

A raccoon is the central focus, sitting on a workbench in a workshop. It is holding a piece of crumpled, foil-wrapped material in its paws. The workshop is filled with various tools, including wrenches, screwdrivers, and pliers, scattered across the workbench. The background shows wooden cabinets and a blurred workshop environment.

Short Shares, Small Coefficients

A New Secret Sharing Scheme and its Applications to Lattice-based
Threshold Cryptography

Guilhem Niot, joint works with *Rafael del Pino, Thomas Espitau, Thomas Prest*

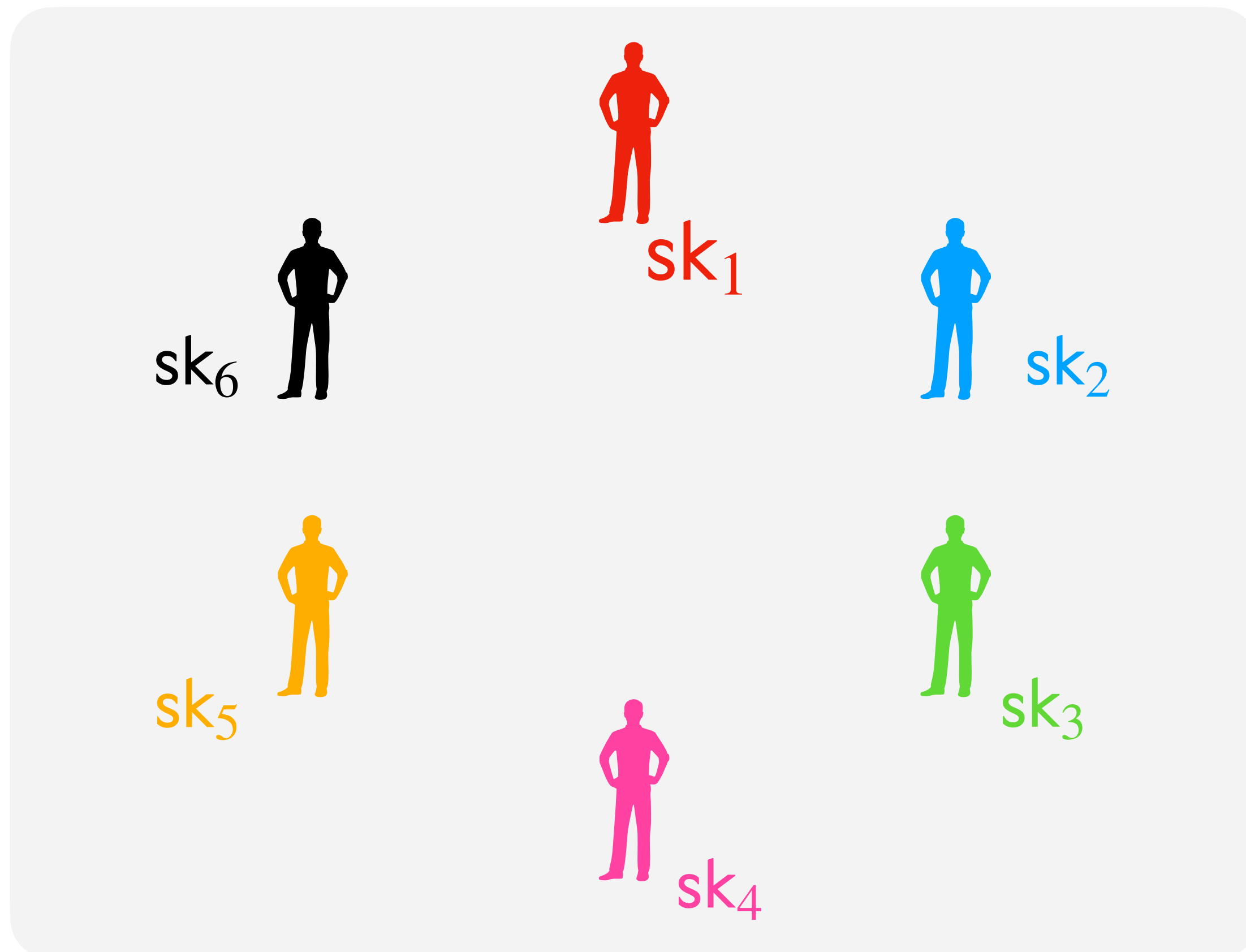
JP Morgan 62nd AlgoCRYPT Seminar - 10. Jan 2025

1. Background

$(T\text{-out-of-}N)$ threshold signatures

What are they?

An interactive protocol to distribute signature generation.

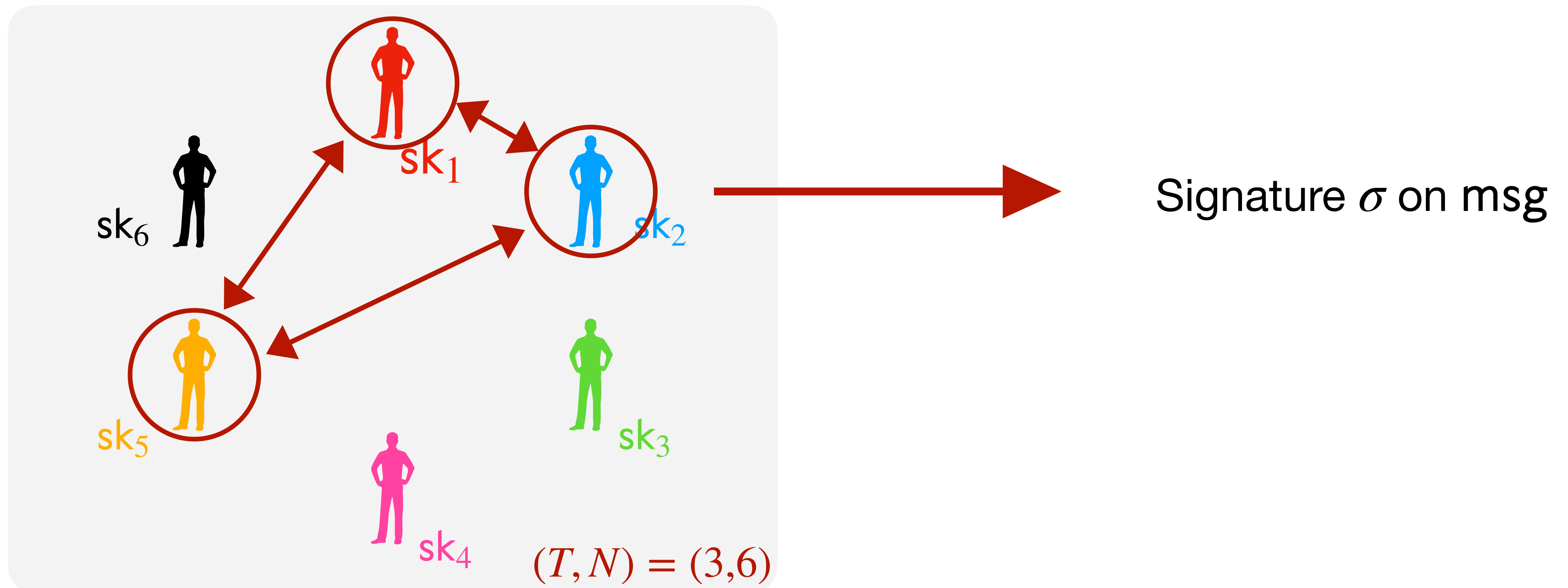


- Global verification key vk
- 1 partial signing key sk_i per party
- T -out-of- N :
 - Any T out of N parties can collaborate to sign a message under vk .
 - $T - 1$ parties cannot sign.

$(T\text{-out-of-}N)$ threshold signatures

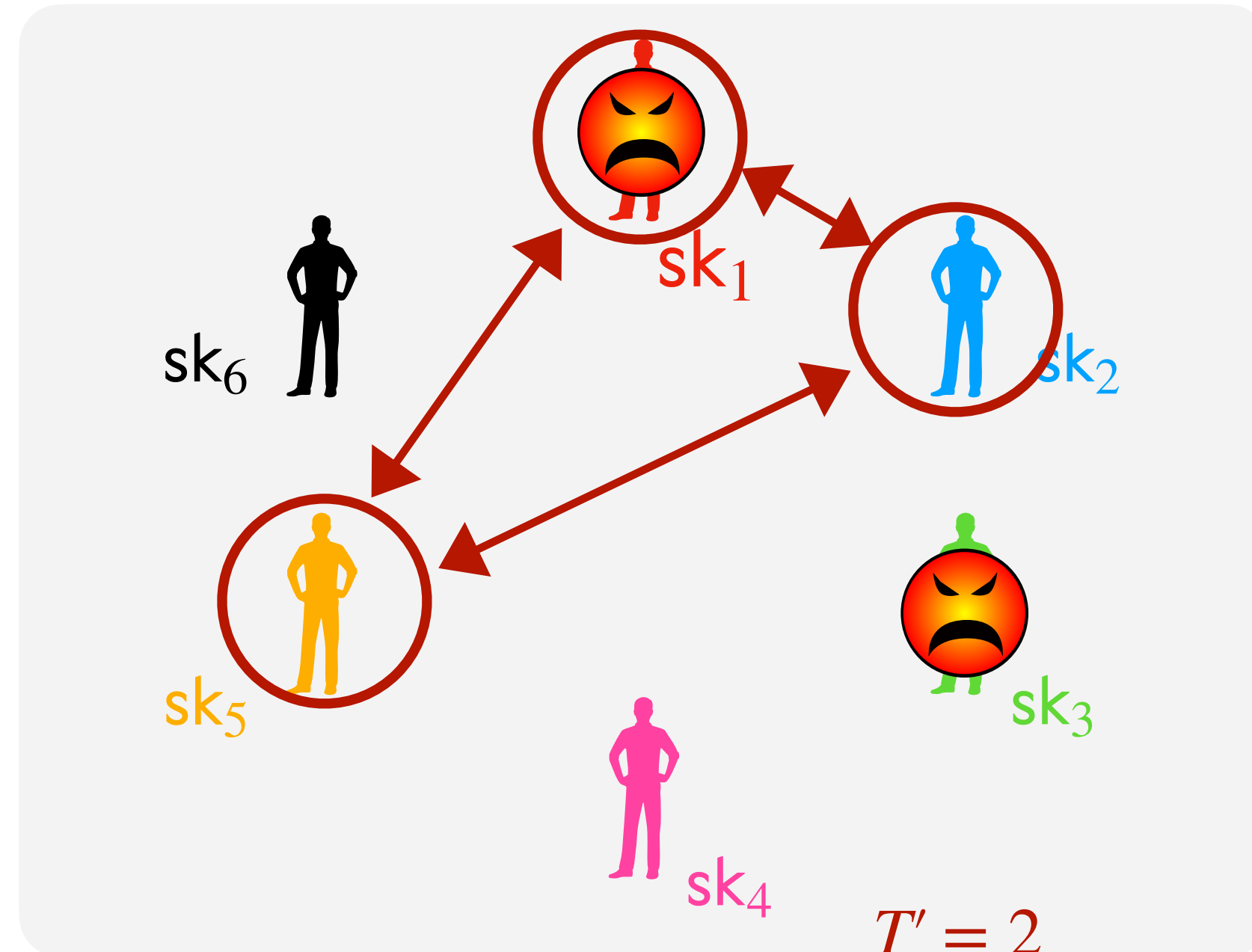
What are they?

An interactive protocol to distribute signature generation.



Core security properties

- **Correctness:** Given at least T -out-of- N partial signing keys, we can sign.
- **(Ramp) Unforgeability:** The signature scheme remains unforgeable even if up to T' parties are corrupted, where $T' \leq T - 1$.



Lattice-based Threshold Signatures

An active field of research.

Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions

Rafael del Pino¹, Shuichi Katsumata^{1,2}, Mary Maller^{1,3}, Fabrice Mouhartem⁴, Thomas Prest¹, Markku-Juhani Saarinen^{1,5}

Two-Round Threshold Signature from Algebraic One-More Learning with Errors

Thomas Espitau¹, Shuichi Katsumata^{1,2}, Kaoru Takemure*^{1,2}

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




Flood and Submerge: Distributed Key Generation and Robust Threshold Signature from Lattices

Thomas Espitau¹ , Guilhem Niot^{1,2} , and Thomas Prest¹ 

Two-round n -out-of- n and Multi-Signatures and Trapdoor Commitment from Lattices*

