Identifiable Aborts in ThRaccoon

Let’s introduce short secret sharings!

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

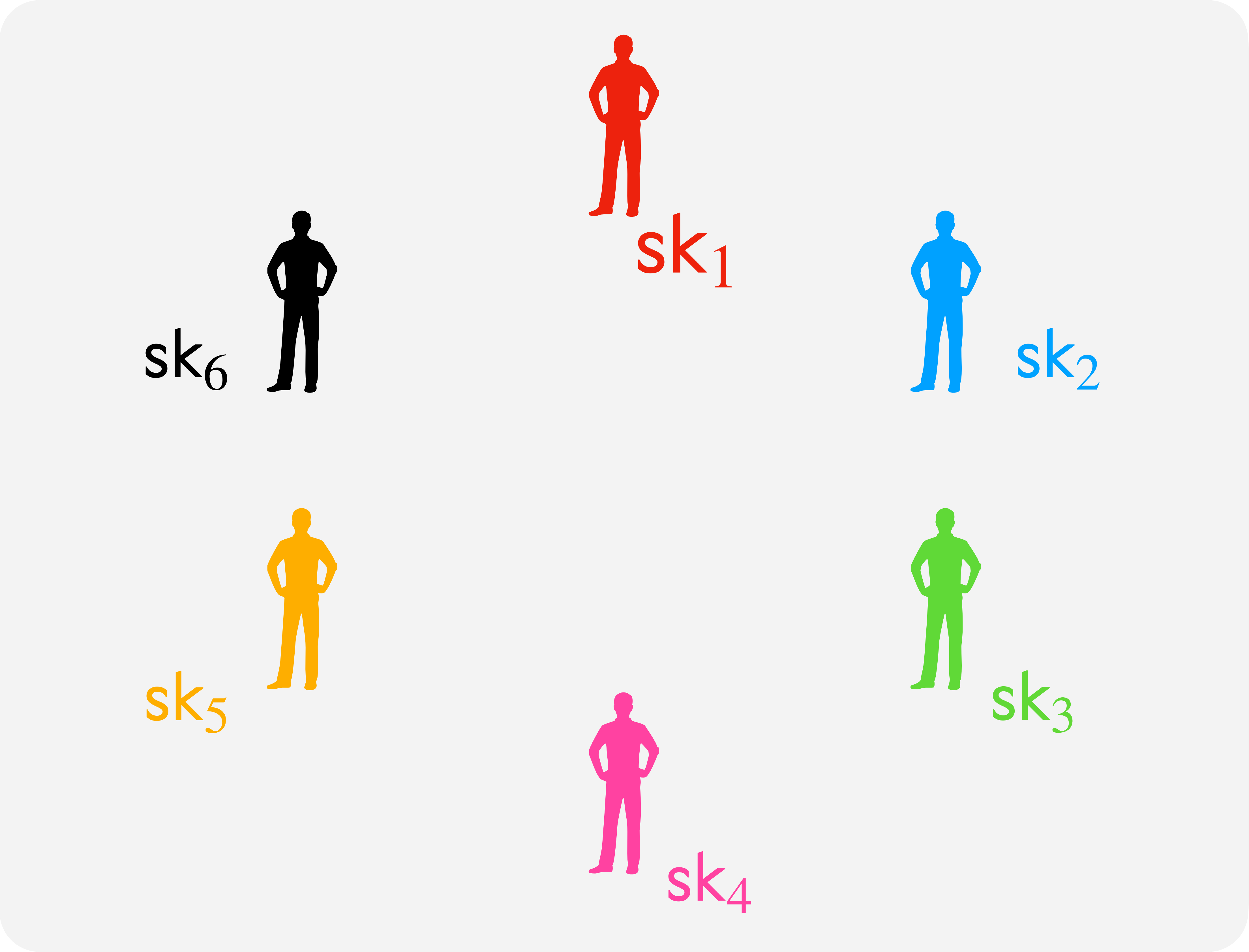
PEPR PQ TLS meeting - 13. Mar 2025

PQ SHIELD

1. Background

(T-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:
  - **Correctness:** Any T out of N parties can collaborate to sign a message under vk.
  - **Unforgeability:** $T - 1$ corrupted parties cannot sign.

Lattice-based Threshold Signatures

An active field of research.

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]

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

Thomas Espitau[1], 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 Submerse*: Distributed Key Generation and Robust Threshold Signature from Lattices**

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

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

Ivan Damgård[1], Claudio Orlandi[1], Akira Takahashi[1], and Mehdi Tibouchi[2]

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

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

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

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

Threshold Raccoon, a practical threshold signature

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]

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

More desirable properties

More desirable properties

- **Distributed Key Generation:** Protocol allowing to distributively sample key material.
- **Abort identification (or robustness):** In the presence of malicious users, the signature protocol can identify misbehaving users (or guarantee a valid output).

More desirable properties

- **Distributed Key Generation:** Protocol allowing to distributively sample key material.
- **Abort identification (or robustness):** In the presence of malicious users, the signature protocol can identify misbehaving users (or guarantee a valid output).

Prior art: Robustness from Verifiable Secret Sharing

> ***Flood and Submerse*: Distributed Key Generation and Robust Threshold Signature from Lattices**
>
> Thomas Espitau[1], Guilhem Niot[1,2], and Thomas Prest[1]

# rounds	Signers per session	| vk |	| sig |	Total comm.
4	3T	4 kB	13 kB	56T kB

2. Threshold 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]

Raccoon signature scheme

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → 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

$\text{Raccoon.Keygen}() \to \text{sk}, \text{vk}$

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

$\text{Raccoon.Sign}(\text{sk}, \text{msg}) \to \text{sig}$

- 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 $\text{sig} = (c, \mathbf{z})$

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

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

Unforgeable assuming

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

Hint-MLWE assumption [KLSS23].

$(\mathbf{A}, \text{vk})$ is pseudorandom even if given Q "hints":

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

As hard as MLWE_σ if

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

Threshold Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → 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

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

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

- $f(0) =$ sk and $\deg f \leq T - 1$
- Partial signing keys sk$_i$ $:= [\![sk]\!]_i = f(i)$

Properties:

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

$$\text{sk} = \sum_{i \in S} L_{S,i} \cdot [\![\text{sk}]\!]_i$$

Threshold Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

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

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

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

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 $\mathsf{cmt}_i = H_{\mathsf{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \mathsf{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot [\![sk]\!]_i + \mathbf{r}_i$

Combine: the final signature is

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

Threshold Raccoon

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

First (insecure) attempt

$\mathsf{ThRaccoon.Sign}(\mathsf{sk}, \mathsf{msg}) \to \mathsf{sig}$

Round 1:

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

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \mathsf{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot [\![sk]\!]_i + \mathbf{r}_i$

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 $L_{S,i} \cdot c \cdot [\![sk]\!]_i$ is large

$\rightarrow$ Leaks $[\![sk]\!]_i$

First (insecure) attempt

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 $\mathsf{cmt}_i = H_{\mathsf{cmt}}(\mathbf{w}_i)$

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \mathsf{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot [\![sk]\!]_i + \mathbf{r}_i$

Combine: the final signature is

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

Threshold Raccoon

- Prevent ROS attack with commit-reveal of $\mathbf{w}_i$
- But, $\mathbf{r}_i$ is small vs $L_{S,i} \cdot c \cdot [\![sk]\!]_i$ is large

  $\rightarrow$ Leaks $[\![sk]\!]_i$

- 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) $\rightarrow$ sig

Round 1:

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

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \mathsf{msg})$
- Broadcast $\mathbf{z}_i = L_{S,i} \cdot c \cdot [\![sk]\!]_i + \mathbf{r}_i + \Delta_i$

Combine: the final signature is

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

3. Detecting aborts in ThRaccoon

How to Shortly Share a Short Vector

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

Rafael del Pino[1], Thomas Espitau[1], Guilhem Niot[1,2], and Thomas Prest[1]

Challenge of detecting malicious behaviour in ThRaccoon

$\mathsf{ThRaccoon.Sign(sk, msg)} \rightarrow \mathsf{sig}$

Round 1:

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

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

Combine: the final signature is

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

Why is it challenging to tackle malicious behaviour to ThRaccoon?

- Main issue: computation of Δ_i using PRF to hide the secret when using Shamir sharing.

Challenge of detecting malicious behaviour in ThRaccoon

$\mathsf{ThRaccoon.Sign}(\mathsf{sk}, \mathsf{msg}) \rightarrow \mathsf{sig}$

Round 1:

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

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

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 $L_{S,i} \cdot [\![\mathsf{sk}]\!]_i$ 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 $L_{S,i} \cdot [\![\mathsf{sk}]\!]_i$.

