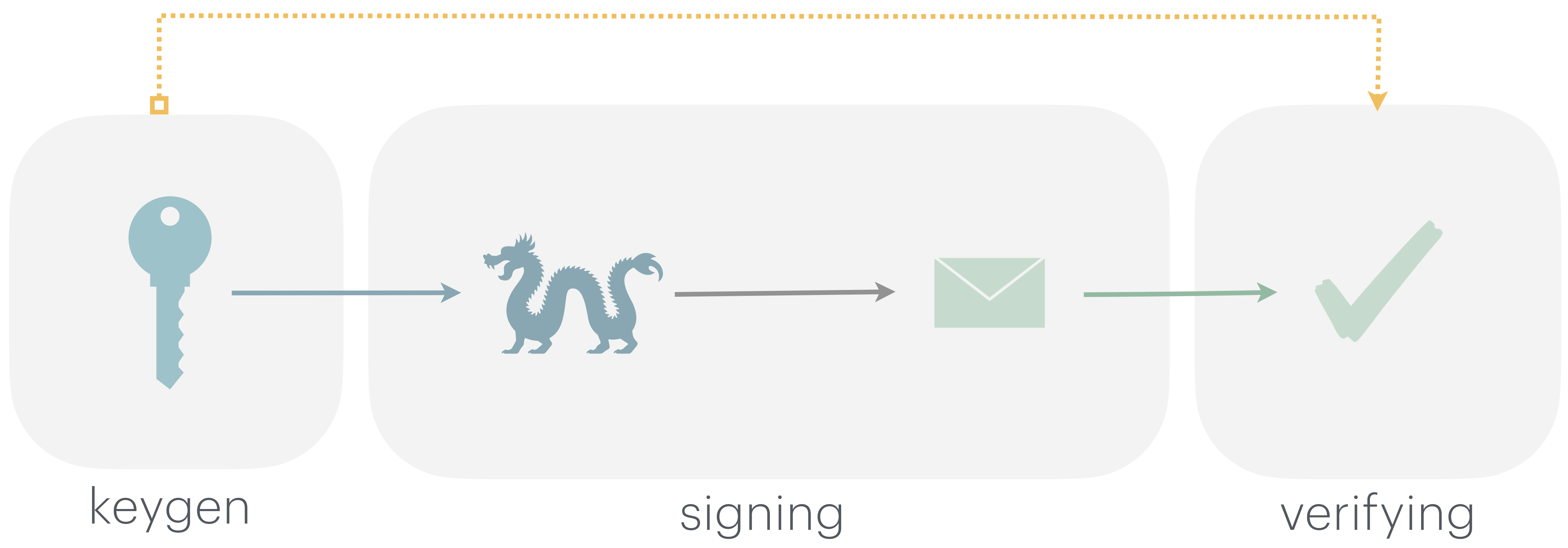
The big picture
of lattice threshold

Thomas Espitau — PQShield

ICMC 2025

What is a signature ?

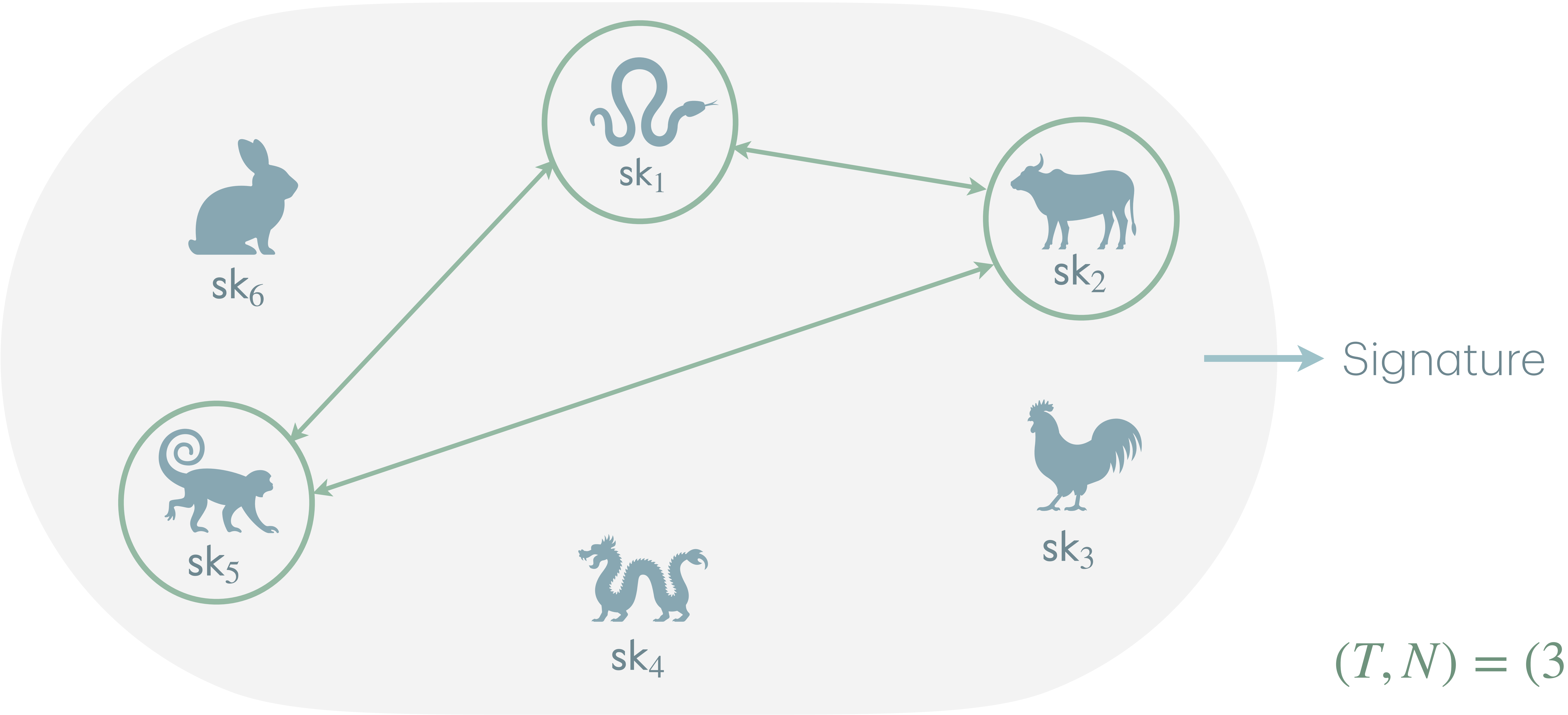
2 party protocol to verify the **authenticity** of a message



What is a threshold signature ?

