Efficient Threshold ML-DSA up to 6 Parties

Post-Quantum Threshold Signatures Compatible with the NIST Standard

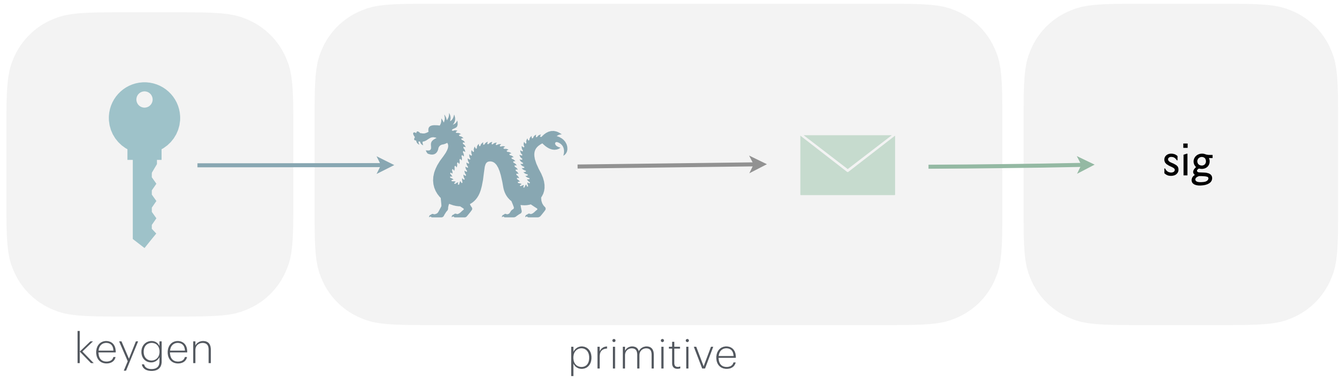
Guilhem Niot

joint work with Rafael del Pino, Sofía Celi, Thomas Espitau, Thomas Prest

NIST Sixth PQC Standardization Conference

Threshold Signatures

Centralized setting



Threshold Signatures

What if the party is corrupted or becomes unresponsive…

Question: can we split the trust among several parties?

Threshold Signatures

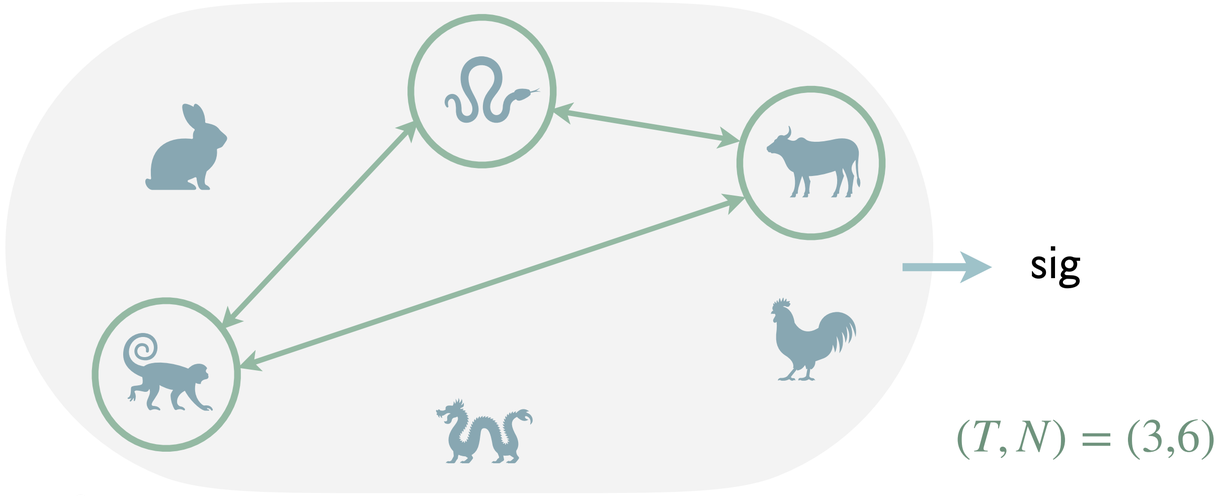
What if the party is corrupted or becomes unresponsive…

Question: can we split the trust among several parties?