Ivan Damgård¹, Claudio Orlandi¹, Akira Takahashi¹, and Mehdi Tibouchi²

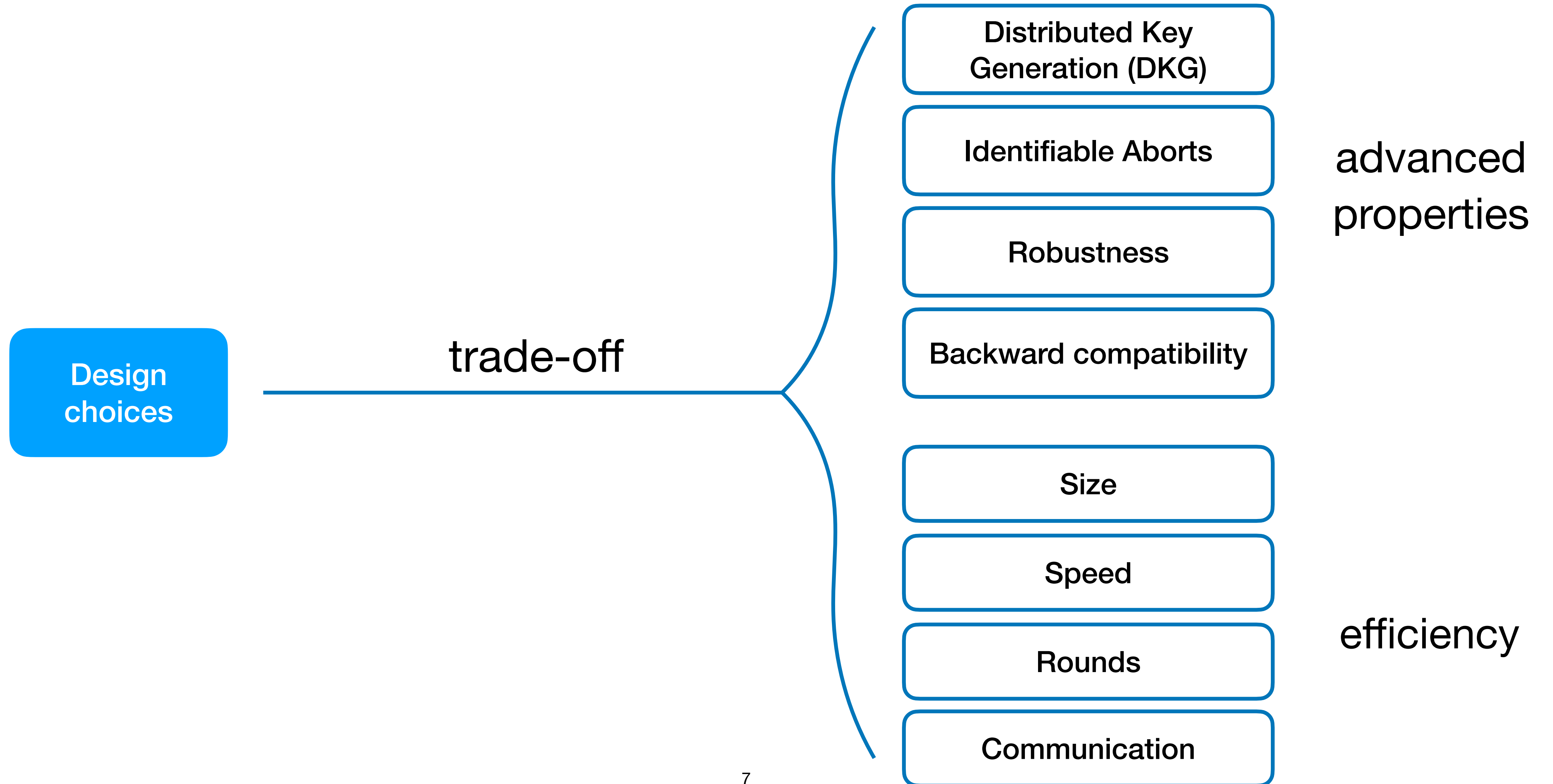
MuSig-L: Lattice-Based Multi-Signature With Single-Round Online Phase*

Cecilia Boschini¹ , Akira Takahashi² , and Mehdi Tibouchi³ 

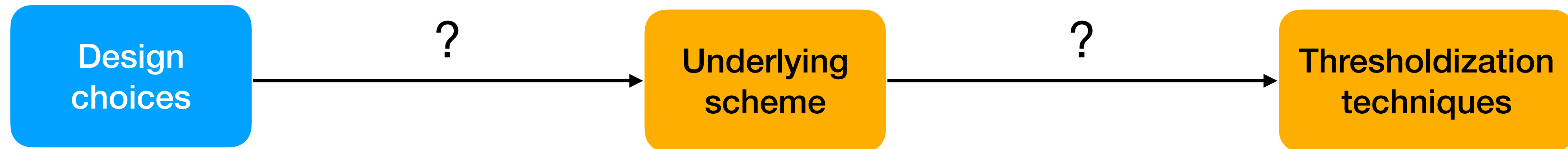
Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption*

Kamil Doruk Gur¹ , Jonathan Katz^{2**} , and Tjerand Silde^{3***} 

Designing a threshold scheme



Designing a threshold scheme



Lattice-based Threshold Signatures

Candidate schemes

*Easier to
thresholdize*

	Hash & Sign	Fiat-Shamir
Gaussian Sampling	Eagle [YJW23]	G+G [DPS23]
Rejection Sampling	Phoenix [JRS24]	Dilithium [LDK+22]
Noise Flooding	Plover [EEN+24]	Raccoon [dEK+24]

*More
compact*

Lattice-based Threshold Signatures

Candidate schemes

	Hash & Sign	Fiat-Shamir
<i>Easier to thresholdize</i> ↓	Gaussian Sampling Eagle [YJW23]	G+G [DPS23]
	Rejection Sampling Phoenix [JRS24]	Dilithium [LDK+22]
	Noise Flooding Plover [EEN+24]	Raccoon [dEK+24]
		↑ <i>More compact</i>

This talk: Raccoon and Dilithium threshold variants.

Lattice-based Threshold Signatures

An active field of research, with different designs.

Thresholdization technique	Size	Speed	Rounds	Comm/party
MPC	S	Slow	15	$\geq 1\text{MB}$
FHE	M	As fast as FHE	2	$\geq 1\text{MB}$
Tailored	S-M	Fast	2-4	20 kB \rightarrow 56T kB

Lattice-based Threshold Signatures

An active field of research, with different designs.

Thresholdization technique	Size	Speed	Rounds	Comm/party
MPC	S	Slow	15	$\geq 1\text{MB}$
FHE	M	As fast as FHE	2	$\geq 1\text{MB}$
Tailored	S-M	Fast	2-4	20 kB \rightarrow 56T kB

This talk: Tailored

Raccoon
 Threshold Raccoon: Practical Threshold Signatures
 from Standard Lattice Assumptions
 Rafael del Pino¹, Shuichi Katsumata^{1,2}, Mary Maller^{1,3}, Fabrice Mouhartem⁴, Thomas
 Prest¹, Markku-Juhani Saarinen^{1,5}

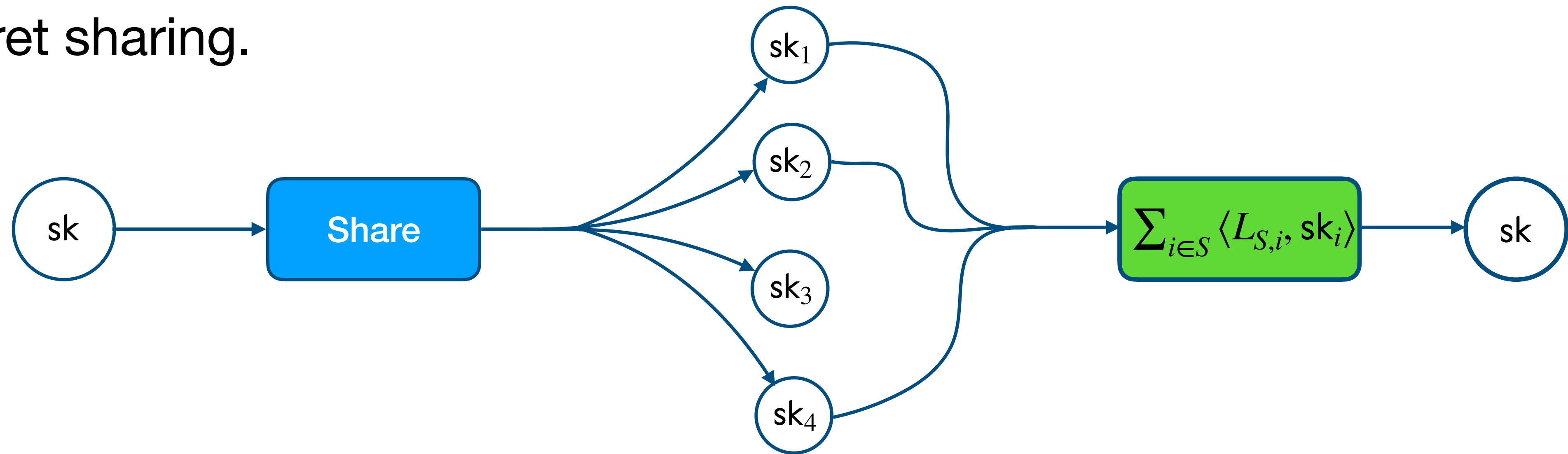
\rightarrow advanced properties?

Dilithium-like
 Two-round n -out-of- n and Multi-Si
 Trapdoor Commitment from Lattices*
 Ivan Damgård¹, Claudio Orlandi¹, Akira Takahashi¹, and Mehdi Tibouchi²

\rightarrow more compact and T -out-of- N ?

Main technique of this talk

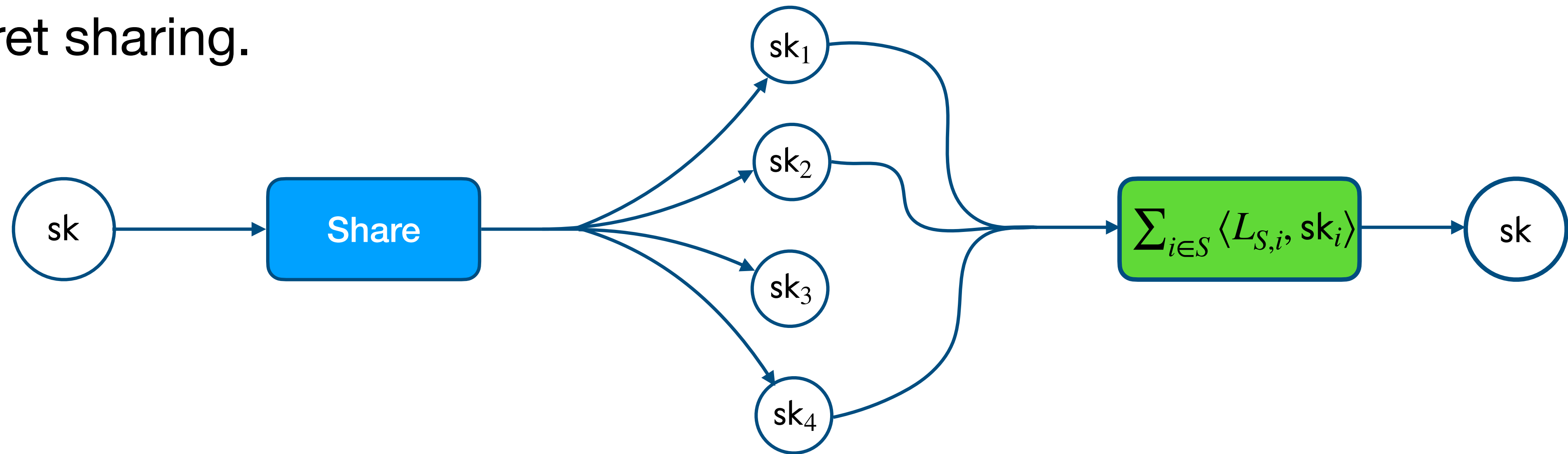
Short secret sharing.



- Individual pool of short shares $sk_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$
- T shares: can recover sk
 - ◆ Reconstruction vector $L_{S,i}$ with small coefficients
- $\leq T - 1$ shares: can't recover sk

Main technique of this talk

Short secret sharing.



- Individual pool of short shares $sk_i = (s_i^{(1)}, s_i^{(2)}, \dots)$
- T shares: can recover sk
 - ◆ Reconstruction vector $L_{S,i}$ with small coefficients
- $\leq T - 1$ shares: can't recover sk

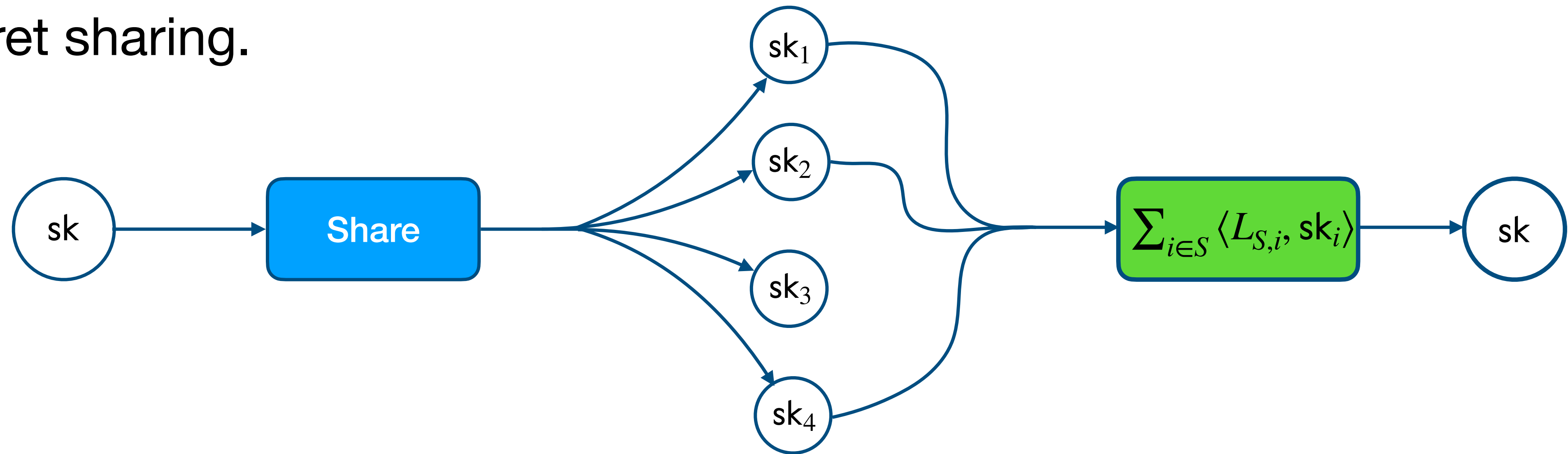
Example: N -out-of- N sharing (one share per party)

- $sk_1, \dots, sk_N \leftarrow \mathcal{D}_\sigma^N$ and $sk = \sum_i sk_i$
- $L_{S,i} = 1$

Extends to T -out-of- N by having several shares per party.

Main technique of this talk

Short secret sharing.