Direction 2: Can we make both $L_{S,i} \cdot [\![\mathsf{sk}]\!]_i$ and $\mathbf{r}_i$ uniform?

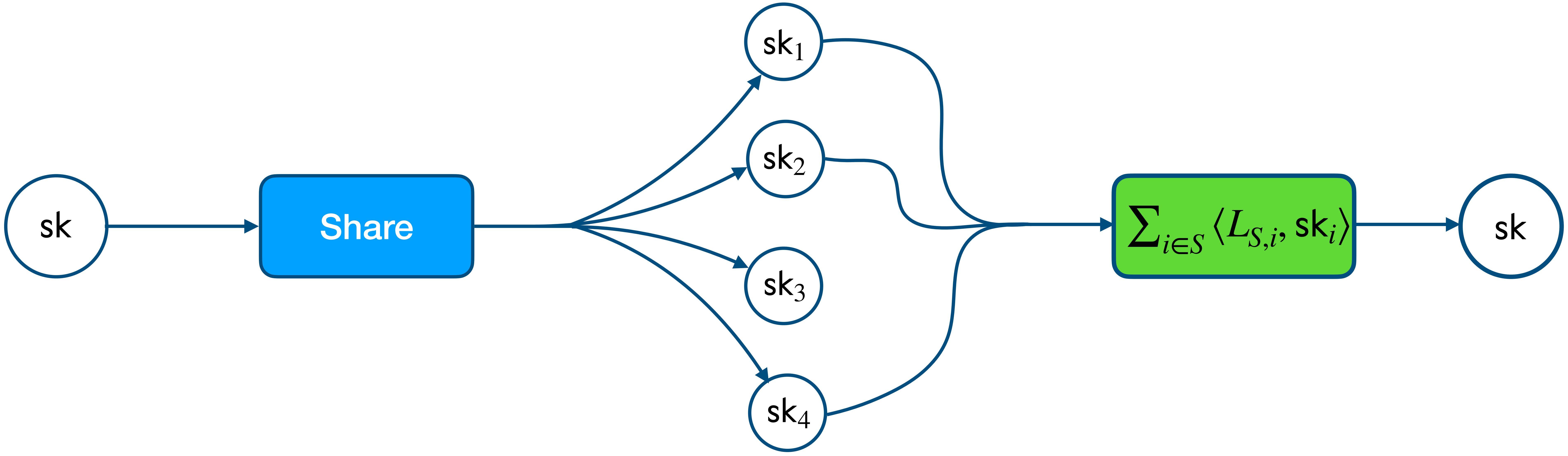
- Use Shamir-sharing for both sk and $\mathbf{r}$ → Flood and submerse [ENP24]

Direction 3: Can we make both $L_{S,i} \cdot [\![\mathsf{sk}]\!]_i$ and $\mathbf{r}_i$ short?

- Can we have **short shares and reconstructions coefficients** for both sk and $\mathbf{r}$?

Introducing Short Secret Sharing

- Our approach relies on sampling a sharing of sk such that we have:
  - Individual pool of short shares $\mathsf{sk}_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \ldots)$
  - 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) → sig

Round 1:

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

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

Combine: the final signature is

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

Security.

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

With Short Secret Sharing

$\mathsf{ShortSS.Sign}(\mathsf{sk}, \mathsf{msg}) \to \mathsf{sig}$

Round 1:

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

Round 2:

- Broadcast $\mathbf{w}_i$

Round 3:

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

Combine: the final signature is

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

Security.

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

Identifiable aborts.

- Each $\mathsf{vk}_i^{(j)} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_i^{(j)}$ is a valid public key ($\mathbf{s}_i^{(j)}$ is short), for $\mathsf{sk}_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \ldots)$

  $\to$ Each $(c, \mathbf{z}_i)$ is a valid signature for $\langle L_{S,i}, (\mathsf{vk}_i^{(j)})_j \rangle$
- Identifiable abort is as easy as verifying partial signatures!
- *Akin to abort identification in Sparkle (Threshold Schnorr): perform partial verifications.*

4. How to concretely sample short sharings

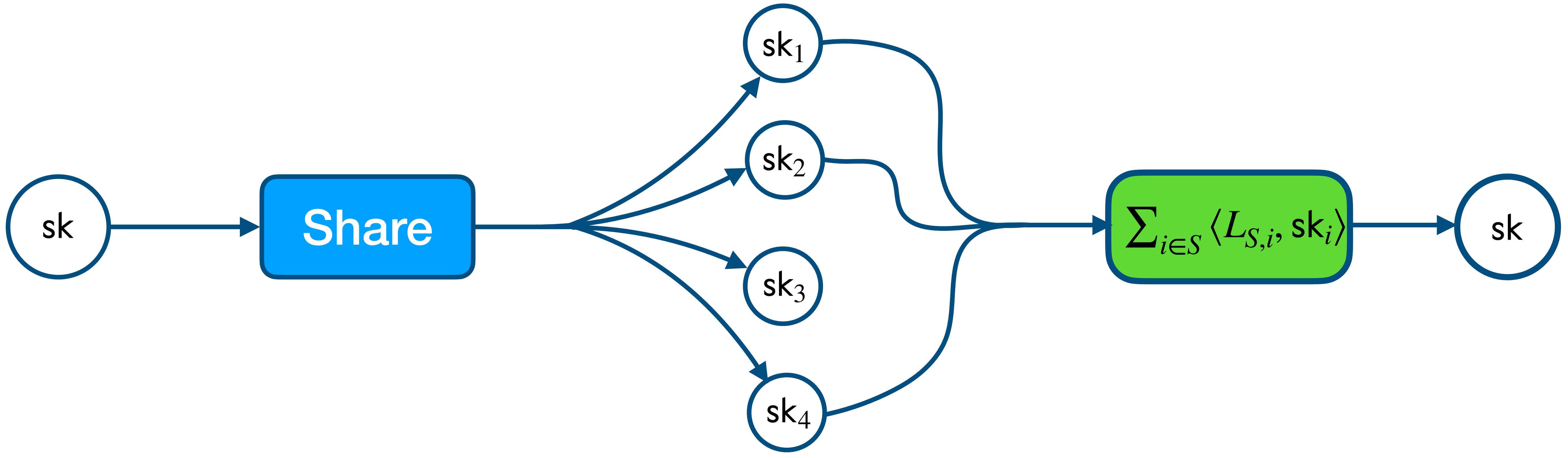
How to Shortly Share a Short Vector

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

Rafael del Pino[1], Thomas Espitau[1], Guilhem Niot[1,2], and Thomas Prest[1]

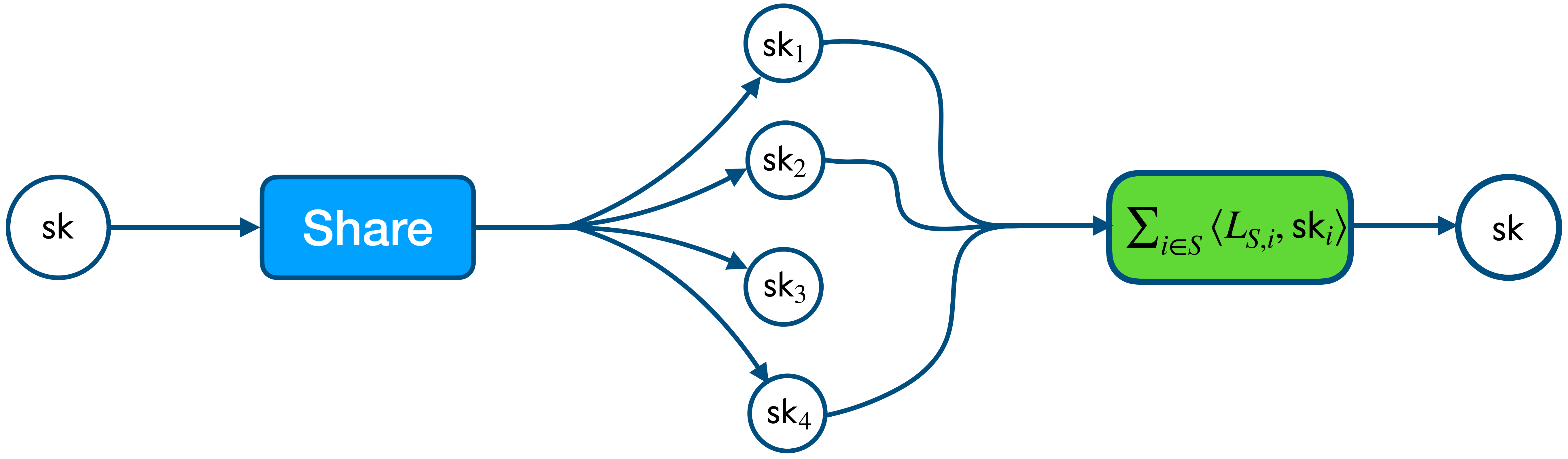
Short Secret Sharing

- Individual pool of short shares $\mathsf{sk}_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \ldots)$
- 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 $\mathsf{sk}_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \ldots)$
- 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…