Interactive protocol to distribute the scheme:

T-out-of-N parties can collaborate to sign and

$T-1$ parties cannot.



NIST Call for Threshold Schemes

PUBLICATIONS

NIST IR 8214C (2nd Public Draft)

NIST First Call for Multi-Party Threshold Schemes

Date Published: March 27, 2025
Comments Due: April 30, 2025
Email Comments to: nistir-8214C-comments@nist.gov

Author(s)

Luís T. A. N. Brandão (NIST, Strativia), Rene Peralta (NIST)

Announcement

This is a second public draft. Threshold schemes should NOT be submitted until the final version of this report is published. However, the present draft can be used as a baseline to prepare for future submissions.

The scope of the call is organized into categories related to signing (Sign), public-key encryption (PKE), symmetric-key cryptography and hashing (Symm), key generation (KeyGen), fully homomorphic encryption

Post-Quantum Threshold Signatures?

Raccoon

Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions

Rafael del Pino[1], Shuichi Katsumata[1,2], Mary Maller[1,3], Fabrice Mouhartem[4], Thomas Prest[1], Markku-Juhani Saarinen[1,5]

***Flood and Submerse*: Distributed Key Generation and Robust Threshold Signature from Lattices**

Thomas Espitau[1], Guilhem Niot[1,2], and Thomas Prest[1]

Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption*

Kamil Doruk Gur[1], Jonathan Katz[2**], and Tjerand Silde[3***]

Ringtail**: Practical Two-Round Threshold Signatures from Learning with Errors**

Cecilia Boschini *ETH Zürich, Switzerland*
Darya Kaviani *UC Berkeley, USA*
Russell W. F. Lai *Aalto University, Finland*
Giulio Malavolta *Bocconi University, Italy*
Akira Takahashi *JPMorgan AI Research & AlgoCRYPT CoE, USA*
Mehdi Tibouchi *NTT Social Informatics Laboratories, Japan*

Post-Quantum Threshold Signatures?

Raccoon

Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions

Rafael del Pino[1], Shuichi Katsumata[1,2], Mary Maller[1,3], Fabrice Mouhartem[4], Thomas Prest[1], Markku-Juhani Saarinen[1,5]

***Flood and Submerse*: Distributed Key Generation and Robust Threshold Signature from Lattices**

Thomas Espitau[1], Guilhem Niot[1,2], and Thomas Prest[1]

Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption*

Kamil Doruk Gur[1], Jonathan Katz[2**], and Tjerand Silde[3***]

Ringtail**: Practical Two-Round Threshold Signatures from Learning with Errors**

Cecilia Boschini, *ETH Zürich, Switzerland*
Darya Kaviani, *UC Berkeley, USA*
Russell W. F. Lai, *Aalto University, Finland*
Giulio Malavolta, *Bocconi University, Italy*
Akira Takahashi, *JPMorgan AI Research & AlgoCRYPT CoE, USA*
Mehdi Tibouchi, *NTT Social Informatics Laboratories, Japan*

In 2023, NIST selected 3 signature schemes for standardization.

ML-DSA

FN-DSA

Based on lattices

SLH-DSA

Based on hash functions

Thresholdizing ML-DSA

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 for $\mathbf{A}$ wide enough (more inputs than outputs)

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 for $\mathbf{A}$ wide enough (more inputs than outputs)

To sign: prove knowledge of sk, $\mathbf{e}$, without revealing sk, $\mathbf{e}$. *(Fiat-Shamir type signature)*

