Secret Sharing Schemes for Lattice-Based Threshold Cryptography

Guilhem Niot, joint works with PQShield & Friends

Workshop on secret sharing schemes - May 2025

1. Background

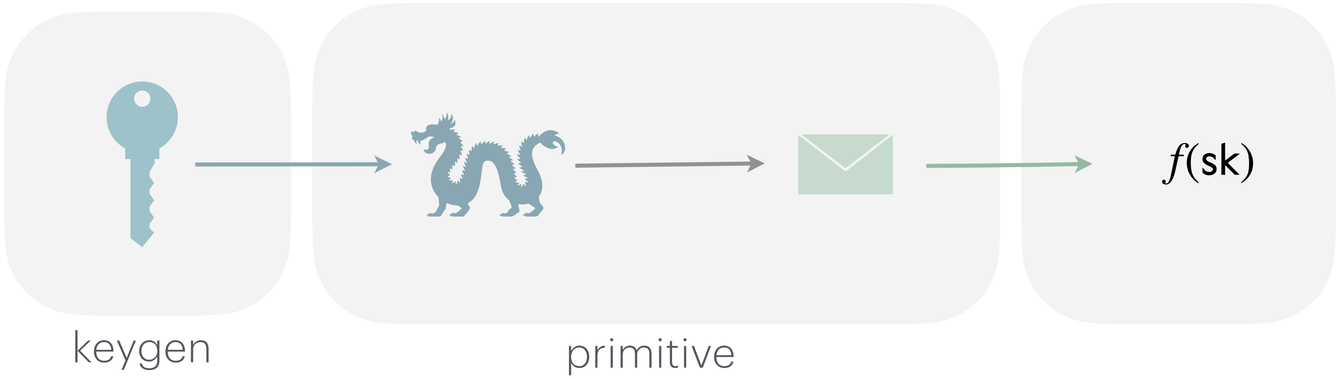
Problem: distributing cryptographic primitives

Cryptographic primitive of interest

Problem: distributing cryptographic primitives

Cryptographic primitive of interest

Centralized setting



Problem: distributing cryptographic primitives

Cryptographic primitive of interest

What if the party is corrupted or becomes unresponsive…

Question: can we split the trust among several parties?

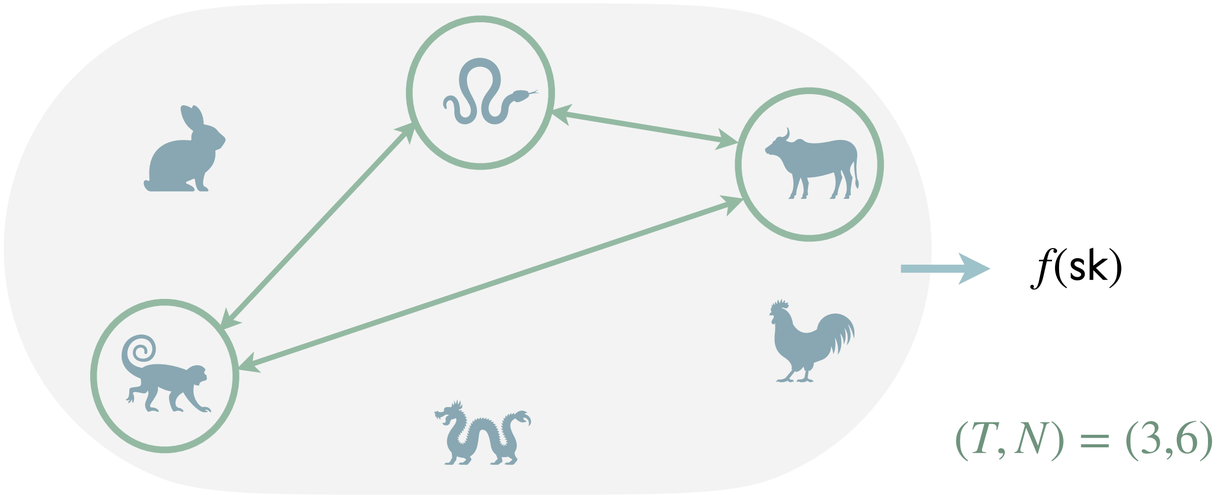
Problem: distributing cryptographic primitives

Cryptographic primitive of interest

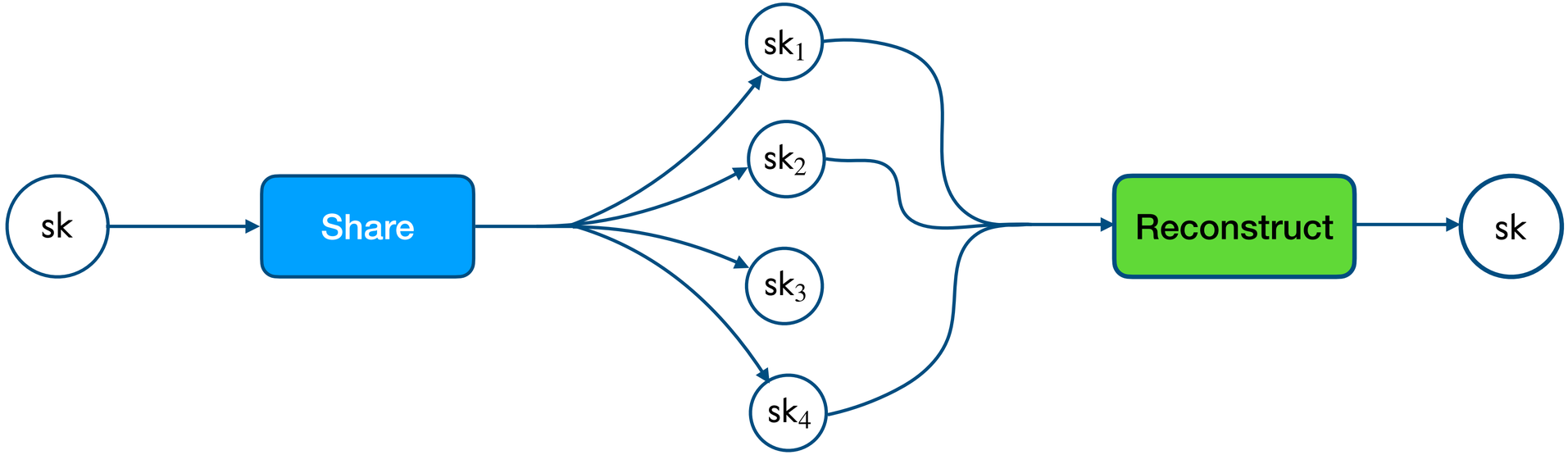
What if the party is corrupted or becomes unresponsive…

Question: can we split the trust among several parties?

Interactive protocol to distribute the primitive: T-out-of-N parties can collaborate to compute the function and $T-1$ parties cannot.



