Lattice-Based Threshold Signatures: Into The Raccoonverse

Thomas Prest (joint work w/ PQShield & friends)

PQSHIELD

November 6, 2024

Lattice Signatures

Lattice signatures

Easier to thresholdize ↓

	Hash-&-Sign	Fiat-Shamir
Convolution	Eagle [YJW23]	G+G [DPS23]
Rejection sampling	Phoenix [JRS24]	Dilithium [LDK+22]
Noise flooding	Plover [EEN+24]	Raccoon [dEK+23]

↑ More compact

Lattice signatures

PQ SHIELD

Easier to thresholdize ↓

	Hash-&-Sign	Fiat-Shamir
Convolution	Eagle [YJW23]	G+G [DPS23]
Rejection sampling	Phoenix [JRS24]	Dilithium [LDK+22]
Noise flooding	Plover [EEN+24]	Raccoon [dEK+23]

↑ More compact

This talk: focus on Raccoon 🦝

→ Masking-friendly [dPKPR24] and threshold-friendly [DKM+24]

→ NIST PQC candidate [dEK+23], 2023-2024 (RIP in peace ⚰️)

→ Similar design also found in [ASY22, GKS24]

Raccoon: Schnorr over lattices

PQ SHIELD

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

1. $\mathtt{vk} = [\mathbf{A} \quad 1] \cdot \mathtt{sk}$, for $\mathtt{sk}$ short.

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

1. Sample a short $\mathbf{r}$
2. $\mathbf{w} = [\mathbf{A} \quad 1] \cdot \mathbf{r}$
3. $c = H(\mathbf{w}, \mathtt{msg})$
4. $\mathbf{z} = \mathbf{r} + c \cdot \mathtt{sk}$
5. Output $\mathtt{sig} = (c, \mathbf{z})$

Raccoon.Verify($\mathtt{vk}, \mathtt{msg}, \mathtt{sig}$)

1. $\mathbf{w}' = [\mathbf{A} \quad 1] \cdot \mathbf{z} - c \cdot \mathtt{vk}$
2. Assert $H(\mathbf{w}', \mathtt{msg}) = c$
3. Assert $\mathbf{z}$ is short

Schnorr.Keygen() $\to \mathtt{sk}, \mathtt{vk}$

1. $\mathtt{vk} = g^{\mathtt{sk}}$, for $\mathtt{sk}$ uniform.

Schnorr.Sign($\mathtt{sk}, \mathtt{msg}$) $\to \mathtt{sig}$

1. Sample r
2. $w = g^r$
3. $c = H(w, \mathtt{msg})$
4. $z = r + c \cdot \mathtt{sk}$
5. Output $\mathtt{sig} = (c, z)$

Schnorr.Verify($\mathtt{vk}, \mathtt{msg}, \mathtt{sig}$)

1. $w' = g^z \cdot \mathtt{vk}^{-c}$
2. Assert $H(\mathbf{w}', \mathtt{msg}) = c$

Security of Raccoon

PQ SHIELD

Raccoon.Keygen() $\to$ sk, vk

1. $\mathsf{vk} = [\mathbf{A} \quad 1] \cdot \mathsf{sk}$, for sk short.

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

1. Sample a short $\mathbf{r}$
2. $\mathbf{w} = [\mathbf{A} \quad 1] \cdot \mathbf{r}$
3. $c = H(\mathbf{w}, \mathsf{msg})$
4. $\mathbf{z} = \mathbf{r} + c \cdot \mathsf{sk}$
5. Output $\mathsf{sig} = (c, \mathbf{z})$

Raccoon.Verify(vk, msg, sig)

1. $\mathbf{w}' = [\mathbf{A} \quad 1] \cdot \mathbf{z} - c \cdot \mathsf{vk}$
2. Assert $H(\mathbf{w}', \mathsf{msg}) = c$
3. Assert $\mathbf{z}$ is short

Raccoon is EUF-CMA assuming:

1. **Hint-MLWE [KLSS23]**
2. **Self-target MSIS [KLS18]**

Hint-MLWE assumption

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

$$(c_i, z_i = \mathbf{r}_i + c_i \cdot \mathsf{sk}), \quad i \in [Q] \tag{1}$$

Note. Hint-MLWE $\geq$ MLWE$_\sigma$ if:

$$\sigma_\mathbf{r} \geq \|c\| \cdot \sqrt{Q} \cdot \sigma \tag{2}$$

Threshold Cryptography

Why threshold cryptography?

Devices can be **compromised** by...

- 🏴‍☠️ Malwares
- 🏴‍☠️ Zero-day exploits
- 🏴‍☠️ Human error
- 🏴‍☠️ ...

Devices can be made **out of order** by...

- ⚰️ Network or energy failure
- ⚰️ Attack on the infrastructure
- ⚰️ Destruction
- ⚰️ ...

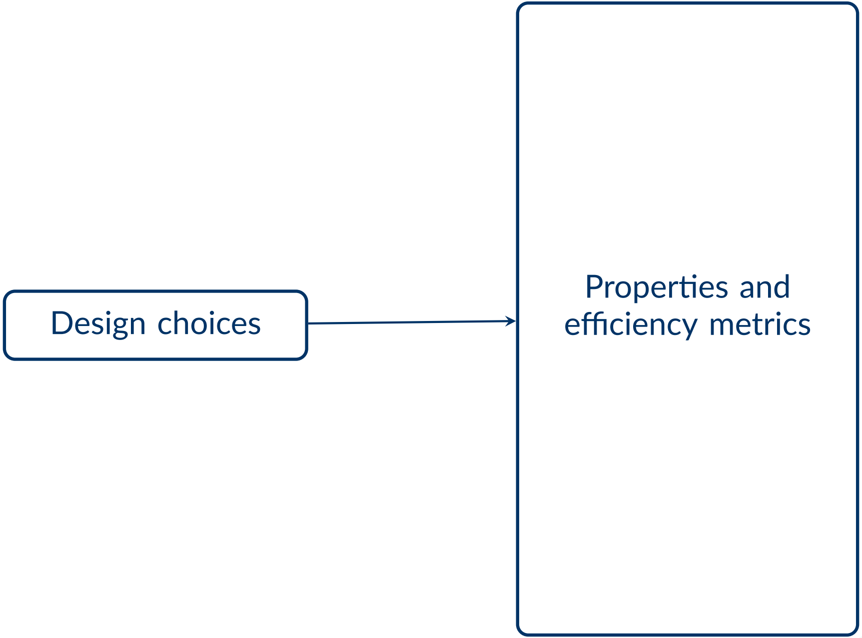
The solution is redundancy

Key idea: distribute trust across several devices

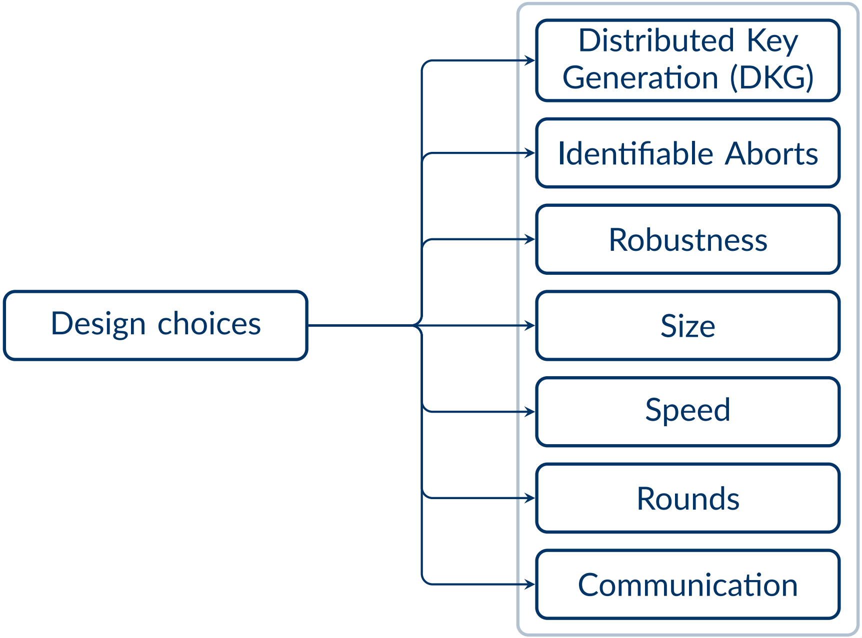
		☠ Attacker: how many devices to compromise?	⚰ Attacker: how many devices to destroy?
1 device	1 key	1 / 1	1 / 1
N devices	1 key	1 / N	N / N
N devices	**N** keys	N / N	1 / N
N devices	**T-out-of-N** keys	T / N	(N - T + 1) / N