Prover **Challenger**

1 Sample short $\mathbf{r}$

$\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$

$\mathbf{w}$ →

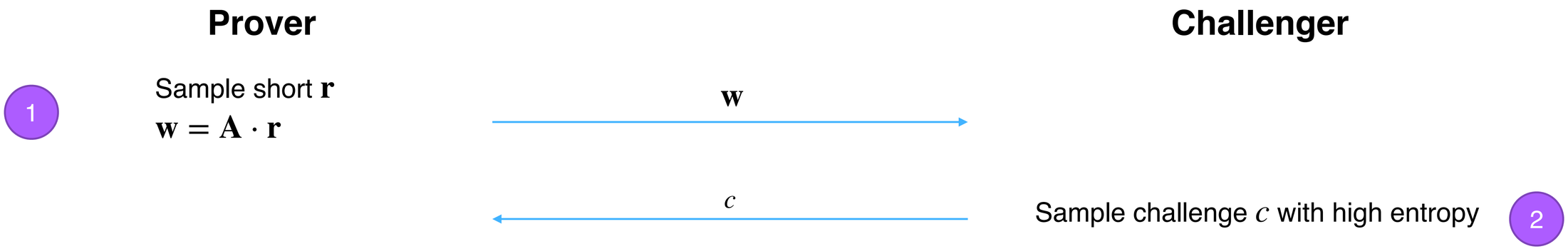
ML-DSA signatures

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 for $\mathbf{A}$ wide enough (more inputs than outputs)

To sign: prove knowledge of sk, $\mathbf{e}$, without revealing sk, $\mathbf{e}$. *(Fiat-Shamir type signature)*



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 for $\mathbf{A}$ wide enough (more inputs than outputs)

To sign: prove knowledge of sk, $\mathbf{e}$, without revealing sk, $\mathbf{e}$. *(Fiat-Shamir type signature)*

	Prover		**Challenger**
1	Sample short $\mathbf{r}$ $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$	$\mathbf{w}$ →	
		c ←	Sample challenge c with high entropy 2
3	Compute response $\mathbf{z} = c \cdot \mathrm{sk} + \mathbf{r}$	$\mathbf{z}$ →	Verify that $\mathbf{z}$ is short Verify $\mathbf{w} - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathrm{vk})$ 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 for $\mathbf{A}$ wide enough (more inputs than outputs)

To sign: prove knowledge of sk, $\mathbf{e}$, without revealing sk, $\mathbf{e}$. *(Fiat-Shamir type signature)*

	Prover		**Challenger**
1	Sample short $\mathbf{r}$ $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$	$\mathbf{w}$ →	
		c ←	Sample challenge c with high entropy 2
3	Compute response $\mathbf{z} = c \cdot \mathrm{sk} + \mathbf{r}$ If $\mathbf{z} \notin S$, restart	$\mathbf{z}$ →	Verify that $\mathbf{z}$ is short Verify $\mathbf{w} - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathrm{vk})$ is short

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 for $\mathbf{A}$ wide enough (more inputs than outputs)

To sign: prove knowledge of sk, $\mathbf{e}$, without revealing sk, $\mathbf{e}$. *(Fiat-Shamir type signature)*

	Prover		**Challenger**
1	Sample short $\mathbf{r}$ $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$	$\lfloor \mathbf{w} \rceil$ →	
		← c	Sample challenge c with high entropy 2
3	Compute response $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$ If $\mathbf{z} \notin S$, restart If $\mathbf{w} - c \cdot \mathbf{e} \notin S'$, restart	$\mathbf{z}$ →	Verify that $\mathbf{z}$ is short Verify $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

