Efficient Threshold ML-DSA up to 6 parties

Post-Quantum Threshold Signatures Compatible with the NIST Standard

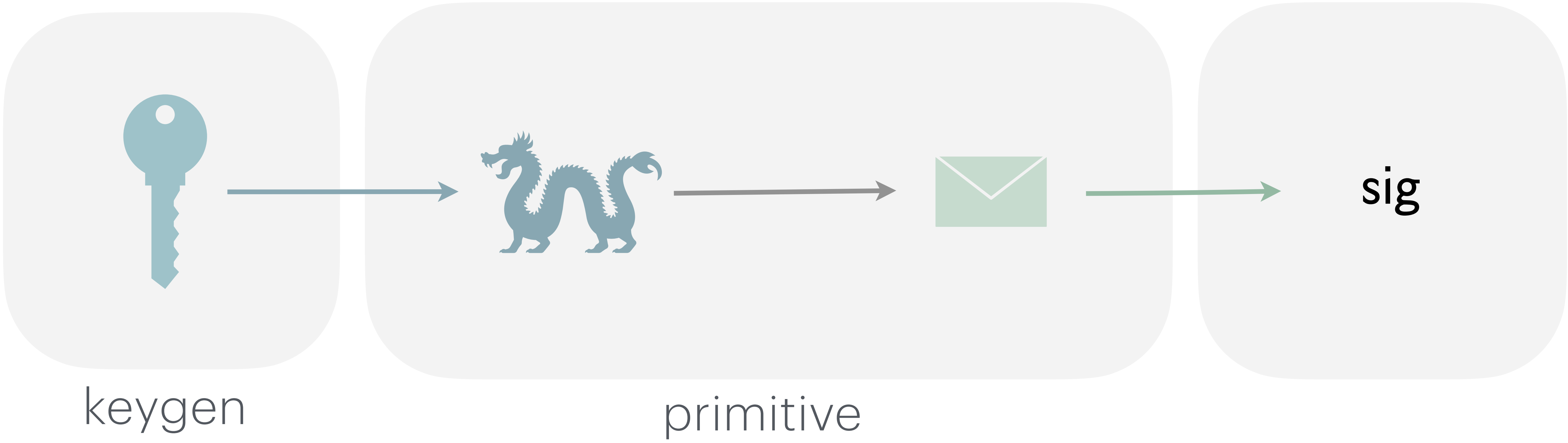
Guilhem Niot, joint works with PQShield & Friends

Seminar JPMorgan - New York, US

PQ SHIELD

Threshold Signatures

Centralized setting



Threshold Signatures

What if the party is corrupted or becomes unresponsive…

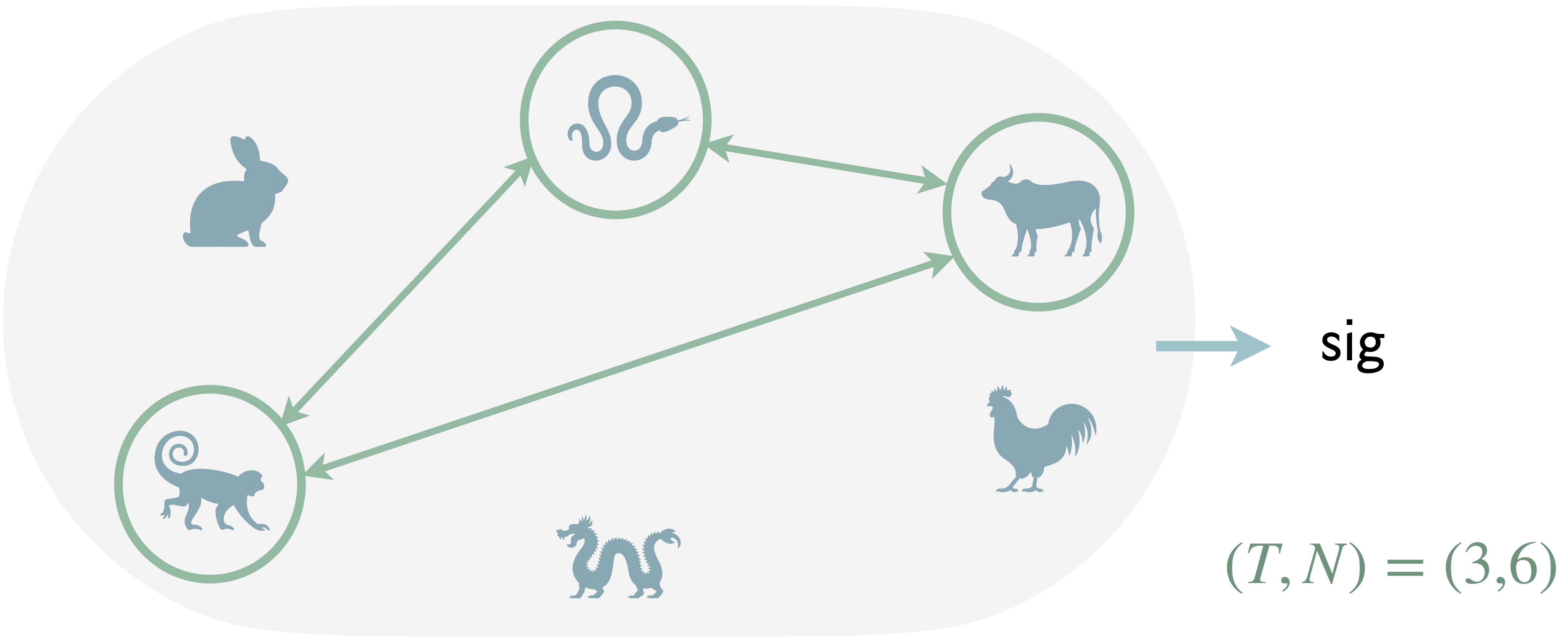
Question: can we split the trust among several parties?