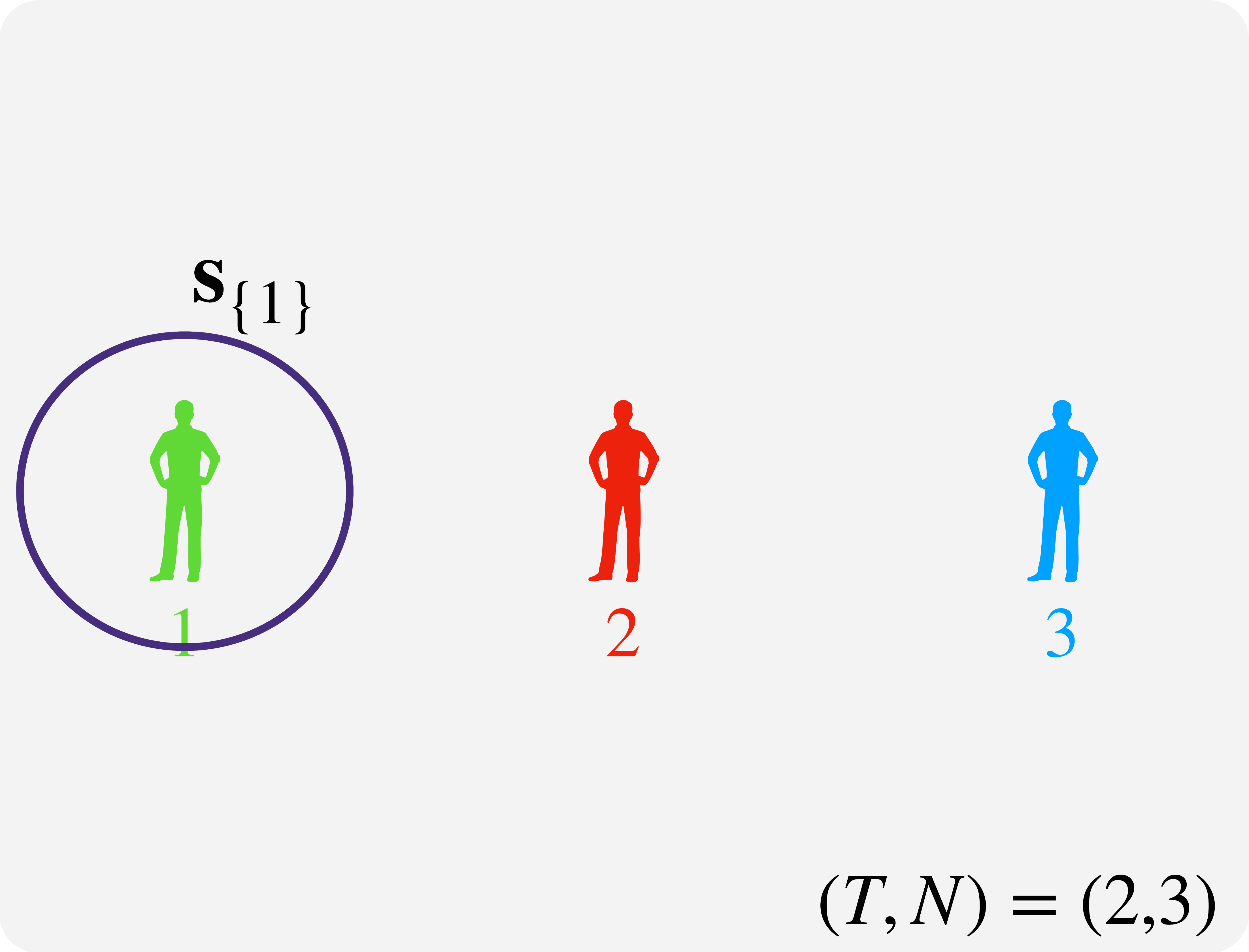
But, lattice-based schemes, often just need $[\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{sk}$ to look uniform.
We can:

- Leak an offset of the secret: $\mathsf{sk} = \mathsf{sk}_{\mathsf{safe}} + \mathsf{sk}_{\mathsf{leak}}$
- Leak hints on the secrets $h = c \cdot \mathsf{sk} + y$, for large enough y

Solution 1: Replicated Secret Sharing

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

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

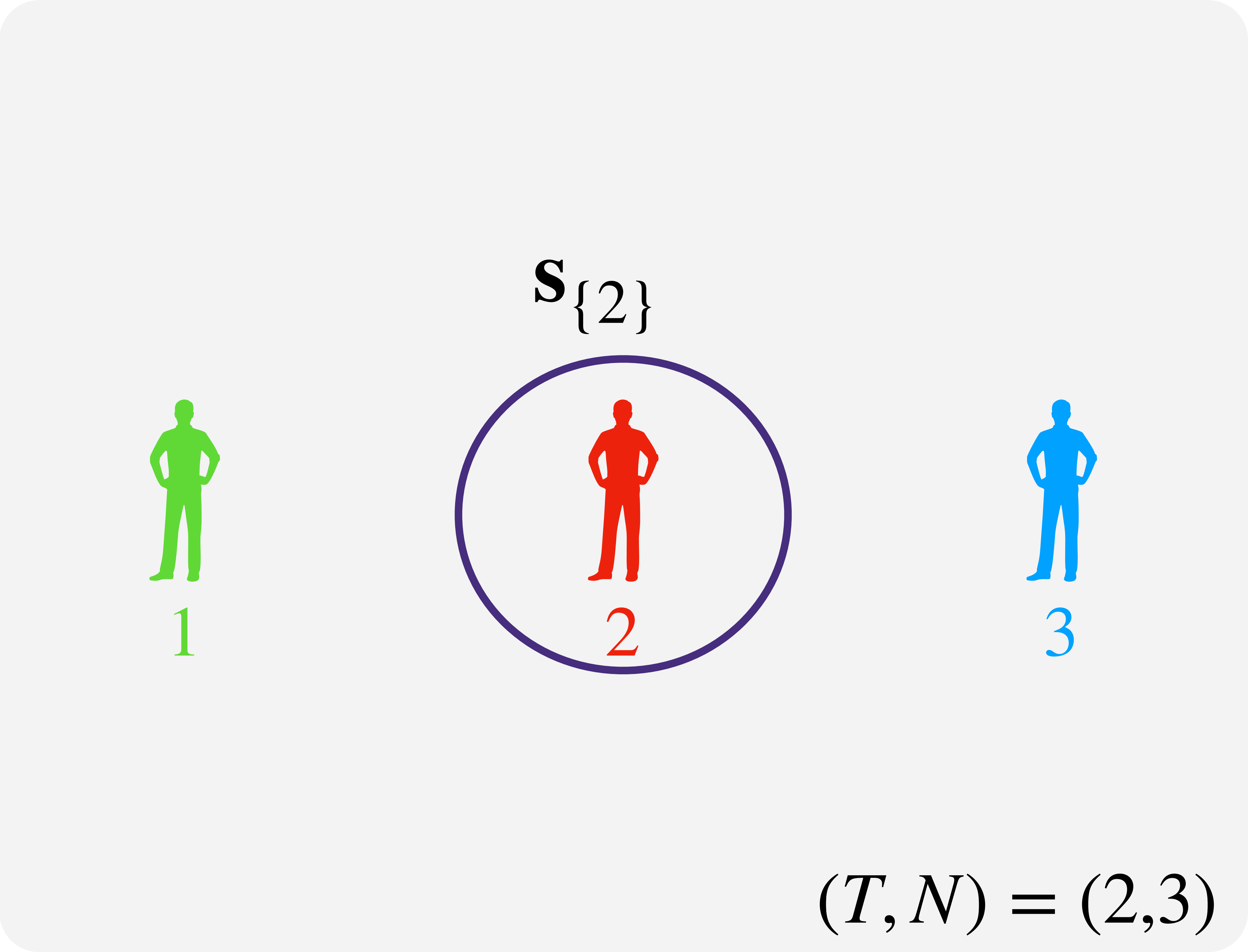


Solution 1: Replicated Secret Sharing

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 $\mathbf{s}_{\mathcal{T}}$.