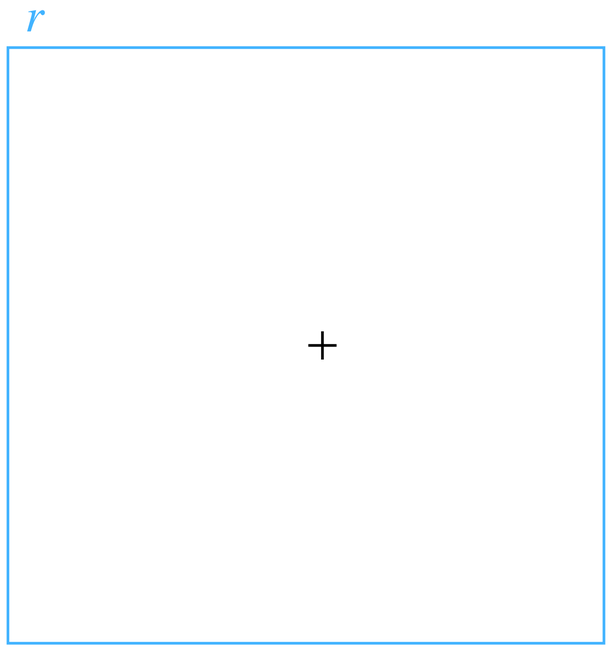
MLWE **assumption:** vk appears uniformly distributed for $\mathbf{A}$ wide enough (more inputs than outputs)

To sign: prove knowledge of sk, $\mathbf{e}$, without revealing sk, $\mathbf{e}$. *(Fiat-Shamir type signature)*

	Prover		**Challenger**
1	Sample short $\mathbf{r}$ $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$		
2	$c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$		
3	Compute response $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$ If $\mathbf{z} \notin S$, restart If $\mathbf{w} - c \cdot \mathbf{e} \notin S'$, restart	$\mathbf{z}$ →	Verify that $\mathbf{z}$ is short Verify $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short

Rejection sampling

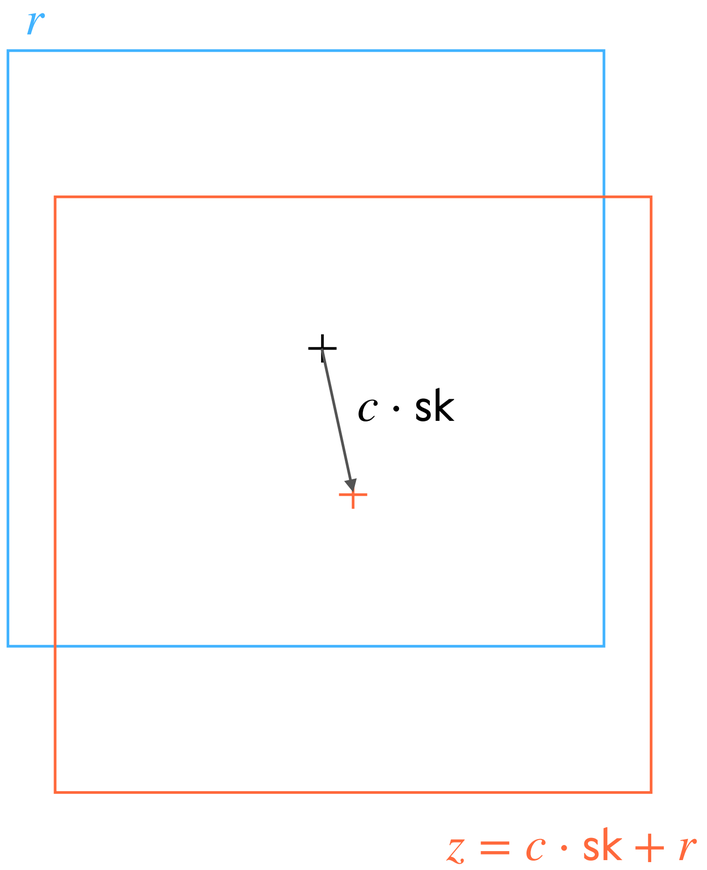
Sample r in a centered hypercube.



Rejection sampling

Sample r in a centered hypercube.

Then, the distribution of z depends on the secret.

