSA parameter sets