- Individual pool of short shares $sk_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$
- T shares: can recover sk
 - ◆ Reconstruction vector $L_{S,i}$ with small coefficients
- $\leq T - 1$ shares: can't recover sk

Applications:

- Identifiable aborts in Threshold Raccoon
- A compact Dilithium-like Threshold Signature

2. Threshold Raccoon

**Threshold Raccoon: Practical Threshold Signatures
from Standard Lattice Assumptions**

Rafael del Pino¹, Shuichi Katsumata^{1,2}, Mary Maller^{1,3}, Fabrice Mouhartem⁴, Thomas
Prest¹, Markku-Juhani Saarinen^{1,5}

Raccoon signature scheme

Raccoon.Keygen() \rightarrow sk, vk

- $\text{vk} = [\mathbf{A} \quad \mathbf{I}] \cdot \text{sk}$, for sk short

Raccoon.Sign(sk, msg) \rightarrow sig

- Sample a short \mathbf{r}
- $\mathbf{w} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = c \cdot \text{sk} + \mathbf{r}$
- Output sig = (c, \mathbf{z})

Raccoon.Verify(vk, msg, sig = (c, \mathbf{z}))

- $\mathbf{w} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z} - c \cdot \text{vk}$
- Assert $c = H(\mathbf{w}, \text{msg})$
- Assert \mathbf{z} short



* omitting usual rounding techniques

Raccoon signature scheme

`Raccoon.Keygen()` \rightarrow sk, vk

- $vk = [A \quad I] \cdot sk$, for sk short

`Raccoon.Sign(sk, msg)` \rightarrow sig

- Sample a short \mathbf{r}
- $\mathbf{w} = [A \quad I] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, msg)$
- $\mathbf{z} = c \cdot sk + \mathbf{r}$
- Output $sig = (c, \mathbf{z})$

`Raccoon.Verify(vk, msg, sig = (c, z))`

- $\mathbf{w} = [A \quad I] \cdot \mathbf{z} - c \cdot vk$
- Assert $c = H(\mathbf{w}, msg)$
- Assert \mathbf{z} short

Unforgeable assuming

- ◆ **Hint-MLWE**
- ◆ **SelfTargetMSIS**

Hint-MLWE assumption [KLSS23].

(A, vk) is pseudorandom even if given Q “hints”:

$$(c_i, \mathbf{z}_i := c_i \cdot sk + \mathbf{r}_i) \text{ for } i \in [Q]$$

As hard as $MLWE_\sigma$ if

$$\sigma_{\mathbf{r}} \geq \sqrt{Q} \cdot \|c\| \cdot \sigma$$

Threshold Raccoon

Raccoon.Keygen() \rightarrow sk, vk

- $\text{vk} = [\mathbf{A} \ \mathbf{I}] \cdot \text{sk}$, for sk short

Raccoon.Sign(sk, msg) \rightarrow sig

- Sample a short \mathbf{r}
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = c \cdot \text{sk} + \mathbf{r}$
- Output sig = (c, \mathbf{z})

Raccoon.Verify(vk, msg, sig = (c, \mathbf{z}))

- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{z} - c \cdot \text{vk}$
- Assert $c = H(\mathbf{w}, \text{msg})$
- Assert \mathbf{z} short

Shamir sharing on secret $\text{sk} \in \mathcal{R}_q^\ell$

Sample polynomial $f \in \mathcal{R}_q^\ell[X]$ s.t.

- $f(0) = \text{sk}$ and $\deg f \leq T - 1$
- Partial signing keys $\text{sk}_i := \llbracket \text{sk} \rrbracket_i = f(i)$

Properties:

- with $< T$ shares, sk is perfectly hidden
- with a set S of $\geq T$ shares, reconstruct sk via Lagrange interpolation

$$\text{sk} = \sum_{i \in S} L_{S,i} \cdot \llbracket \text{sk} \rrbracket_i$$

Threshold Raccoon

Raccoon.Keygen() \rightarrow sk, vk

- $\text{vk} = [\mathbf{A} \ \mathbf{I}] \cdot \text{sk}$, for sk short

Raccoon.Sign(sk, msg) \rightarrow sig

- Sample a short \mathbf{r}
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = c \cdot \text{sk} + \mathbf{r}$
- Output sig = (c, \mathbf{z})

Raccoon.Verify(vk, msg, sig = (c, \mathbf{z}))

- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{z} - c \cdot \text{vk}$
- Assert $c = H(\mathbf{w}, \text{msg})$
- Assert \mathbf{z} short

First (insecure) attempt

ThRaccoon.Sign(sk, msg) \rightarrow sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket \text{sk} \rrbracket_i + \mathbf{r}_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold Raccoon

- ◆ Prevent ROS attack with commit-reveal of \mathbf{w}_i

First (insecure) attempt

ThRaccoon . Sign(sk, msg) \rightarrow sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket sk \rrbracket_i + \mathbf{r}_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold Raccoon

- ◆ Prevent ROS attack with commit-reveal of \mathbf{w}_i
- ◆ But, \mathbf{r}_i is small vs $L_{S,i} \cdot c \cdot \llbracket sk \rrbracket_i$ is large
→ Leaks $\llbracket sk \rrbracket_i$

First (insecure) attempt

ThRaccoon . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket sk \rrbracket_i + \mathbf{r}_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold Raccoon

- ◆ Prevent ROS attack with commit-reveal of \mathbf{w}_i
- ◆ But, \mathbf{r}_i is small vs $L_{S,i} \cdot c \cdot \llbracket sk \rrbracket_i$ is large
→ Leaks $\llbracket sk \rrbracket_i$
- ◆ Solution: add a zero-share Δ_i :
 - Derived with a PRF, using pre-shared pairwise keys
 - Any set of $< T$ values Δ_i is uniformly random
 - $\sum_{i \in S} \Delta_i = 0$

ThRaccoon . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket sk \rrbracket_i + \mathbf{r}_i + \Delta_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold Raccoon, a practical threshold signature

Speed	Rounds	vk	sig	Total communication
Fast	3	4 kB	13 kB	40 kB

... but does not provide a DKG, or robustness / identifiable aborts.

3. Another direction for ThRaccoon

Flood and Submerge: Distributed Key
Generation and Robust Threshold Signature
from Lattices

Thomas Espitau¹ , Guilhem Niot^{1,2} , and Thomas Prest¹ 

How to Shortly Share a Short Vector
DKG with Short Shares and Application to Lattice-Based
Threshold Signatures with Identifiable Aborts

Rafael del Pino¹ , Thomas Espitau¹ , Guilhem Niot^{1,2} , and Thomas
Prest¹ 

Challenge of making ThRaccoon robust

ThRaccoon . Sign(sk, msg) \rightarrow sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Compute zero-share Δ_i
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket \text{sk} \rrbracket_i + \mathbf{r}_i + \Delta_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Why is it challenging to add robustness to ThRaccoon?

- Incompatibility of the sharings of sk and \mathbf{r}_i , that prevents a simple verification of computations
- Additional non-linearity introduced by Δ_i

Challenge of making ThRaccoon robust

ThRaccoon . Sign(sk, msg) \rightarrow sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Compute zero-share Δ_i
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket \text{sk} \rrbracket_i + \mathbf{r}_i + \Delta_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Let's take a step back!

The key challenge in ThRaccoon is to hide a secret $L_{S,i} \cdot \llbracket \text{sk} \rrbracket_i$ with the randomness \mathbf{r}_i .

Direction 1 (Threshold Raccoon):

- The shares of sk are **uniform**
- The randomness shares \mathbf{r}_i are **short**

A **uniform** zero-share Δ_i is added to partial signatures to hide $L_{S,i} \cdot \llbracket \text{sk} \rrbracket_i$.

Direction 2: Can we make both $L_{S,i} \cdot \llbracket \text{sk} \rrbracket_i$ and \mathbf{r}_i uniform?

- Use Shamir-sharing for both sk and \mathbf{r} \rightarrow Flood and submerge [ENP24]

Direction 3: Can we make both $L_{S,i} \cdot \llbracket \text{sk} \rrbracket_i$ and \mathbf{r}_i short?

- Use a **short secret-sharing** for both sk and \mathbf{r}

Flood and submerge

ThRaccoon . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Compute zero-share Δ_i
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot \llbracket \text{sk} \rrbracket_i + \mathbf{r}_i + \Delta_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

[ENP24] . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i , and Shamir sharing $\llbracket \mathbf{r}_i \rrbracket$
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$
- Privately send $\llbracket \mathbf{r}_i \rrbracket_j$ to user j

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- $\llbracket \mathbf{r} \rrbracket_i = \sum_j \llbracket \mathbf{r}_j \rrbracket_i$
- Broadcast $\llbracket \mathbf{z} \rrbracket_i = c \cdot \llbracket \text{sk} \rrbracket_i + \llbracket \mathbf{r} \rrbracket_i$

Combine: the final signature is

$$(c, \sum_{i \in S} L_{s,i} \cdot \llbracket \mathbf{z} \rrbracket_i)$$

Flood and submerge

- **Security:** $[[\mathbf{r}]]_i$ is **uniform** and hides $[[\text{sk}]]_i$
- This protocol can be augmented to achieve robustness:
 - ◆ Add a complaints round
 - ◆ Use of a V3S (Verifiable Short Secret Sharing) to prove shortness of \mathbf{r} , and correct Shamir-sharing
 - ◆ Can also be used to implement DKG

[ENP24] . $\text{Sign}(\text{sk}, \text{msg}) \rightarrow \text{sig}$

Round 1:

- Sample a short \mathbf{r}_i , and Shamir sharing $[[\mathbf{r}_i]]$
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$
- Privately send $[[\mathbf{r}_i]]_j$ to user j

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- $[[\mathbf{r}]]_i = \sum_j [[\mathbf{r}_j]]_i$
- Broadcast $[[\mathbf{z}]]_i = c \cdot [[\text{sk}]]_i + [[\mathbf{r}]]_i$

Combine: the final signature is

$$(c, \sum_{i \in S} L_{s,i} \cdot [[\mathbf{z}]]_i)$$

Flood and submerge

- **Security:** $[[\mathbf{r}]]_i$ is **uniform** and hides $[[\text{sk}]]_i$
- This protocol can be augmented to achieve robustness:
 - ◆ Add a complaints round
 - ◆ Use of a V3S (Verifiable Short Secret Sharing) to prove shortness of \mathbf{r} , and correct Shamir-sharing
 - ◆ Can also be used to implement DKG

Speed: Fast

Rounds: 4

Communication: $T \cdot 56$ kB

DKG + Robustness 😊

[ENP24] . $\text{Sign}(\text{sk}, \text{msg}) \rightarrow \text{sig}$

Round 1:

- Sample a short \mathbf{r}_i , and Shamir sharing $[[\mathbf{r}_i]]$
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$
- Privately send $[[\mathbf{r}_i]]_j$ to user j

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

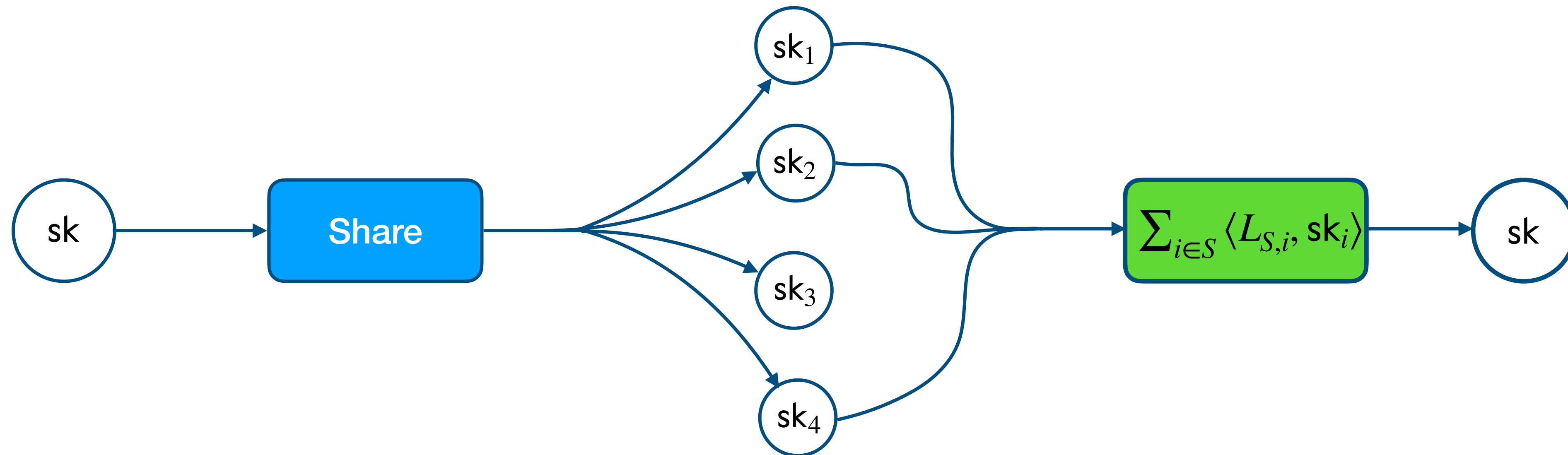
- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- $[[\mathbf{r}]]_i = \sum_j [[\mathbf{r}_j]]_i$
- Broadcast $[[\mathbf{z}]]_i = c \cdot [[\text{sk}]]_i + [[\mathbf{r}]]_i$

Combine: the final signature is

$$(c, \sum_{i \in S} L_{s,i} \cdot [[\mathbf{z}]]_i)$$

With Short Secret Sharing

- Another approach relies on sampling a sharing of sk such that we have:
 - ◆ Individual pool of short shares $sk_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$
 - ◆ T shares: can recover sk + reconstruction vector $L_{S,i}$ with small coefficients
 - ◆ $\leq T - 1$ shares: can't recover sk



With Short Secret Sharing

ShortSS . Sign(sk, msg) \rightarrow sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = c \cdot \langle L_{S,i}, \text{sk}_i \rangle + \mathbf{r}_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

For simplicity, we consider one share per party.

Security.

- $c \cdot \langle L_{S,i}, \text{sk}_i \rangle$ is short $\rightarrow \mathbf{r}_i$ hides it.
 - Prove security with Hint-MLWE

With Short Secret Sharing

ShortSS . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = c \cdot \langle L_{S,i}, \text{sk}_i \rangle + \mathbf{r}_i$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

For simplicity, we consider one share per party.

Security.

- $c \cdot \langle L_{S,i}, \text{sk}_i \rangle$ is short → \mathbf{r}_i hides it.
 - Prove security with Hint-MLWE

Identifiable aborts.

- Each $\text{vk}_i^{(j)} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{s}_i^{(j)}$ is a valid public key ($\mathbf{s}_i^{(j)}$ is short), for $\text{sk}_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$
 - Each (c, \mathbf{z}_i) is a valid signature for $\langle L_{S,i}, (\text{vk}_i^{(j)})_j \rangle$
- Identifiable abort is as easy as verifying partial signatures!
- *Akin to abort identification in Sparkle (Threshold Schnorr): perform partial verifications.*

With Short Secret Sharing

Instantiating this scheme.