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.



Applications of Threshold Signatures

Cryptocurrency wallets & DeFi

Distributed signing for CDNs

Distributed consensus in Tor

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?

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*

MuSig-L: Lattice-Based Multi-Signature With Single-Round Online Phase*

Cecilia Boschini[1], Akira Takahashi[2], and Mehdi Tibouchi[3]

Post-Quantum Threshold Signatures?

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*

MuSig-L: Lattice-Based Multi-Signature With Single-Round Online Phase*

Cecilia Boschini[1], Akira Takahashi[2], and Mehdi Tibouchi[3]

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() $\rightarrow$ 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)

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

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

To sign: prove knowledge of $\mathsf{sk}, \mathbf{e}$, without revealing $\mathsf{sk}, \mathbf{e}$. *(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() $\to$ 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 $\mathbf{r}$ $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$	$\xrightarrow{\mathbf{w}}$		
		$\xleftarrow{c}$	Sample challenge c with high entropy	2

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

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

To sign: prove knowledge of $\mathsf{sk}, \mathbf{e}$, without revealing $\mathsf{sk}, \mathbf{e}$. *(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 $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$	$\mathbf{z}$ →	Verify $\mathbf{w} - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short Verify that $\mathbf{z}$ is short	

ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

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

To sign: prove knowledge of $\mathsf{sk}, \mathbf{e}$, without revealing $\mathsf{sk}, \mathbf{e}$. *(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 $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$ If $\mathbf{z} \notin S$, restart	$\mathbf{z}$ →	Verify $\mathbf{w} - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short Verify that $\mathbf{z}$ is short	

ML-DSA signatures

$\mathsf{ML\text{-}DSA.Keygen}() \to \mathsf{sk}, \mathsf{vk}$

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

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

To sign: prove knowledge of $\mathsf{sk}, \mathbf{e}$, without revealing $\mathsf{sk}, \mathbf{e}$. *(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 $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short Verify that $\mathbf{z}$ is short	

ML-DSA signatures

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

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

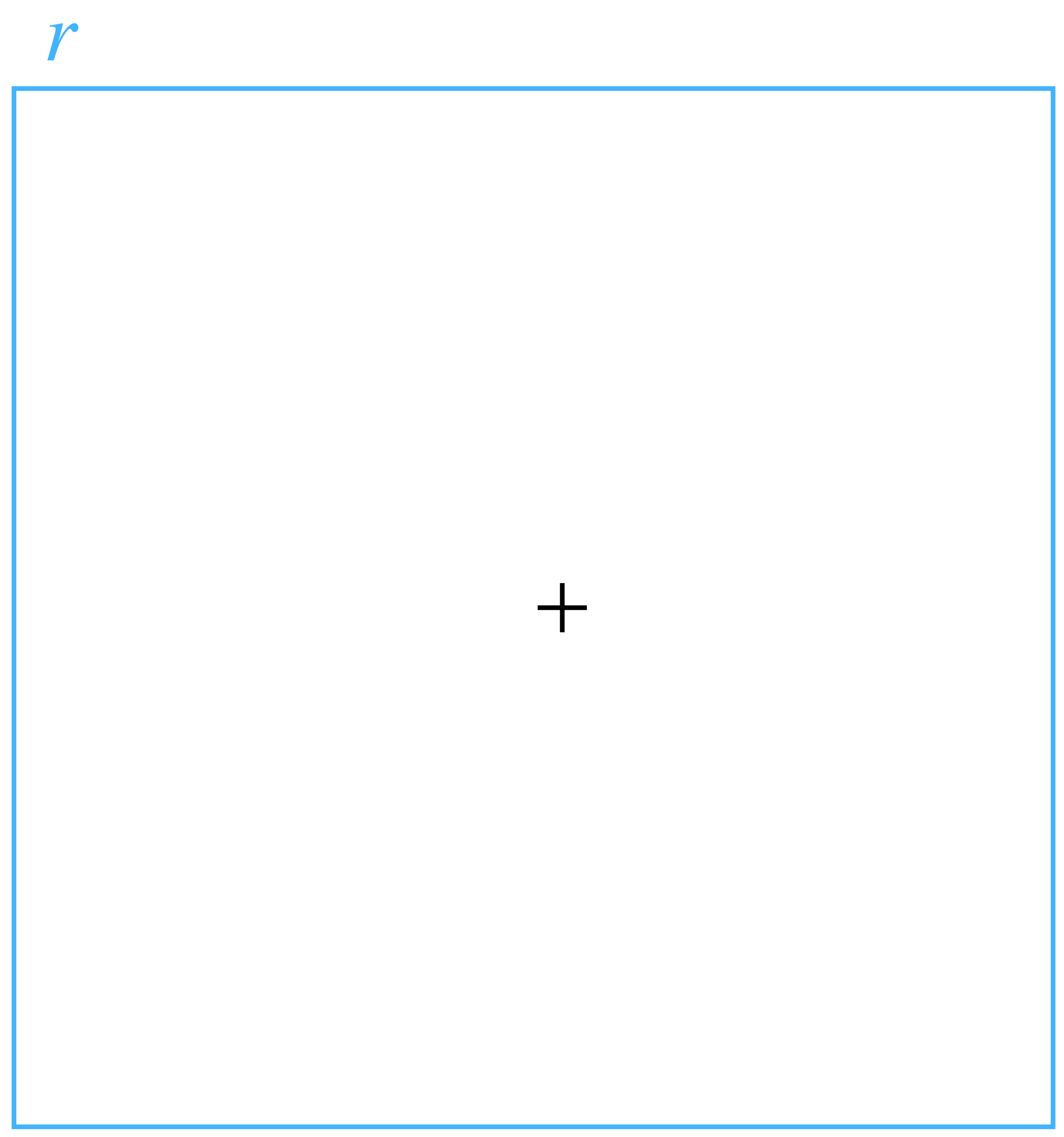
MLWE **assumption:** vk appears uniformly distributed for $\mathbf{A}$ wide enough (more inputs than outputs)