Interactive protocol to distribute signature generation such that:

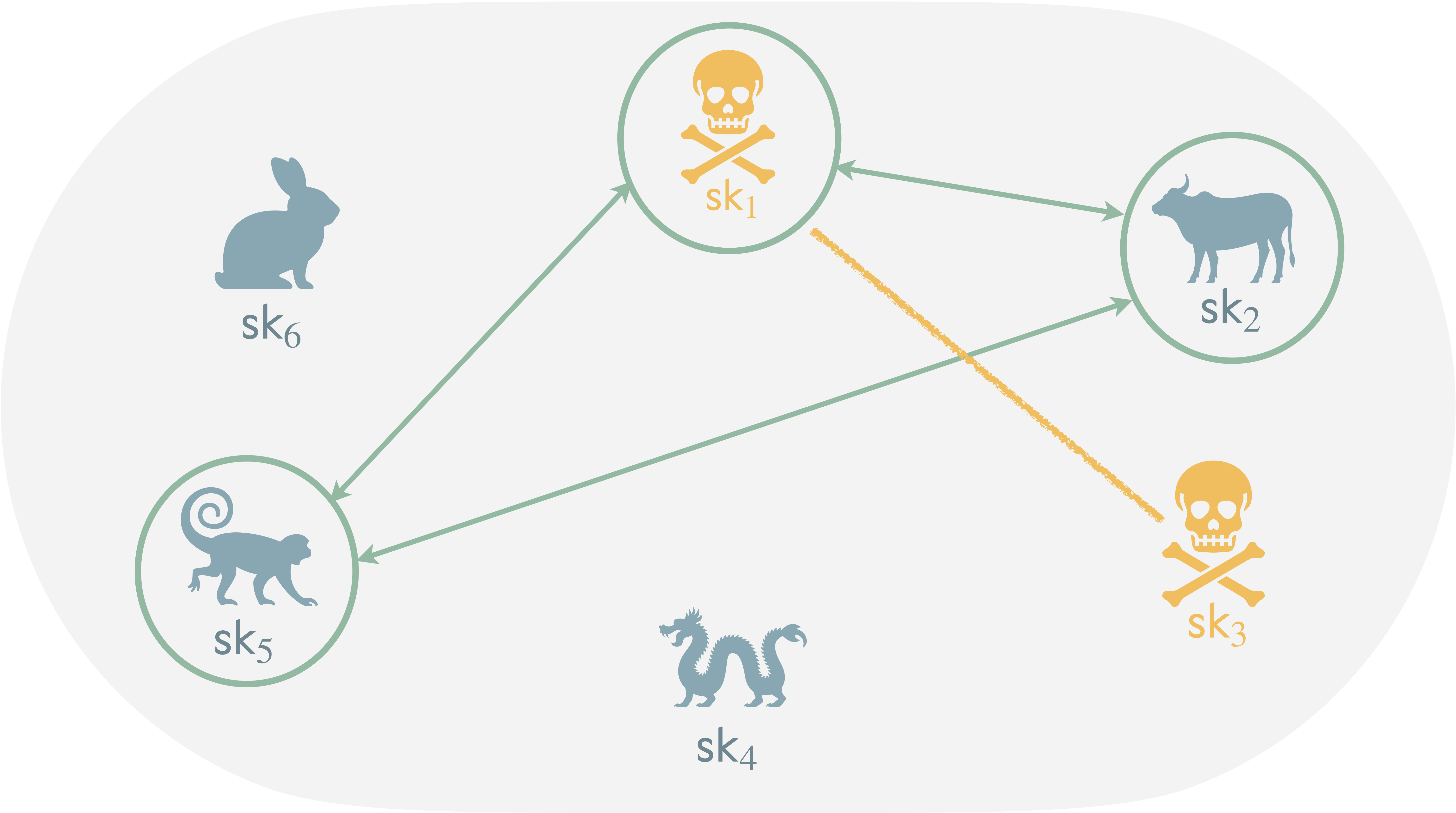
T out of N parties can **collaborate** to sign a message and $T-1$ parties **cannot** sign.