- In the T -out-of- N setting, the number of shares grows with $\binom{N}{T-1}$, this scheme thus only supports a small number of parties.

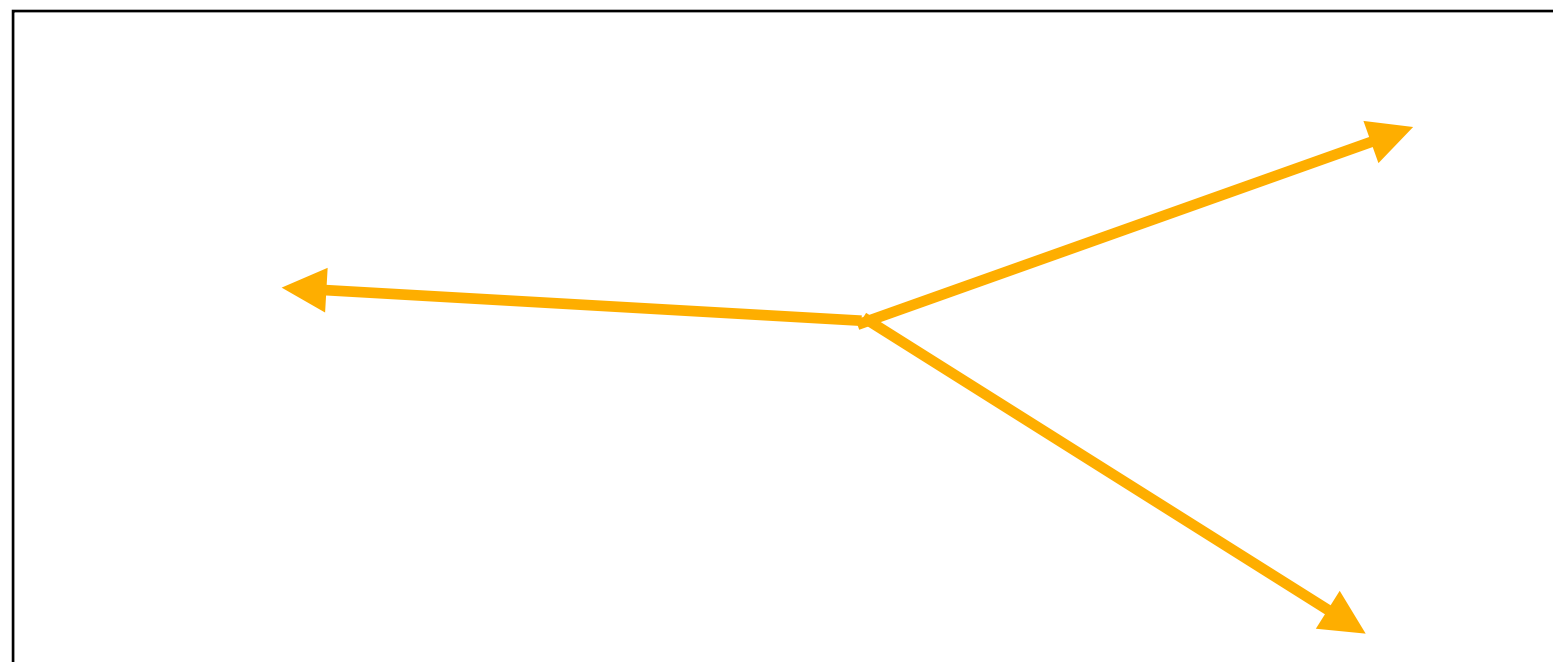
For $N \leq 16$,

Phase	# rounds	vk	sig	Total communication
Signing	3	4 kB	11 kB	25 kB
Abort Identification	0			

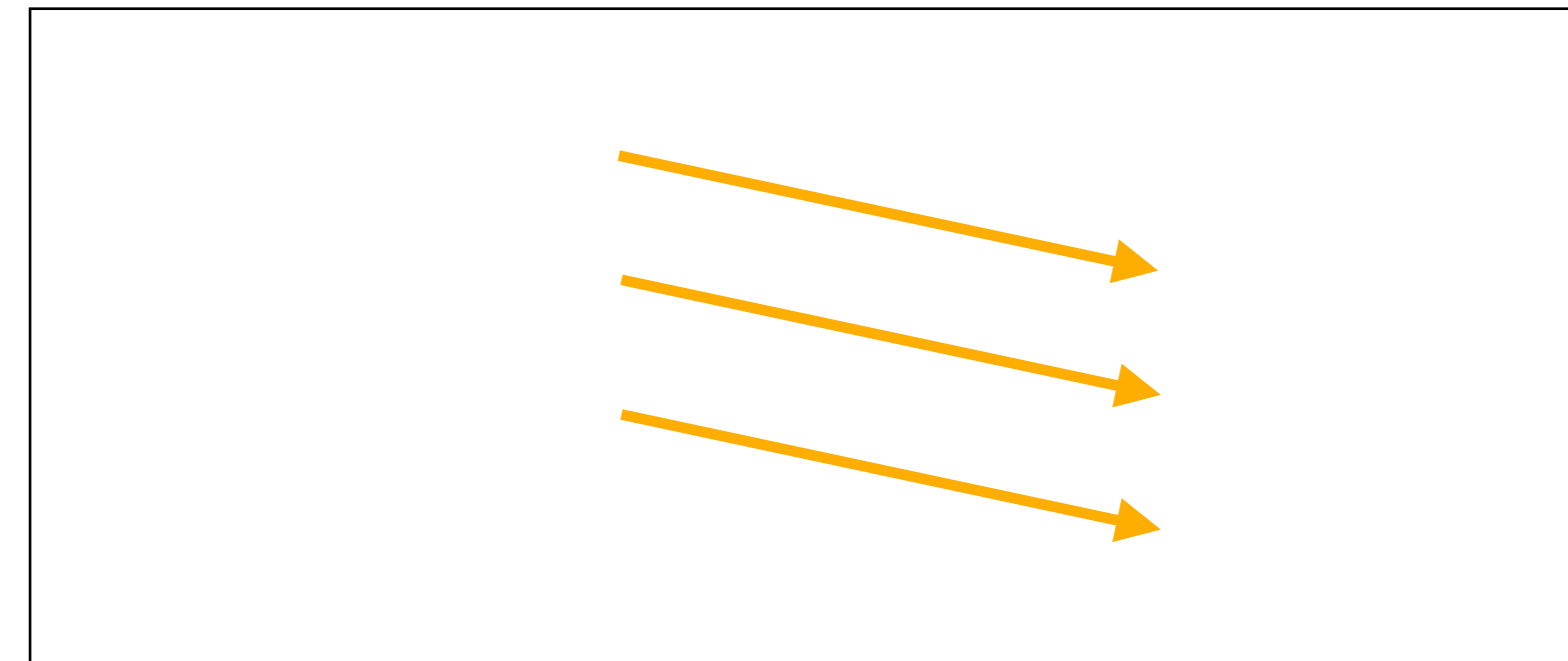
Bonus: tighter check bounds using Short SS

Looking in more detail, the correctness of the previous schemes relies on the shortness of $\mathbf{z} = \sum_i \mathbf{z}_i$.

What can we say about the norm of T Gaussians?



Average-case: $O(\sqrt{T})$



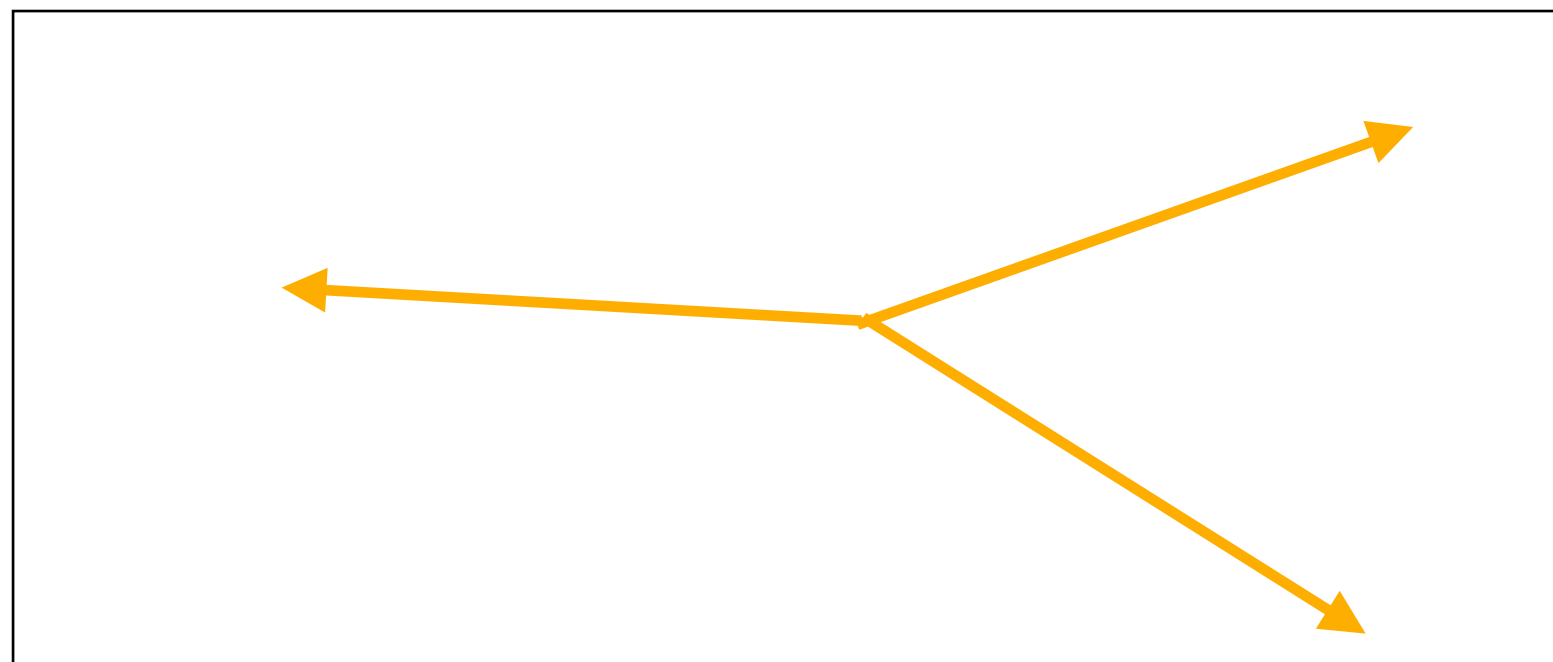
Worst-case: $O(T)$

- When users are honest: average-case.
- Colliding malicious users can force worst-case.

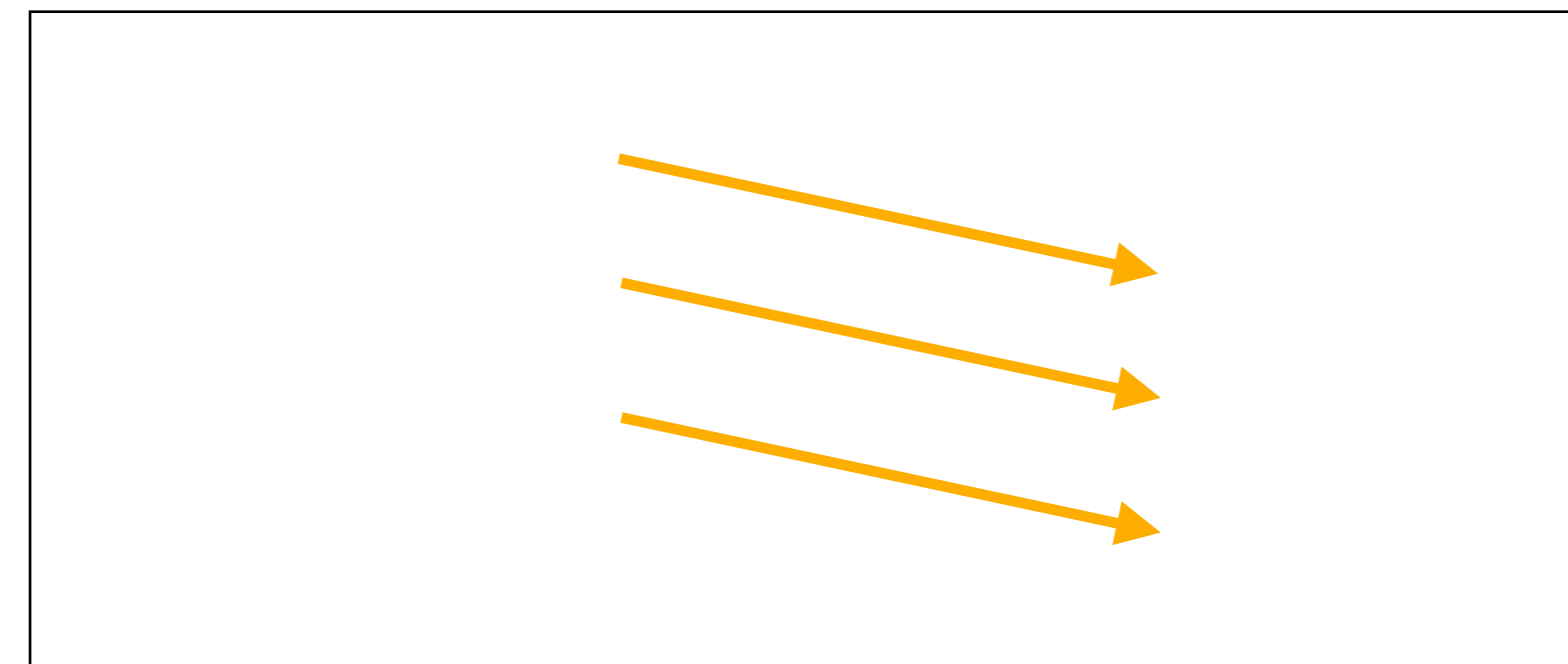
Bonus: tighter check bounds using Short SS

Looking in more detail, the correctness of the previous schemes relies on the shortness of $\mathbf{z} = \sum_i \mathbf{z}_i$.

What can we say about the norm of T Gaussians?