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	$\mathbf{z}$ $\longrightarrow$	Verify $\lfloor \mathbf{w} \rceil - (\mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{vk})$ is short Verify that $\mathbf{z}$ is short

Rejection sampling

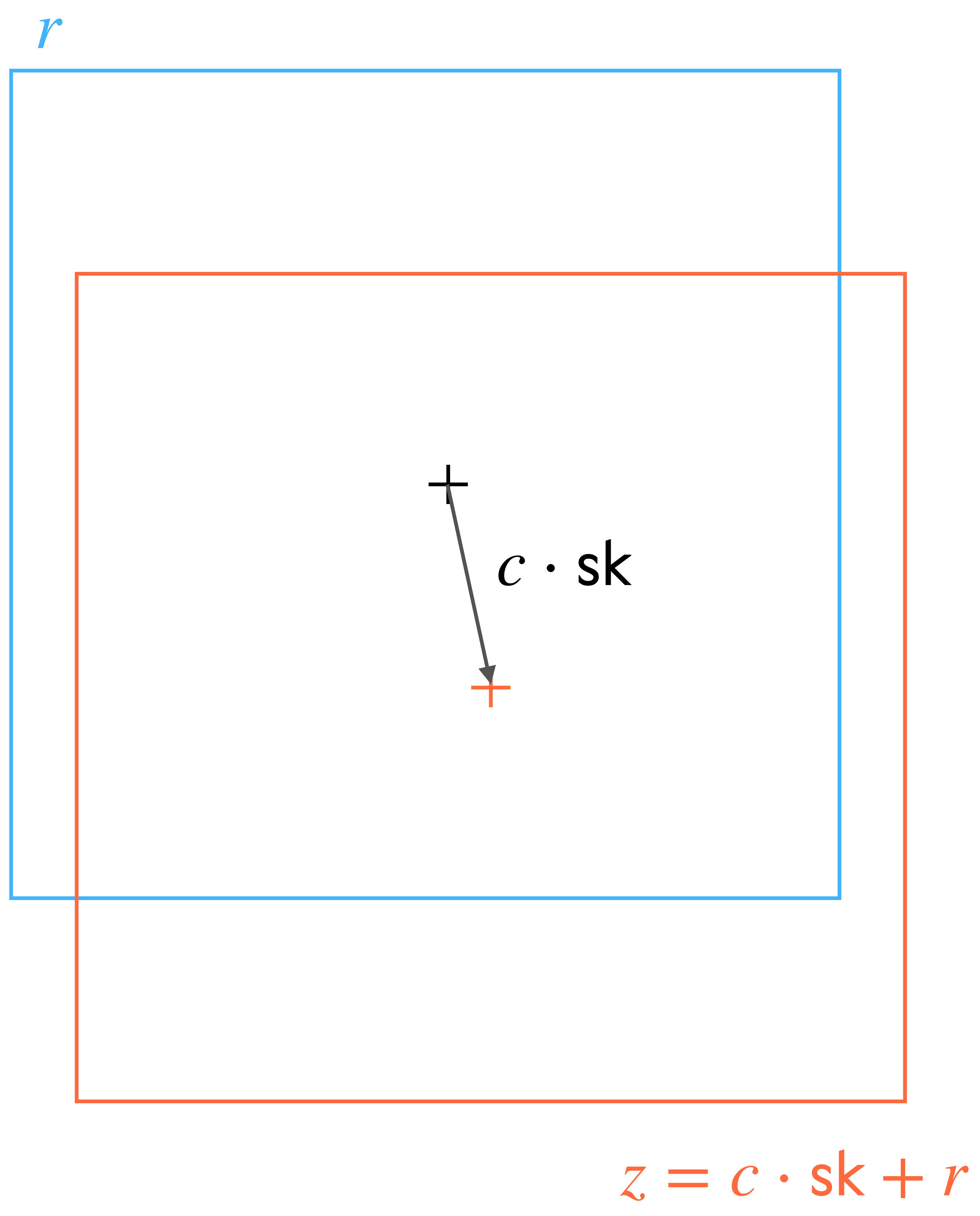
Sample r in a centered hypercube.



Rejection sampling

Sample r in a centered hypercube.

Then, the distribution of z depends on the secret.