- → The two last solutions fall under **threshold cryptography**
- → Main focus of the NIST MPTC programme (see Luis' talk tomorrow)
- → Reminiscent of masking, but key differences in the attack model and properties

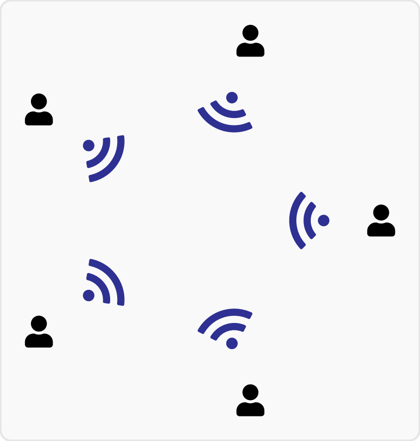
How design choices impact properties



How design choices impact properties



Model



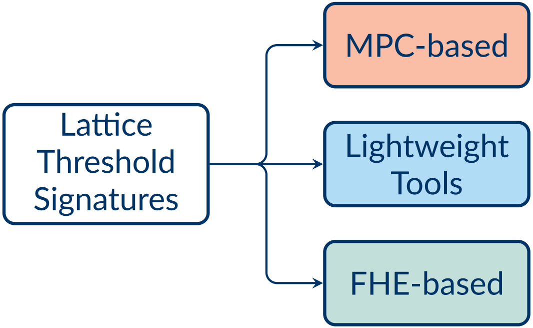
Communication

- Authenticated, reliable & synchronous broadcast channel
- Each *i* and *j* may share an authenticated private channel (via AEAD)

Syntax

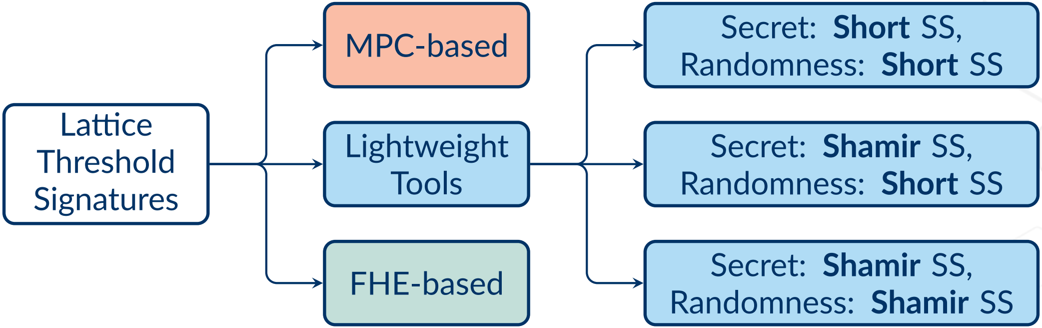
- One public key $\texttt{vk}$
- Each user *i* has a secret key share $\texttt{sk}_i$
- Signing is an interactive protocol between $|\mathcal{S}|$ signers
  - Our protocols are 3-4 rounds
  - $(|\mathcal{S}| < T) \Rightarrow \bot$
  - $(|\mathcal{S}| = T) \Rightarrow \texttt{sig}$ a valid signature

Design choices



Paradigm	Size	Speed	Rounds	Comm/party
MPC	S	Slow	15	≥ 1000 KB
Lightweight	S-M	Fast	2-4	$20 \rightarrow 56 \cdot T$ KB
FHE	M	As fast as FHE	2	≥ 1000 KB

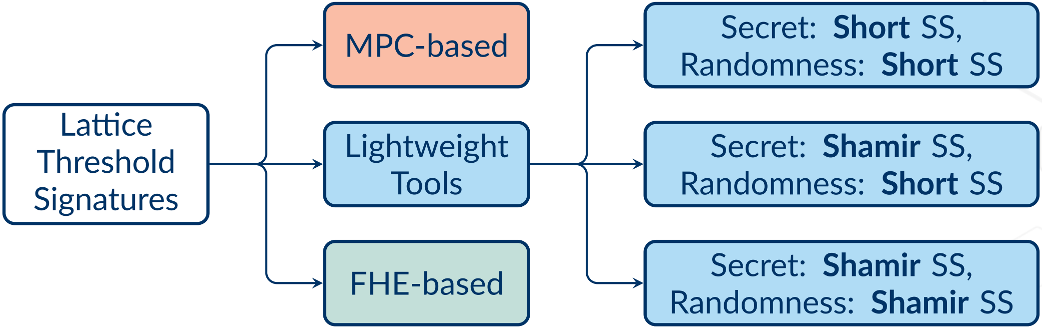
Design choices



Paradigm	Size	Speed	Rounds	Comm/party
MPC	S	Slow	15	$\geq$ 1000 KB
Lightweight	S-M	Fast	2-4	$20 \rightarrow 56 \cdot T$ KB
FHE	M	As fast as FHE	2	$\geq$ 1000 KB

Design choices

PQ SHIELD

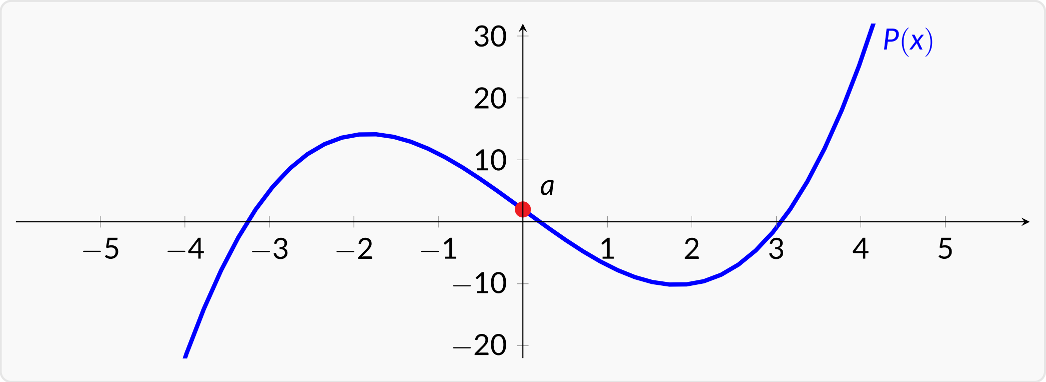


Paradigm	**Size**	**Speed**	**Rounds**	**Comm/party**
MPC	S	Slow	15	$\geq$ 1000 KB
Lightweight	S-M	Fast	2-4	$20 \to 56 \cdot T$ KB
FHE	M	As fast as FHE	2	$\geq$ 1000 KB

This talk.