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

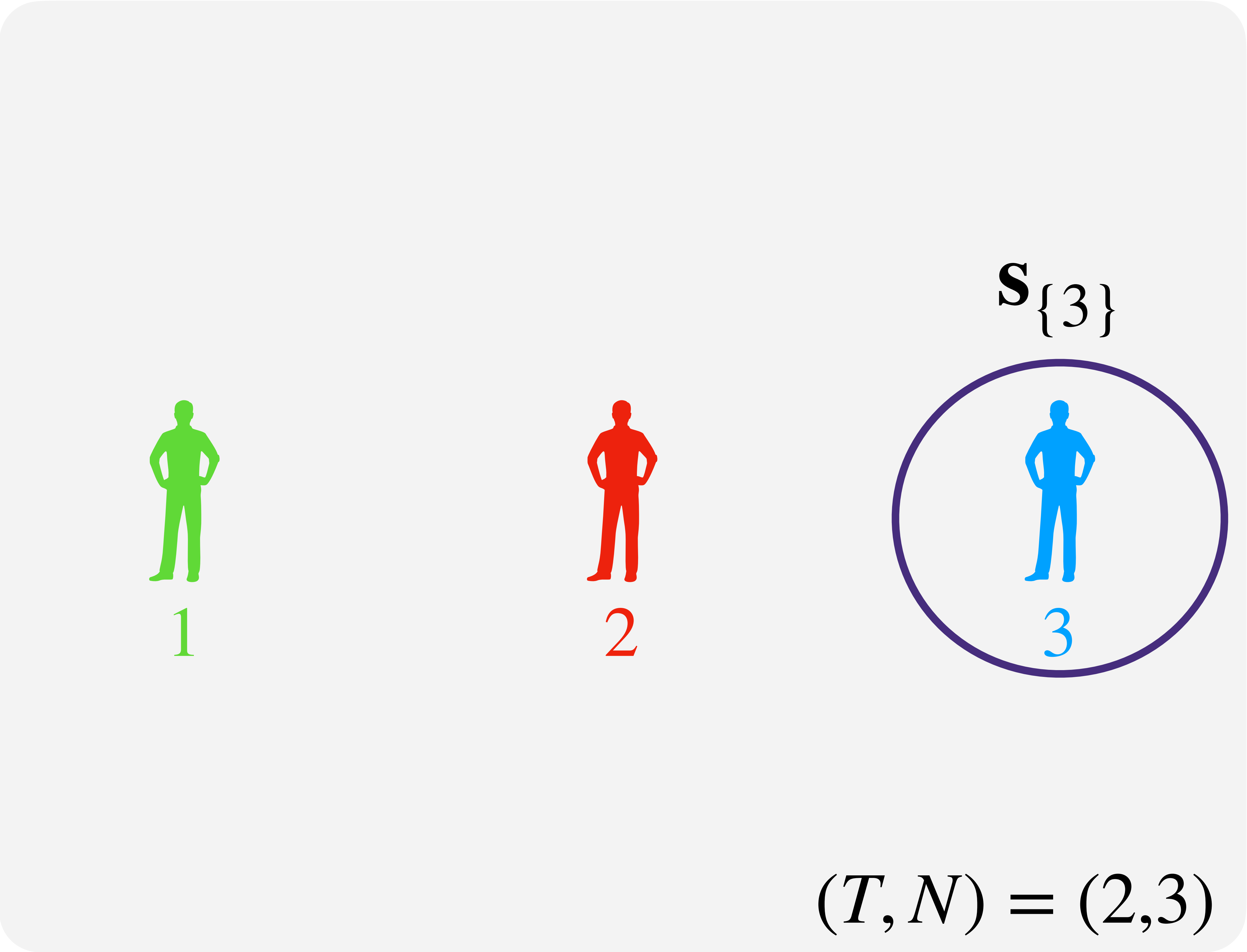


Solution 1: Replicated Secret Sharing

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 $\mathbf{s}_{\mathcal{T}}$.

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

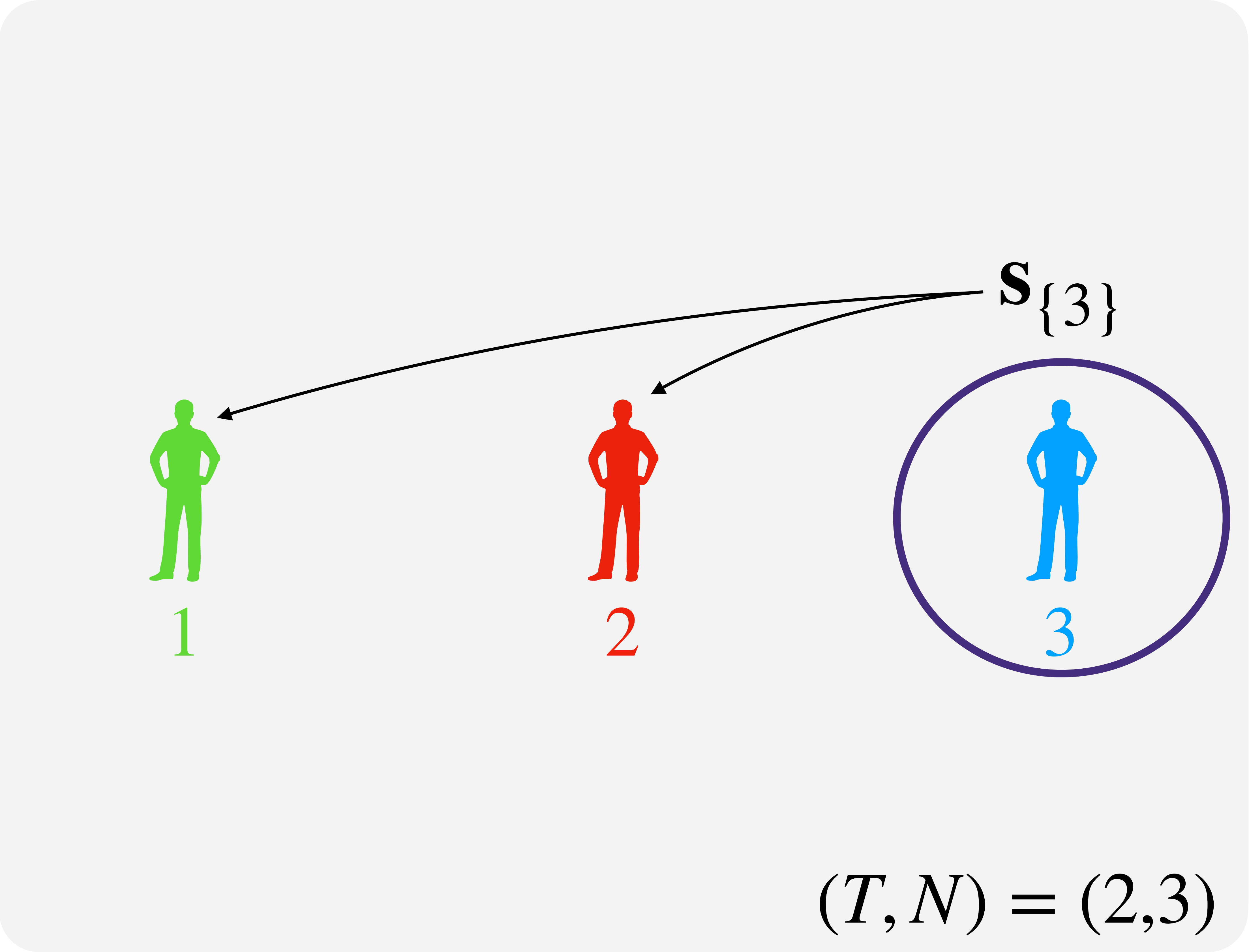


Solution 1: Replicated Secret Sharing

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 $[N]\backslash\mathcal{T}$.