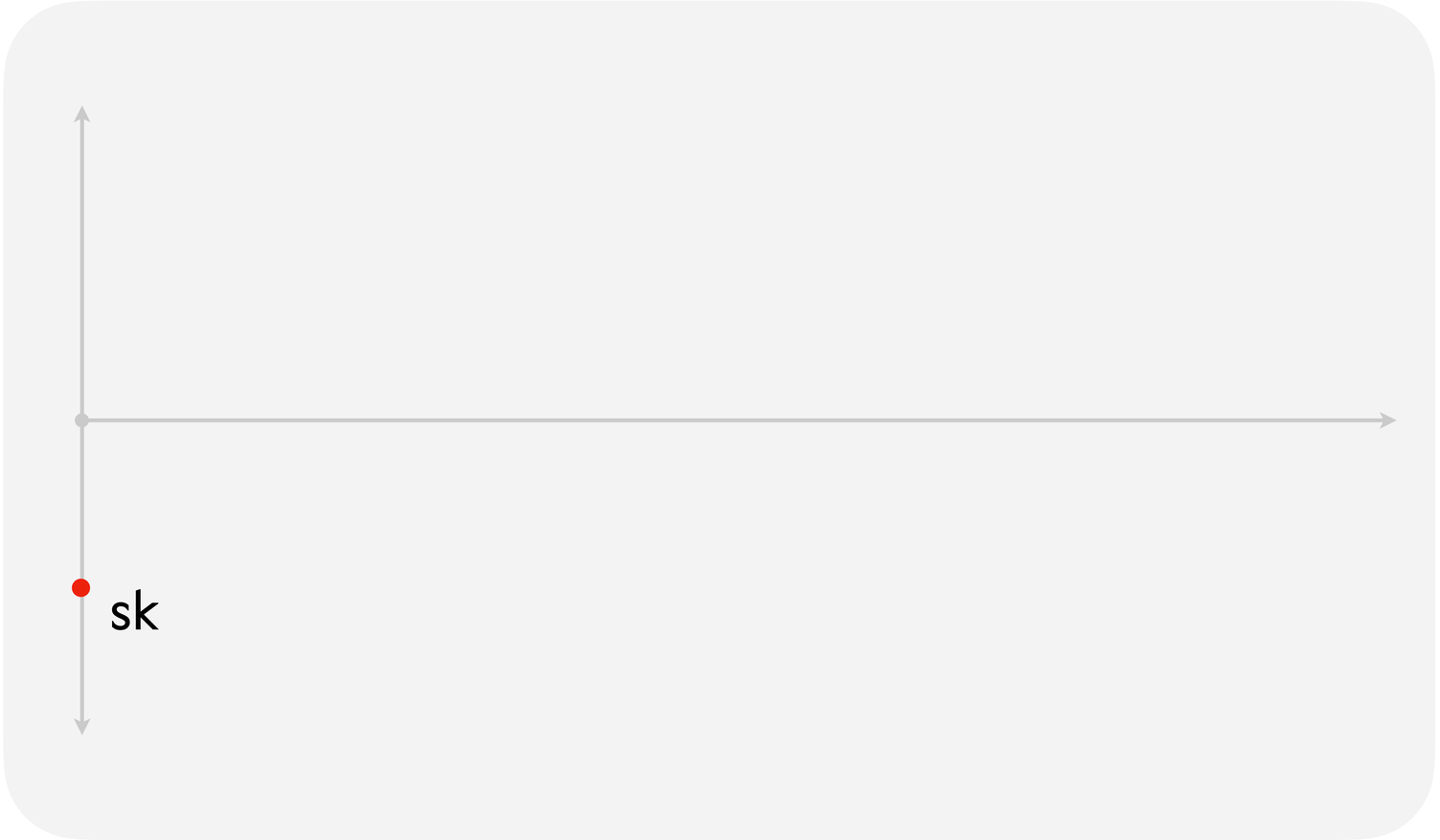
Building block: secret sharings



- Individual shares $(sk_1, \ldots, sk_N)$
- T shares: can reconstruct sk
- $\leq T - 1$ shares: sk is hidden