$(T, N) = (3,6)$

Security requirements

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



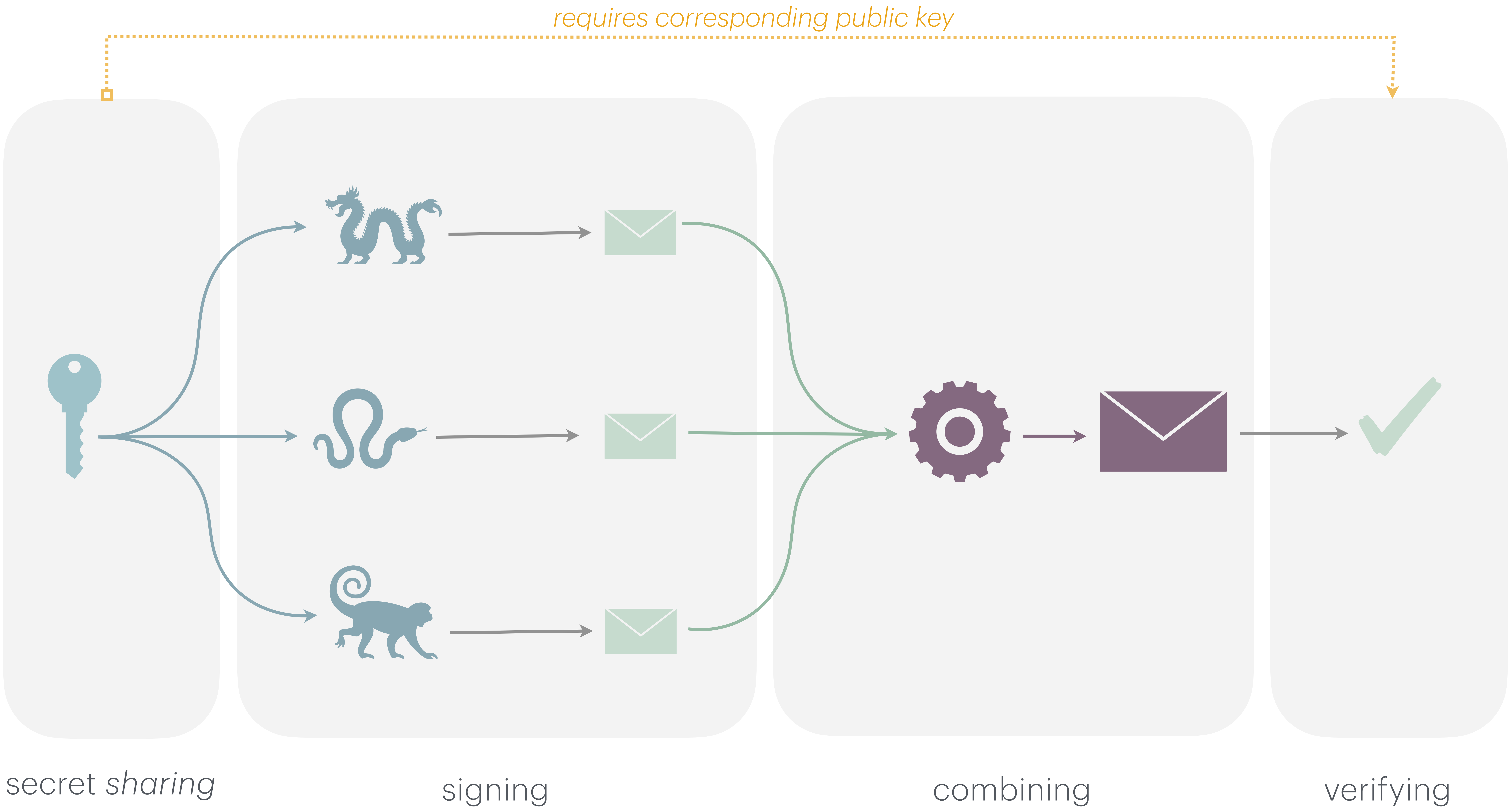

Threshold Lattice-Based Signatures
Homomorphic Encryption*
and Tjerand Silde3***
Two-round n-out-of-n and Multi-Signatures and
Trapdoor Commitment from Lattices*
Ivan Damgård1, Claudio Orlandi1, Akira Takahashi1, and Mehdi Tibouchi2
Two-Round
Algebraic One-
Thomas Espitau1, Shuichi Katsumata1,2, Kaoru Taker
Flood and Submerse: Distributed Key
Generation and Robust Threshold Signature
from Lattices
Thomas Espitau1, Guilhem Niot1,2, and Thomas Prest1
Ringtail: Practic
Cecilia Boschini
ETH Zürich, Switzerland
Akira Takahashi
JPMorgan AI Research & AlgoCRYPT CoE,
Threshold Raccoon: Practical Thre
from Standard Lattice Assump
Rafael del Pino1, Shuichi Katsumata1,2, Mary Maller1,3, Fabrice Mouh
Prest1, Markku-Juhani Saarinen1,5
attice-Based Multi-Signatu
gle-Round Online Phase*
Akira Takahashi2, and Mehdi Tibouchi3


Tailored
FHE
MPC

Family of techniques
different designs choices, different pros/cons

Thresholdization technique	Signature size	Speed	Rounds	Comm cost/party
MPC	Small	Slow	15	≥ 1MB
FHE	Medium	As fast as FHE	2	≥ 1MB
Tailored	S—M	Fast	2-4	20 kB

What is the rationale of (*tailored*) threshold?


requires corresponding public key
secret sharing
signing
combining
verifying

SIGNATURE SCHEME

+

KEY DISTRIBUTION / SHARING

=

THRESHOLD SIGNATURE

Fiat-Shamir

NIST standard (ML-DSA — “Dilithium”)

Hash-and-Sign

NIST standard (FN-DSA — “Falcon”)

(*flooded*) Fiat-Shamir 101

Simple identification protocol

Keygen() → sk, vk

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

Prover

Commit

- Sample a short $\mathbf{r}$
- $\mathbf{w} = A \cdot \mathbf{r}$

$\mathbf{w}$

Challenge

- Sample a short $\mathbf{c}$

$\mathbf{c}$

Answer

- $\mathbf{z} = \mathbf{c} \cdot$ sk $+ \mathbf{r}$

$\mathbf{z}$

Verify

- $\mathbf{w}* = A \cdot \mathbf{z} - c \cdot$ vk
- Assert $\mathbf{w}* = \mathbf{w}$
- Assert $\mathbf{z}$ short

Verifier

(all operations are mod q)

Simple identification protocol

Keygen() → sk, vk

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

Prover

Commit

- Sample a short $\mathbf{r}$
- $\mathbf{w} = A \cdot \mathbf{r}$

Challenge

- Sample a short $\mathbf{c}$

Answer

- $\mathbf{z} = \mathbf{c} \cdot \text{sk} + \mathbf{r}$

$\xrightarrow{\mathbf{z}}$

Verifier

Verify

- $\mathbf{w}^* = A \cdot \mathbf{z} - c \cdot \text{vk}$
- Assert $\mathbf{w}^* = \mathbf{w}$
- Assert $\mathbf{z}$ short

(all operations are mod q)

Simple identification protocol

Keygen() → sk, vk

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

Prover

Commit

- Sample a short $\mathbf{r}$
- $\mathbf{w} = A \cdot \mathbf{r}$

Challenge

- $\mathbf{c} =$ Hash$(\mathbf{w})$

Answer

- $\mathbf{z} = \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}$

$\mathbf{c}, \mathbf{z}$

Verify

- $\mathbf{w}* = A \cdot \mathbf{z} - c \cdot \mathsf{vk}$
- Assert $\mathbf{w}* = \mathbf{w}$
- Assert $\mathbf{z}$ short

Verifier

(all operations are mod q)