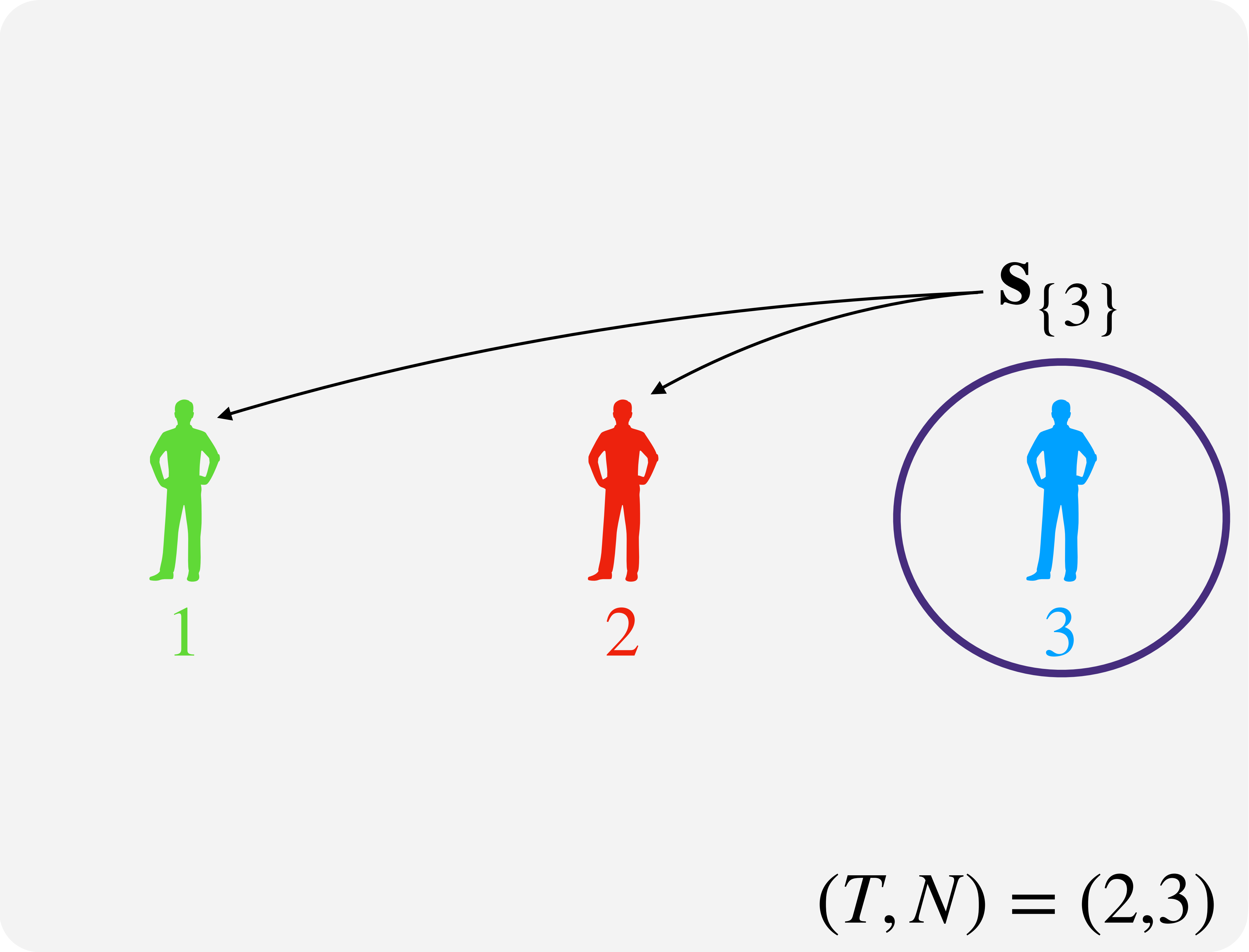
Solution 1: Replicated Secret Sharing

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 $[N]\backslash\mathcal{T}$.

3. Define 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 $\mathscr{T}$ of $T-1$ parties, sample a uniform share $\mathbf{s}_{\mathscr{T}}$.
2. Distribute $\mathbf{s}_{\mathscr{T}}$ to the parties in $[N]\backslash\mathscr{T}$.
3. Define $\mathsf{sk} = \sum_{\mathscr{T}} \mathbf{s}_{\mathscr{T}}$.

Properties:

- Reconstruction coefficients 0 or 1
- When $< T$ corrupted parties, at least one $\mathbf{s}_{\mathscr{T}}$ remains hidden.

$\rightarrow$ 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 $[N]\backslash\mathcal{T}$.
3. Define 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.

$\rightarrow$ guarantees that $[\mathbf{A} \quad \mathbf{I}] \cdot$ 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 $\mathscr{T}$
   sample a short

2. Distribute $\mathbf{s}_{\mathscr{T}}$ t
   $[N]\backslash\mathscr{T}$.

3. Define sk $= \sum_{\mathscr{T}} \mathbf{s}_{\mathscr{T}}$.

Caveat: This scheme has a number of shares that is equal to $\binom{N}{T-1}$.

efficients 0 or 1

ted parties, at least one $\mathbf{s}_{\mathscr{T}}$ remains hidden.

→ guarantees that $[\mathbf{A} \quad \mathbf{I}] \cdot$ 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