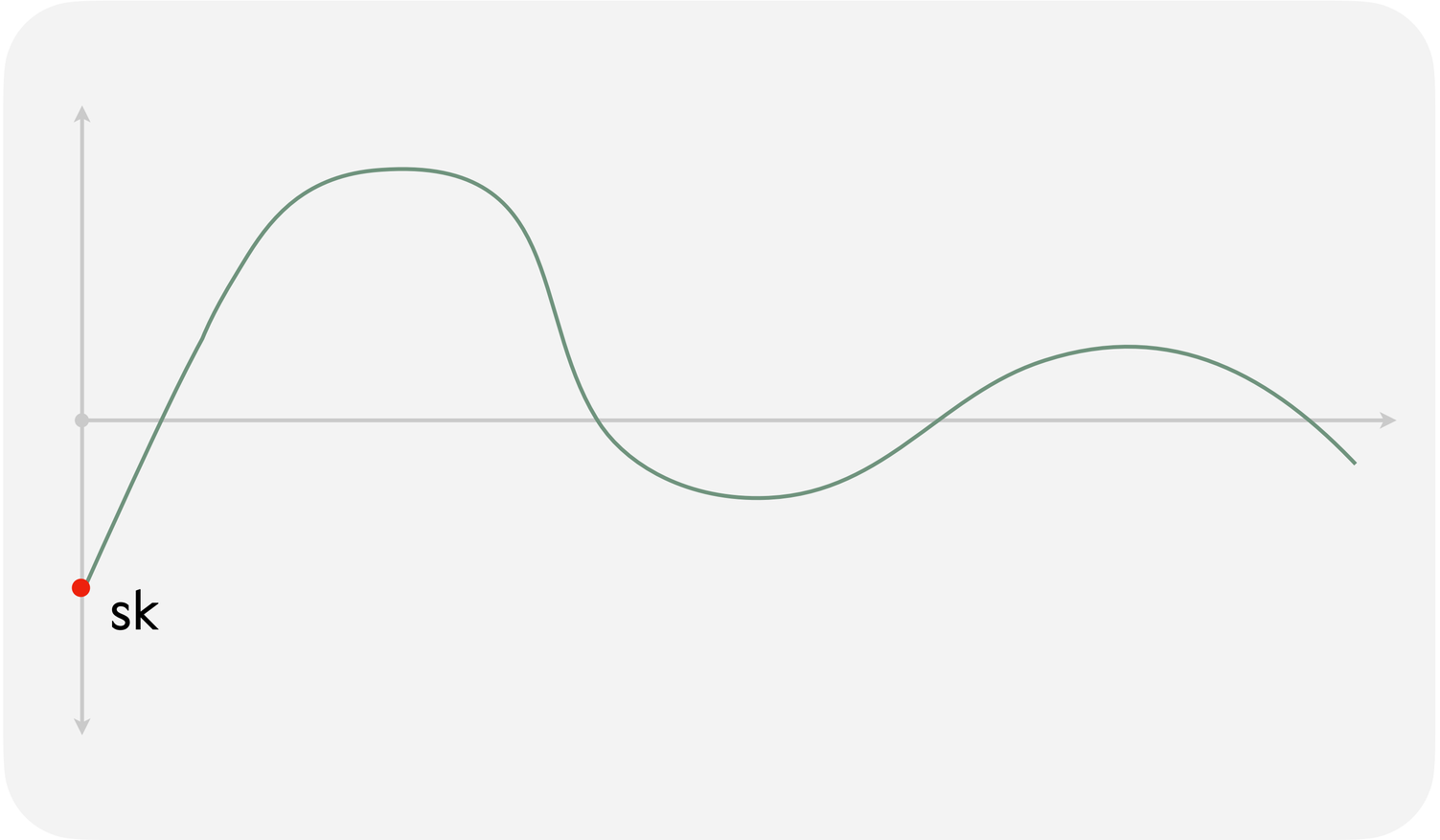
Building block: Shamir secret sharing

- secret sk



Building block: Shamir secret sharing

- secret sk
- curve of degree $T - 1$



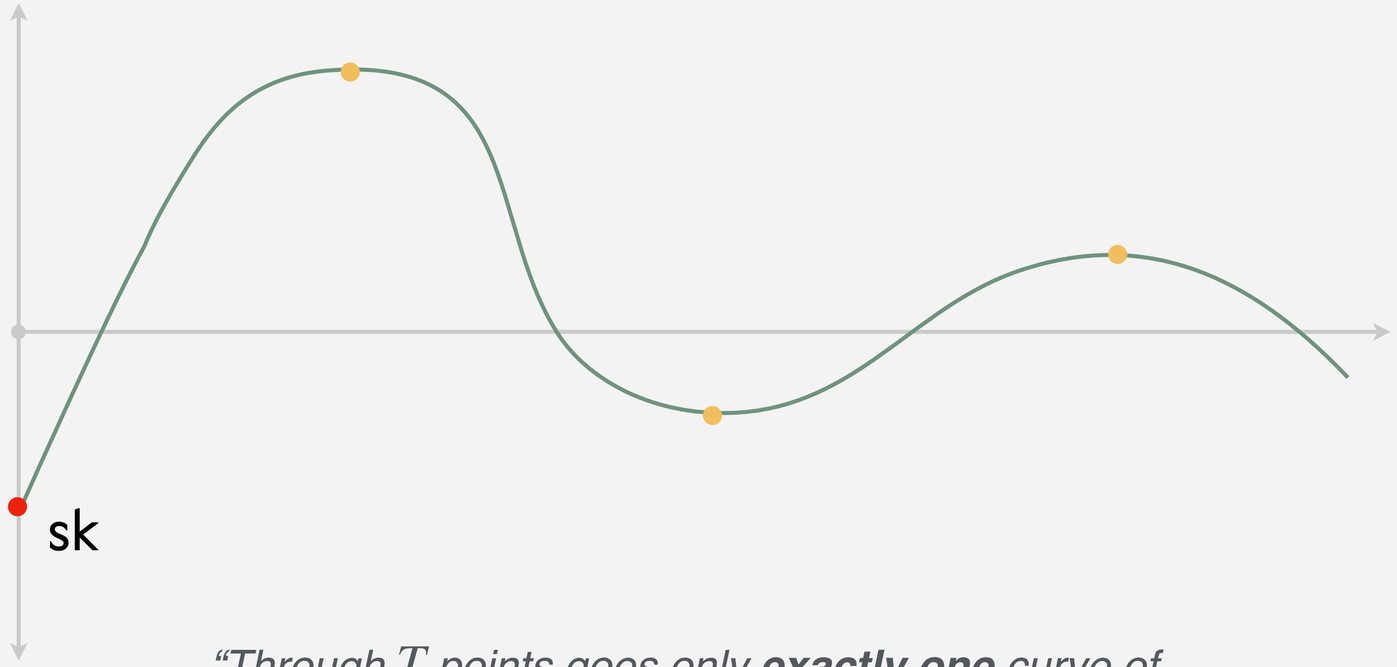
Building block: Shamir secret sharing

secret sk

curve of degree $T - 1$

shares = points of

Here, reconstruction of **sk** is **linear**



*"Through T points goes only **exactly one** curve of degree $T - 1$"*

Building block: more secret sharings…

- Additive sharing
- CRT-based sharing
- Error correcting code based sharing
  - Shamir secret sharing

Which sharing should I choose?

It depends on the function f:

- Some operations are easier with given secret sharings

Which sharing should I choose?

It depends on the function f:

- Some operations are easier with given secret sharings

It depends on the security model and access structure:

- Some sharings allow error detection, or more complex access structures

2. Lattice-based cryptography

Let’s first try to distribute it with Shamir’s sharing!

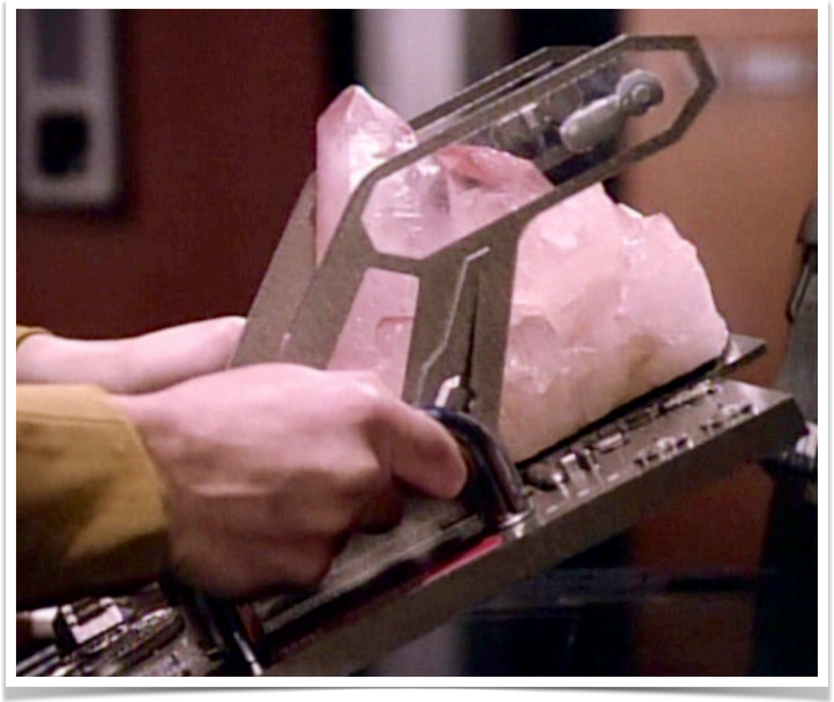
Example: ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

ML-DSA . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- If $\mathbf{z}$ not in S, **restart**
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$



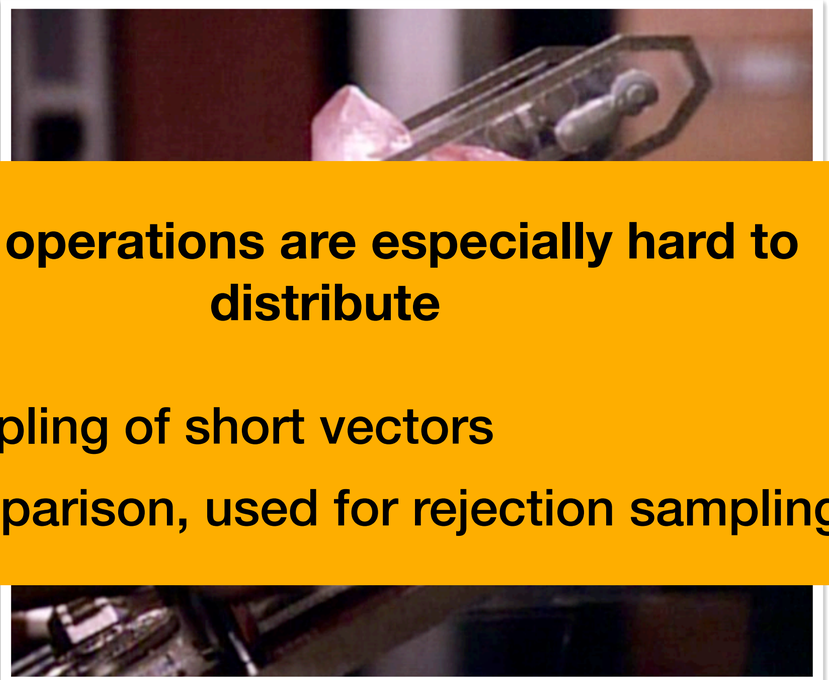
Example: ML-DSA signatures

ML-DSA . Keygen() → sk, vk

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

ML-DSA . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- If $\mathbf{z}$ not in S, **restart**
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$



Some operations are especially hard to distribute

- Sampling of short vectors
- Comparison, used for rejection sampling

A first step towards a solution: Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}, \mathbf{y}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{y}$
- $\mathbf{w}_\mathsf{T} = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\mathsf{T}, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- ~~If $\mathbf{z}$ not in S, **restart**~~
- $\mathbf{h} = \mathbf{w}_\mathsf{T} - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output sig $= (c, \mathbf{z}, \mathbf{h})$

Let's remove the rejection sampling!

A first step towards a solution: Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}, \mathbf{y}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{y}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- ~~If $\mathbf{z}$ not in S, **restart**~~
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$

Unforgeable under

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

| vk |	| sig |
2.3 kB	11.5 kB

Hint-MLWE assumption [KLSS23].