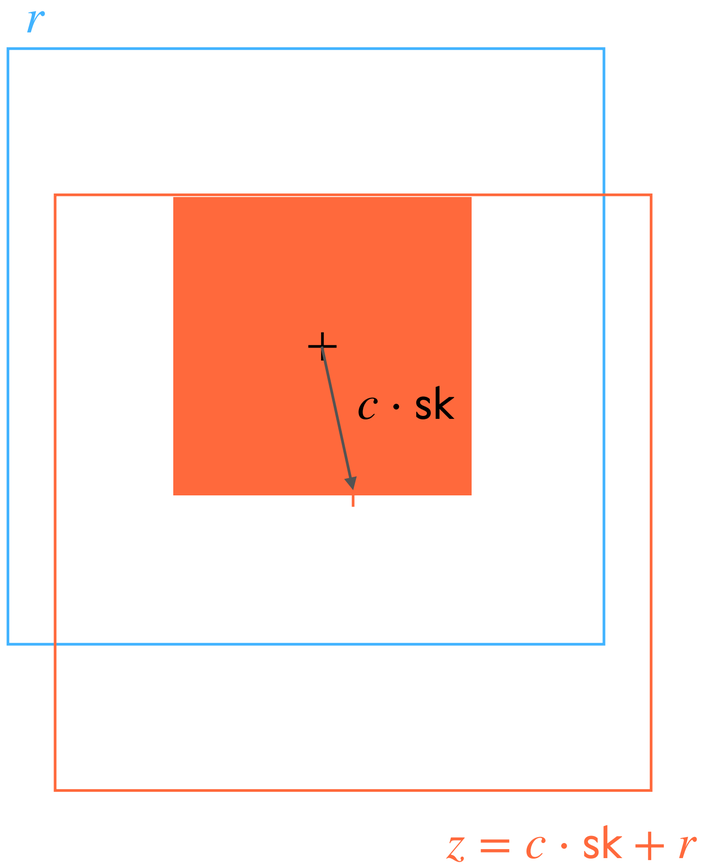


Rejection sampling

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.



ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

MLWE **assumption:** vk appears uniformly distributed for $\mathbf{A}$ wide enough (more inputs than outputs)

ML-DSA . Sign(sk, msg) → sig

- Sample short $\mathbf{r}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$
- $c = H(\lfloor \mathbf{w} \rceil, \text{msg})$
- $\mathbf{z} = c \cdot \text{sk} + \mathbf{r}$
- If $\mathbf{z}$ not in S, **restart**
- If $\mathbf{z} - c \cdot \mathbf{e}$ not in S', **restart**
- Output $\text{sig} = (\mathbf{z}, \lfloor \mathbf{w} \rceil)$

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \text{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \text{vk})$ is short
- Assert $\mathbf{z}$ is small

Our approach

1. Design a variant of ML-DSA that is threshold-friendly for $T = N$ parties.

   Idea: Sample N independent ML-DSA secrets in Keygen and prove their knowledge independently.

Our approach

1. Design a variant of ML-DSA that is threshold-friendly for $T = N$ parties.

   Idea: Sample N independent ML-DSA secrets in Keygen and prove their knowledge independently.

   Note: The success probability decreases exponentially with N, works well for $N \leq 6$.

Our approach

1. Design a variant of ML-DSA that is threshold-friendly for $T = N$ parties.

   Idea: Sample N independent ML-DSA secrets in Keygen and prove their knowledge independently.

   Note: The success probability decreases exponentially with N, works well for $N \leq 6$.

2. N-out-of-N Threshold ML-DSA.

Our approach

1. Design a variant of ML-DSA that is threshold-friendly for $T = N$ parties.

   Idea: Sample N independent ML-DSA secrets in Keygen and prove their knowledge independently.

   Note: The success probability decreases exponentially with N, works well for $N \leq 6$.

2. N-out-of-N Threshold ML-DSA.

3. Extend to $T \neq N$.

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \leq i \leq N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

Sample N secrets, and aggregate the knowledge proofs.

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short
- Assert $\mathbf{z}$ is small

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$

Sample N secrets, and aggregate the knowledge proofs.

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \mathrm{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short
- Assert $\mathbf{z}$ is small

Sample a $\mathbf{w}_i$ for each secret, and do not rely on rounding for security:
reintroduce error in $\mathbf{w}_i$ for rejection sampling on $\mathbf{e}$

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$

Sample N secrets, and aggregate the knowledge proofs.