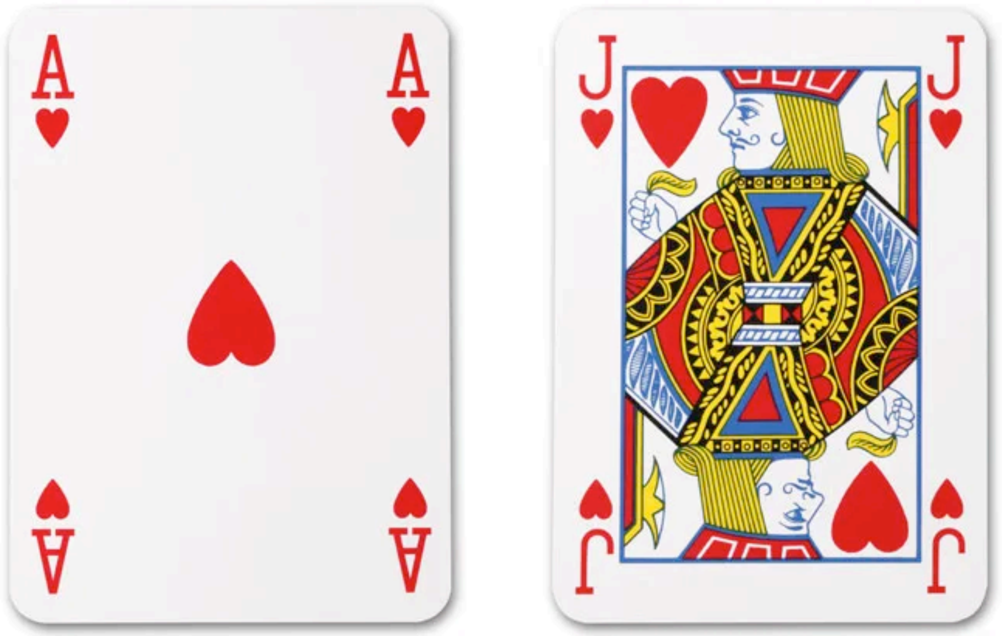
1

Solution 2: Coupon collector problem

Full collection

N cards

Draw with replacement



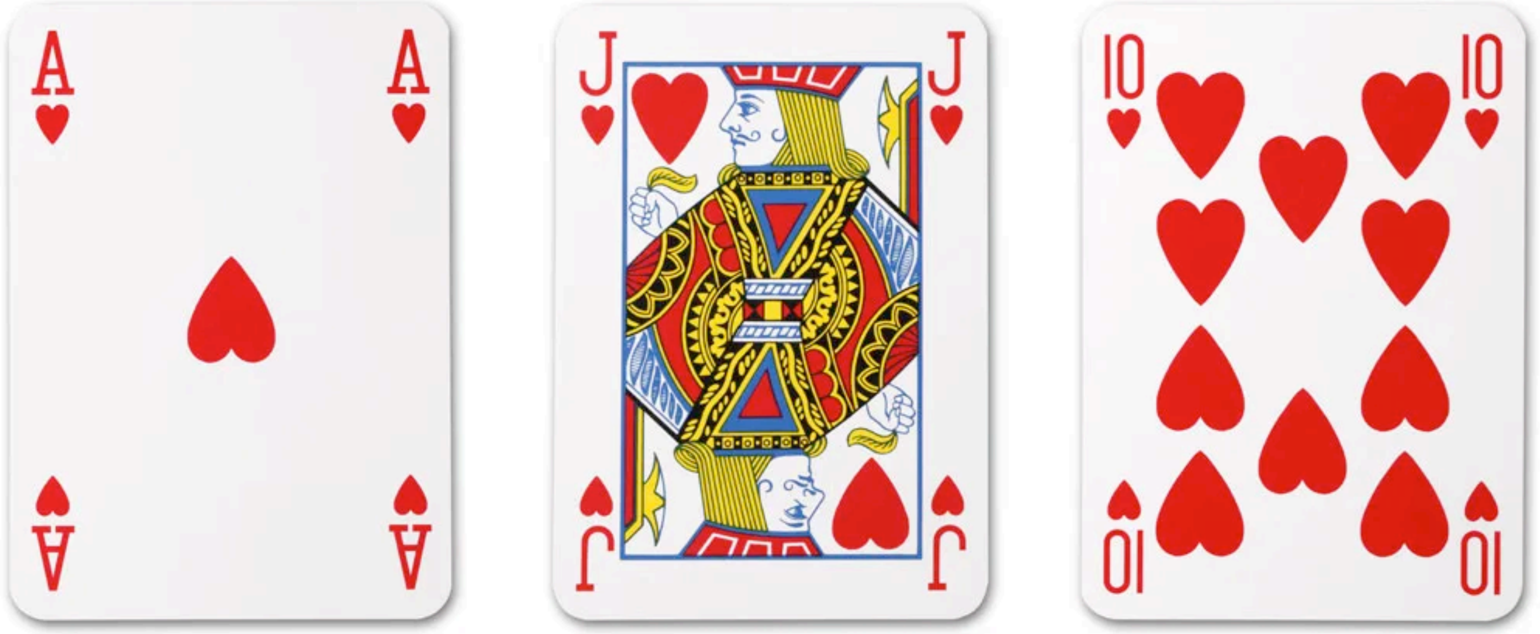
1 2

Solution 2: Coupon collector problem

Full collection

N cards

Draw with replacement



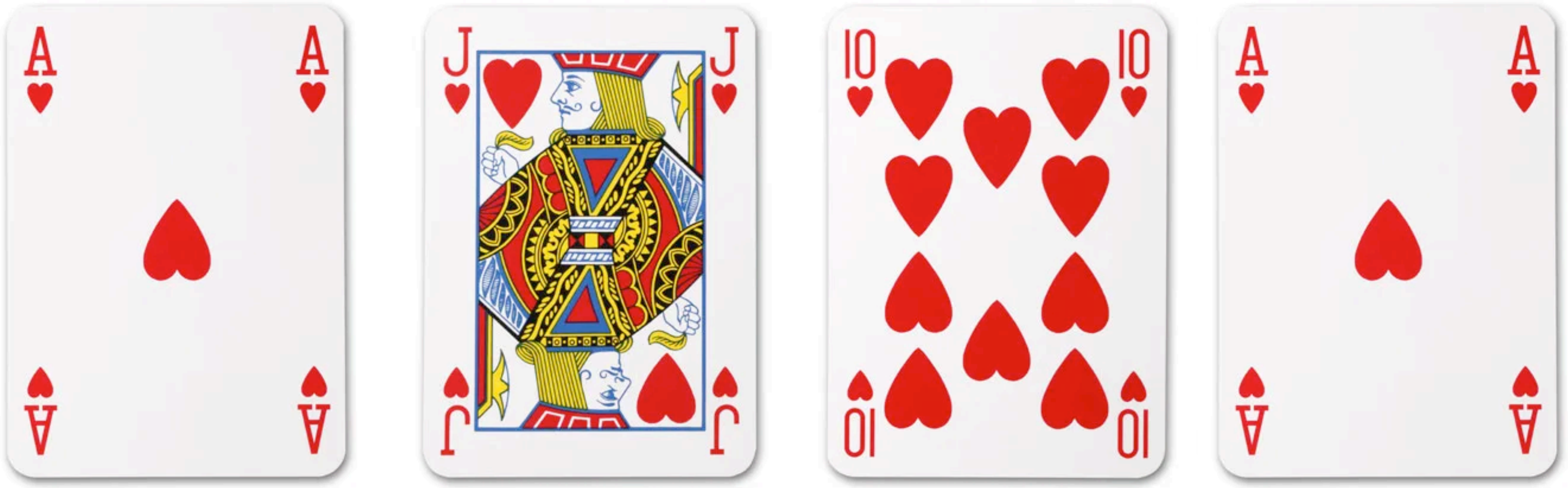
1 2 3

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 $\mathsf{sk} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3 + \mathbf{s}_4$

N shares

Example:

- $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1} \leftarrow \mathscr{D}_\sigma^{N-1}$ and $\mathbf{s}_N = \mathsf{sk} - \sum_{j<N} \mathbf{s}_i$

Solution 2: Coupon collector problem

Full collection $\mathsf{sk} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3 + \mathbf{s}_4$

N shares

Idea: Randomly distribute one share per party.

Example:

- $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1} \leftarrow \mathscr{D}_\sigma^{N-1}$ and $\mathbf{s}_N = \mathsf{sk} - \sum_{j<N} \mathbf{s}_i$

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 $\mathsf{sk} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3 + \mathbf{s}_4$

N shares

Example:

- $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1} \leftarrow \mathscr{D}_\sigma^{N-1}$ and $\mathbf{s}_N = \mathsf{sk} - \sum_{j<N} \mathbf{s}_i$

Idea: Randomly distribute one share per party.

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 $\underset{N\to\infty}{\approx} 1 + 128/\log N$

Solution 2: Coupon collector problem

Full collection $\mathsf{sk} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3 + \mathbf{s}_4$

N shares

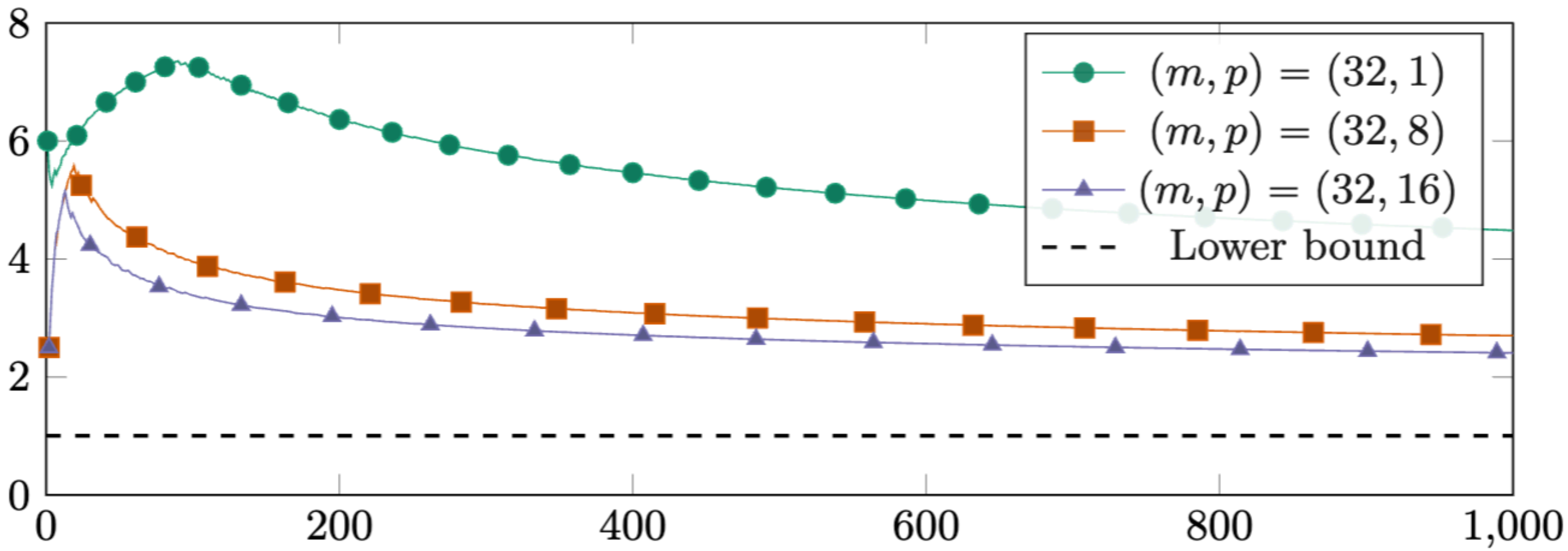
Better parameters by amplifying properties:

- **Reconstruction threshold:** Share same sk m times, just need at least one sharing fully known to recover sk.
- **Security threshold:** Share multiple secrets sk

$$\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2 + \ldots + \mathsf{sk}_p$$

An adversary must know all the secrets to forge.

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.

5. Let’s instantiate it!

ThRaccoon with Identifiable aborts

Instantiating our scheme with short secret sharings.