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

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

As hard as MLWE_σ if

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

A first step towards a solution: Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}, \mathbf{y}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{y}$
- $\mathbf{w}_\mathsf{T} = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\mathsf{T}, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- ~~If $\mathbf{z}$ not in S, **restart**~~
- $\mathbf{h} = \mathbf{w}_\mathsf{T} - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$

Now, can we distribute short vector sampling?

i.e. sample short $\mathbf{r}$*, such that* $T - 1$ *parties do not learn* $\mathbf{r}$

A first step towards a solution: Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}, \mathbf{y}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{y}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- ~~If $\mathbf{z}$ not in S, **restart**~~
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$

Now, can we distribute short vector sampling?

i.e. sample short $\mathbf{r}$*, such that* $T - 1$ *parties do not learn* $\mathbf{r}$

Idea: sample T short vectors, and sum them

- Party i samples short $\mathbf{r}_i$
- Define $\mathbf{r} = \sum \mathbf{r}_i$

Threshold Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}, \mathbf{y}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{y}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$

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

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

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

With a set S of $\geq T$ shares, reconstruct sk via Lagrange interpolation

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

Threshold Raccoon

Raccoon . Keygen() → sk, vk

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

Raccoon . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}, \mathbf{y}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{y}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \mathsf{msg})$
- $\mathbf{z} = c \cdot \mathsf{sk} + \mathbf{r}$
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \mathsf{vk} \rceil_\nu$
- Output $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$

First (insecure) attempt

ThRaccoon . Sign(sk, msg) → sig

Round 1:

- Sample a short $\mathbf{r}_i, \mathbf{y}_i$
- $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{y}_i$
- 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 $\mathbf{z}_i = L_{S,i} \cdot c \cdot [\![\mathsf{s}k]\!]_i + \mathbf{r}_i$

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) → sig

Round 1:

- Sample a short $\mathbf{r}_i, \mathbf{y}_i$
- $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{y}_i$
- 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 $\mathbf{z}_i = L_{S,i} \cdot c \cdot [\![\mathsf{s}k]\!]_i + \mathbf{r}_i$

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 $\mathbf{r}_i, \mathbf{y}_i$
- $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{y}_i$
- 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 $[\![sk]\!]_i$

- 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 $\mathbf{r}_i, \mathbf{y}_i$
- $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{y}_i$
- 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. Can we do more?

Can we do more?

Further problems to solve:

- Detecting malicious behaviour in ThRaccoon
  - The use of zero-shares during signing prevents partial verification as in the classical setting.
  - Can’t use NIZK either as PRF are inefficient to prove.

- Can we really not distribute rejection sampling?

Let’s generalize our solution for short vector sampling

Idea: sample T short vectors, and sum them

- Party i samples short $\mathbf{r}_i$
- Define $\mathbf{r} = \sum \mathbf{r}_i$

What we did: perform the complex operation locally, then aggregate the results

Let’s generalize our solution for short vector sampling

What we did: perform the complex operation locally, then aggregate the results

Can we

- In ThRaccoon, apply this to the entire signature computation to remove the use of zero shares?

Let’s generalize our solution for short vector sampling

What we did: perform the complex operation locally, then aggregate the results

Can we

- In ThRaccoon, apply this to the entire signature computation to remove the use of zero shares?

  Answer: Yes, but only if we possess a short partial secret for party i.

Let’s generalize our solution for short vector sampling

What we did: perform the complex operation locally, then aggregate the results

Can we

- In ThRaccoon, apply this to the entire signature computation to remove the use of zero shares?

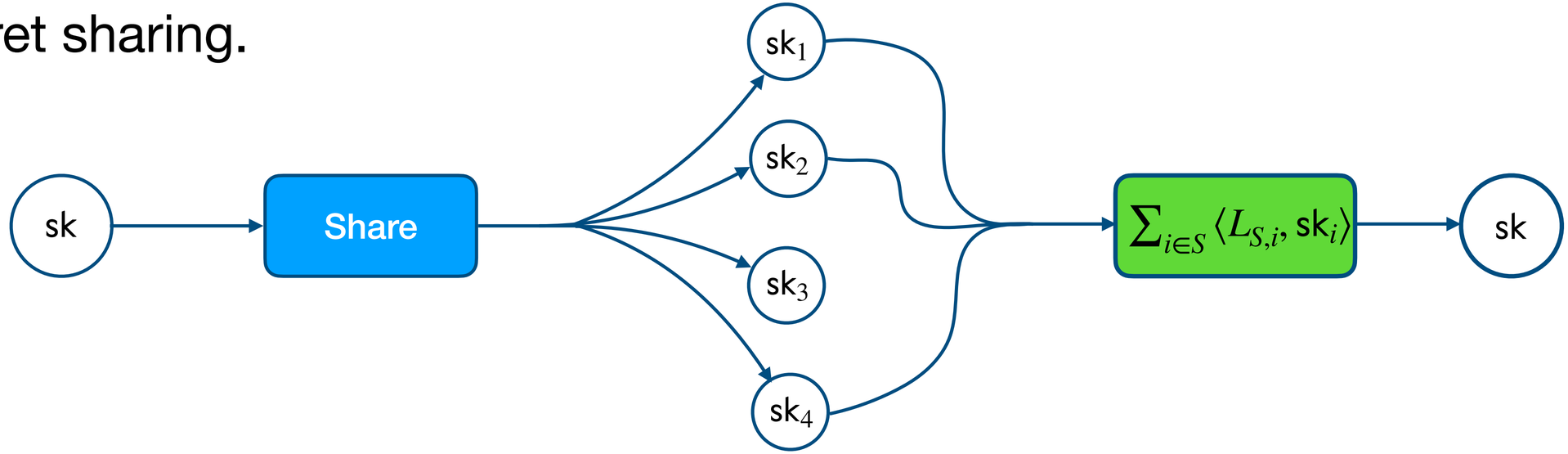
  Answer: Yes, but only if we possess a short partial secret for party i.

- Apply this to rejection sampling?

  Answer: Yes, but again only if we possess a short partial secret.

A new class of secret sharings

Short secret sharing.



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

ThRaccoon 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

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]

ThRaccoon 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

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 $\mathsf{sk}_i = (\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}, \dots)$

  → 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.*

Threshold ML-DSA-like

ML-DSA . Sign(sk, msg) → sig

- Sample a short $\mathbf{r}$
- $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$
- $\mathbf{w}_\top = \lfloor \mathbf{w} \rceil_\nu$
- $c = H(\mathbf{w}_\top, \text{msg})$
- $\mathbf{z} = c \cdot \text{sk} + \mathbf{r}$
- If $\mathbf{z}$ not in S, **restart**
- $\mathbf{h} = \mathbf{w}_\top - \lfloor \mathbf{A}\mathbf{z} - c \cdot \text{vk} \rceil_\nu$
- Output $\text{sig} = (c, \mathbf{z}, \mathbf{h})$

Finally! A Compact Lattice-Based Threshold Signature

Rafael del Pino[1] and Guilhem Niot[1,2]

Finally . 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:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \text{msg})$
- $\mathbf{z}_i = c \cdot \langle L_{S,i}, \text{sk}_i \rangle + \mathbf{r}_i$
- If $\mathbf{z}_i$ not in S, **restart**

Combine: the final signature is

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

Threshold ML-DSA-like

For $N \leq 8$,

| vk |	| sig |
2.6 kB	2.6 kB

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

Finally! A Compact Lattice-Based Threshold Signature

Rafael del Pino[1] and Guilhem Niot[1,2]

Finally . 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:

- $\mathbf{w} = \sum_i \mathbf{w}_i$
- $c = H(\mathbf{w}, \mathsf{msg})$
- $\mathbf{z}_i = c \cdot \langle L_{S,i}, \mathsf{sk}_i \rangle + \mathbf{r}_i$
- If $\mathbf{z}_i$ not in S, **restart**