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short
- Assert $\mathbf{z}$ is small

Sample a $\mathbf{w}_i$ for each secret, and do not rely on rounding for security:
reintroduce error in $\mathbf{w}_i$ for rejection sampling on $\mathbf{e}$

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**

Sample N secrets, and aggregate the knowledge proofs.

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short
- Assert $\mathbf{z}$ is small

Sample a $\mathbf{w}_i$ for each secret, and do not rely on rounding for security:
reintroduce error in $\mathbf{w}_i$ for rejection sampling on $\mathbf{e}$

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Sample N secrets, and aggregate the knowledge proofs.

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short
- Assert $\mathbf{z}$ is small

Sample a $\mathbf{w}_i$ for each secret, and do not rely on rounding for security:
reintroduce error in $\mathbf{w}_i$ for rejection sampling on $\mathbf{e}$

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Sample N secrets, and aggregate the knowledge proofs.

ML-DSA . Verify(vk, msg, sig $= (\mathbf{z}, \lfloor \mathbf{w} \rceil)$)

- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short
- Assert $\mathbf{z}$ is small

We use more compact distributions than ML-DSA to still pass verification → supports up to 6 parties

Threshold ML-DSA for N parties ($T = N$)

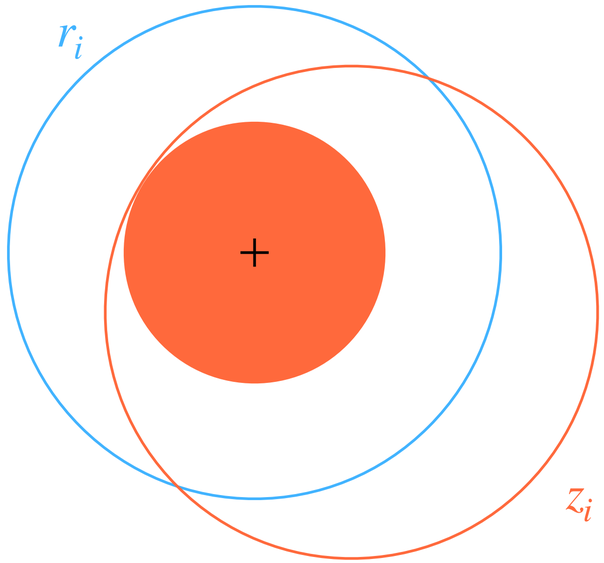
ML-DSA* . Keygen() →

- For $1 \le i \le N$, vk
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg

- For $1 \le i \le N$
  - Sample short
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i +$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathrm{ms}$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i =$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Rejection sampling with hyperballs



e knowledge proofs.

We use more compact distributions than ML-DSA to still pass verification → supports up to 6 parties

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Th-ML-DSA . Sign(sk, msg) → sig

Round 1:

- Sample short $\mathbf{r}_i, \mathbf{e}'_i$
- Broadcast $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$

Round 2:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathsf{e}_i + \mathbf{e}'_i$
- If $(\mathbf{z}_i, \mathbf{y}_i)$ in S, **broadcast $\mathbf{z}_i$, else abort**

Combine:

- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where $\mathsf{sk}, \mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

But, the scheme is only secure if corrupted parties do not bias $\mathbf{w}$

Th-ML-DSA . Sign(sk, msg) → sig

Round 1:

- Sample short $\mathbf{r}_i, \mathbf{e}'_i$
- Broadcast $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$

Round 2:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathsf{e}_i + \mathbf{e}'_i$
- If $(\mathbf{z}_i, \mathbf{y}_i)$ in S, **broadcast $\mathbf{z}_i$, else abort**

Combine:

- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Threshold ML-DSA for N parties ($T = N$)

ML-DSA* . Keygen() → sk, vk

- For $1 \le i \le N$, $\mathsf{vk}_i = \mathbf{A} \cdot \mathsf{sk}_i + \mathbf{e_i}$, where sk, $\mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA* . Sign(sk, msg) → sig

- For $1 \le i \le N$
  - Sample short $\mathbf{r}_i, \mathbf{e}'_i$
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- For $1 \le i \le N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Th-ML-DSA . Sign(sk, 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 $\mathsf{commit}_i = H(\mathbf{w}_i)$

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

Combine:

- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Techniques from [dPN25].