Threshold Raccoon

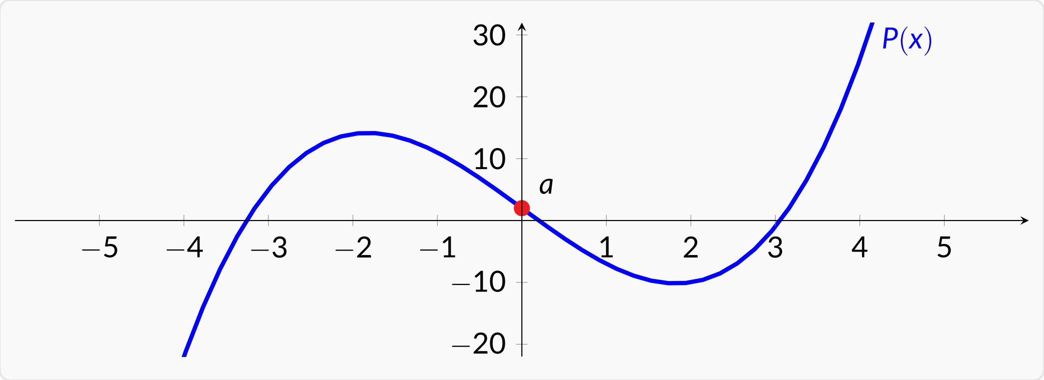
Shamir secret sharing



Secret-sharing a secret $a \in \mathbb{Z}_p$:

- Generate $P(x)$ of degree at most $T - 1$ such that $P(0) = a$
- Each party $i \in \mathbb{Z}_p$ receives a share $a_i P(i)$

Shamir secret sharing



Properties:

- With $< T$ shares, a is perfectly hidden
- With a set $\mathcal{S}$ of T shares, a can be recovered via Lagrange interpolation:

$$a = \sum_{i \in \mathcal{S}} \lambda_{i,\mathcal{S}} \cdot a_i, \quad \text{where} \quad \lambda_{i,\mathcal{S}} = \prod_{j \in \mathcal{S} \setminus \{i\}} \frac{j}{i-j} \tag{3}$$

Threshold Schnorr signatures

Sparkle

Each signer i knows a share sk_i of sk.

- **Round 1:**
  1. Sample r_i
  2. $w_i = g^{r_i}$
  3. $\mathtt{com}_i = H_{\mathtt{com}}(w_i, \mathtt{msg}, \mathcal{S})$
  4. Broadcast $\mathtt{com}_i$
- **Round 2:**
  1. Broadcast w_i
- **Round 3:**
  1. $w = \prod_i w_i$
  2. $c = H(\mathsf{vk}, \mathtt{msg}, w)$
  3. $z_i = r_i + c \cdot \lambda_{i,\mathcal{S}} \cdot \mathsf{sk}_i$
  4. Broadcast z_i
- **Combine:** the final signature is $(c, z = \sum_{i \in \mathcal{S}} z_i)$

See [BN06, CKM23]

This produces valid Schnorr signatures:

$$\begin{aligned} g^z &= g^{\sum_i z_i} \\ &= \left(g^{\sum_i r_i}\right) \cdot \left(g^{c \sum_i \lambda_{i,\mathcal{S}} \cdot \mathsf{sk}_i}\right) \\ &= w \cdot \mathsf{vk}^c \end{aligned}$$

Security: in z_i, r_i is uniform and perfectly hides $c \cdot \lambda_{i,\mathcal{S}} \cdot \mathsf{sk}_i$

We commit to w_i before revealing it to avoid ROS attacks [DEF+19, BLL+22]

Can we transpose this to Raccoon?

First attempt

PQ SHIELD

Insecure Threshold Raccoon

- **Round 1:**
  1. Sample short $\mathbf{r}_i$
  2. $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
  3. $\mathtt{com}_i = H_{\mathtt{com}}(\mathbf{w}_i, \mathtt{msg}, \mathcal{S})$
  4. Broadcast $\mathtt{com}_i$
- **Round 2:**
  1. Broadcast $\mathbf{w}_i$
- **Round 3:**
  1. $\mathbf{w} = \sum_i \mathbf{w}_i$
  2. $c = H(\mathsf{vk}, \mathtt{msg}, \mathbf{w})$
  3. $\mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_i \cdot \mathsf{sk}_i$
  4. Broadcast $\mathbf{z}_i$
- **Combine:** the final signature is $(c, \mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i)$

✓ This gives valid Raccoon signatures (up to slight parameter changes)

⚠ Issue: when we consider

$$\mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_i \cdot \mathsf{sk}_i, \qquad (4)$$

$\mathbf{r}_i$ is small whereas $c \cdot \lambda_i \cdot \mathsf{sk}_i$ is large.

- Breaks the security proof
- For a fixed i, with enough $\mathbf{z}_i$ of the form in (4) one can recover sk_i

This is the crossroads of the talk

? Can we add to each $\mathbf{z}$ a value Δ_i such that:

1. Any set of $< T$ values Δ_i is uniformy random?
2. $\sum_{i \in \mathcal{S}} \Delta_i = \mathbf{0}$?

Lets call $(\Delta_i)_{i \in \mathcal{S}}$ a zero-share.

Building a Zero-Share

	1	2	3	4	5
1	$\mathbf{m}_{1,1}$	$\mathbf{m}_{1,2}$	$\mathbf{m}_{1,3}$	$\mathbf{m}_{1,4}$	$\mathbf{m}_{1,5}$
2	$\mathbf{m}_{2,1}$	$\mathbf{m}_{2,2}$	$\mathbf{m}_{2,3}$	$\mathbf{m}_{2,4}$	$\mathbf{m}_{2,5}$
3	$\mathbf{m}_{3,1}$	$\mathbf{m}_{3,2}$	$\mathbf{m}_{3,3}$	$\mathbf{m}_{3,4}$	$\mathbf{m}_{3,5}$
4	$\mathbf{m}_{4,1}$	$\mathbf{m}_{4,2}$	$\mathbf{m}_{4,3}$	$\mathbf{m}_{4,4}$	$\mathbf{m}_{4,5}$
5	$\mathbf{m}_{5,1}$	$\mathbf{m}_{5,2}$	$\mathbf{m}_{5,3}$	$\mathbf{m}_{5,4}$	$\mathbf{m}_{5,5}$

- Users i and j share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

$$
\begin{array}{c|ccccccccccc}
 & 1 & & 2 & & 3 & & 4 & & 5 & & \\
\hline
1 & \mathbf{m}_{1,1} & + & \mathbf{m}_{1,2} & + & \mathbf{m}_{1,3} & + & \mathbf{m}_{1,4} & + & \mathbf{m}_{1,5} & = & \mathbf{m}_1 \\
 & + & & + & & + & & + & & + & & + \\
2 & \mathbf{m}_{2,1} & + & \mathbf{m}_{2,2} & + & \mathbf{m}_{2,3} & + & \mathbf{m}_{2,4} & + & \mathbf{m}_{2,5} & = & \mathbf{m}_2 \\
 & + & & + & & + & & + & & + & & + \\
3 & \mathbf{m}_{3,1} & + & \mathbf{m}_{3,2} & + & \mathbf{m}_{3,3} & + & \mathbf{m}_{3,4} & + & \mathbf{m}_{3,5} & = & \mathbf{m}_3 \\
 & + & & + & & + & & + & & + & & + \\
4 & \mathbf{m}_{4,1} & + & \mathbf{m}_{4,2} & + & \mathbf{m}_{4,3} & + & \mathbf{m}_{4,4} & + & \mathbf{m}_{4,5} & = & \mathbf{m}_4 \\
 & + & & + & & + & & + & & + & & + \\
5 & \mathbf{m}_{5,1} & + & \mathbf{m}_{5,2} & + & \mathbf{m}_{5,3} & + & \mathbf{m}_{5,4} & + & \mathbf{m}_{5,5} & = & \mathbf{m}_5 \\
 & \| & & \| & & \| & & \| & & \| & & \| \\
 & \mathbf{m}_1^* & + & \mathbf{m}_2^* & + & \mathbf{m}_3^* & + & \mathbf{m}_4^* & + & \mathbf{m}_5^* & = & \mathbf{m}
\end{array}
$$