Combine: the final signature is

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

4. How to concretely sample short sharings

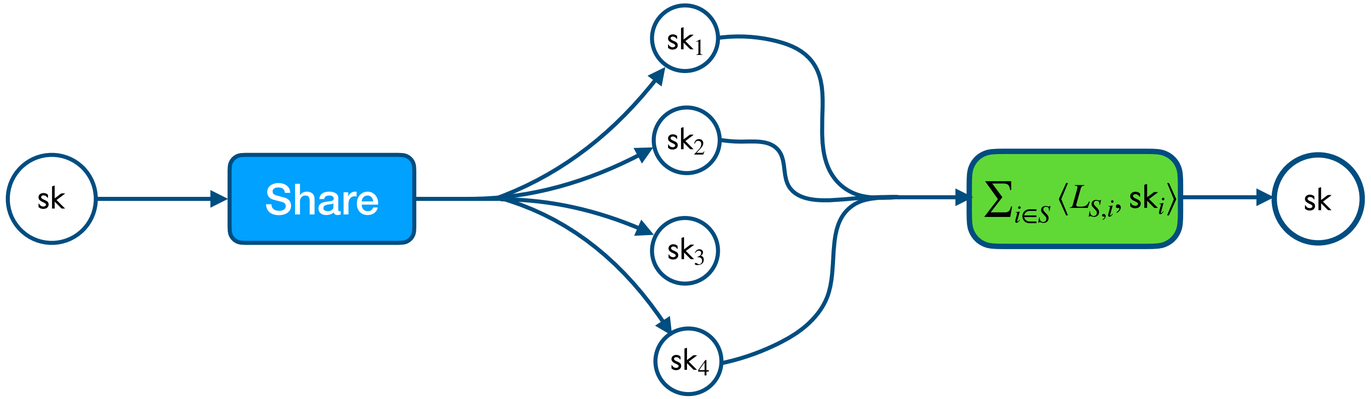
How to Shortly Share a Short Vector

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]

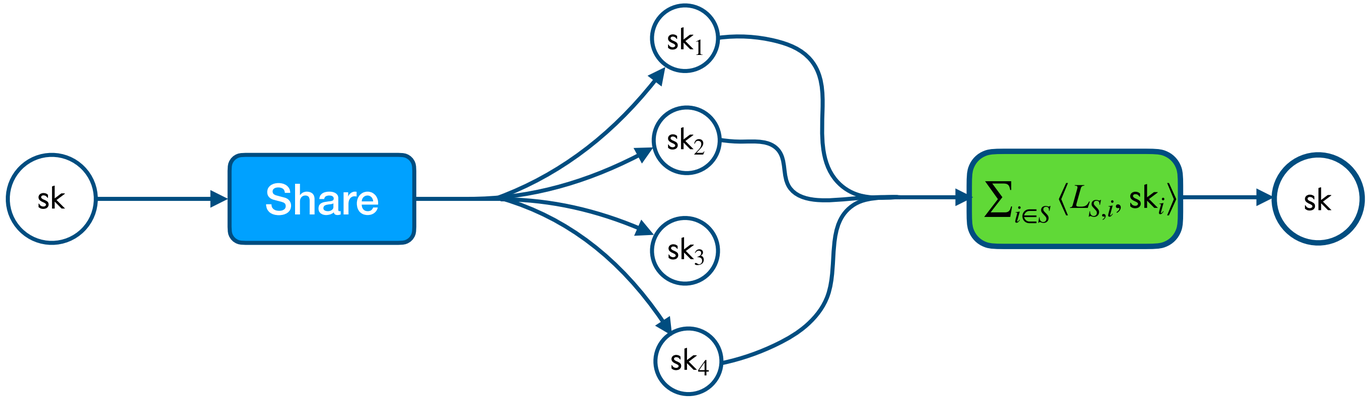
Short Secret Sharing

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

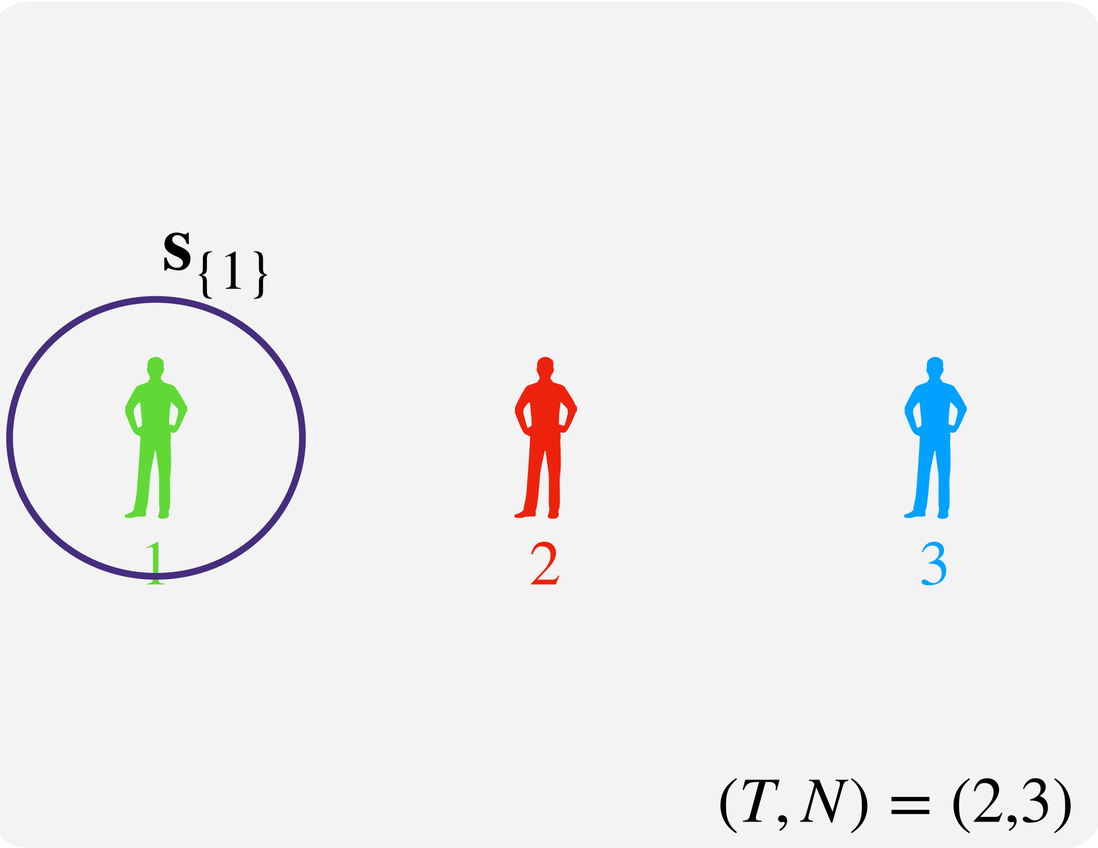
- Leak an offset of the secret: $\mathsf{sk} = \mathsf{sk}_{\mathsf{safe}} + \mathsf{sk}_{\mathsf{leak}}$
- Leak hints on the secrets $h = c \cdot \mathsf{sk} + y$, for large enough y

$\rightarrow$ We just need $[\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{sk}$ to look uniform