Average-case: $O(\sqrt{T})$



Worst-case: $O(T)$

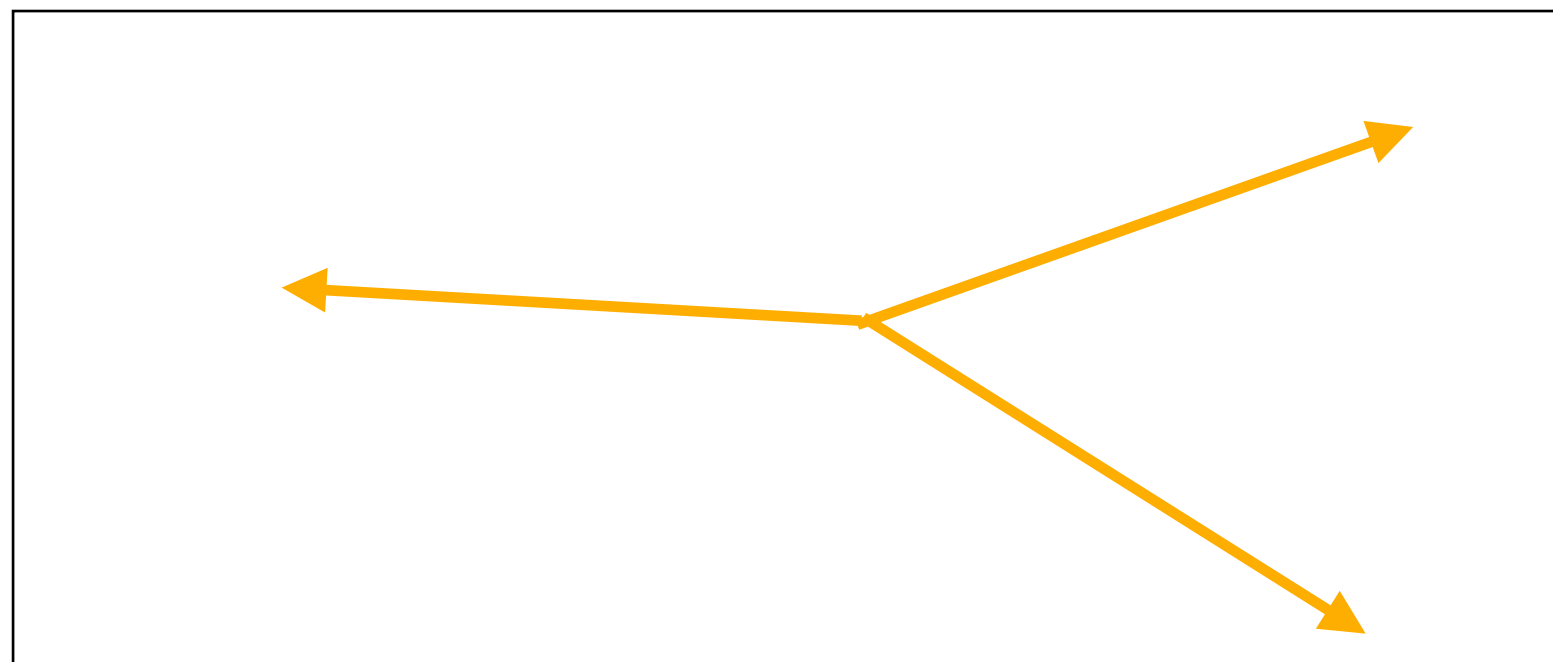
- When users are honest: average-case.
- Colliding malicious users can force worst-case.

In Flood and Submerge, \mathbf{z}_i is masked (uniform-looking sharings), hard to detect worst-case
→ bound in $O(T)$ that reduces security 😞

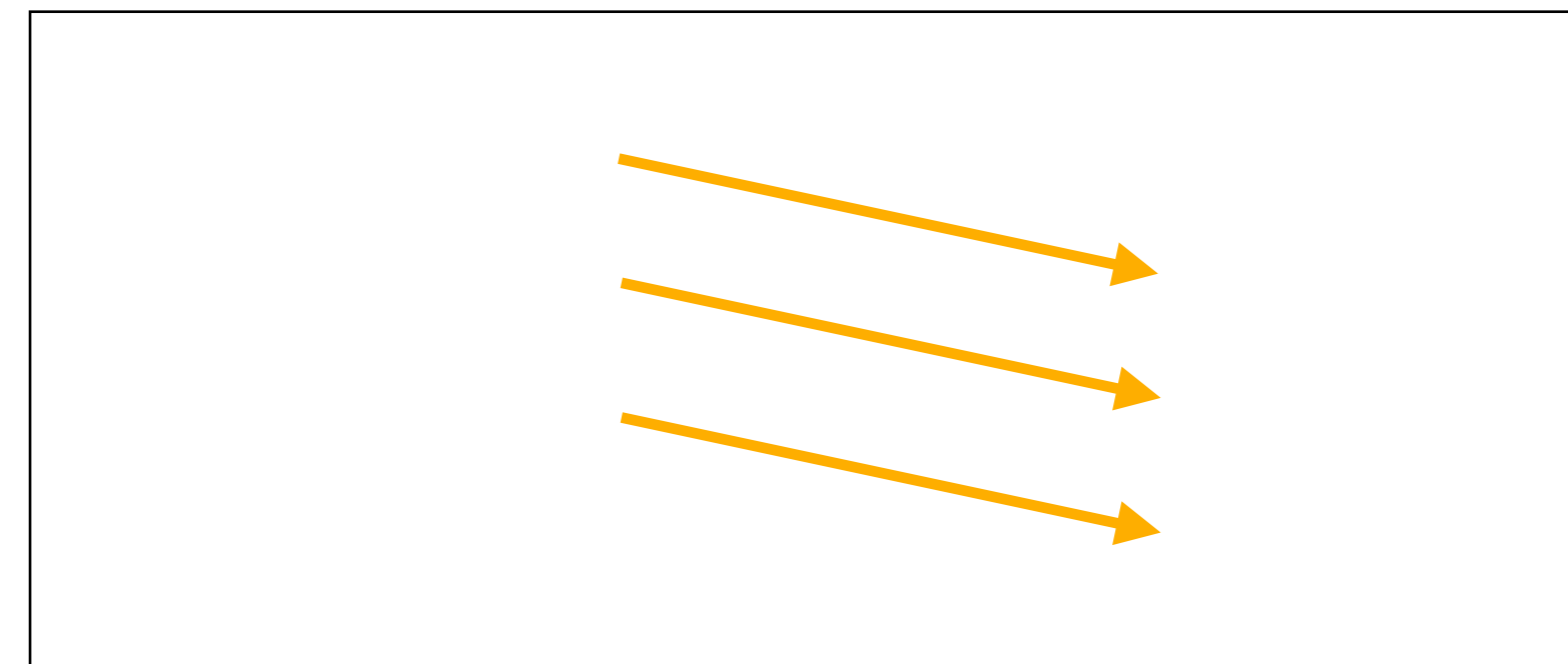
Bonus: tighter check bounds using Short SS

Looking in more detail, the correctness of the previous schemes relies on the shortness of $\mathbf{z} = \sum_i \mathbf{z}_i$.

What can we say about the norm of T Gaussians?



Average-case: $O(\sqrt{T})$



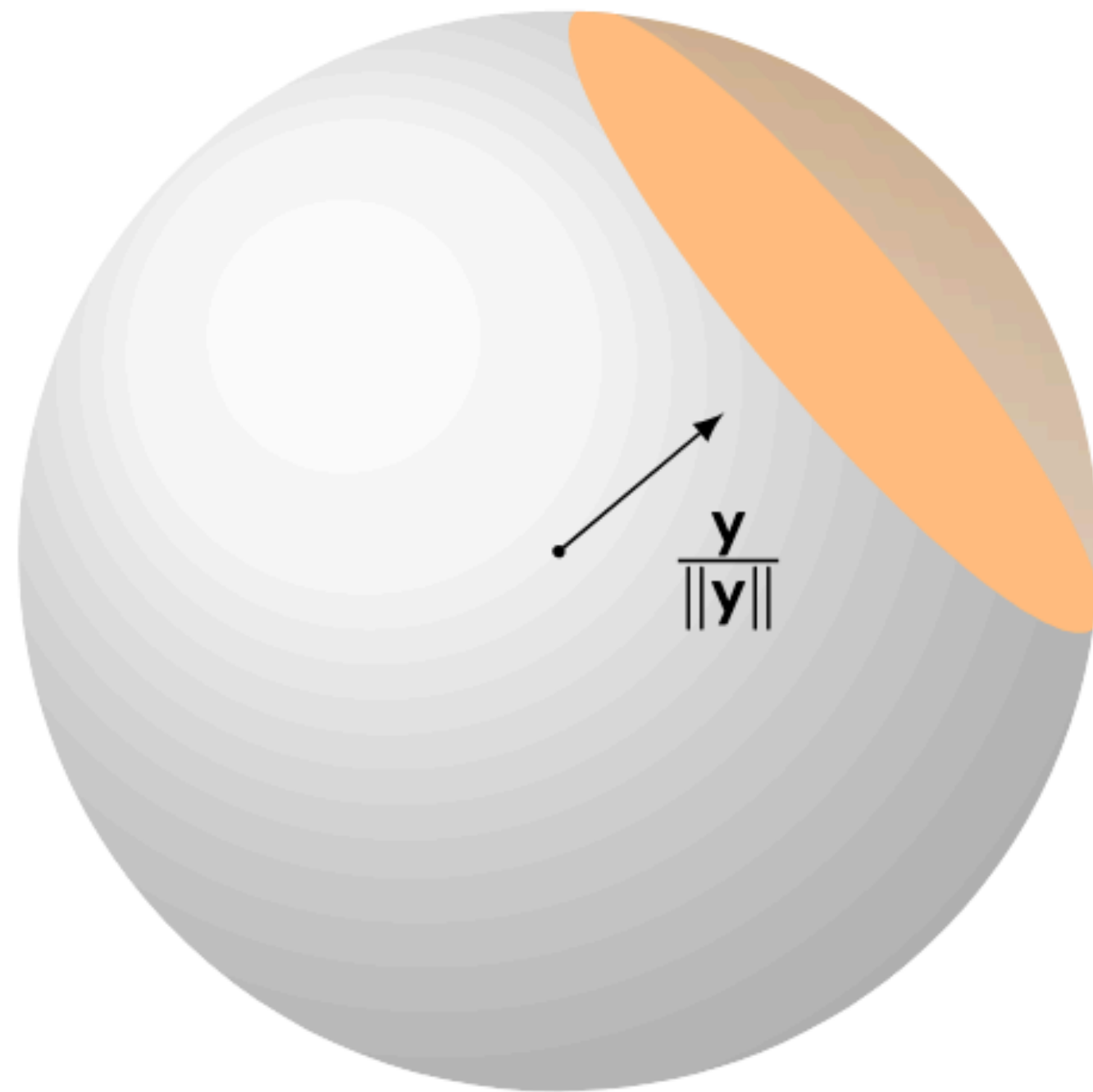
Worst-case: $O(T)$

- When users are honest: average-case.
- Colliding malicious users can force worst-case.

In Flood and Submerge, \mathbf{z}_i is masked (uniform-looking sharings), hard to detect worst-case
→ bound in $O(T)$ that reduces security 😞

With Short SS, \mathbf{z}_i is short and we can detect collusions and worst-case behaviour!

The Death Star Algorithm



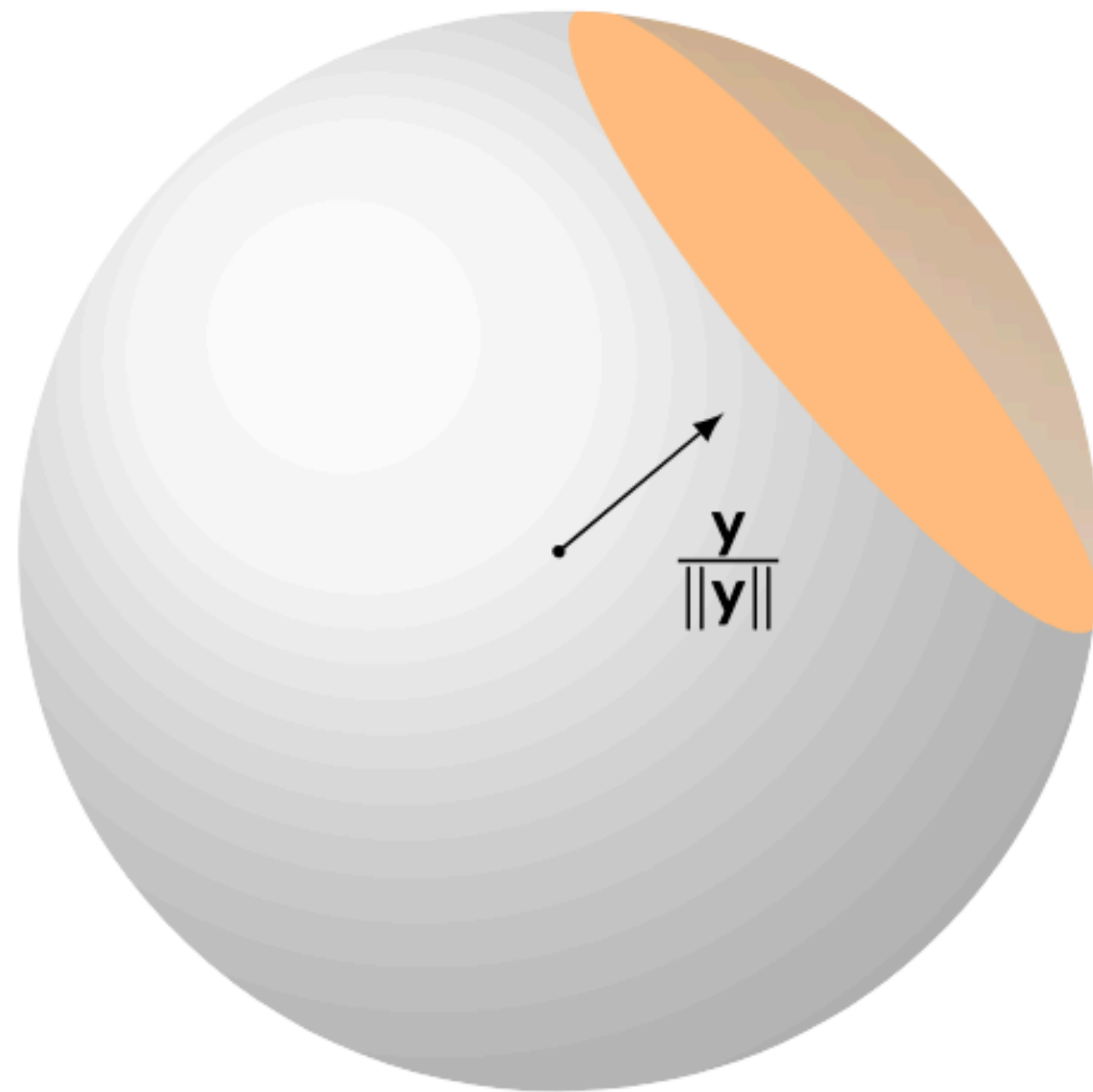
If $\mathbf{x} \leftarrow \mathcal{D}_\sigma$,

- $\|\mathbf{x}\|$ is concentrated around its expected value $\sqrt{n}\sigma$
- For any vector \mathbf{y} ,

$$\langle \mathbf{x}, \mathbf{y} \rangle < \sigma \sqrt{O(\lambda)} \cdot \|\mathbf{y}\|$$

except with probability $2^{-\lambda}$.