- Users *i* and *j* share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

$$\begin{array}{c|ccccccccccc}
 & 1 & & 2 & & 3 & & 4 & & 5 & & \\
\hline
1 & \mathbf{m}_{1,1} & + & \mathbf{m}_{1,2} & + & \mathbf{m}_{1,3} & + & \mathbf{m}_{1,4} & + & \mathbf{m}_{1,5} & = & \mathbf{m}_1 \\
 & + & & + & & + & & + & & + & & + \\
2 & \mathbf{m}_{2,1} & + & \mathbf{m}_{2,2} & + & \mathbf{m}_{2,3} & + & \mathbf{m}_{2,4} & + & \mathbf{m}_{2,5} & = & \mathbf{m}_2 \\
 & + & & + & & + & & + & & + & & + \\
3 & \mathbf{m}_{3,1} & + & \mathbf{m}_{3,2} & + & \mathbf{m}_{3,3} & + & \mathbf{m}_{3,4} & + & \mathbf{m}_{3,5} & = & \mathbf{m}_3 \\
 & + & & + & & + & & + & & + & & + \\
4 & \mathbf{m}_{4,1} & + & \mathbf{m}_{4,2} & + & \mathbf{m}_{4,3} & + & \mathbf{m}_{4,4} & + & \mathbf{m}_{4,5} & = & \mathbf{m}_4 \\
 & + & & + & & + & & + & & + & & + \\
5 & \mathbf{m}_{5,1} & + & \mathbf{m}_{5,2} & + & \mathbf{m}_{5,3} & + & \mathbf{m}_{5,4} & + & \mathbf{m}_{5,5} & = & \mathbf{m}_5 \\
 & \| & & \| & & \| & & \| & & \| & & \| \\
 & \mathbf{m}_1^* & + & \mathbf{m}_2^* & + & \mathbf{m}_3^* & + & \mathbf{m}_4^* & + & \mathbf{m}_5^* & = & \mathbf{m}
\end{array}$$

- Users i and j share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

$$
\begin{array}{c|ccccccccccc}
 & 1 & & 2 & & 3 & & 4 & & 5 & & \\
1 & \mathbf{m}_{1,1} & + & \mathbf{m}_{1,2} & + & \mathbf{m}_{1,3} & + & \mathbf{m}_{1,4} & + & \mathbf{m}_{1,5} & = & \mathbf{m}_1 \\
 & + & & + & & + & & + & & + & & + \\
2 & \mathbf{m}_{2,1} & + & \mathbf{m}_{2,2} & + & \mathbf{m}_{2,3} & + & \mathbf{m}_{2,4} & + & \mathbf{m}_{2,5} & = & \mathbf{m}_2 \\
 & + & & + & & + & & + & & + & & + \\
3 & \mathbf{m}_{3,1} & + & \mathbf{m}_{3,2} & + & \mathbf{m}_{3,3} & + & \mathbf{m}_{3,4} & + & \mathbf{m}_{3,5} & = & \mathbf{m}_3 \\
 & + & & + & & + & & + & & + & & + \\
4 & \mathbf{m}_{4,1} & + & \mathbf{m}_{4,2} & + & \mathbf{m}_{4,3} & + & \mathbf{m}_{4,4} & + & \mathbf{m}_{4,5} & = & \mathbf{m}_4 \\
 & + & & + & & + & & + & & + & & + \\
5 & \mathbf{m}_{5,1} & + & \mathbf{m}_{5,2} & + & \mathbf{m}_{5,3} & + & \mathbf{m}_{5,4} & + & \mathbf{m}_{5,5} & = & \mathbf{m}_5 \\
 & \parallel & & \parallel & & \parallel & & \parallel & & \parallel & & \parallel \\
 & \mathbf{m}_1^* & + & \mathbf{m}_2^* & + & \mathbf{m}_3^* & + & \mathbf{m}_4^* & + & \mathbf{m}_5^* & = & \mathbf{m}
\end{array}
$$

- Users i and j share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

PQ SHIELD

$$
\begin{array}{c|ccccccccccccc}
 & 1 & & 2 & & 3 & & 4 & & 5 & & \\
\hline
1 & \mathbf{m}_{1,1} & + & \mathbf{m}_{1,2} & + & \mathbf{m}_{1,3} & + & \mathbf{m}_{1,4} & + & \mathbf{m}_{1,5} & = & \mathbf{m}_1 \\
 & + & & + & & + & & + & & + & & + \\
2 & \mathbf{m}_{2,1} & + & \mathbf{m}_{2,2} & + & \mathbf{m}_{2,3} & + & \mathbf{m}_{2,4} & + & \mathbf{m}_{2,5} & = & \mathbf{m}_2 \\
 & + & & + & & + & & + & & + & & + \\
3 & \mathbf{m}_{3,1} & + & \mathbf{m}_{3,2} & + & \mathbf{m}_{3,3} & + & \mathbf{m}_{3,4} & + & \mathbf{m}_{3,5} & = & \mathbf{m}_3 \\
 & + & & + & & + & & + & & + & & + \\
4 & \mathbf{m}_{4,1} & + & \mathbf{m}_{4,2} & + & \mathbf{m}_{4,3} & + & \mathbf{m}_{4,4} & + & \mathbf{m}_{4,5} & = & \mathbf{m}_4 \\
 & + & & + & & + & & + & & + & & + \\
5 & \mathbf{m}_{5,1} & + & \mathbf{m}_{5,2} & + & \mathbf{m}_{5,3} & + & \mathbf{m}_{5,4} & + & \mathbf{m}_{5,5} & = & \mathbf{m}_5 \\
 & \| & & \| & & \| & & \| & & \| & & \| \\
 & \mathbf{m}_1^* & + & \mathbf{m}_2^* & + & \mathbf{m}_3^* & + & \mathbf{m}_4^* & + & \mathbf{m}_5^* & = & \mathbf{m}
\end{array}
$$

- Users *i* and *j* share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

	1		2		3		4		5		
1	$\mathbf{m}_{1,1}$	$+$	$\mathbf{m}_{1,2}$	$+$	$\mathbf{m}_{1,3}$	$+$	$\mathbf{m}_{1,4}$	$+$	$\mathbf{m}_{1,5}$	$=$	$\mathbf{m}_1$
	$+$		$+$		$+$		$+$		$+$		$+$
2	$\mathbf{m}_{2,1}$	$+$	$\mathbf{m}_{2,2}$	$+$	$\mathbf{m}_{2,3}$	$+$	$\mathbf{m}_{2,4}$	$+$	$\mathbf{m}_{2,5}$	$=$	$\mathbf{m}_2$
	$+$		$+$		$+$		$+$		$+$		$+$
3	$\mathbf{m}_{3,1}$	$+$	$\mathbf{m}_{3,2}$	$+$	$\mathbf{m}_{3,3}$	$+$	$\mathbf{m}_{3,4}$	$+$	$\mathbf{m}_{3,5}$	$=$	$\mathbf{m}_3$
	$+$		$+$		$+$		$+$		$+$		$+$
4	$\mathbf{m}_{4,1}$	$+$	$\mathbf{m}_{4,2}$	$+$	$\mathbf{m}_{4,3}$	$+$	$\mathbf{m}_{4,4}$	$+$	$\mathbf{m}_{4,5}$	$=$	$\mathbf{m}_4$
	$+$		$+$		$+$		$+$		$+$		$+$
5	$\mathbf{m}_{5,1}$	$+$	$\mathbf{m}_{5,2}$	$+$	$\mathbf{m}_{5,3}$	$+$	$\mathbf{m}_{5,4}$	$+$	$\mathbf{m}_{5,5}$	$=$	$\mathbf{m}_5$
	$\parallel$		$\parallel$		$\parallel$		$\parallel$		$\parallel$		$\parallel$
	$\mathbf{m}_1^*$	$+$	$\mathbf{m}_2^*$	$+$	$\mathbf{m}_3^*$	$+$	$\mathbf{m}_4^*$	$+$	$\mathbf{m}_5^*$	$=$	$\mathbf{m}$