Threshold ML-DSA for $T \neq N$ parties

Use Replicated Secret Sharing [dPN25]

ML-DSA* . Keygen() → 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.

Threshold ML-DSA for $T \neq N$ parties

Use Replicated Secret Sharing [dPN25]

ML-DSA* . Keygen() → 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 $\sqcup_{i \in \mathsf{SS}} m_i = \{I \text{ s.t. } |I| = N - T + 1\}$:

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

Threshold ML-DSA for $T \neq N$ parties

Use Replicated Secret Sharing [dPN25]

ML-DSA* . Keygen() → 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 $\sqcup_{i \in SS} m_i = \{I \text{ s.t. } |I| = N - T + 1\}$:

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

Th-ML-DSA . Sign(sk, 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 $\mathsf{commit}_i = H(\mathbf{w}_i)$

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

Combine:

- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Techniques from [dPN25].

Threshold ML-DSA for $T \neq N$ parties

Use Replicated Secret Sharing [dPN25]

ML-DSA* . Keygen() → sk, vk

- For every possible set I of $N - T + 1$ pa
  - $\mathsf{vk}_I = \mathbf{A} \cdot \mathsf{sk}_I + \mathbf{e}_I$, where $\mathsf{sk}_I, \mathbf{e}_I$ sho
  - 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
at least one of these secrets remai

2. T parties can collaboratively reconstruct the full secret.

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

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

… plus some other optimizations to make parameters as tight as possible

Th-ML-DSA . Sign(sk, msg) → sig

Round 1:

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

:

cast $\mathbf{w}_i$

:

$\sum_i \mathbf{w}_i$ + abort if inconsistent commit_i

$H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$

$c \cdot \sum_{I \in m_i} \mathsf{sk}_I + \mathbf{r}_i, \mathbf{y}_i = c \cdot \sum_{I \in m_i} \mathbf{e}_I + \mathbf{e}'_i$

- If $(\mathbf{z}_i, \mathbf{y}_i)$ in S, **broadcast $\mathbf{z}_i$, else abort**

Combine:

- $\mathsf{sig} = (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

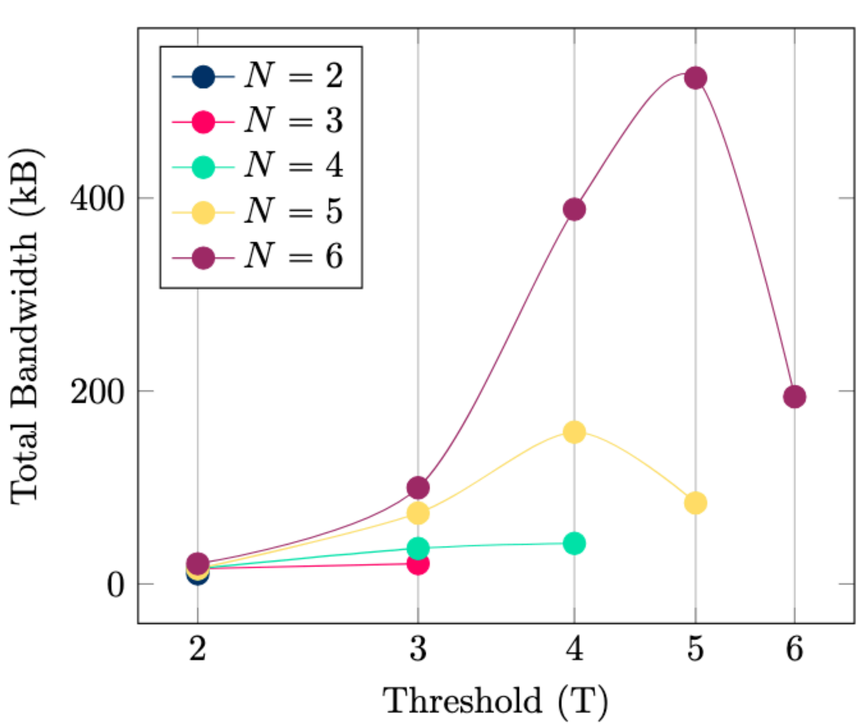
Techniques from [dPN25].