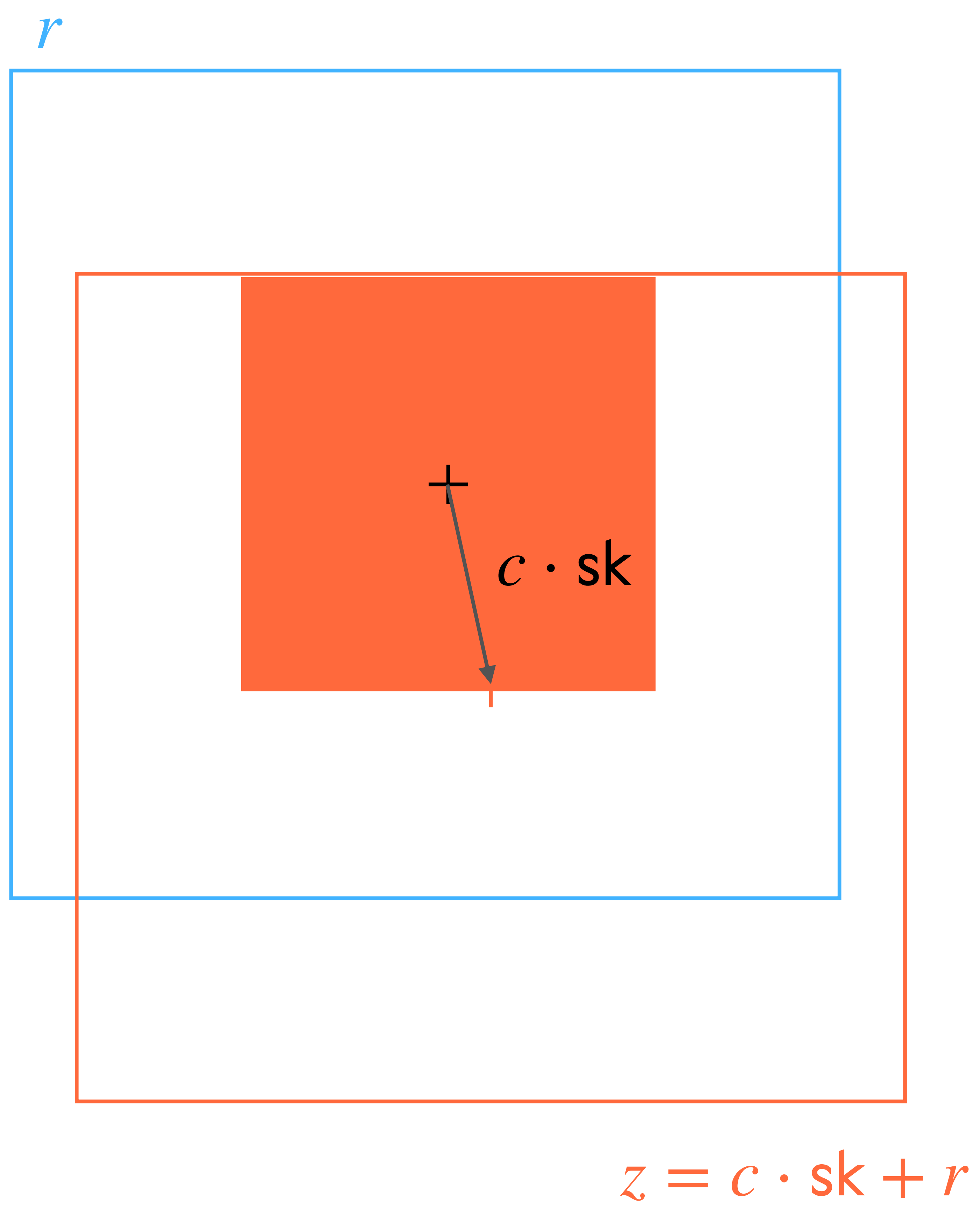


Rejection sampling

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() $\rightarrow$ sk, vk

- vk $= \mathbf{A} \cdot$ 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) $\rightarrow$ 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 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

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

$\text{ML-DSA}^*.\text{Keygen}() \to \text{sk}, \text{vk}$

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

Sample N secrets, and aggregate the knowledge proofs.

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

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

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 $\mathsf{sk}, \mathbf{e}_i$ short
- $\mathsf{vk} = \sum_i \mathsf{vk}_i$

ML-DSA*.Sign(sk, msg) → sig

- For $1 \leq i \leq N$
  - 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 $1 \leq i \leq N$,
  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}'_i$
- 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

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() $\rightarrow$ sk, vk

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

ML-DSA*. Sign(sk, msg) $\rightarrow$ sig

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

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

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

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

ML-DSA*. Keygen() →

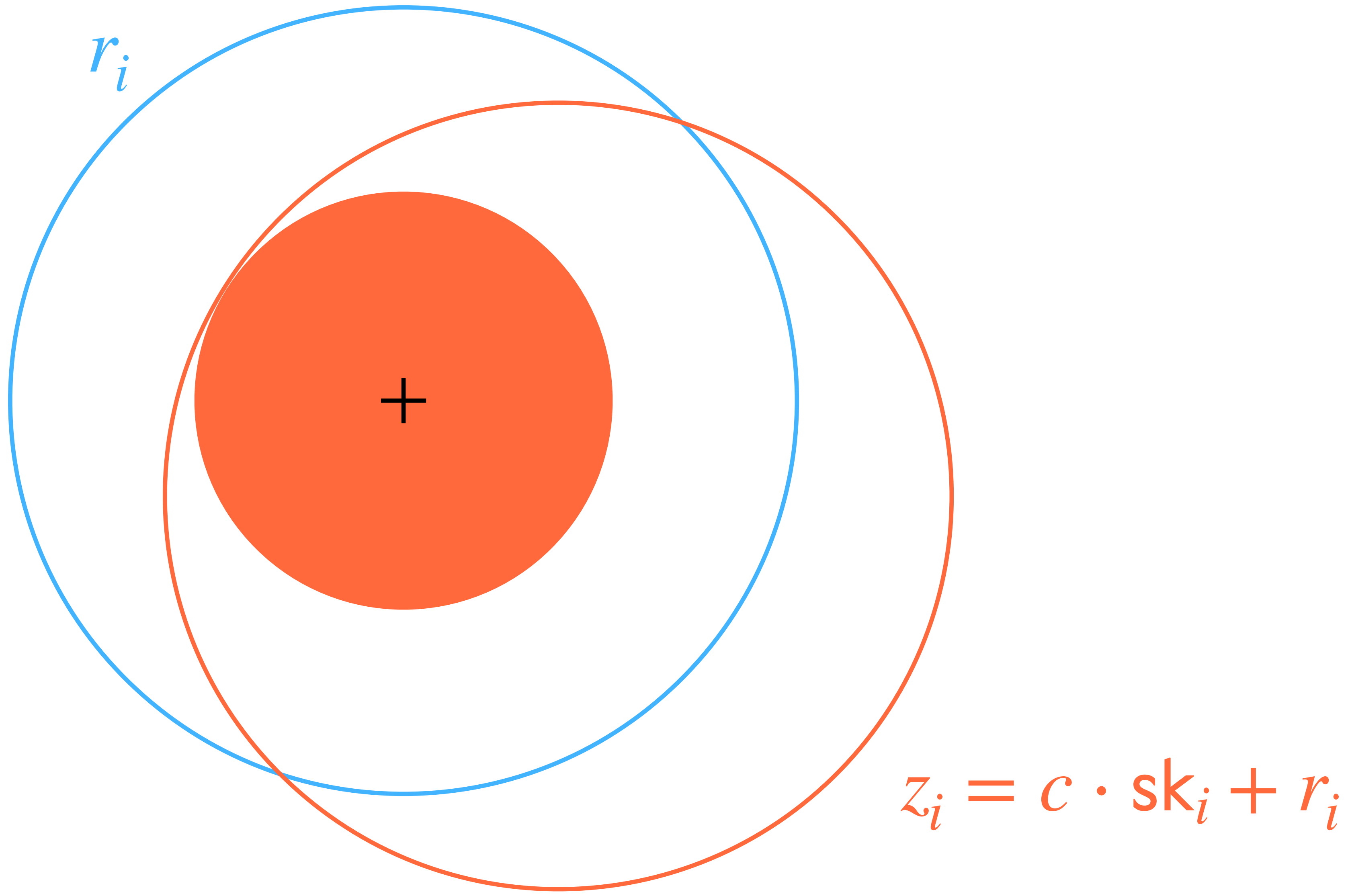
- For $1 \leq i \leq N$, vk
- vk $= \sum_i$ vk$_i$