- Users i and j share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

$$
\begin{array}{c|ccccccccccc}
 & 1 & & 2 & & 3 & & 4 & & 5 & & \\
\hline
1 & \mathbf{m}_{1,1} & + & \mathbf{m}_{1,2} & + & \mathbf{m}_{1,3} & + & \mathbf{m}_{1,4} & + & \mathbf{m}_{1,5} & = & \mathbf{m}_1 \\
 & + & & + & & + & & + & & + & & + \\
2 & \mathbf{m}_{2,1} & + & \mathbf{m}_{2,2} & + & \mathbf{m}_{2,3} & + & \mathbf{m}_{2,4} & + & \mathbf{m}_{2,5} & = & \mathbf{m}_2 \\
 & + & & + & & + & & + & & + & & + \\
3 & \mathbf{m}_{3,1} & + & \mathbf{m}_{3,2} & + & \mathbf{m}_{3,3} & + & \mathbf{m}_{3,4} & + & \mathbf{m}_{3,5} & = & \mathbf{m}_3 \\
 & + & & + & & + & & + & & + & & + \\
4 & \mathbf{m}_{4,1} & + & \mathbf{m}_{4,2} & + & \mathbf{m}_{4,3} & + & \mathbf{m}_{4,4} & + & \mathbf{m}_{4,5} & = & \mathbf{m}_4 \\
 & + & & + & & + & & + & & + & & + \\
5 & \mathbf{m}_{5,1} & + & \mathbf{m}_{5,2} & + & \mathbf{m}_{5,3} & + & \mathbf{m}_{5,4} & + & \mathbf{m}_{5,5} & = & \mathbf{m}_5 \\
 & \parallel & & \parallel & & \parallel & & \parallel & & \parallel & & \parallel \\
 & \mathbf{m}_1^* & + & \mathbf{m}_2^* & + & \mathbf{m}_3^* & + & \mathbf{m}_4^* & + & \mathbf{m}_5^* & = & \mathbf{m}
\end{array}
$$

- Users i and j share a symmetric key $K_{i,j}$, and generate a fresh $\mathbf{m}_{i,j} = PRF(K_{i,j}, sid)$ each signing session
- Each user knows all $\mathbf{m}_{i,j}$'s on their corrresponding row and column

Building a Zero-Share

PQ SHIELD

	1		2		3		4		5		
1	$\mathbf{m}_{1,1}$	$+$	$\mathbf{m}_{1,2}$	$+$	$\mathbf{m}_{1,3}$	$+$	$\mathbf{m}_{1,4}$	$+$	$\mathbf{m}_{1,5}$	$=$	$\mathbf{m}_1$
	$+$		$+$		$+$		$+$		$+$		$+$
2	$\mathbf{m}_{2,1}$	$+$	$\mathbf{m}_{2,2}$	$+$	$\mathbf{m}_{2,3}$	$+$	$\mathbf{m}_{2,4}$	$+$	$\mathbf{m}_{2,5}$	$=$	$\mathbf{m}_2$
	$+$		$+$		$+$		$+$		$+$		$+$
3	$\mathbf{m}_{3,1}$	$+$	$\mathbf{m}_{3,2}$	$+$	$\mathbf{m}_{3,3}$	$+$	$\mathbf{m}_{3,4}$	$+$	$\mathbf{m}_{3,5}$	$=$	$\mathbf{m}_3$
	$+$		$+$		$+$		$+$		$+$		$+$
4	$\mathbf{m}_{4,1}$	$+$	$\mathbf{m}_{4,2}$	$+$	$\mathbf{m}_{4,3}$	$+$	$\mathbf{m}_{4,4}$	$+$	$\mathbf{m}_{4,5}$	$=$	$\mathbf{m}_4$
	$+$		$+$		$+$		$+$		$+$		$+$
5	$\mathbf{m}_{5,1}$	$+$	$\mathbf{m}_{5,2}$	$+$	$\mathbf{m}_{5,3}$	$+$	$\mathbf{m}_{5,4}$	$+$	$\mathbf{m}_{5,5}$	$=$	$\mathbf{m}_5$
	$\parallel$		$\parallel$		$\parallel$		$\parallel$		$\parallel$		$\parallel$
	$\mathbf{m}_1^*$	$+$	$\mathbf{m}_2^*$	$+$	$\mathbf{m}_3^*$	$+$	$\mathbf{m}_4^*$	$+$	$\mathbf{m}_5^*$	$=$	$\mathbf{m}$

- ✓ $(\Delta_1, \ldots, \Delta_T)$, where each $\Delta_i = \mathbf{m}_i - \mathbf{m}_i^*$, is a secret-sharing of $\mathbf{0}$
- For each (i, j), the mask $\mathbf{m}_{i,j}$ remains secret if i and j are not corrupted

Second attempt

PQ SHIELD

Threshold Raccoon

→ **Round 1:**

1. Sample short $\mathbf{r}_i$
2. $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
3. $\mathtt{com}_i = H_{\mathtt{com}}(\mathbf{w}_i, \mathtt{msg}, \mathcal{S})$
4. Broadcast $\mathtt{com}_i$

→ **Round 2:** Broadcast $\mathbf{w}_i$

→ **Round 3:**

1. $\mathbf{w} = \sum_i \mathbf{w}_i$
2. $c = H(\mathsf{vk}, \mathtt{msg}, \mathbf{w})$
3. $\Delta_i = \sum_j (\mathbf{m}_{j,i} - \mathbf{m}_{i,j})$
4. $\mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_i \cdot \mathsf{sk}_i + \Delta_i$
5. Broadcast $\mathbf{z}_i$

→ **Combine:** the final signature is $(c, \mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i)$

✔ This gives valid Raccoon signatures:

$$\begin{aligned} \mathbf{z} &= \sum_{i \in \mathcal{S}} \mathbf{z}_i + \Delta_i \\ &= \sum_{i \in \mathcal{S}} (\mathbf{r}_i + c \cdot \lambda_i \cdot \mathsf{sk}_i + \Delta_i) \\ &= c \cdot \mathsf{sk} + \sum_{i \in \mathcal{S}} \mathbf{r}_i \end{aligned}$$

🔒 This negates the previous attack

Second attempt

PQ SHIELD

Threshold Raccoon

- **Round 1:**
  1. Sample short $\mathbf{r}_i$
  2. $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
  3. $\mathtt{com}_i = H_{\mathtt{com}}(\mathbf{w}_i, \mathtt{msg}, \mathcal{S})$
  4. Broadcast $\mathtt{com}_i$
- **Round 2:** Broadcast $\mathbf{w}_i$ and signature of view of Round 1
- **Round 3:**
  1. $\mathbf{w} = \sum_i \mathbf{w}_i$
  2. $c = H(\mathrm{vk}, \mathtt{msg}, \mathbf{w})$
  3. $\Delta_i = \sum_j (\mathbf{m}_{j,i} - \mathbf{m}_{i,j})$
  4. $\mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_i \cdot \mathrm{sk}_i + \Delta_i$
  5. Broadcast $\mathbf{z}_i$
- **Combine:** the final signature is $(c, \mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i)$