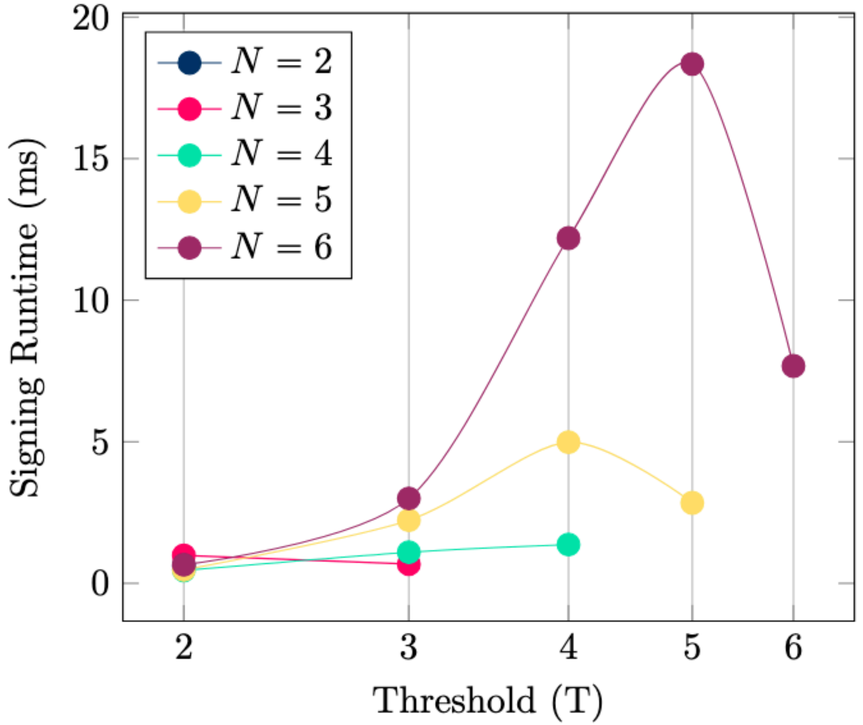
Evaluation

Evaluation

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)

Evaluation

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

(T,N)	Locations	Signing (ms)
(2,6)	T - S	27
(2,6)	T - V	620
(4,6)	T - V - L - L	750
(6,6)	T - V - L - L - S - S	659

Conclusion

Conclusion

Scheme	# Parties	# Rounds	Comm (MB)	Computation	Paradigm	Security
Our work	6	6	0.021 to 1.05	Lightweight	Game-based	Dishonest Majority
Bienstock et al.	Unlimited	96	>1.2*	Online lightweight*	UC	Honest Majority
		24	>2.3*			
Trilithium	2	60	234	Heavy	UC	Trusted Party

Average # rounds, and communication per party to obtain a valid signature

** Communication and computation exclude cost of offline correlated randomness generation.*

Conclusion

Scheme	# Parties	# Rounds	Comm (MB)	Computation	Paradigm	Security
Our work	6	6	0.021 to 1.05	Lightweight	Game-based	Dishonest Majority
Bienstock et al.	Unlimited	96	>1.2*	Online lightweight*	UC	Honest Majority
		24	>2.3*			
Trilithium	2	60	234	Heavy	UC	Trusted Party

Scheme	# Parties	# Rounds	Comm (MB)
Threshold ECDSA	Unlimited	7	> 4.1T kB
		3	> 45.3T kB

Questions?

Evaluation

Other ML-DSA parameter sets