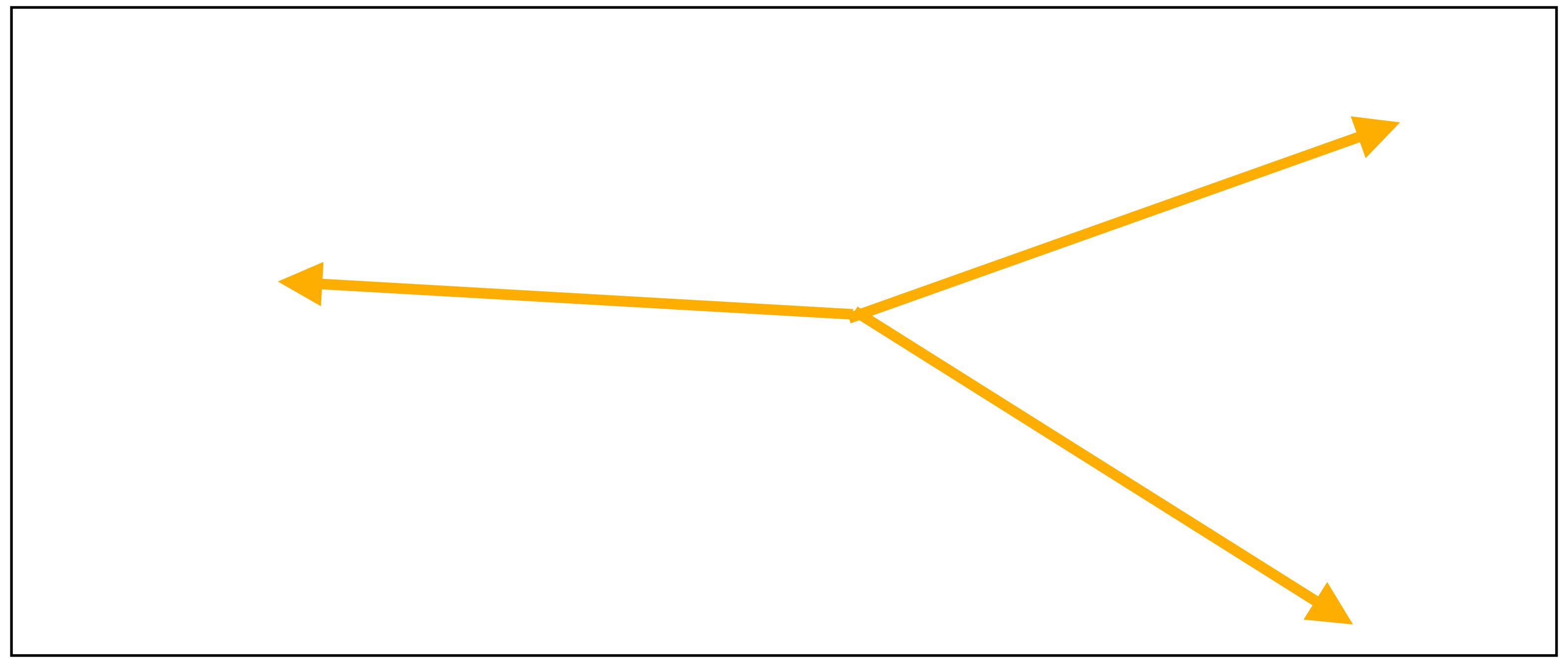
- Small thresholds $N \leq 16$ with replicated secret sharing
- Or, large thresholds $N \leq 1024$ (but with security/reconstruction gap) with ramp secret sharing based on coupon collector

Phase	# rounds	| vk |	| sig |	Total communication
Signing	3	4 kB	11.9 kB	25 kB
Abort Identification	0			

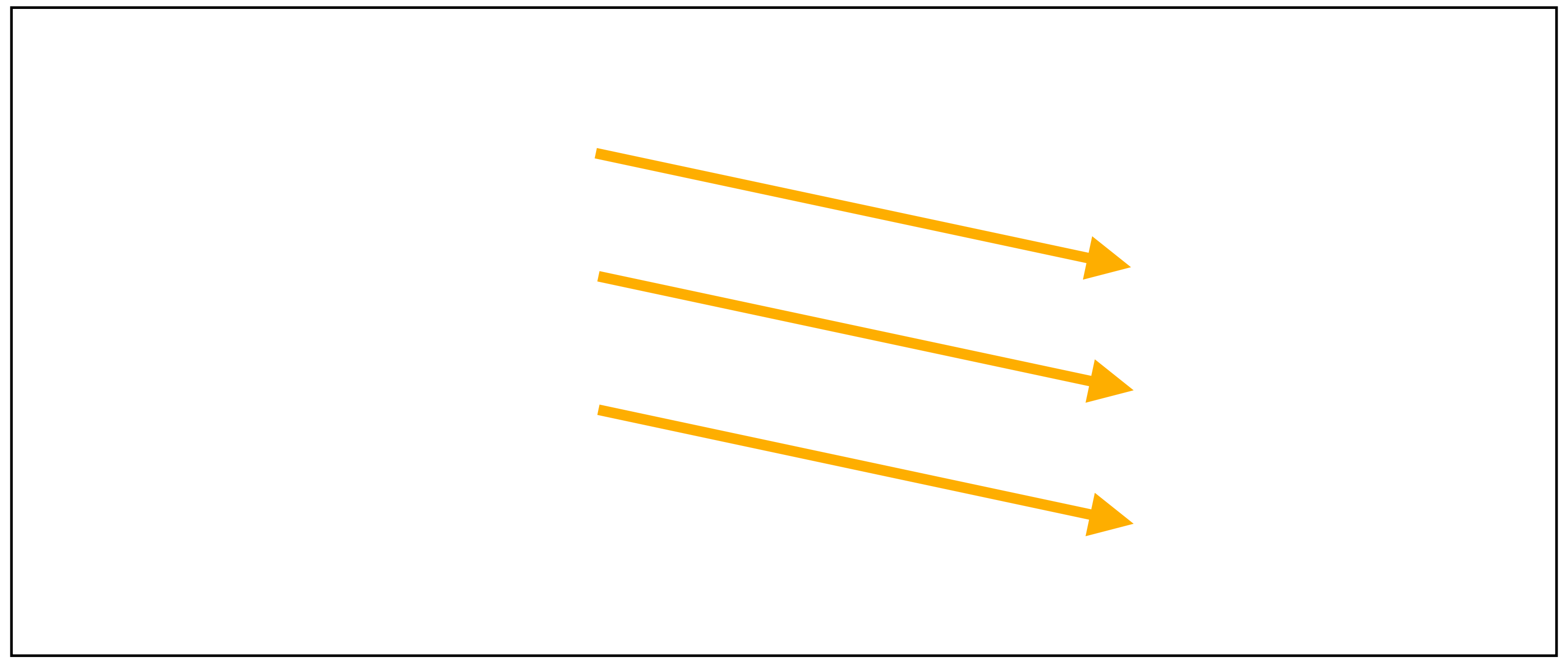
Bonus: tighter check bounds using Short SS

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?