✓ This gives valid Raccoon signatures:

$$\mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i + \Delta_i$$
$$= \sum_{i \in \mathcal{S}} (\mathbf{r}_i + c \cdot \lambda_i \cdot \mathrm{sk}_i + \Delta_i)$$
$$= c \cdot \mathrm{sk} + \sum_{i \in \mathcal{S}} \mathbf{r}_i$$

- This negates the previous attack
- One last thing: we sign the view of Round 1 to avoid a fork attack
  - In [KRT24], the PRF is tweaked so that no signature is needed

Second attempt

PQ SHIELD

Threshold Raccoon

- **Round 1:**
  1. Sample short $\mathbf{r}_i$
  2. $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
  3. $\mathtt{com}_i = H_{\mathtt{com}}(\mathbf{w}_i, \mathtt{msg}, \mathcal{S})$
  4. Broadcast $\mathtt{com}_i$
- **Round 2:** Broadcast $\mathbf{w}_i$ and signature of view of Round 1
- **Round 3:**
  1. $\mathbf{w} = \sum_i \mathbf{w}_i$
  2. $c = H(\mathsf{vk}, \mathtt{msg}, \mathbf{w})$
  3. $\Delta_i = \sum_j (\mathbf{m}_{j,i} - \mathbf{m}_{i,j})$
  4. $\mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_i \cdot \mathsf{sk}_i + \Delta_i$
  5. Broadcast $\mathbf{z}_i$
- **Combine:** the final signature is $(c, \mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i)$

✓ This gives valid Raccoon signatures:

$$\mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i + \Delta_i$$
$$= \sum_{i \in \mathcal{S}} (\mathbf{r}_i + c \cdot \lambda_i \cdot \mathsf{sk}_i + \Delta_i)$$
$$= c \cdot \mathsf{sk} + \sum_{i \in \mathcal{S}} \mathbf{r}_i$$

- This negates the previous attack
- One last thing: we sign the view of Round 1 to avoid a fork attack
  - In [KRT24], the PRF is tweaked so that no signature is needed
- We can prove security under MSIS and Hint-MLWE

Final observations for Threshold Raccoon

- **Sizes:** about 10 KB
- **Speed:** very fast (bottleneck is generating T pseudorandom vectors per user)
- **Rounds:** 3 rounds
  - Reduced to 2 in [EKT24, BKL$^+$24], but communications increases by a factor $\times 10$
- **Communication:** 40 KB per user
- **Distributed key generation:** ?
- **Robustness or IA:** How do we check the computation $PRF(K_{i,j}, sid)$?

Further reading:

- del Pino, Katsumata, Maller, Mouhartem, Prest, Saarinen. *Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions.* EUROCRYPT 2024 [DKM$^+$24]
- Espitau, Katsumata, Takemure. *Two-Round Threshold Signature from Algebraic One-More Learning with Errors.* CRYPTO 2024 [EKT24]
- Katsumata, Reichle, Takemure. *Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding.* CRYPTO 2024 [KRT24]

Flood and Submerse

The key technical challenge is to mask a secret ($\lambda_i \cdot \mathsf{sk}_i$) with the randomness $\mathbf{r}_i$.

1. **Direction 1** (Threshold Raccoon):
   - The shares of the secret are **uniform**
   - The randomness shares $\mathbf{r}_i$ are **short**

   A **uniform** zero-share Δ_i is added to partial signatures in order to hide $\lambda_i \cdot \mathsf{sk}_i$.

2. **Direction 2:** Can we make both $\lambda_i \cdot \mathsf{sk}_i$ and $\mathbf{r}_i$ **uniform**?
   - Use Shamir secret sharing for both sk and $\mathbf{r}$ $\Rightarrow$ This section

3. **Direction 3:** Can we make both $\lambda_i \cdot \mathsf{sk}_i$ and $\mathbf{r}_i$ **short**?
   - Use short secret sharing for both sk and $\mathbf{r}$ $\Rightarrow$ Next section

Shamir Everywhere

Flood and Submerse

- **Round 1:**
  1. Sample short $\mathbf{r}_i$
  2. $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
  3. $\mathtt{com}_i = H_{\mathtt{com}}(\mathbf{w}_i, \mathtt{msg}, \mathcal{S})$
  4. Broadcast $\mathtt{com}_i$
  5. $([\![\mathbf{r}_i]\!]_j)_{j \in [\mathcal{S}]} \leftarrow \mathtt{Shamir.Share}(\mathbf{r}_i)$
  6. Encrypt $[\![\mathbf{r}_i]\!]_j$ to each party j
- **Round 2:** Broadcast $\mathbf{w}_i$
- **Round 3:**
  1. $\mathbf{w} = \sum_i \mathbf{w}_i$
  2. $c = H(\mathtt{vk}, \mathtt{msg}, \mathbf{w})$
  3. $[\![\mathbf{r}]\!]_i = \sum_{j \in [\mathcal{S}]} [\![\mathbf{r}_i]\!]_j$
  4. $\mathbf{z}_i = [\![\mathbf{r}]\!]_i + c \cdot \mathtt{sk}_i$
  5. Broadcast $\mathbf{z}_i$
- **Combine:** the final signature is $(c, \mathbf{z} = \sum_{i \in \mathcal{S}} \lambda_i \cdot \mathbf{z}_i)$

Similar to [CGJ+99, JL00, AF04]

Security: $[\![\mathbf{r}]\!]_i$ is uniform and therefore hides $\mathtt{sk}_i$

This protocol can be augmented to achieve **robustness**

- Adds a *complaint* round
- Adds a V3S (Verifiable Short Secret Sharing) inspired from [ABCP23, GHL22]
  - Lighter than NIZK
- Same ideas can be used for **DKG**

Final observations for Flood-and-Submerse

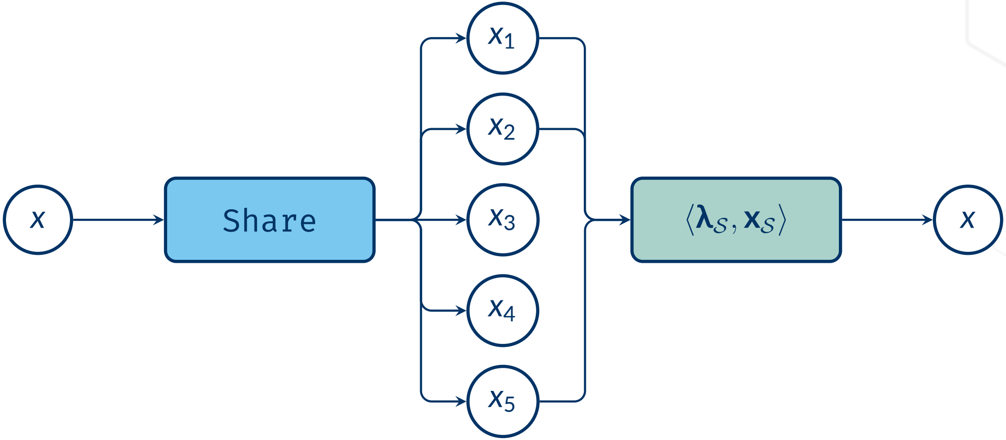
- **Sizes:** About 12 KB
- **Speed:** Very fast (bottleneck is generating T ciphertext per user)
- **Rounds:** 4 rounds
- **Communication:** $56 \cdot T$ KB per user
- **Distributed key generation:** Yes
- **Robustness:** Yes