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

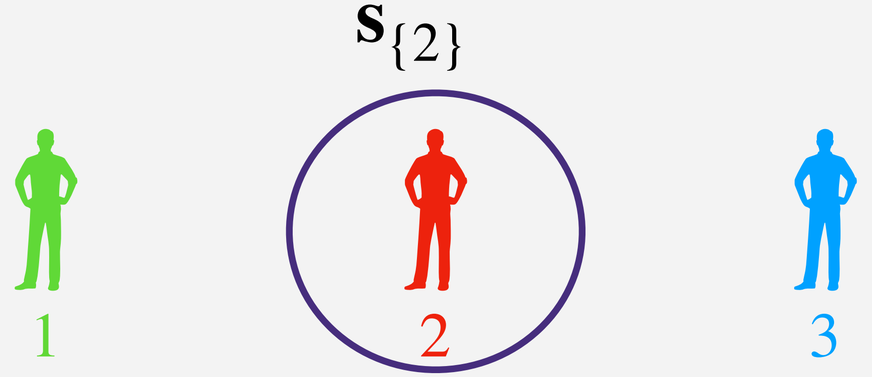


Solution 1: Replicated Secret Sharing

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

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



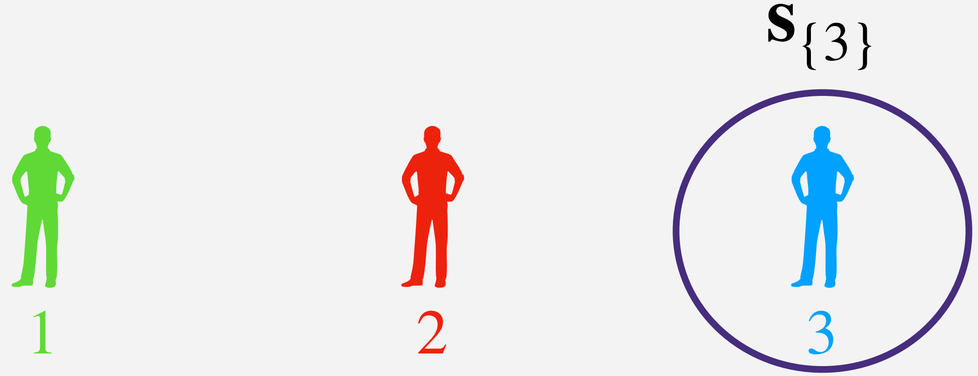
$(T, N) = (2,3)$

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

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

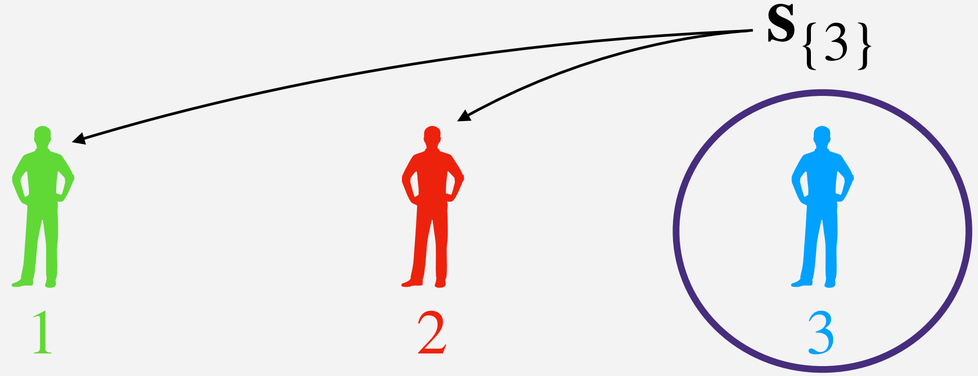


$(T, N) = (2,3)$

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 $\mathbf{s}_{\mathcal{T}}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to the parties in $[N]\backslash\mathcal{T}$.

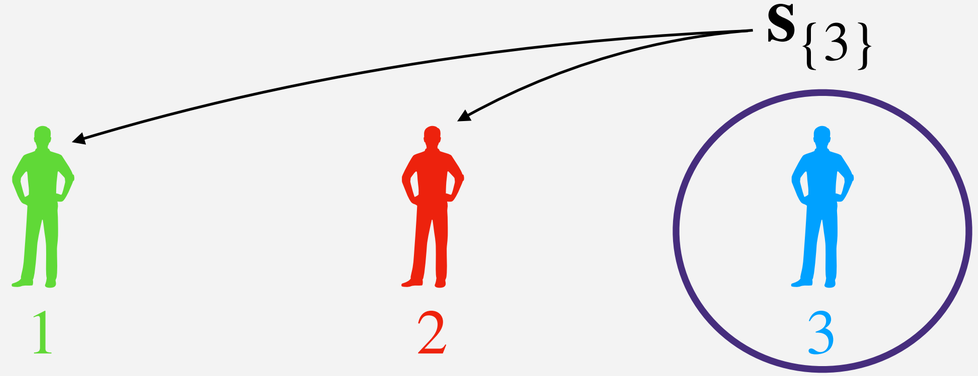


$(T, N) = (2,3)$

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 $\mathbf{s}_{\mathcal{T}}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to the parties in $[N]\backslash\mathcal{T}$.
3. Define $\mathsf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.



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

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 $\mathbf{s}_{\mathcal{T}}$.
2. Distribute $\mathbf{s}_{\mathcal{T}}$ to the parties in $[N]\backslash\mathcal{T}$.
3. Define $\mathsf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.

Properties:

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

  $\rightarrow$ guarantees that 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 $\mathsf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.

Properties:

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

  $\rightarrow$ guarantees that $[\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{sk}$ 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 $\mathcal{T}$ sample a short
2. Distribute $\mathbf{s}_{\mathcal{T}}$ t $[N]\backslash\mathcal{T}$.
3. Define $\mathsf{sk} = \sum_{\mathcal{T}} \mathbf{s}_{\mathcal{T}}$.

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

$\rightarrow$ guarantees that $[\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{sk}$ 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

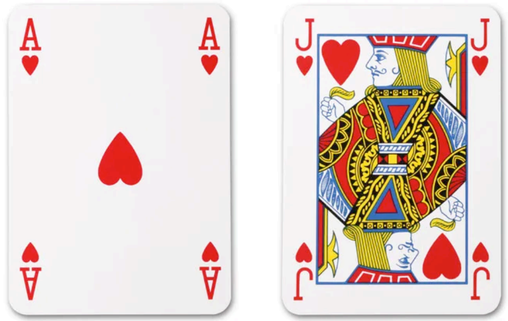
1

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

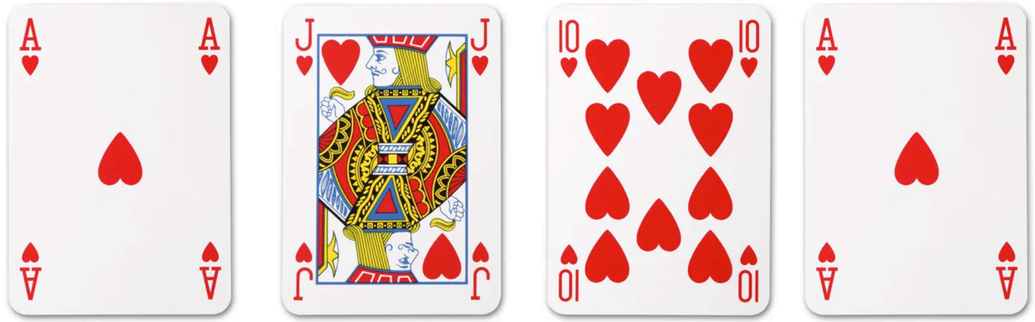
1 2 3

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

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

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

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

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)

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

N shares

Example:

- $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1} \leftarrow \mathscr{D}_\sigma^{N-1}$ and

Idea: Randomly distribute one share per party.

Caveat: This secret sharing is ramp (gap between correctness and privacy thresholds) and only holds in a selective security model.