ML-DSA*. Sign(sk, msg

- For $1 \leq i \leq N$
  - Sample short
  - $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i +$
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\lfloor \mathbf{w} \rceil$, ms
- For $1 \leq i \leq N$,

  $\mathbf{z}_i = c \cdot \mathsf{sk}_i + \mathbf{r}_i, \mathbf{y}_i = c \cdot \mathbf{e}_i + \mathbf{e}_i$
- If any $(\mathbf{z}_i, \mathbf{y}_i)$ not in S, **restart**
- sig $= (\sum_i \mathbf{z}_i, \lfloor \mathbf{w} \rceil)$
- If sig not in S', **restart**
- **return** sig

Rejection sampling with hyperballs



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

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

$\text{ML-DSA}^*.\text{Keygen}() \rightarrow \text{sk}, \text{vk}$

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

$\text{ML-DSA}^*.\text{Sign}(\text{sk}, \text{msg}) \rightarrow \text{sig}$

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

$\text{Th-ML-DSA}.\text{Sign}(\text{sk}, \text{msg}) \rightarrow \text{sig}$

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:

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

Combine:

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

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

$\text{ML-DSA}^*.\text{Keygen}() \rightarrow \text{sk}, \text{vk}$

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

$\text{ML-DSA}^*.\text{Sign}(\text{sk}, \text{msg}) \rightarrow \text{sig}$

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

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

$\text{Th-ML-DSA}.\text{Sign}(\text{sk}, \text{msg}) \rightarrow \text{sig}$

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:

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

Combine:

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

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

$\text{ML-DSA}^*.\text{Keygen}() \to \text{sk}, \text{vk}$

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

$\text{ML-DSA}^*.\text{Sign}(\text{sk}, \text{msg}) \to \text{sig}$

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

$\text{Th-ML-DSA}.\text{Sign}(\text{sk}, \text{msg}) \to \text{sig}$

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

Combine:

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

Techniques from [dPN25].

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

ML-DSA*. Keygen() → sk, vk

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

ML-DSA*. Sign(sk, msg) → sig

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

Is it safe to reveal $\mathbf{w}_i$ in case of abort?

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 $\text{commit}_i = H(\mathbf{w}_i)$

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

Combine:

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

Techniques from [dPN25].

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

Recent result from [dPN25]:

Lemma: Rejected $\mathbf{w}_i$ is indistinguishable from uniform if:

- MLWE is hard over $\chi_{\mathbf{r}}$
- MLWE is hard over $\chi_{\mathbf{z}}$

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

Use Replicated Secret Sharing [dPN25]

ML-DSA* . Keygen() $\rightarrow$ 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.

Th-ML-DSA . Sign(sk, msg) $\rightarrow$ 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
- $c = H(\lfloor \mathbf{w} \rceil, \mathsf{msg})$
- $\mathbf{z}_i = c \cdot \sum_{I \in m_i} \mathsf{sk}_I + \mathbf{r}_i, \mathbf{y}_i = c \cdot \sum_{I \in m_i} \mathsf{e}_I + \mathbf{e}'_i$
- 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]

$\text{ML-DSA}^*.\text{Keygen}() \to \text{sk}, \text{vk}$

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

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

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

$\text{Th-ML-DSA}.\text{Sign}(\text{sk}, \text{msg}) \to \text{sig}$

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

Combine:

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

Techniques from [dPN25].

Optimizing for ML-DSA

Optimizing for ML-DSA

1. We can accept a somewhat low success probability by performing K attempts in parallel.

Optimizing for ML-DSA

1. We can accept a somewhat low success probability by performing K attempts in parallel.

2. **Unbalanced constraints:** The aggregated signature must be small enough for ML-DSA verification.

- For the first half $\mathbf{z}$: infinite norm constraint
- For the second half $\mathbf{y}$ + rounding: (smaller) infinite norm constraint + deserialization constraint for the recovery of $\lfloor \mathbf{w} \rceil$

$\rightarrow$ stronger constraint on second half: we want to use smaller $\mathbf{y}$ than $\mathbf{z}$

Optimizing for ML-DSA

1 We can accept a somewhat low success probability by performing K attempts in parallel.

2 **Unbalanced constraints:** The aggregated signature must be small enough for ML-DSA verification.