Further reading:

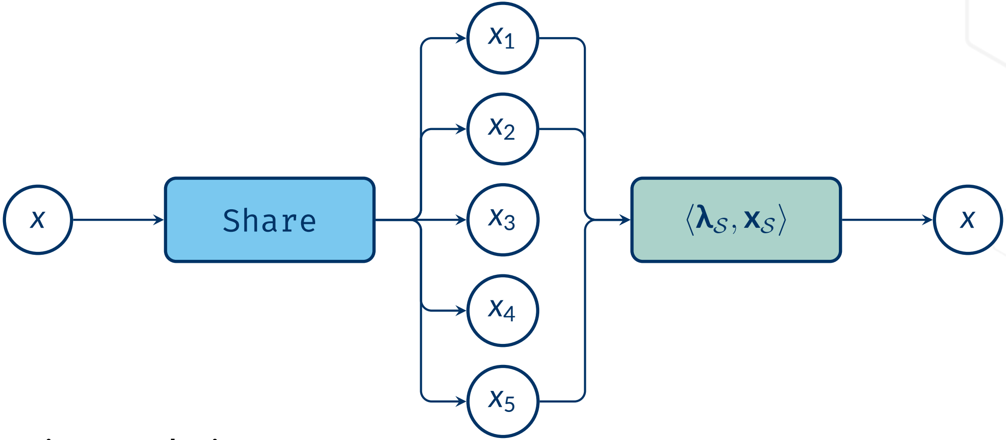
- Thomas Espitau, Guilhem Niot, Thomas Prest. *Flood and Submerse: Distributed Key Generation and Robust Threshold Signature from Lattices.* CRYPTO 2024 [ENP24]

The Death Star Algorithm

Different types of secret sharings



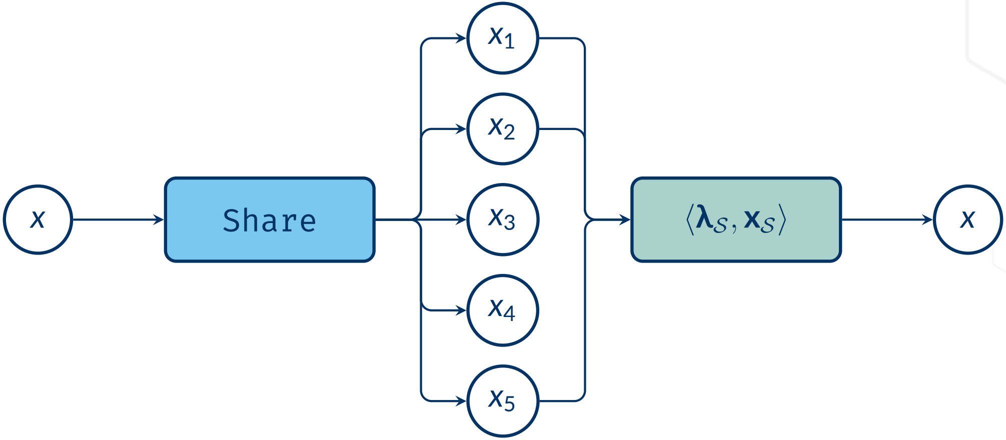
Different types of secret sharings



Shamir secret sharing:

- → `Share`: $x_i = P(i)$, where $P(0) = x$
- → The shares x_i and reconstruction vector $\boldsymbol{\lambda}_{\mathcal{S}}$ may be large

Different types of secret sharings



"Short" secret sharing: we require that:

1. If x is short, the shares x_i are short
2. The reconstruction vector $\boldsymbol{\lambda}_{\mathcal{S}}$ is short

Example: N-out-of-N sharing where:

- → $x_1, \ldots, x_{N-1} \leftarrow D_\sigma^{N-1}$, and $x_N = x - \sum_{i<N} x_i$
- → $\boldsymbol{\lambda}_{\mathcal{S}} = (1, \ldots, 1)$

Extensible to T-out-of-N via replicated SS, requires $\binom{N}{T-1}$ shares per party.

Threshold Raccoon with short shares

Threshold Raccoon, short shares

- **Round 1:**
  1. Sample short $\mathbf{r}_i$
  2. $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
  3. $\mathtt{com}_i = H_{\mathtt{com}}(\mathbf{w}_i, \mathtt{msg}, \mathcal{S})$
  4. Broadcast $\mathtt{com}_i$
- **Round 2:**
  1. Broadcast $\mathbf{w}_i$
- **Round 3:**
  1. $\mathbf{w} = \sum_i \mathbf{w}_i$
  2. $c = H(\mathtt{vk}, \mathtt{msg}, \mathbf{w})$
  3. $\mathbf{z}_i = \mathbf{r}_i + c \cdot \mathtt{sk}_i$
  4. Broadcast $\mathbf{z}_i$
- **Combine:** the final signature is $(c, \mathbf{z} = \sum_{i \in \mathcal{S}} \mathbf{z}_i)$

- ✔ For simplicity, we consider $T = N$
  - Each $\lambda_i = 1$

Identifiable aborts

- Each $\mathtt{vk}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathtt{sk}_i$ is a valid public key
- Therefore each $(c, \mathbf{z}_i)$ is a valid partial signature
- We get identifiable aborts for free!

Security

- $\mathbf{r}_i$ hides $c \cdot \mathtt{sk}_i$ as both are short
- We argue security via Hint-MLWE

How large is the sum of T vectors?

Consider the sum of T i.i.d. Gaussian vectors $\mathbf{x}_i \leftarrow D_\sigma^n$.
What can se say about its norm?

How large is the sum of T vectors?

Consider the sum of T i.i.d. Gaussian vectors $\mathbf{x}_i \leftarrow D_\sigma^n$.
What can se say about its norm?

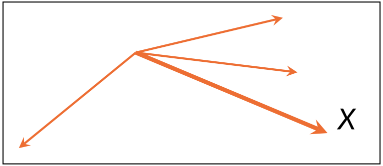


Figure 1: Average-case: $O(\sqrt{T})$

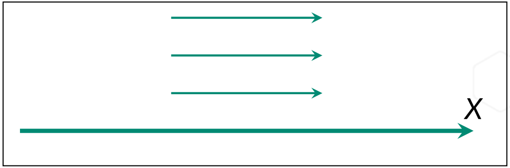
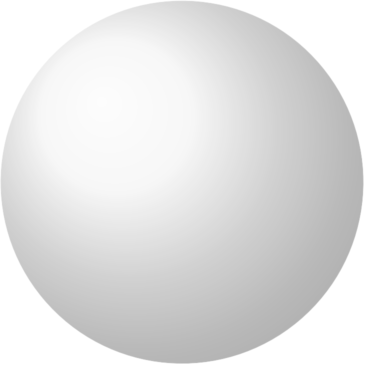


Figure 2: Worst-case: $O(T)$

- ✓ Signatures by honest signers would end up in Fig. 2
- ✗ But colluding signers could force the Fig. 1

This will decrease security. Can we do better?