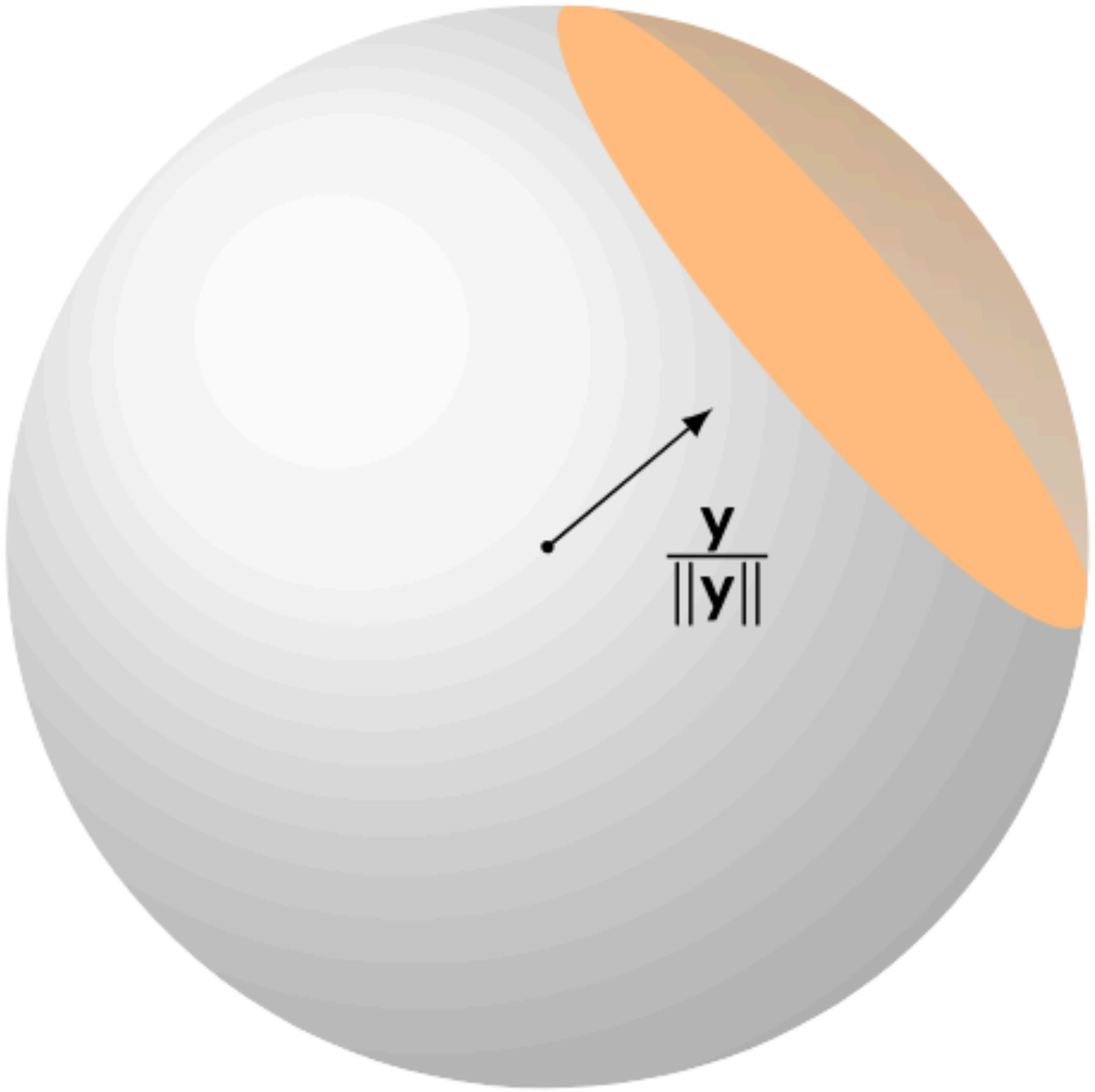
Average-case: $O(\sqrt{T})$



Worst-case: $O(T)$

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

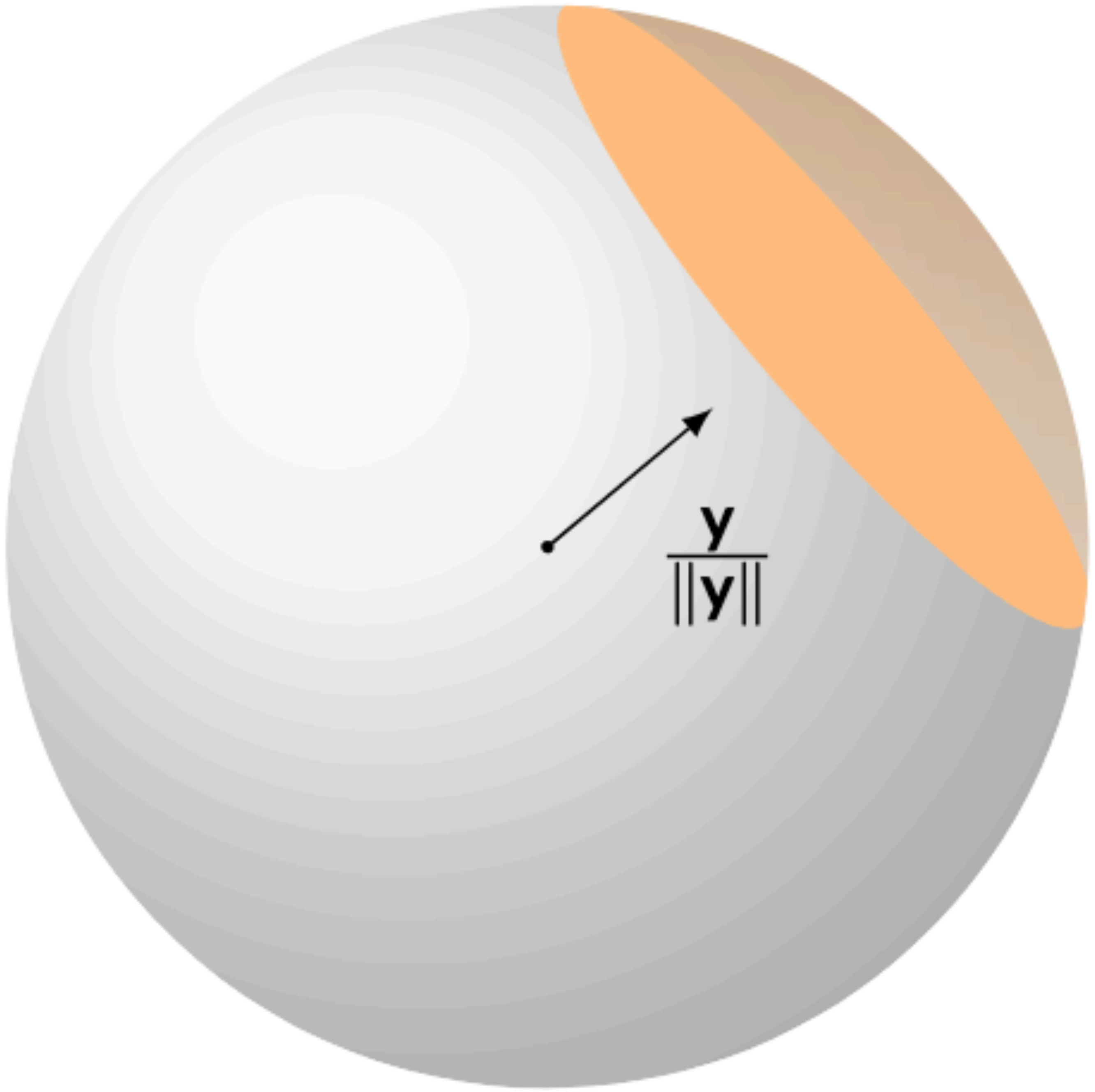
The Death Star Algorithm



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

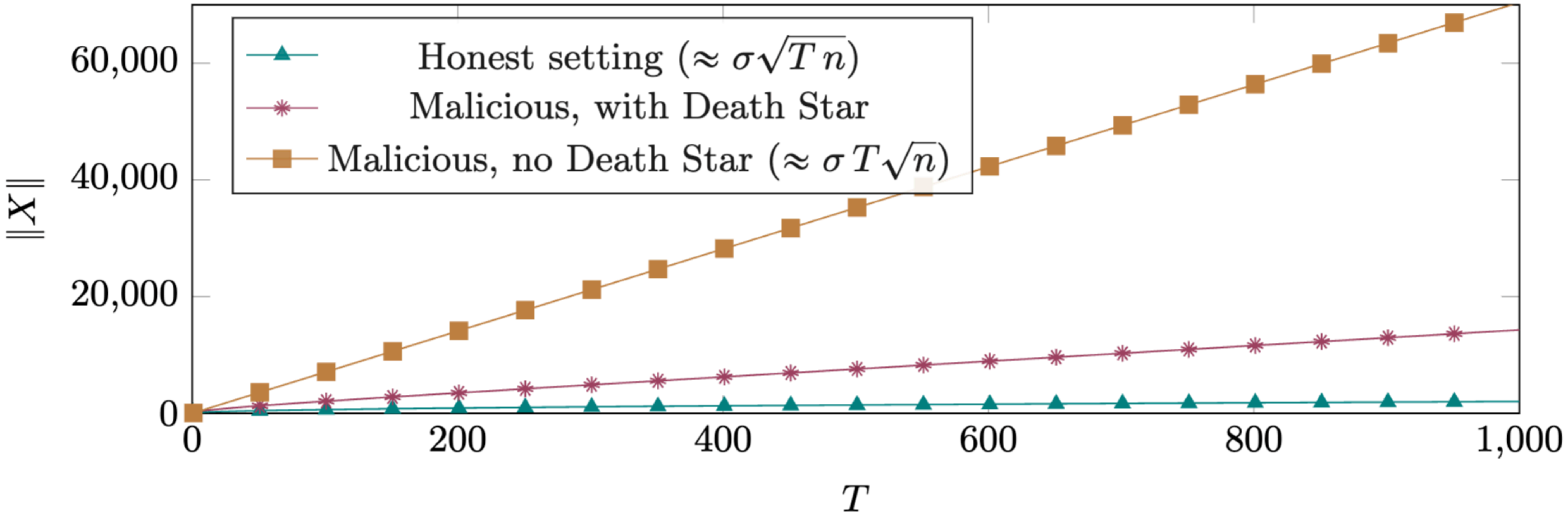
- For any vector $\mathbf{y}$, $\langle \mathbf{x}, \mathbf{y} \rangle \lessapprox \|x\| \|y\| / \sqrt{n/\lambda}$ except with probability $2^{-\lambda}$.

The Death Star Algorithm



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

- For any vector $\mathbf{y}$, $\langle \mathbf{x}, \mathbf{y} \rangle \lessapprox \|x\| \|y\| / \sqrt{n/\lambda}$ except with probability $2^{-\lambda}$.



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

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'
- **Application to Threshold Raccoon with identifiable aborts (using partial verification keys)**
  - Tighter norm bound for the sum of T *potentially malicious* contributions with Death Star algorithm

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'
- **Application to Threshold Raccoon with identifiable aborts (using partial verification keys)**
  - Tighter norm bound for the sum of T *potentially malicious* contributions with Death Star algorithm
- **Future work?**
  - Better short secret sharings? → work in progress
  - Other applications? → Compact threshold signature for less than 8 parties (2.7kB), to appear at PKC 2025 + talk at JC2 2025

Questions?