- For the first half $\mathbf{z}$: infinite norm constraint
- For the second half $\mathbf{y}$ + rounding: (smaller) infinite norm constraint + deserialization constraint for the recovery of $\lfloor \mathbf{w} \rceil$

$\rightarrow$ stronger constraint on second half: we want to use smaller $\mathbf{y}$ than $\mathbf{z}$

Solution: We perform hyperball rejection sampling on $(s, \nu \cdot \mathbf{e})$ for $\nu > 1$: reduces the second half contribution.

Optimizing for ML-DSA

1. We can accept a somewhat low success probability by performing K attempts in parallel.

2. **Unbalanced constraints:** The aggregated signature must be small enough for ML-DSA verification.
   - For the first half $\mathbf{z}$: infinite norm constraint
   - For the second half $\mathbf{y}$ + rounding: (smaller) infinite norm constraint + deserialization constraint for the recovery of $\lfloor \mathbf{w} \rceil$

   $\rightarrow$ stronger constraint on second half: we want to use smaller $\mathbf{y}$ than $\mathbf{z}$

3. The size of the hyperball used is **proportional to the norm of the partial secret** to hide: we minimize the number of secrets used by each party in a session.

Optimizing for ML-DSA

3 The size of the hyperball used is **proportional to the norm of the partial secret** to hide: we minimize the number of secrets used by each party in a session.

$\binom{N}{N-T+1}$ secrets to partition among T parties.

Ideally, at most $\left\lceil \binom{N}{N-T+1} \Big/ T \right\rceil$ secrets each.

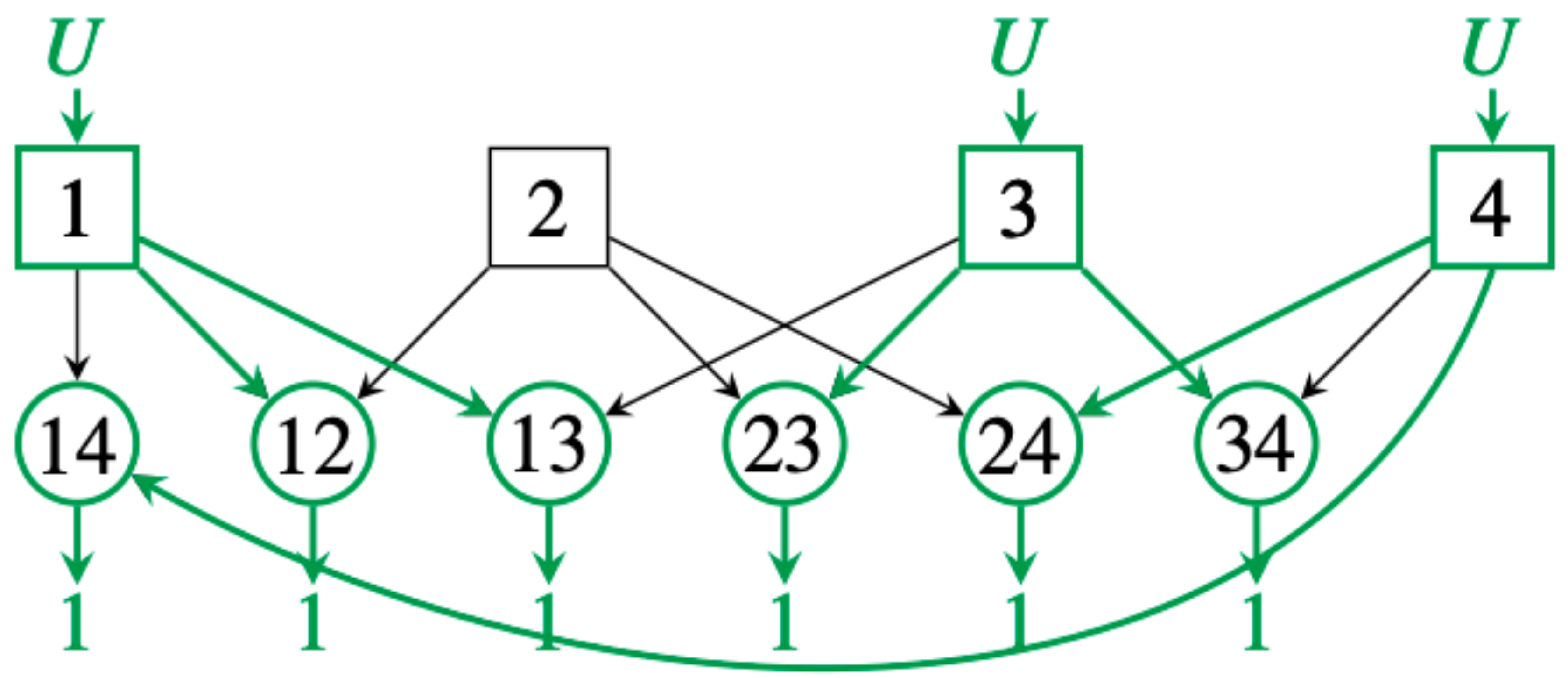
Optimizing for ML-DSA

3 The size of the hyperball used is **proportional to the norm of the partial secret** to hide: we minimize the number of secrets used by each party in a session.

$\binom{N}{N-T+1}$ secrets to partition among T parties.

Ideally, at most $\left\lceil \binom{N}{N-T+1} / T \right\rceil$ secrets each.

We find an optimal partition with a max-flow algorithm.