Simple identification protocol

Keygen() → sk, vk

- $\text{vk} = \mathbf{A} \cdot \textbf{sk}$, for short $\textbf{sk}$

Prover

Commit

- Sample a short $\mathbf{r}$
- $\mathbf{w} = A \cdot \mathbf{r}$

Challenge

- $\mathbf{c} = \text{Hash}(\mathbf{w})$

Answer

- $\mathbf{z} = \mathbf{c} \cdot \text{sk} + \mathbf{r}$

$\mathbf{c}, \mathbf{z}$

Verify

- $\mathbf{w}^* = A \cdot \mathbf{z} - c \cdot \text{vk}$
- Assert $\mathbf{c} = \text{Hash}(\mathbf{w}^*)$
- Assert $\mathbf{z}$ short

Verifier

(all operations are mod q)

Simple identification protocol

Keygen() $\rightarrow$ sk, vk

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

Prover

Commit

- Sample a short $\mathbf{r}$
- $\mathbf{w} = A \cdot \mathbf{r}$
- $\mathbf{c} =$ Hash$(\mathbf{w})$
- $\mathbf{z} = \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}$

$\mathbf{c}, \mathbf{z}$

Verifier

Verify

- $\mathbf{w}* = A \cdot \mathbf{z} - c \cdot \mathsf{vk}$
- Assert $\mathbf{c} =$ Hash$(\mathbf{w}*)$
- Assert $\mathbf{z}$ short

(all operations are mod q)

Fiat-Shamir on lattices

Keygen() → sk, vk

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

Sign

- Sample a short $\mathbf{r}$
- $\mathbf{w} = A \cdot \mathbf{r}$
- $\mathbf{c} = \text{Hash}(m, \mathbf{w})$
- $\mathbf{z} = \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}$
- Output $\mathbf{c}, \mathbf{z}$

Verify

- $\mathbf{w}* = A \cdot \mathbf{z} - c \cdot \mathsf{vk}$
- Assert $\mathbf{c} = \text{Hash}(m, \mathbf{w}*)$
- Assert $\mathbf{z}$ short

(all operations are mod q)

What about hardness ?

	Keygen() $\to$ sk, vk
Short integer solution (SIS)	• $\mathsf{vk} = \mathbf{A} \cdot \mathsf{sk}$, for short sk
	Sign
	• Sample a short $\mathbf{r}$
Short integer solution (SIS)	• $\mathbf{w} = A \cdot \mathbf{r}$
	• $\mathbf{c} =$ Hash$(m, \mathbf{w})$
Hint-LWE	• $\mathbf{z} = \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}$
	• Output $\mathbf{c}, \mathbf{z}$
	Verify
Short integer solution (SIS)	• $\mathbf{w}* = A \cdot \mathbf{z} - c \cdot \mathsf{vk}$
	• Assert $\mathbf{c} =$ Hash$(m, \mathbf{w}*)$
	• Assert $\mathbf{z}$ short

(all operations are mod q)

SIGNATURE SCHEME

+

KEY DISTRIBUTION / SHARING

=

THRESHOLD SIGNATURE

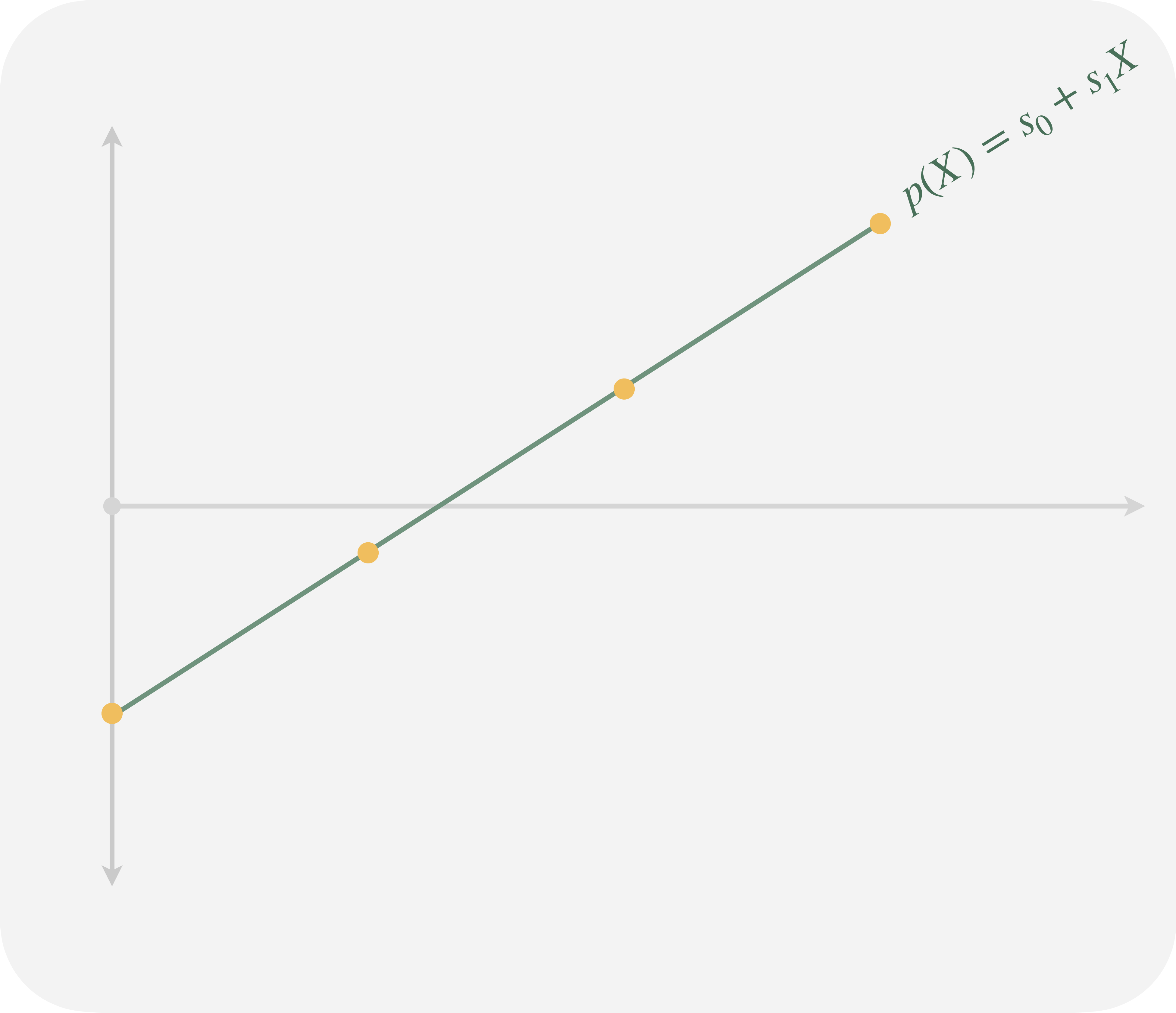
How to share a secret?

Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Secret : line

Shares : points of

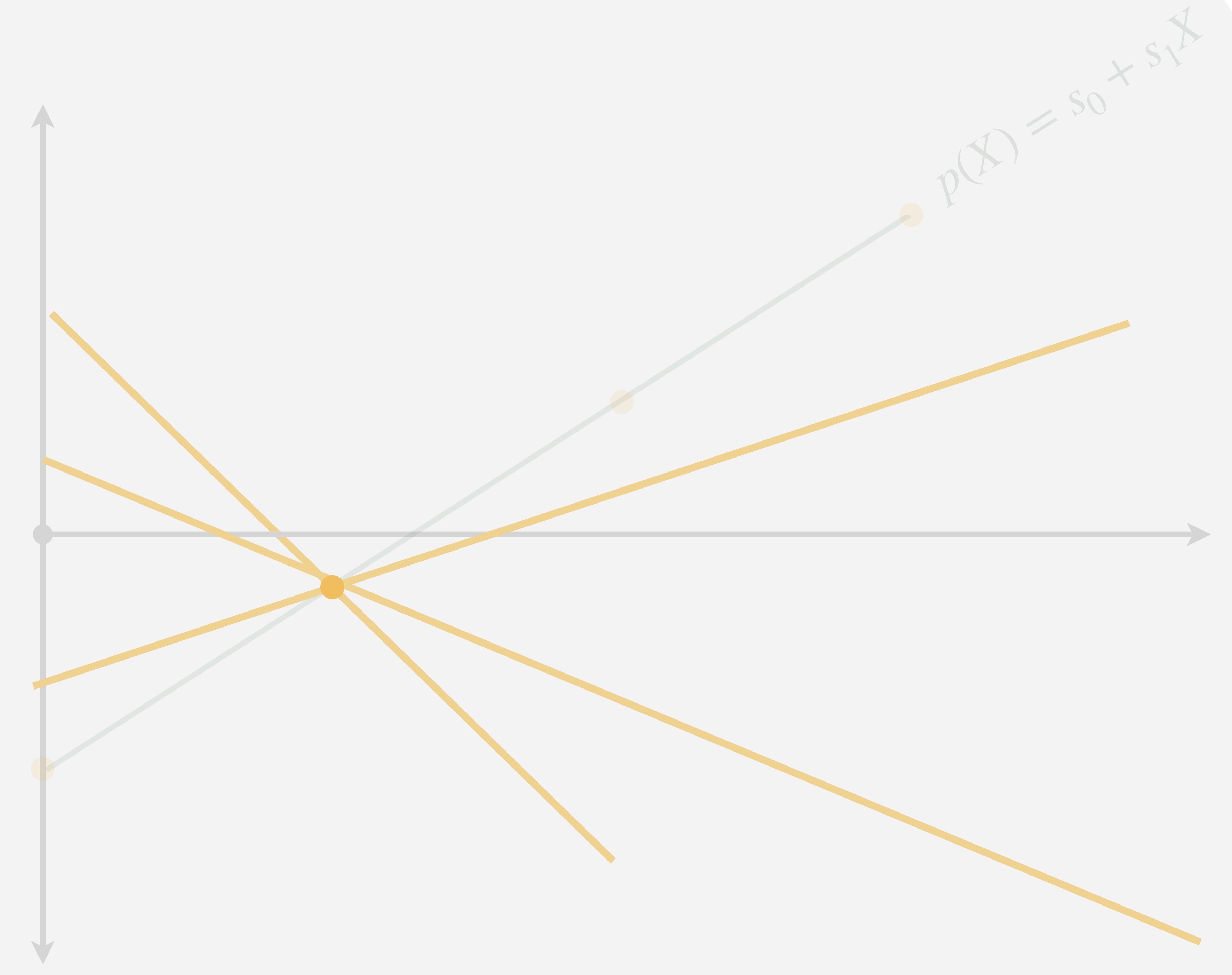


Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Secret : line

Shares : points of



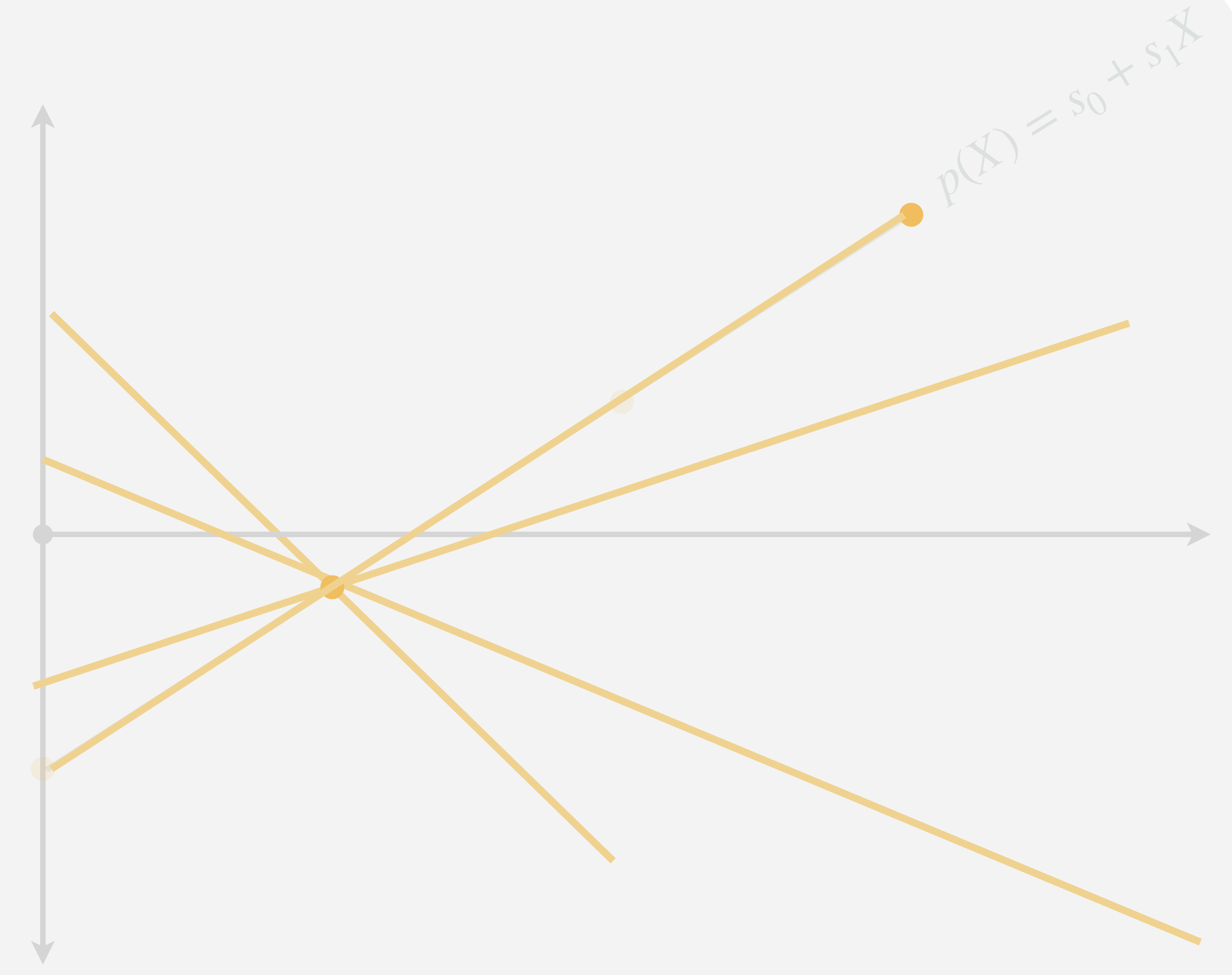