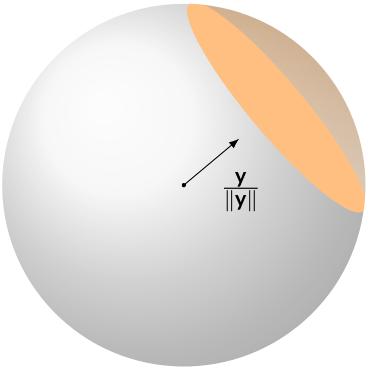
If $\mathbf{x} \leftarrow D_{\sigma}^{n}$, it is well known that™:



If $\mathbf{x} \leftarrow D_\sigma^n$, it is well known that™:

1. $\|\mathbf{x}\|$ is concentrated around its expected value $\sigma\sqrt{n}$

The Death Star Algorithm

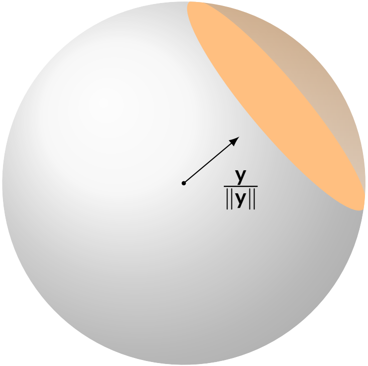


If $\mathbf{x} \leftarrow D_\sigma^n$, it is well known that™:

1. $\|\mathbf{x}\|$ is concentrated around its expected value $\sigma\sqrt{n}$
2. For any vector $\mathbf{y}$:

$$\langle \mathbf{x}, \mathbf{y} \rangle < \sigma\sqrt{O(\lambda)}\,\|\mathbf{y}\| \qquad (5)$$

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



The Death Star Algorithm

1. For each signer i:
   1. If $\|\mathbf{x}_i\| \geq (1 + o(1))\sigma\sqrt{n}$, reject i
   2. 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

Lemma: for a set of non-rejected $(\mathbf{x}_i)_{i \in [T]}$, the sum $\mathbf{x} = \sum_i \mathbf{x}_i$ satistifes:

$$\|\mathbf{x}\| \leq \sigma \cdot T \cdot \sqrt{2 \log 2 \cdot \lambda} \tag{5}$$

$$+ \sigma \cdot \sqrt{T \cdot d} \cdot (1 + \varepsilon) \tag{6}$$

Comparison with standard approaches

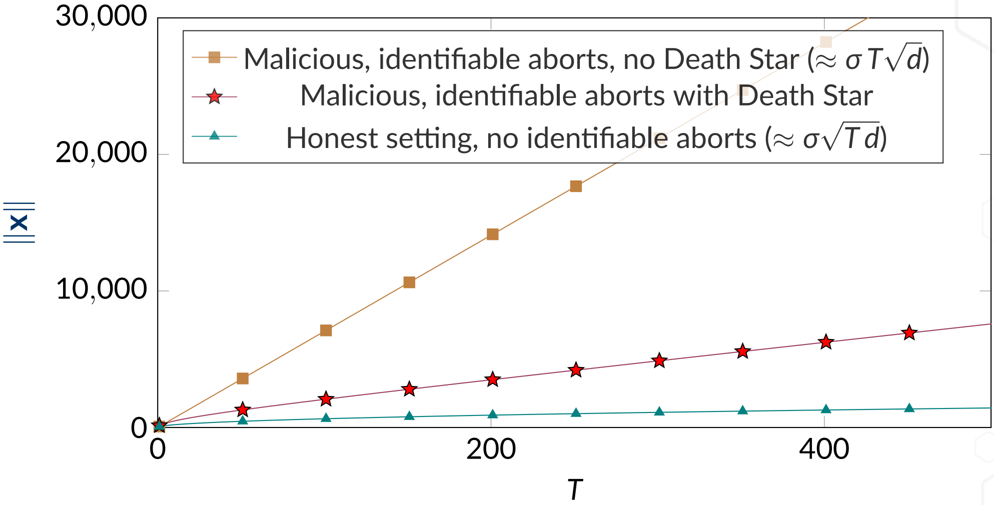


Figure 3: Norm of $\mathbf{x} = \sum_{i\in[T]} \mathbf{x}_i$, for $\sigma = 1$, dimension $n = 4096$, $\lambda = 128$ bits of security, and $1 \leq T \leq 1000$.

Conclusion

Approach	Size	Speed	Rounds	Comm/party	IA/Robust	DKG
[DKM+24]	≈10 KB	**O(T)**	3	**40 KB**	No	No
[EKT24]	≈10 KB	**O(T)**	**2**	300 KB	No	No
[ENP24]	≈10 KB	**O(T)**	4	$56 \cdot T$ KB	**Yes**	**Yes**
"Death Star"	≈10 KB	$O\binom{N}{T}$	3	**20 KB**	**Yes**	**Yes**

Questions?

Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen.
VSS from distributed ZK proofs and applications.
In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 405–440. Springer, Singapore, December 2023.

Masayuki Abe and Serge Fehr.
Adaptively secure feldman VSS and applications to universally-composable threshold cryptography.
In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 317–334. Springer, Berlin, Heidelberg, August 2004.

Shweta Agrawal, Damien Stehlé, and Anshu Yadav.
Round-optimal lattice-based threshold signatures, revisited.
In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.

Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi.
Ringtail: Practical two-round threshold signatures from learning with errors.
Cryptology ePrint Archive, Report 2024/1113, 2024.

Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova.
On the (in)security of ROS.
Journal of Cryptology, 35(4):25, October 2022.

Mihir Bellare and Gregory Neven.
Multi-signatures in the plain public-key model and a general forking lemma.
In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin.
Adaptive security for threshold cryptosystems.
In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115. Springer, Berlin, Heidelberg, August 1999.

Elizabeth C. Crites, Chelsea Komlo, and Mary Maller.
Fully adaptive Schnorr threshold signatures.
In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023.

Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs.
On the security of two-round multi-signatures.
In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019.

Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani Saarinen.
Raccoon.
Technical report, National Institute of Standards and Technology, 2023.

available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen.
Threshold raccoon: Practical threshold signatures from standard lattice assumptions.
In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi.
Raccoon: A masking-friendly signature proven in the probing model.
In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 409–444. Springer, Cham, August 2024.

Julien Devevey, Alain Passelègue, and Damien Stehlé.
G+G: A fiat-shamir lattice signature based on convolved gaussians.
In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 37–64. Springer, Singapore, December 2023.

Muhammed F. Esgin, Thomas Espitau, Guilhem Niot, Thomas Prest, Amin Sakzad, and Ron Steinfeld.
Plover: Masking-friendly hash-and-sign lattice signatures.
In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 316–345. Springer, Cham, May 2024.

Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure.

Two-round threshold signature from algebraic one-more learning with errors.
In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024*, *Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024.

Thomas Espitau, Guilhem Niot, and Thomas Prest.
Flood and submerse: Distributed key generation and robust threshold signature from lattices.
In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024*, *Part VII*, volume 14926 of *LNCS*, pages 425–458. Springer, Cham, August 2024.

Craig Gentry, Shai Halevi, and Vadim Lyubashevsky.
Practical non-interactive publicly verifiable secret sharing with thousands of parties.
In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022*, *Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, Cham, May / June 2022.

Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde.
Two-round threshold lattice-based signatures from threshold homomorphic encryption.
In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 266–300. Springer, Cham, June 2024.

Stanislaw Jarecki and Anna Lysyanskaya.
Adaptively secure threshold cryptography: Introducing concurrency, removing erasures.
In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Berlin, Heidelberg, May 2000.

Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders.

Phoenix: Hash-and-sign with aborts from lattice gadgets.
In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I*, pages 265–299. Springer, Cham, June 2024.

Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner.
A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model.
In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Cham, April / May 2018.

Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song.
Toward practical lattice-based proof of knowledge from hint-MLWE.
In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.

Shuichi Katsumata, Michael Reichle, and Kaoru Takemure.
Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding.
In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.

Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai.
CRYSTALS-DILITHIUM.
Technical report, National Institute of Standards and Technology, 2022.
available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.

Yang Yu, Huiwen Jia, and Xiaoyun Wang.
Compact lattice gadget and its applications to hash-and-sign signatures.
In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 390–420. Springer, Cham, August 2023.