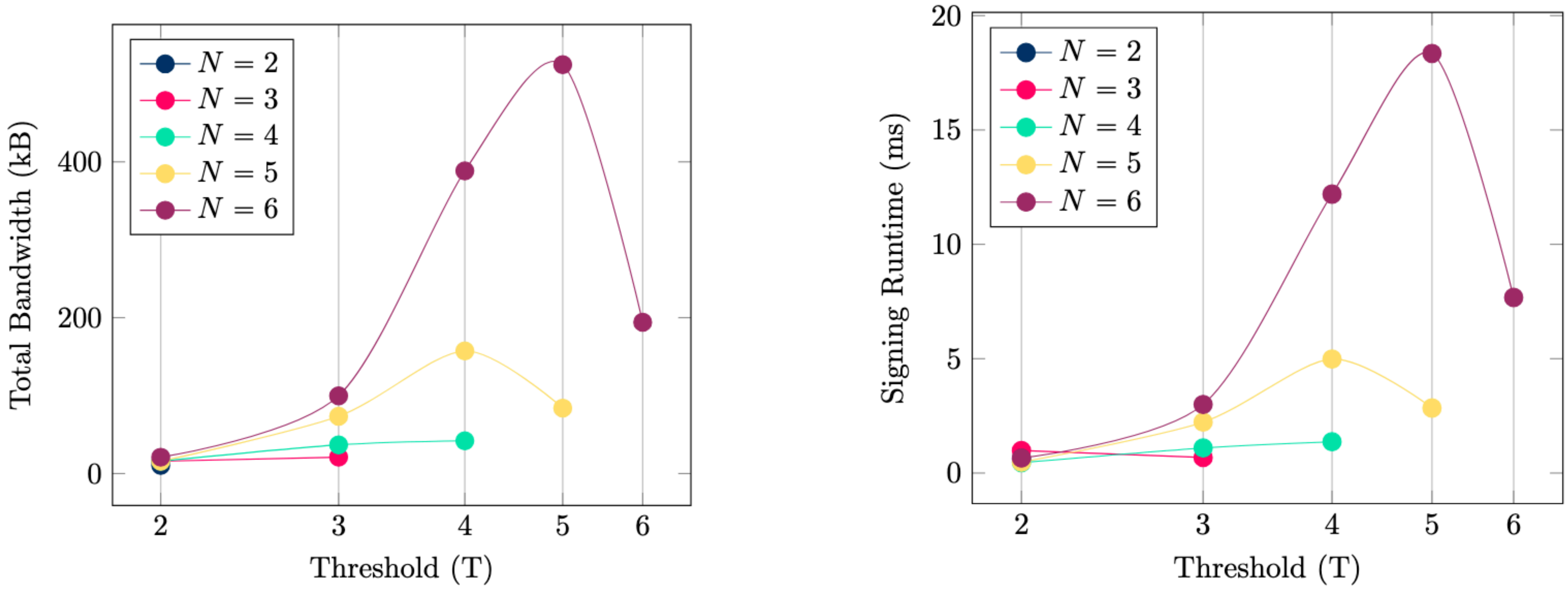


Evaluation

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)