The Death Star Algorithm



The Death Star Algorithm

For each signer i ,

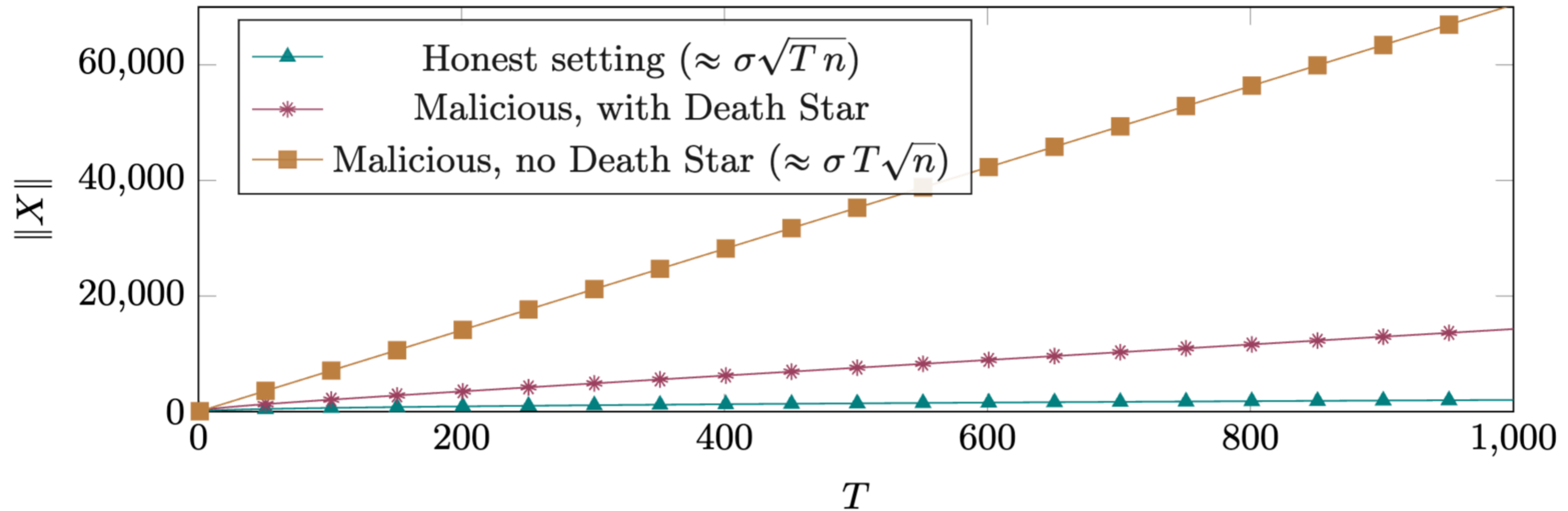
- If $\|\mathbf{x}_i\| \geq (1 + o(1))\sqrt{n}\sigma$, reject i
- If $\langle \mathbf{x}_i, \mathbf{y}_i \rangle \geq \sigma\sqrt{O(\lambda)}\|\mathbf{y}_i\|$, where $\mathbf{y}_i = \sum_{j \neq i} \mathbf{x}_j$, reject i

Detect exactly cheating parties except with proba $2^{-\lambda}$

When no signer is rejected, the sum $\mathbf{x} = \sum_i \mathbf{x}_i$ verifies

$$\begin{aligned} \|\mathbf{x}\| &\leq \sigma \cdot T \cdot \sqrt{2 \log 2 \cdot \lambda} \\ &\quad + \sigma \cdot \sqrt{T \cdot n} \cdot (1 + \varepsilon) \end{aligned}$$



The Death Star Algorithm



Norm of $\mathbf{x} = \sum_i \mathbf{x}_i$ for $\sigma = 1$, $n = 4096$, 128 bits of security, and $T \leq 1000$

4. Compact Dilithium-like Threshold Signatures

Finally! A Compact Lattice-Based Threshold
Signature

Rafael del Pino¹  and Guilhem Niot^{1,2} 

Fiat-Shamir with Aborts signature

$\text{Rej}(\mathbf{v}, \chi_r, \chi_z, M) \rightarrow \mathbf{z} \mid \perp$

- $\mathbf{r} \leftarrow \chi_r$
- $\mathbf{z} = \mathbf{v} + \mathbf{r}$
- $b \leftarrow \mathcal{B} \left(\max \left(\frac{\chi_z(\mathbf{z})}{M\chi_r(\mathbf{r})}, 1 \right) \right)$
- If $b = 0$ then $\mathbf{z} = \perp$
- Return \mathbf{z}

$\text{Ideal}(\chi_z, M) \rightarrow \mathbf{z} \mid \perp$

- $\mathbf{z} \leftarrow \chi_z$
- $b \leftarrow \mathcal{B} \left(\frac{1}{M} \right)$
- If $b = 0$ then $\mathbf{z} = \perp$
- Return \mathbf{z}

For proper parameters, $\text{Rej}(\mathbf{v}, \chi_r, \chi_z, M) \sim \text{Ideal}(\chi_z, M)$.

→ distribution of \mathbf{z} is independent of the secret value \mathbf{v}

Fiat-Shamir with Aborts signature

$\text{Rej}(\mathbf{v}, \chi_r, \chi_z, M; \mathbf{r}) \rightarrow \mathbf{z} \mid \perp$

- $\mathbf{z} = \mathbf{v} + \mathbf{r}$
- $b \leftarrow \mathcal{B} \left(\max \left(\frac{\chi_z(\mathbf{z})}{M\chi_r(\mathbf{r})}, 1 \right) \right)$
- If $b = 0$ then $\mathbf{z} = \perp$
- Return \mathbf{z}

FSwA . Sign(sk, msg) \rightarrow sig

- $\mathbf{r} \leftarrow \chi_r$
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = \text{Rej}(c \cdot \text{sk}, \chi_r, \chi_z, M; \mathbf{r})$
- If $\mathbf{z} = \perp$ then **restart**
- Return (c, \mathbf{z})

FSwA . Verify(vk, msg, sig = (c, \mathbf{z}))

- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{z} - c \cdot \text{vk}$
- Assert $c = H(\mathbf{w}, \text{msg})$
- Assert \mathbf{z} short

In the ROM, the distribution of signatures of the above scheme is independent of the secret sk.

\rightarrow allows to prove unforgeability

Threshold FSwa signature?

FSwa . Sign(sk, msg) \rightarrow sig

- $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = \text{Rej}(c \cdot \text{sk}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r})$
- If $\mathbf{z} = \perp$ then **restart**
- Return (c, \mathbf{z})

o How to support T -out-of- N ?

TH-FSwA . Sign(sk, msg) \rightarrow sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = \text{Rej}(c \cdot \text{sk}_i, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_i)$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Intuition N -out-of- N setting: take N short secrets sk_i

Threshold FSwa signature?

FSwa . Sign(sk, msg) → sig

- $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = \text{Rej}(c \cdot \text{sk}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r})$
- If $\mathbf{z} = \perp$ then **restart**
- Return (c, \mathbf{z})

o How to support T -out-of- N ?

→ Use short secret sharing

TH-FSwA . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = \text{Rej}(c \cdot \langle L_{S,i}, \text{sk}_i \rangle, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_i)$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold FSWA signature?

FSWA . Sign(sk, msg) → sig

- $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = \text{Rej}(c \cdot \text{sk}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r})$
- If $\mathbf{z} = \perp$ then **restart**
- Return (c, \mathbf{z})

- How to support T -out-of- N ?
 - Use short secret sharing
- \mathbf{w}_i is leaked even in case of rejection
 - ◆ Need proof strategy to show independence of secret
 - ◆ [DOTT22] hides rejected \mathbf{w}_i with a trapdoor commitment scheme
 - ◆ [BTT22] simulates rejected \mathbf{w}_i but with regularity lemma (degraded parameters)

TH-FSWA . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = \text{Rej}(c \cdot \langle L_{S,i}, \text{sk}_i \rangle, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_i)$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold FSWA signature?

FSWA . Sign(sk, msg) → sig

- $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$
- $\mathbf{w} = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z} = \text{Rej}(c \cdot \text{sk}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r})$
- If $\mathbf{z} = \perp$ then **restart**
- Return (c, \mathbf{z})

- How to support T -out-of- N ?
 - Use short secret sharing
- \mathbf{w}_i is leaked even in case of rejection
 - ◆ Need proof strategy to show independence of secret
 - ◆ [DOTT22] hides rejected \mathbf{w}_i with a trapdoor commitment scheme
 - ◆ [BTT22] simulates rejected \mathbf{w}_i but with regularity lemma (degraded parameters)

→ Tighter simulation lemma

TH-FSWA . Sign(sk, msg) → sig

Round 1:

- Sample a short \mathbf{r}_i
- $\mathbf{w}_i = [\mathbf{A} \ \mathbf{I}] \cdot \mathbf{r}_i$
- Broadcast $\text{cmt}_i = H_{\text{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast \mathbf{w}_i

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- Broadcast $\mathbf{z}_i = \text{Rej}(c \cdot \langle L_{S,i}, \text{sk}_i \rangle, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_i)$

Combine: the final signature is

$$(c, \sum_{i \in S} \mathbf{z}_i)$$

Threshold FSWA signature?

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

- $\mathbf{w} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}$, with $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$ is indistinguishable from uniform
- $[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}$, with $\mathbf{z} \leftarrow \chi_{\mathbf{z}}$ is indistinguishable from uniform

Threshold FS_wA signature

For $N \leq 8$,

Distributions	Speed	Rounds	vk	sig	Total communication
Gaussians	Fast	3	2.6 kB	2.6 kB	5.6 kB
Uniforms			2.9 kB	6.3 kB	13.5 kB

Comparable to Dilithium size: 2.4kB at NIST level II!

4. How to concretely sample short sharings

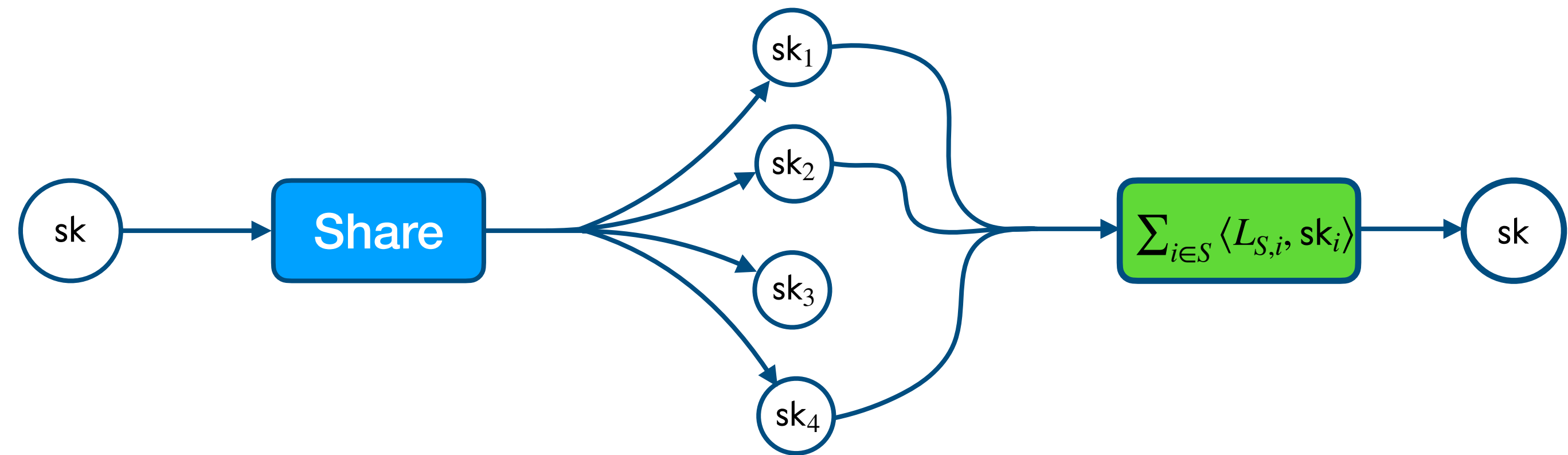
How to Shortly Share a Short Vector

DKG with Short Shares and Application to Lattice-Based
Threshold Signatures with Identifiable Aborts