Desired p

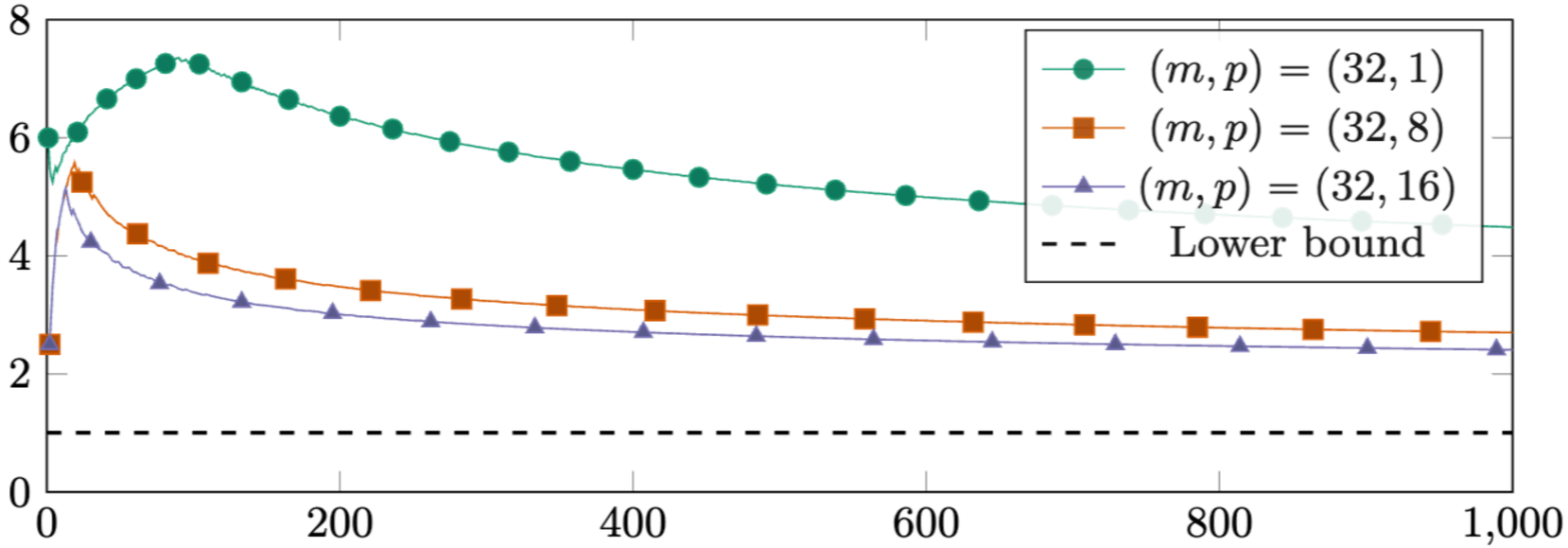
- **Recons** …to gather all the sh…
- **Security threshold:** Maximum number of parties T' such that at least one share is not known (with overwhelming probability)

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

It is possible to amplify the properties for a lower gap. m, p are amplification parameters.



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

Solution 3: Vandermonde sharing

Vandermonde's identity

For $0 \leq c \leq N$:

$$\binom{N}{T} = \sum_{k=0}^{T} \binom{c}{k} \cdot \binom{N-c}{T-k} \tag{6}$$

Distribution theory interpretation: The sum of two binomials is a binomial:

$$B(m,p) + B(n,p) \sim B(m+n,p) \tag{7}$$

Set theory interpretation: let us note $\mathcal{S}_{S,T}$ the subsets of S of cardinality T.

- Any subset $\mathtt{act} \in \mathcal{S}_{\{1,\dots,N\},T}$ can be decomposed uniquely as:

$$\mathtt{act} = \mathtt{act}_L \sqcup \mathtt{act}_R, \quad \text{where} \begin{cases} \mathtt{act}_L \subseteq \{1,\dots,c\} \\ \mathtt{act}_R \subseteq \{c+1,\dots,N\} \end{cases} \tag{8}$$

- Eq. (6) follows from enumerating these decompositions.

Vandermonde secret sharing [DDB95] turns this into a secret sharing:

- Enumerating all the possible disjunctions of the form in Eq. (8)
- For each disjunction, share the secret in two
  - Recursively share the first half across members of $\mathtt{act}_L$
  - Recursively share the second half across members of $\mathtt{act}_L$

Solution 3: Vandermonde sharing

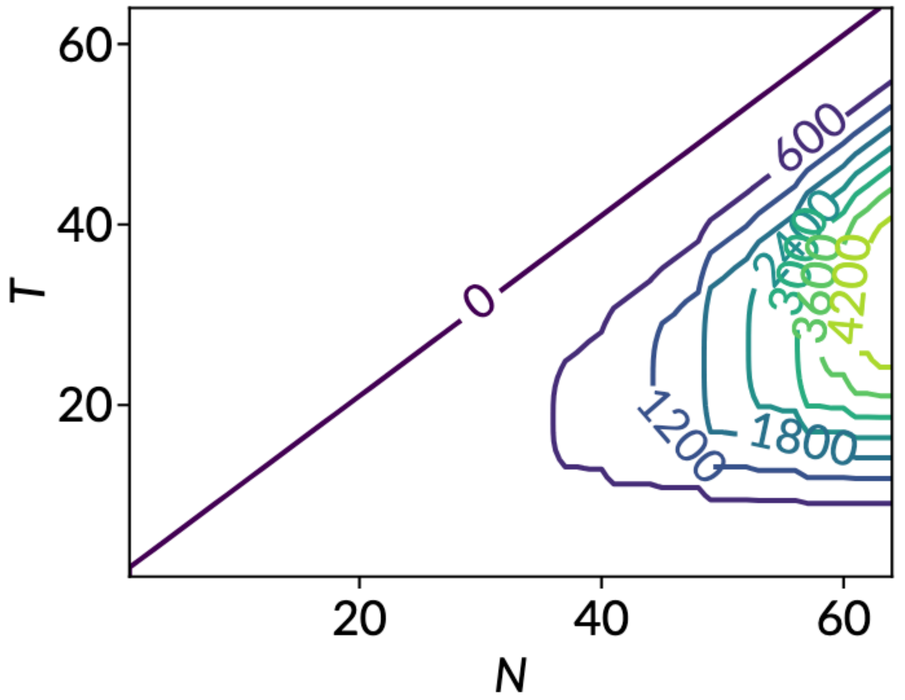
Algorithm 1 Share$(x, \mathcal{P}, T, \texttt{idx} = (T)) \to \texttt{Dict}$

$N = |\mathcal{P}|$
if $T = 1$ **then**
  return $\texttt{Dict} \coloneqq \{user : \{\texttt{idx} : x\} \mid user \in \mathcal{P}\}$
else
  $\texttt{Dict} = \{user : \{:\} \mid user \in \mathcal{P}\}$. $c = \lfloor N/2 \rfloor$
  Parse $\mathcal{P} = \mathcal{P}_L \sqcup \mathcal{P}_R$, with $\mathcal{P}_L$ the c smallest elements of $\mathcal{P}$
  for $k = \max(0, T - N + c), \ldots, \min(c, T)$ **do**
    $\texttt{idx}_L \coloneqq (\texttt{idx}, k)$
    $\texttt{idx}_R \coloneqq (\texttt{idx}, T - k)$
    if $k = 0$ **then**
      $\texttt{Dict} \coloneqq \texttt{Dict} \cup \text{Share}(x, \mathcal{P}_R, T, \texttt{idx}_R)$
    else if $k = T$ **then**
      $\texttt{Dict} \coloneqq \texttt{Dict} \cup \text{Share}(x, \mathcal{P}_L, T, \texttt{idx}_L)$
    else
      $x_0 \leftarrow \chi$
      $x_1 \coloneqq (x - x_0) \bmod q$
      $\texttt{Dict}_L \coloneqq \text{Share}(x, \mathcal{P}_L, k, \texttt{idx}_L)$
      $\texttt{Dict}_R \coloneqq \text{Share}(x, \mathcal{P}_R, T - k, \texttt{idx}_R)$
      $\texttt{Dict} \coloneqq \texttt{Dict} \cup \texttt{Dict}_L \cup \texttt{Dict}_R$
  return Dict