*"Through two points goes only **one** line"*

Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Secret : line

Shares : points of



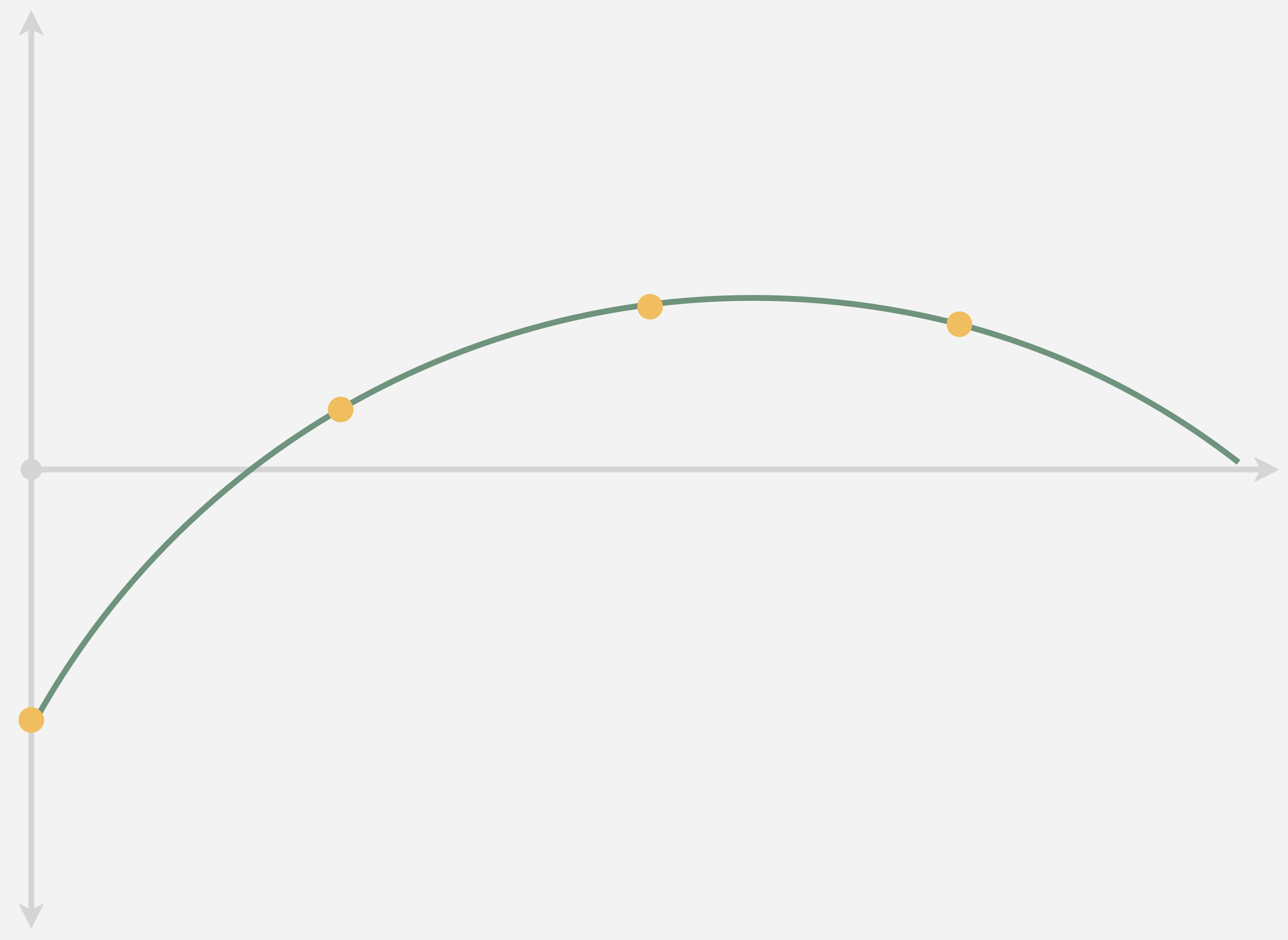
*"Through two points goes only **one** line"*

Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Secret : curve of degree 2

Shares : points of



*"Through 3 points goes only **one** parabola"*

Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Secret : curve of degree 2

Shares : points of

*"Through 3 points goes only **one** parabola"*

Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Secret : curve of degree 2

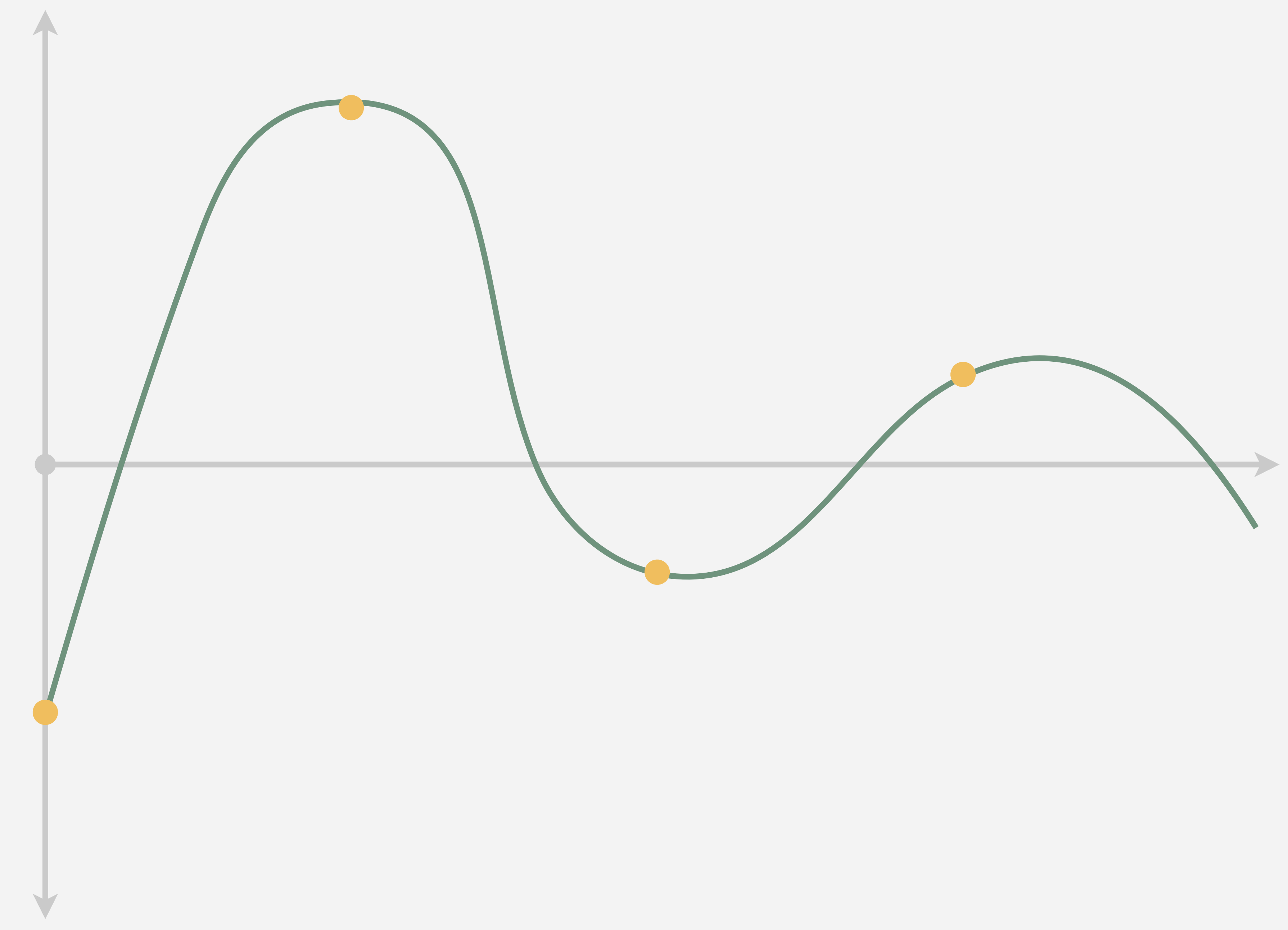
Shares : points of

*"Through 3 points goes only **one** parabola"*