Rafael del Pino¹ , Thomas Espitau¹ , Guilhem Niot^{1,2} , and Thomas
Prest¹ 

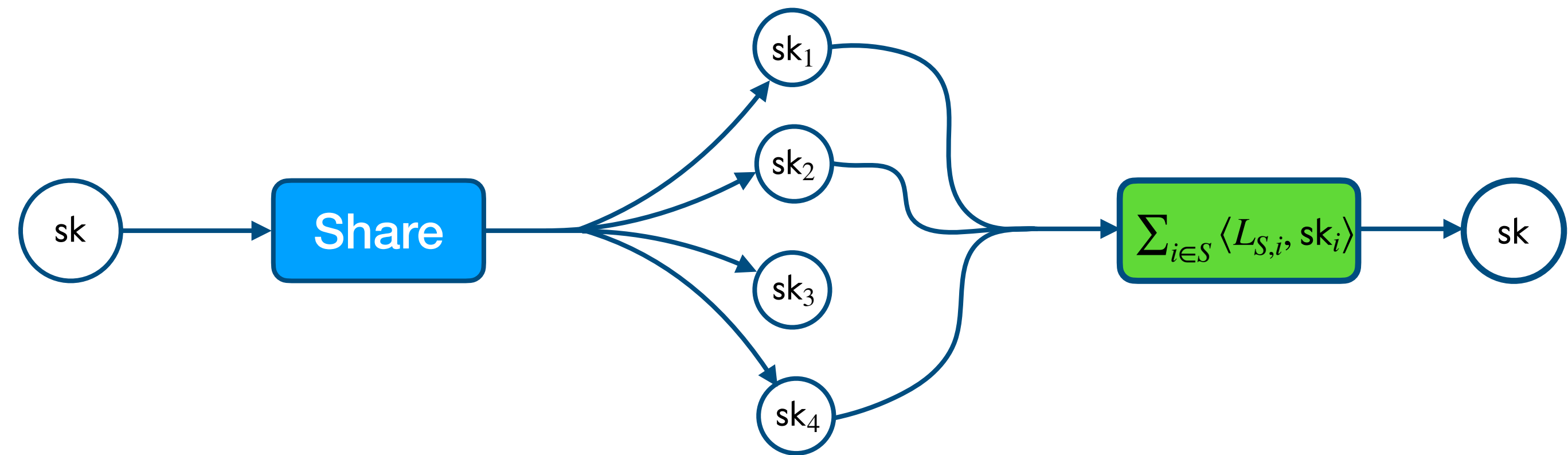
Short Secret Sharing

- Individual pool of short shares $sk_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$
- T shares: can recover sk + reconstruction vector $L_{S,i}$ with small coefficients
- $\leq T - 1$ shares: can't recover sk



Short Secret Sharing

- Individual pool of short shares $sk_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$
- T shares: can recover sk + reconstruction vector $L_{S,i}$ with small coefficients
- $\leq T - 1$ shares: can't recover sk



Observation: hard to not leak the secret with these constraints...

But, in a lattice-based scheme, it is fine to:

- Leak an offset of the secret: $sk = sk_{\text{safe}} + sk_{\text{leak}}$
- Leak hints on the secrets $h = c \cdot sk + y$, for large enough y
→ We just need $[\mathbf{A} \quad \mathbf{I}] \cdot sk$ to look uniform

Short Secret Sharing

Weaken zero-knowledge → Functional simulatability

We are interested in protocols generating sharings such that:

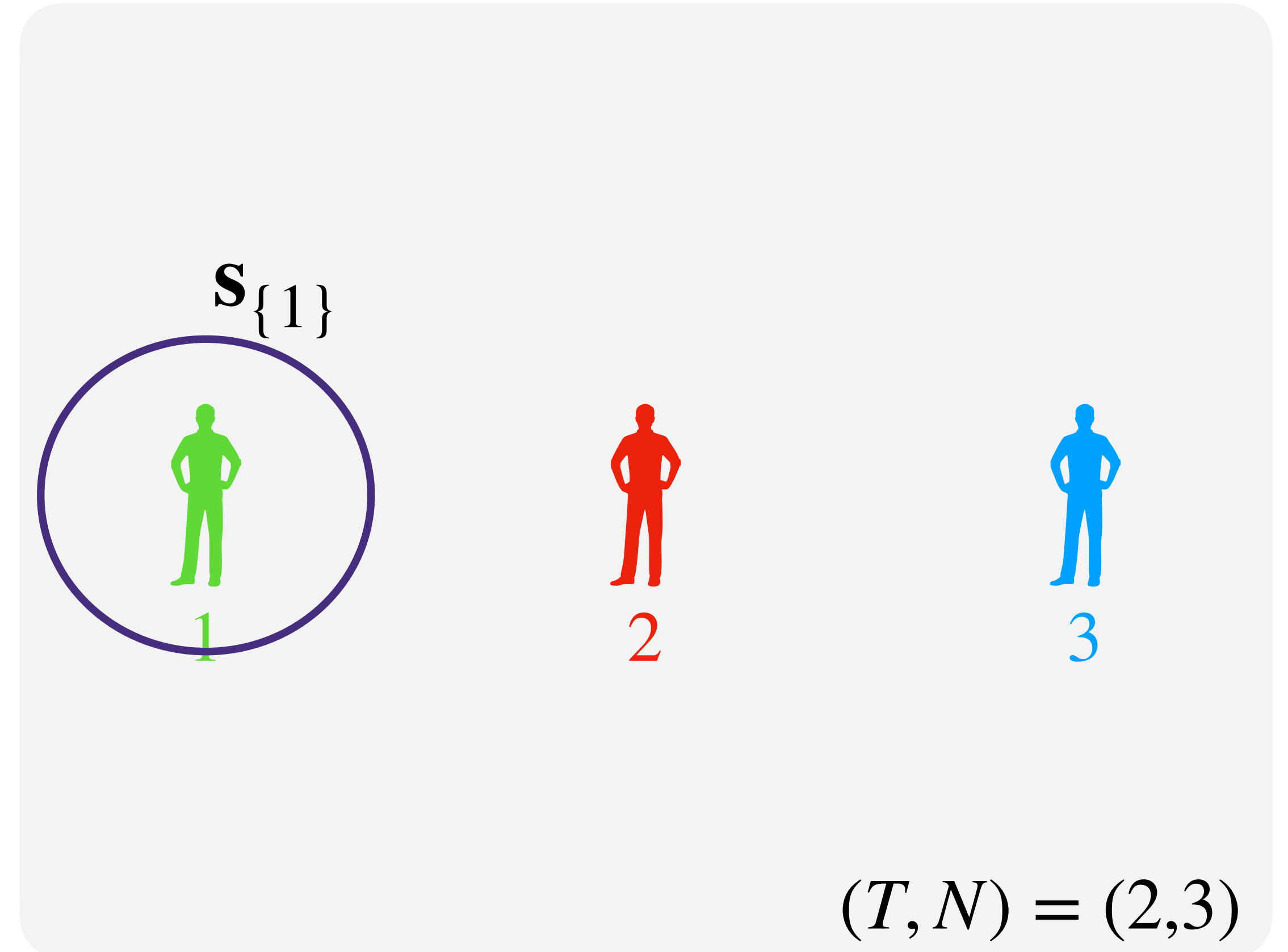
- When $< T$ parties are corrupted,
 - ◆ Their views can be simulated replacing $[\mathbf{A} \quad \mathbf{I}] \cdot \text{sk}$ with a uniform sample
 - ◆ It is possible to simulate a function on honest shares (i.e. obtain a hint on honest shares $h = c \cdot \langle L_{S,i}, \text{sk}_i \rangle + y$)

Inspired by the fractional knowledge notion in [ENP24], introduced for VSS.

Solution 1: Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of $T - 1$ parties, sample a uniform share $s_{\mathcal{T}}$.

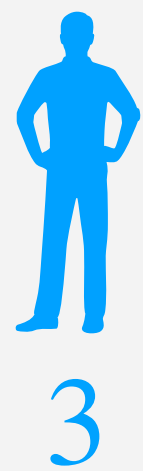
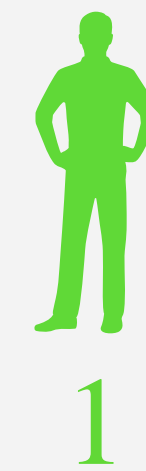


Solution 1: Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of $T - 1$ parties, sample a uniform share $\mathbf{s}_{\mathcal{T}}$.

$\mathbf{s}_{\{1\}}$



$\mathbf{s}_{\{2\}}$

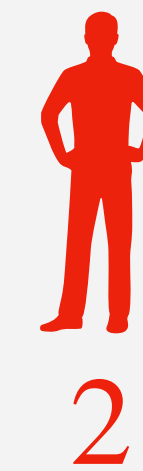
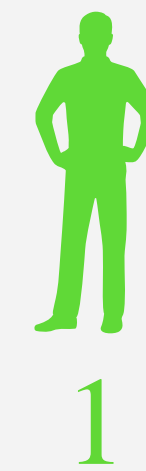
$(T, N) = (2, 3)$

Solution 1: Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of $T - 1$ parties, sample a uniform share $\mathbf{s}_{\mathcal{T}}$.