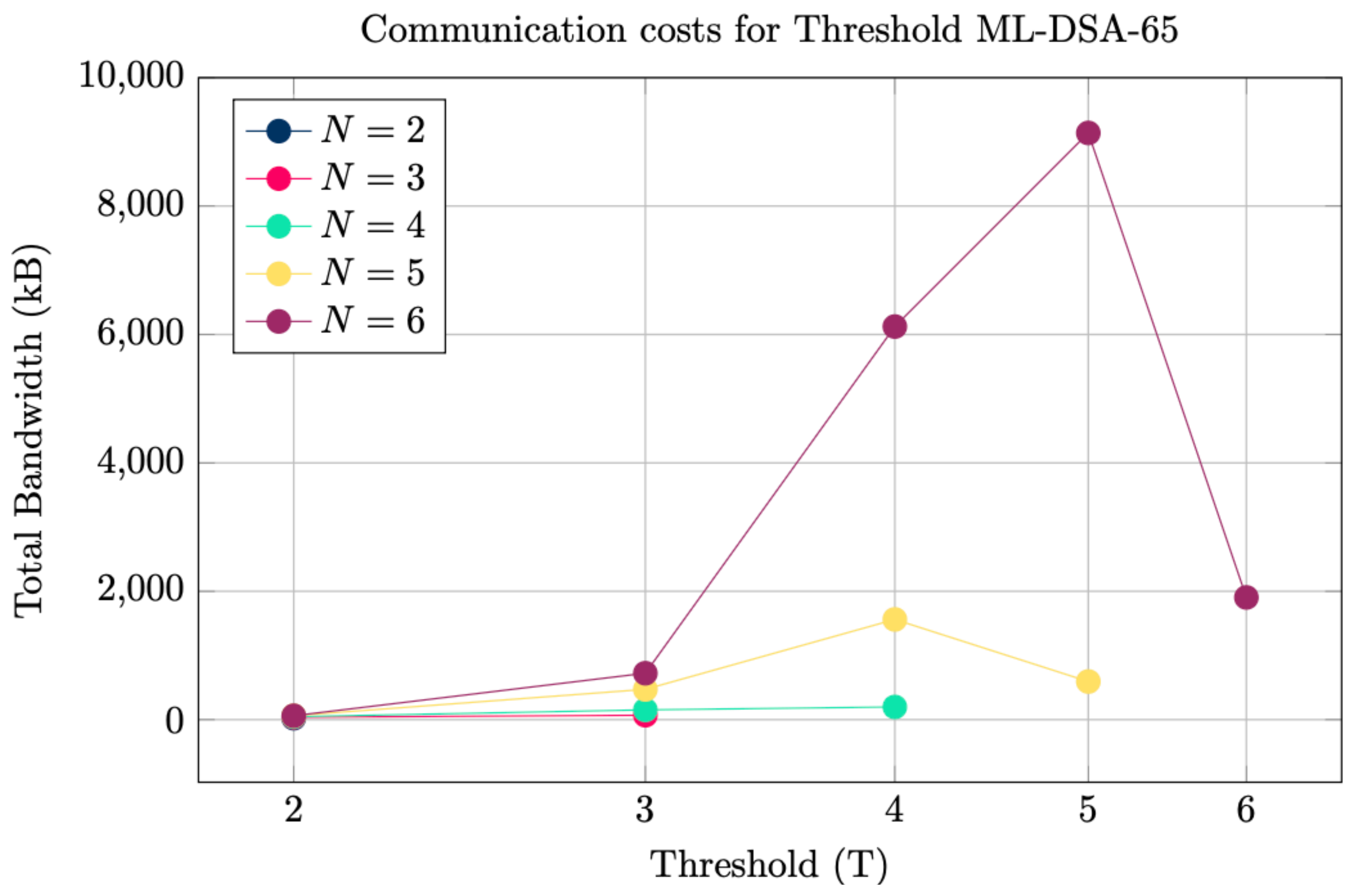
Evaluation

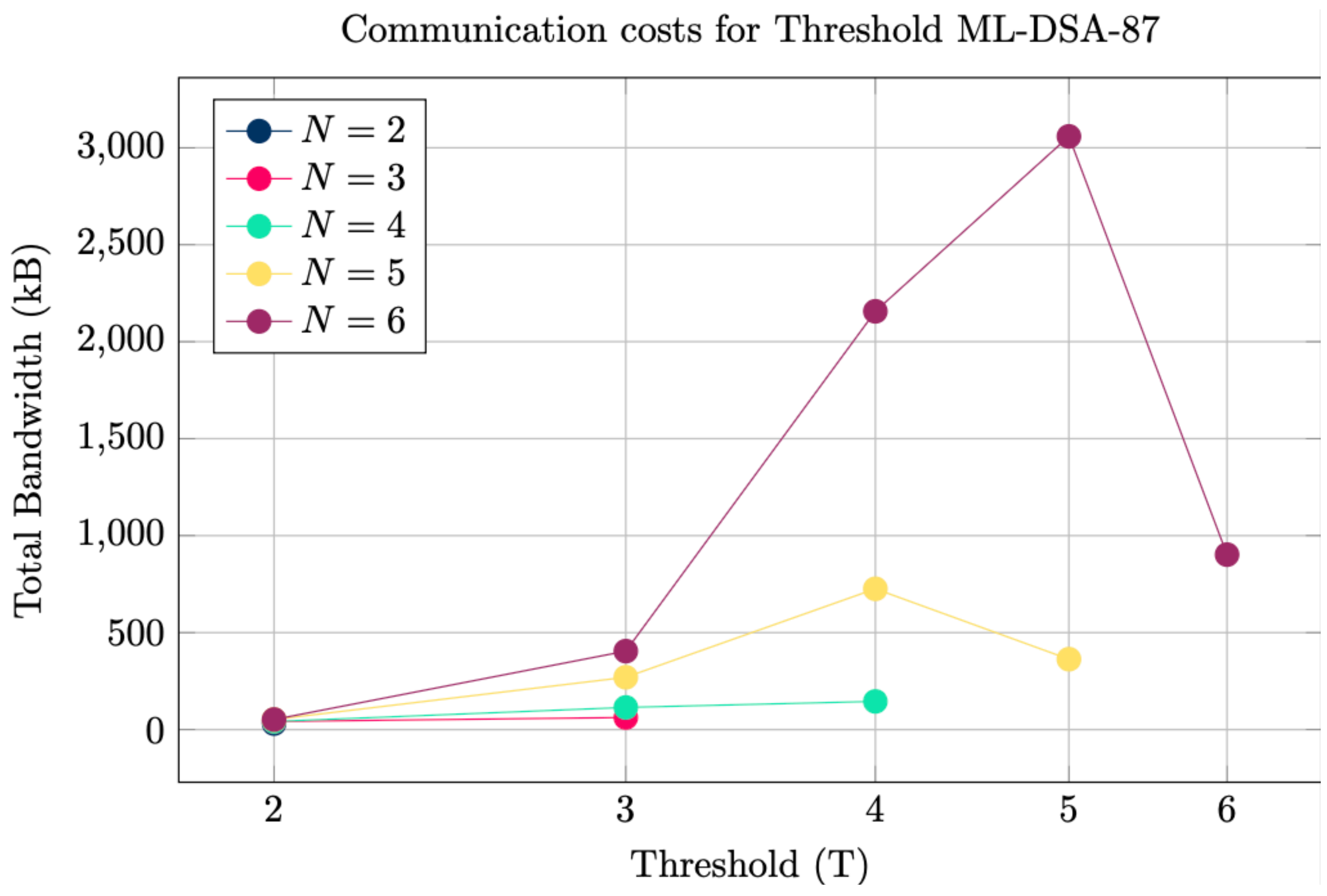
Table 6: WAN signing latency (in ms) for Threshold ML-DSA-44 and T-Raccoon-I across different topologies. L = London, S = Seoul, T = Taipei, V = Virginia.

Scheme	(T, N)	**Locations**	**Signing (ms)**
ML-DSA	(2,6)	T – S	27.34
ML-DSA	(2,6)	T – V	620.43
ML-DSA	(4,6)	T – V – L – L	750.65
ML-DSA	(6,6)	T – V – L – L – S – S	659.55

Evaluation

Other ML-DSA parameter sets





Conclusion

Conclusion

Scheme	Paradigm	# Parties	# Rounds	Communication (MB)	Computation	Security
This work	Tailored	6	6	0.021 to 1.05	Lightweight	Standard
Bienstock et al. [BdCE+25]	MPC	Unlimited	96 24	>1.2* >2.3*	Online lightweight*	Honest majority
Trilithium [DKLS25]	MPC	2	60	234†	Heavy	Trusted party†
Generic MPC [CS19]	MPC	Unlimited	High	High	Impractical	Standard

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

Conclusion

Future questions:

- Support more parties
- Online / offline tradeoff
- More scalable scheme by mixing MPC and tailored techniques?

Questions?