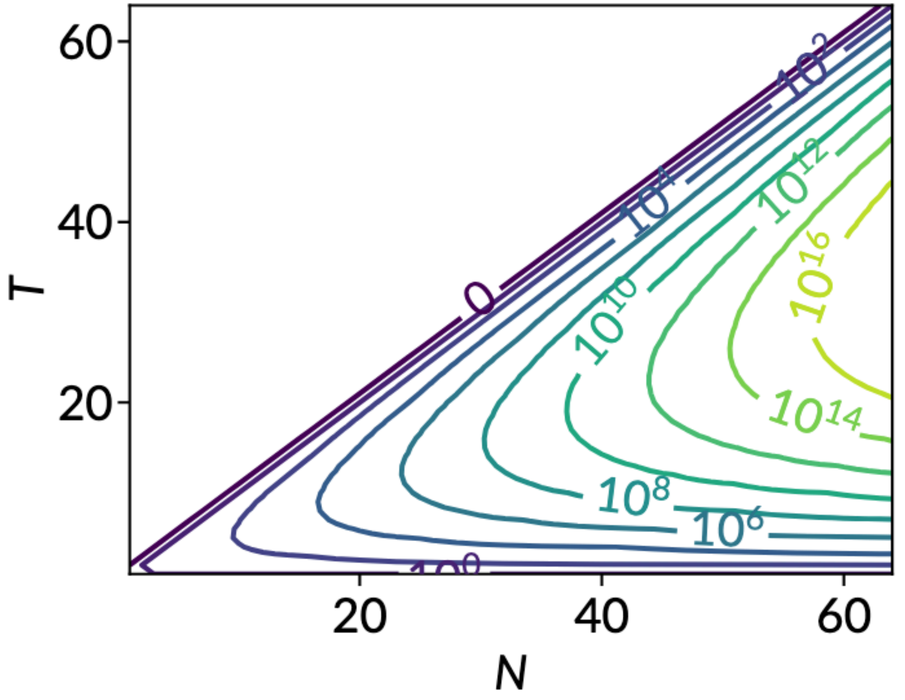
Algorithm 2 Recover$(\mathcal{P}, \texttt{act}, \texttt{idx} = (T)) \to \texttt{Dict}$

$N = |\mathcal{P}|, T = |\texttt{act}|$
if $T = 1$ **then**
  return $\texttt{Dict} \coloneqq \{user : \texttt{idx} \mid user \in \mathcal{P}\}$
else
  $c = \lfloor N/2 \rfloor$. Parse $\mathcal{P} = \mathcal{P}_L \sqcup \mathcal{P}_R$, with $\mathcal{P}_L$ the c smallest elements of $\mathcal{P}$
  $k = |\mathcal{P}_L|, \texttt{act}_L = \texttt{act} \cap \mathcal{P}_L, \texttt{act}_R = \texttt{act} \cap \mathcal{P}_R$
  $\texttt{idx}_L \coloneqq (\texttt{idx}, k)$
  $\texttt{idx}_R \coloneqq (\texttt{idx}, T - k)$
  if $k = 0$ **then**
    return $\text{Recover}(\mathcal{P}_R, \texttt{act}_R, \texttt{idx}_R)$
  else if $k = T$ **then**
    return $\text{Recover}(\mathcal{P}_L, \texttt{act}_L, \texttt{idx}_L)$
  else
    $\texttt{Dict}_L \coloneqq \text{Recover}(\mathcal{P}_L, \texttt{act}_L, \texttt{idx}_L)$
    $\texttt{Dict}_R \coloneqq \text{Recover}(\mathcal{P}_R, \texttt{act}_R, \texttt{idx}_R)$
    return $\texttt{Dict} \coloneqq \texttt{Dict}_L \sqcup \texttt{Dict}_R$

Solution 3: Vandermonde sharing



(a) **Vandermonde:** $O((N/\log N)^{\log N})$ shares/party



(b) **Replicated:** up to $\binom{N-1}{N-T} \approx 2^N$ shares/party

Contour plots of the number of shares/party, as a function of N and T.

DKGs

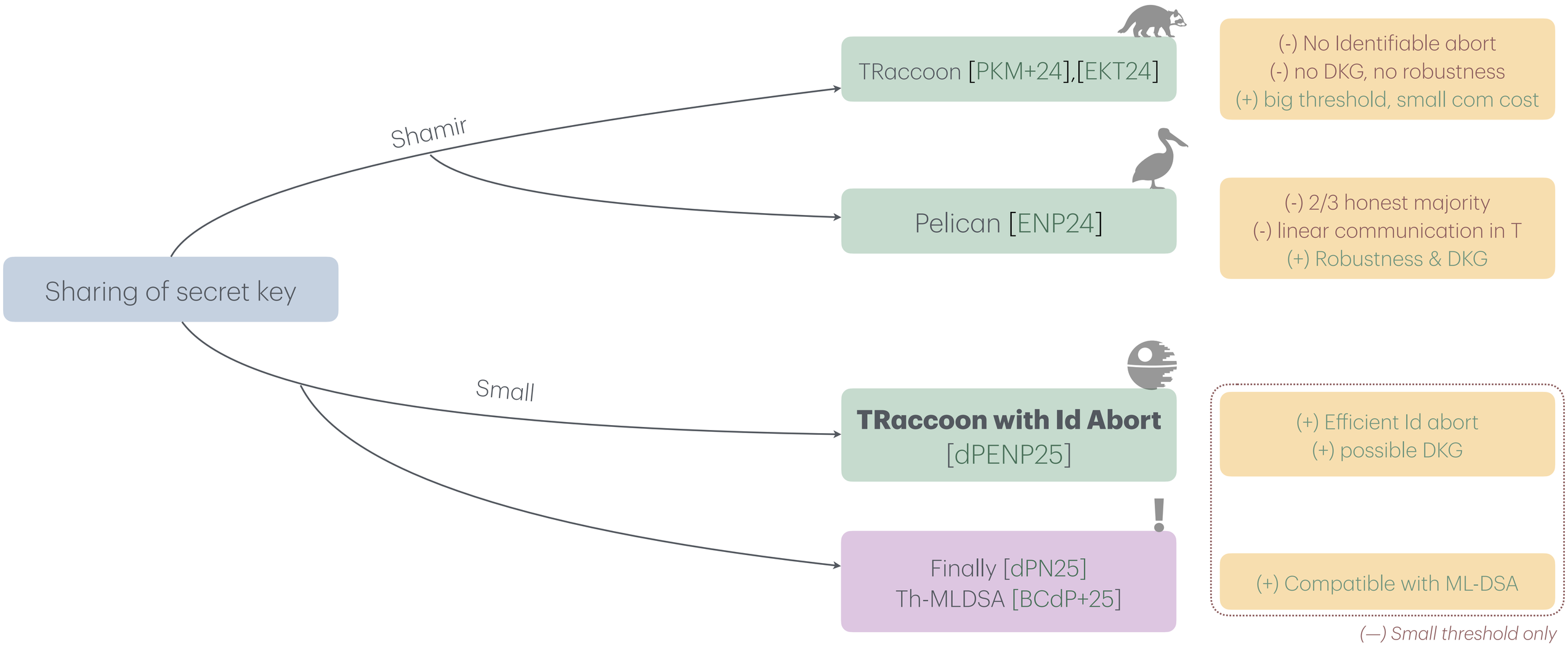
The short secret sharings presented here can be sampled distributively.

Conclusion

Conclusion

Scheme	Shares/party	$\frac{T_{\text{correctness}}}{T_{\text{privacy}}}$
Shamir	1	1
Replicated	2^N	1
Coupon collector	$m \cdot p$	$1 + O\left(\frac{\lambda/m + \ln \lambda}{\ln(np)}\right)$
Vandermonde	$O\left(\left(\frac{N}{\log N}\right)^{\log N}\right)$	1

Shamir vs tailored secret sharing in lattice-based threshold cryptography



Conclusion

- **We can use Shamir sharing to distribute the Raccoon signature scheme efficiently**
- **Take-away: we can tailor secret sharings to lattice-based constructions for more properties**
  - Shortness requirements are natural in this setting
  - We can weaken privacy to only pseudo-uniformity of the public key
  - Our instantiations:
    - Replicated secret sharing (up to 16 parties)
    - Vandermonde sharing (up to 64 parties)
    - Coupon collector problem: scales to larger thresholds, but has a gap between privacy and correctness thresholds
- Applications:
  - DKG + Identifiable aborts in Threshold Raccoon (using partial verification keys)
  - A compact threshold ML-DSA-like signature scheme for $N \leq 8$

Questions?