$\mathbf{s}_{\{1\}}$ $\mathbf{s}_{\{2\}}$

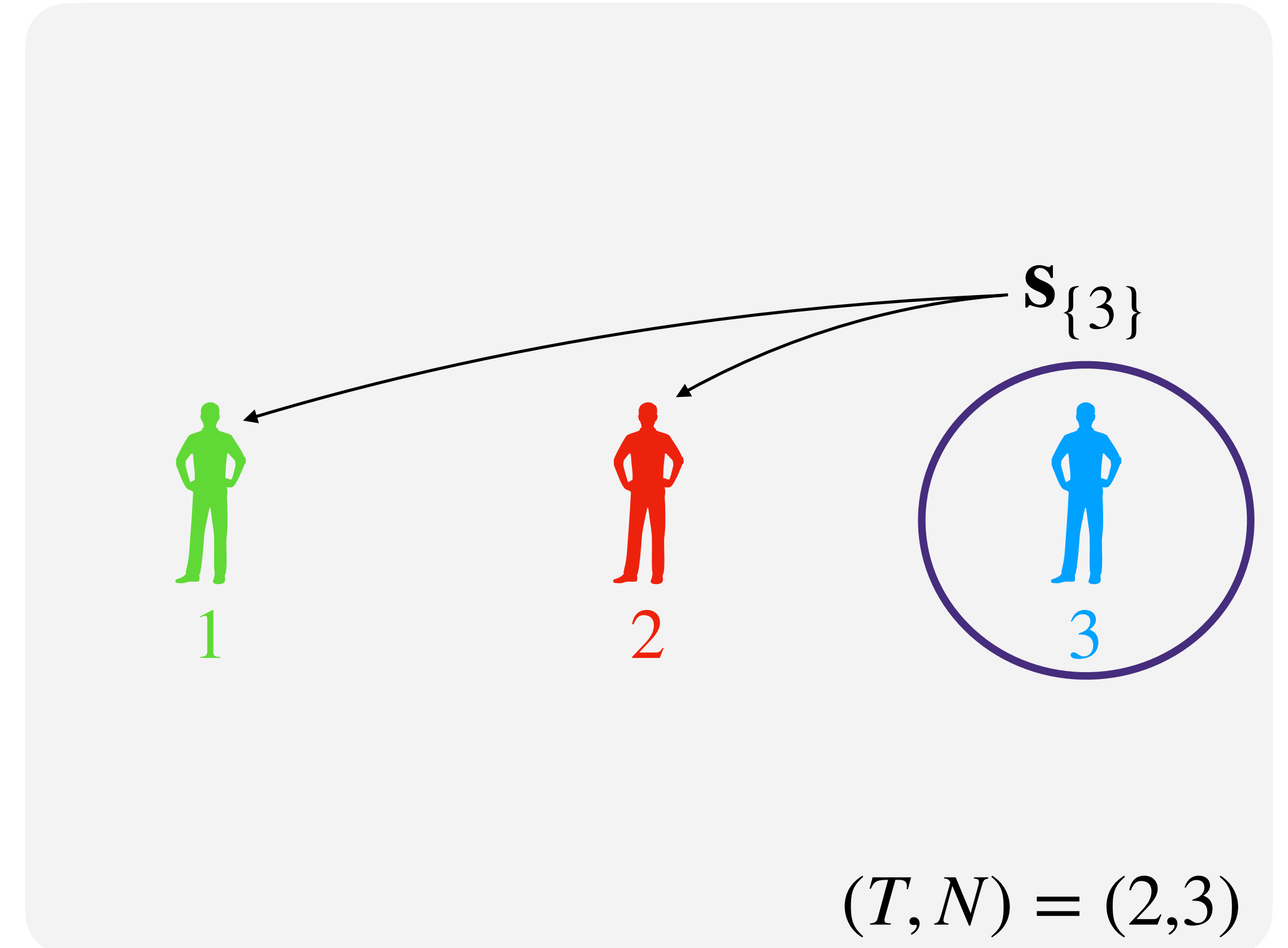


$(T, N) = (2, 3)$

Solution 1: Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

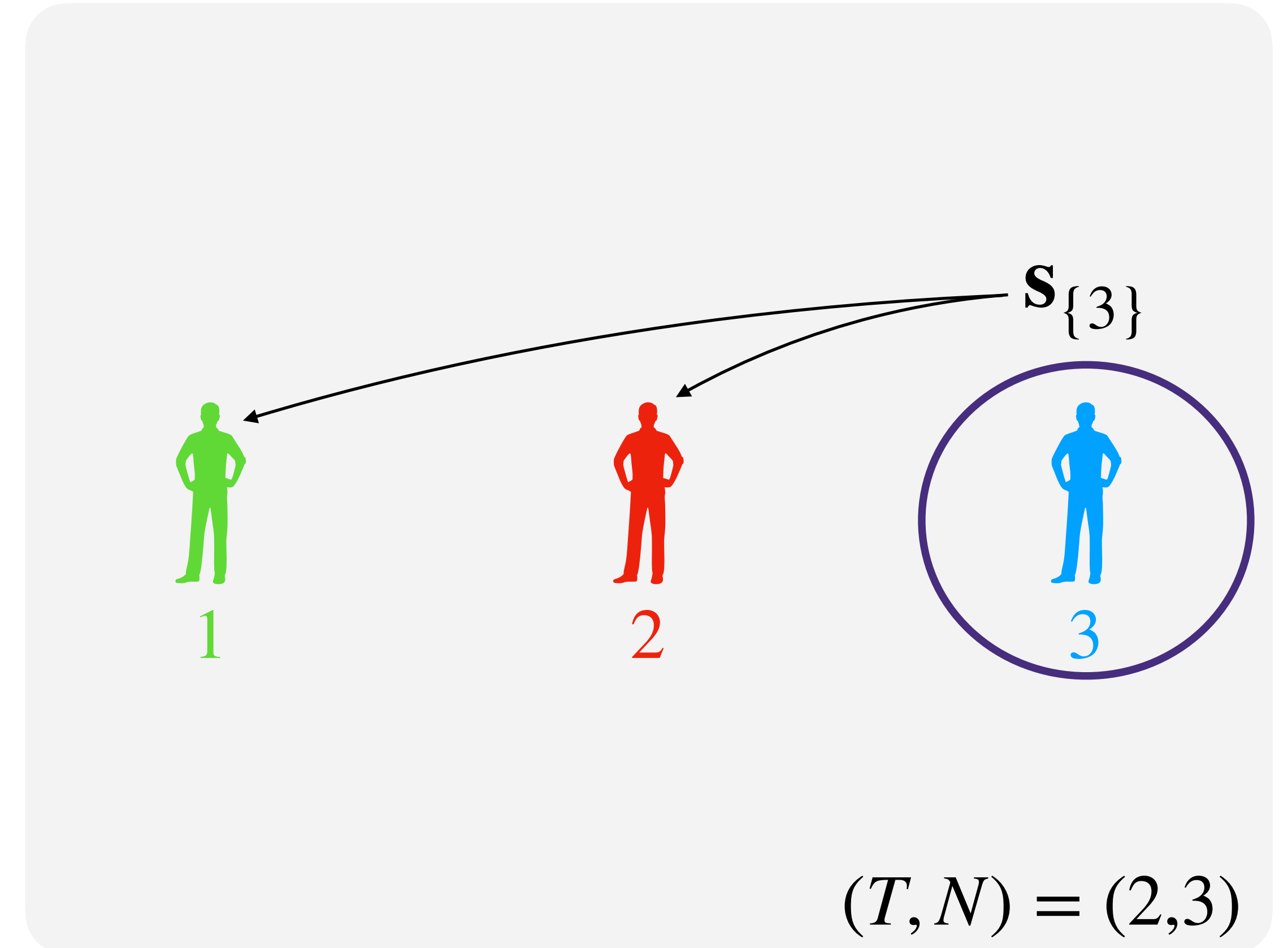
1. For any set \mathcal{T} of $T - 1$ parties, sample a uniform share $s_{\mathcal{T}}$.
2. Distribute $s_{\mathcal{T}}$ to the parties in $[N] \setminus \mathcal{T}$.



Solution 1: Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of $T - 1$ parties, sample a uniform share $\mathbf{s}_{\mathcal{T}}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to the parties in $[N] \setminus \mathcal{T}$.
3. Define $\mathbf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.



Solution 1: Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of $T - 1$ parties, sample a uniform share $\mathbf{s}_{\mathcal{T}}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to the parties in $[N] \setminus \mathcal{T}$.
3. Define $\text{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.

Properties:

- Reconstruction coefficients 0 or 1
- When $< T$ corrupted parties, at least one $\mathbf{s}_{\mathcal{T}}$ remains hidden.
→ guarantees that sk remains protected

Solution 1: **Short** Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of $T - 1$ parties, sample a **short** share $\mathbf{s}_{\mathcal{T}}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to the parties in $[N] \setminus \mathcal{T}$.
3. Define $\mathbf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.

Properties:

- Reconstruction coefficients 0 or 1
- When $< T$ corrupted parties, at least one $\mathbf{s}_{\mathcal{T}}$ remains hidden.
→ guarantees that $[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{sk}$ looks uniform (MLWE assumption)

Solution 1: **Short** Replicated Secret Sharing

Idea: sample a share for any possible set of corrupted parties.

1. For any set \mathcal{T} of corrupted parties, sample a **short** share $\mathbf{s}_{\mathcal{T}}$ with coefficients 0 or 1. **Caveat:** This scheme has a number of shares that is equal to $\binom{N}{T-1}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to all parties in $[N] \setminus \mathcal{T}$. For each set of corrupted parties, at least one $\mathbf{s}_{\mathcal{T}}$ remains hidden.
3. Define $\mathbf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.
→ guarantees that $[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{sk}$ looks uniform (MLWE assumption)

Solution 2: Coupon collector problem

Full collection

N cards



Solution 2: Coupon collector problem

Full collection

N cards



**Draw with
replacement**



1

Solution 2: Coupon collector problem

Full collection

N cards



**Draw with
replacement**



1



2

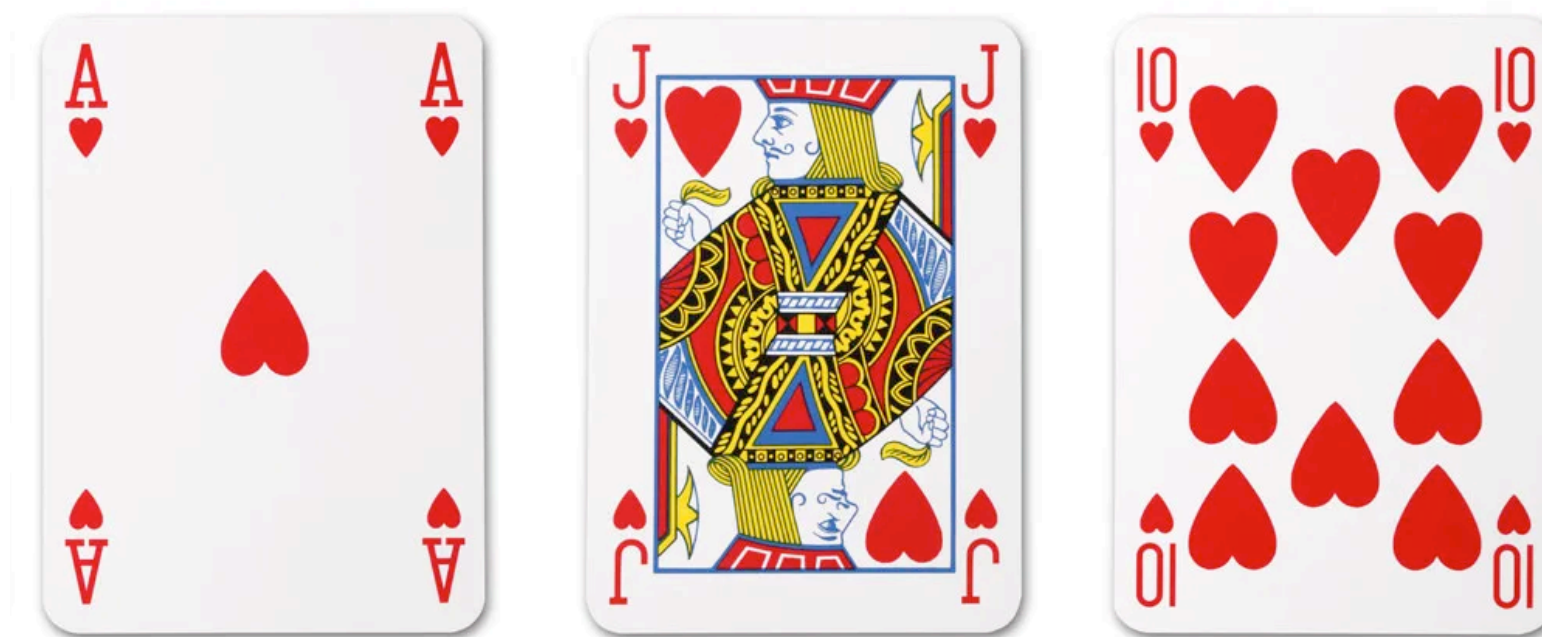
Solution 2: Coupon collector problem

Full collection

N cards



**Draw with
replacement**



1

2

3

Solution 2: Coupon collector problem

Full collection

N cards



Draw with replacement



1



2



3



4

...

How many draws to get the full collection?

$$\sim N \log N$$

Solution 2: Coupon collector problem

Full collection

N shares

$$sk = s_1 + s_2 + s_3 + s_4$$

Example:

- $s_1, \dots, s_{N-1} \leftarrow \mathcal{D}_\sigma^{N-1}$ and
 $s_N = sk - \sum_{j < N} s_j$

Solution 2: Coupon collector problem

Full collection

N shares

$$sk = s_1 + s_2 + s_3 + s_4$$

Idea: Randomly distribute one share per party.

Example:

- $s_1, \dots, s_{N-1} \leftarrow \mathcal{D}_\sigma^{N-1}$ and
 $s_N = sk - \sum_{j < N} s_j$

Desired properties:

- **Reconstruction threshold:** Minimum number of parties T needed to gather all the shares? (with overwhelming probability)
- **Security threshold:** Maximum number of parties T' such that at least one share is not known (with overwhelming probability)

Solution 2: Coupon collector problem

Full collection

N shares

$$sk = s_1 + s_2 + s_3 + s_4$$

Idea: Randomly distribute one share per party.

Example:

- $s_1, \dots, s_{N-1} \leftarrow \mathcal{D}_\sigma^{N-1}$ and
 $s_N = sk - \sum_{j < N} s_j$

Desired properties:

- **Reconstruction threshold:** Minimum number of parties T needed to gather all the shares? (with overwhelming probability)
- **Security threshold:** Maximum number of parties T' such that at least one share is not known (with overwhelming probability)

Bounds T, T' are exactly bounds of the coupon collector problem.

Both $T, T' \sim N \log N$, with gap $\underset{N \rightarrow \infty}{\approx} 1 + 128/\log N$

Solution 2: Coupon collector problem

Full collection

N shares

$$sk = s_1 + s_2 + s_3 + s_4$$

Better parameters by amplifying properties:

- **Reconstruction threshold:** If for given T , proba $1/2$ of reconstructing sk

$$\begin{aligned} sk &= s_1^1 + s_2^1 + s_3^1 + s_4^1 \\ &= \dots \\ &= s_1^m + s_2^m + s_3^m + s_4^m \end{aligned}$$

Share sk multiple times \rightarrow proba $1 - 1/2^m$

Solution 2: Coupon collector problem

Full collection $sk = s_1 + s_2 + s_3 + s_4$
N shares

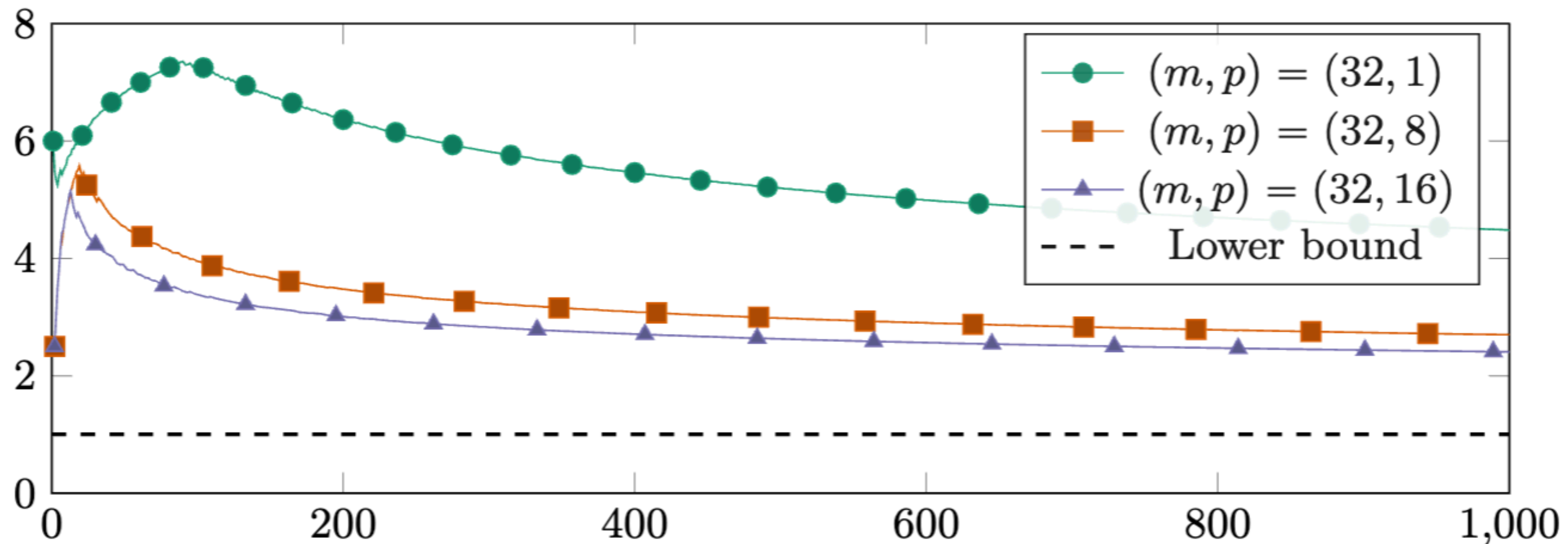
Better parameters by amplifying properties:

- **Reconstruction threshold:** Share sk multiple times \rightarrow proba $1 - 1/2^m$
- **Security threshold:** Share multiple secrets sk

$$sk = sk_1 + sk_2 + \dots + sk_p$$

If for given T' , proba $1/2$ of leaking sk_i , proba of leaking all the sk_i is $1/2^p$

Solution 2: Coupon collector problem



Ratio T/T' achieved by our sharing as a function of T' . The dotted line corresponds to an ideal asymptotic $T/T' = 1$.

Recall: m, p correspond respectively to amplification for reconstruction and security thresholds.

Solution 2: Coupon collector problem

Full collection

N shares

$$sk = s_1 + s_2 + s_3 + s_4$$

Example:

- $s_1, \dots, s_{N-1} \leftarrow \mathcal{D}_\sigma^{N-1}$ and
 $s_N = sk - \sum_{j < N} s_j$

Security:

We can prove that when $\leq T'$ parties are corrupted, leaked shares can be seen as hints on sk ($s_n = sk + y$).

→ Reduce security to Hint-MLWE

Use case: can be used for ThRaccoon with id abort without degrading parameters.

Short secret sharing

This presentation assumes a trusted dealer to sample the short secret sharing.

But, in our paper, we show that it is quite easy to design DKGs.

Conclusion

Conclusion

- ◆ **Introduced two short secret sharing methods**
 - Based on replicated secret sharing (exponential number of shares → for small number of parties)
 - Based on coupon collector problem: scales to larger thresholds, but has a gap between T and T'
- ◆ **Two applications**
 - Threshold Raccoon with identifiable aborts (using partial verification keys)
 - A compact threshold FS_WA signature scheme for $N \leq 8$

Questions?