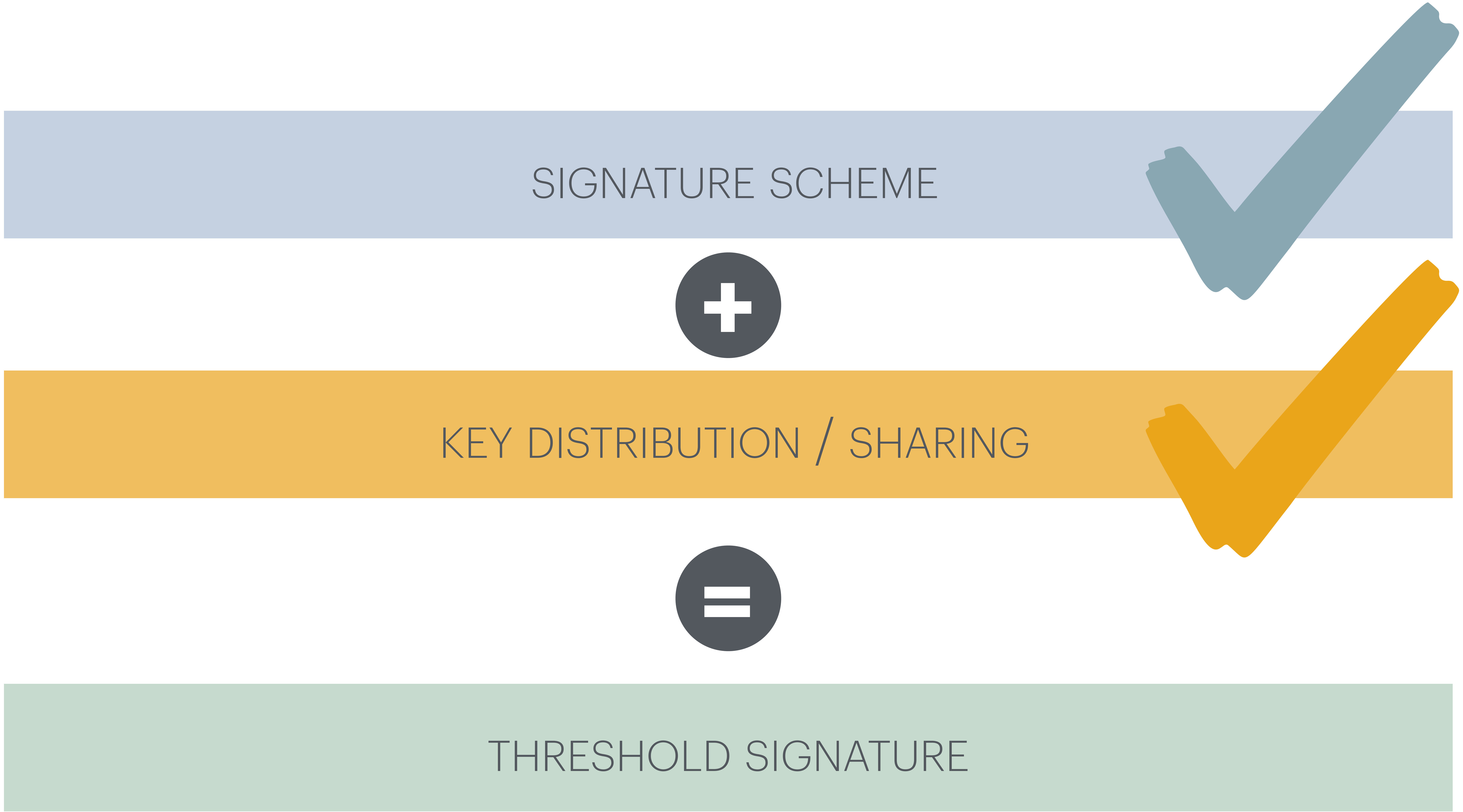
Introducing Shamir Secret Sharing

T out of N parties can **collaborate** to recover a message and $T-1$ parties **cannot**.

Reconstruction here is **linear**

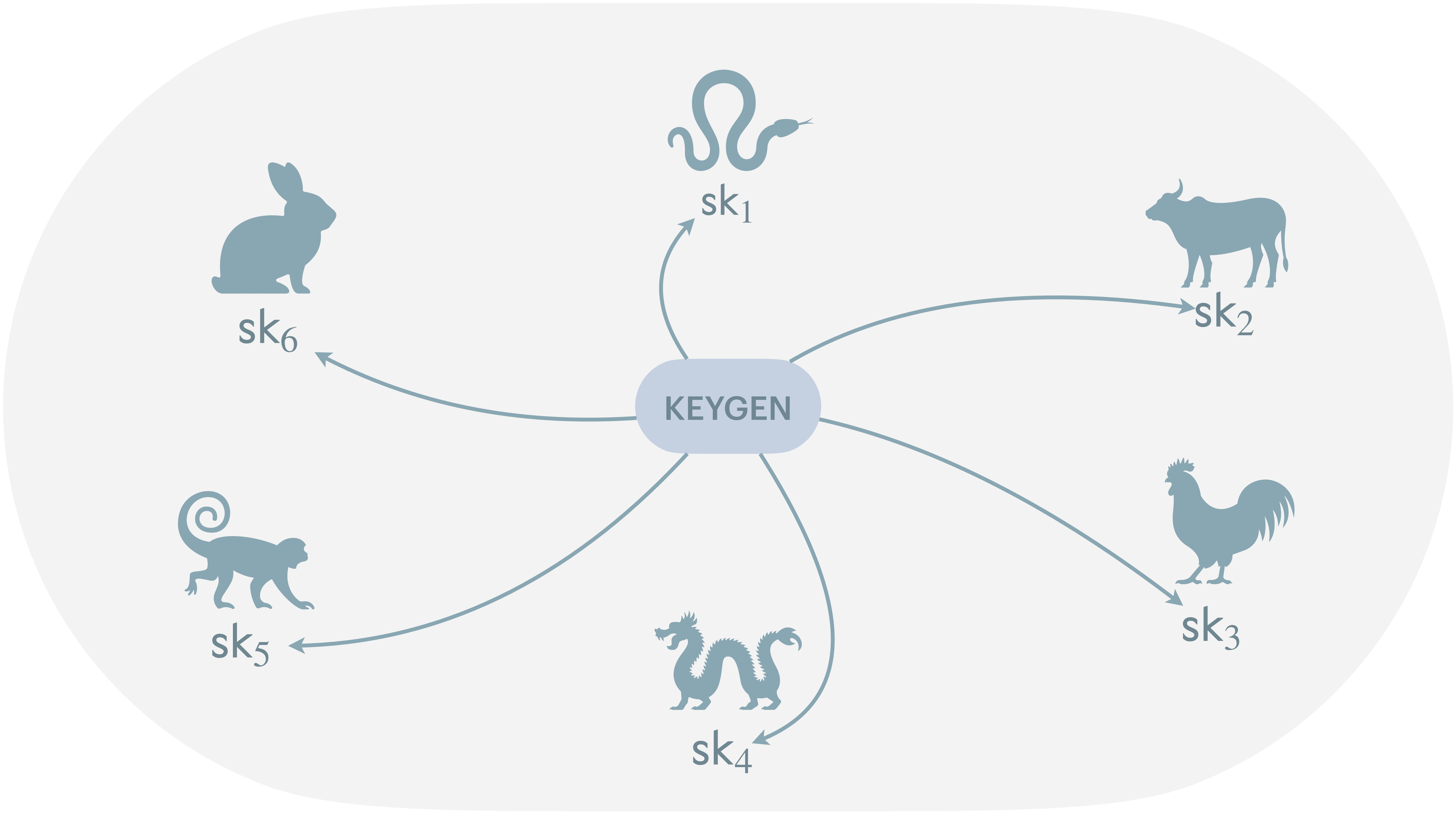


*“Through T points goes only **one** curve of degree T-1”*


SIGNATURE SCHEME
+
KEY DISTRIBUTION / SHARING
=
THRESHOLD SIGNATURE

BACK TO THE THRESHOLD

$$\mathsf{sk} = L_1 \mathsf{sk}_1 + L_2 \mathsf{sk}_2 + L_5 \mathsf{sk}_5$$



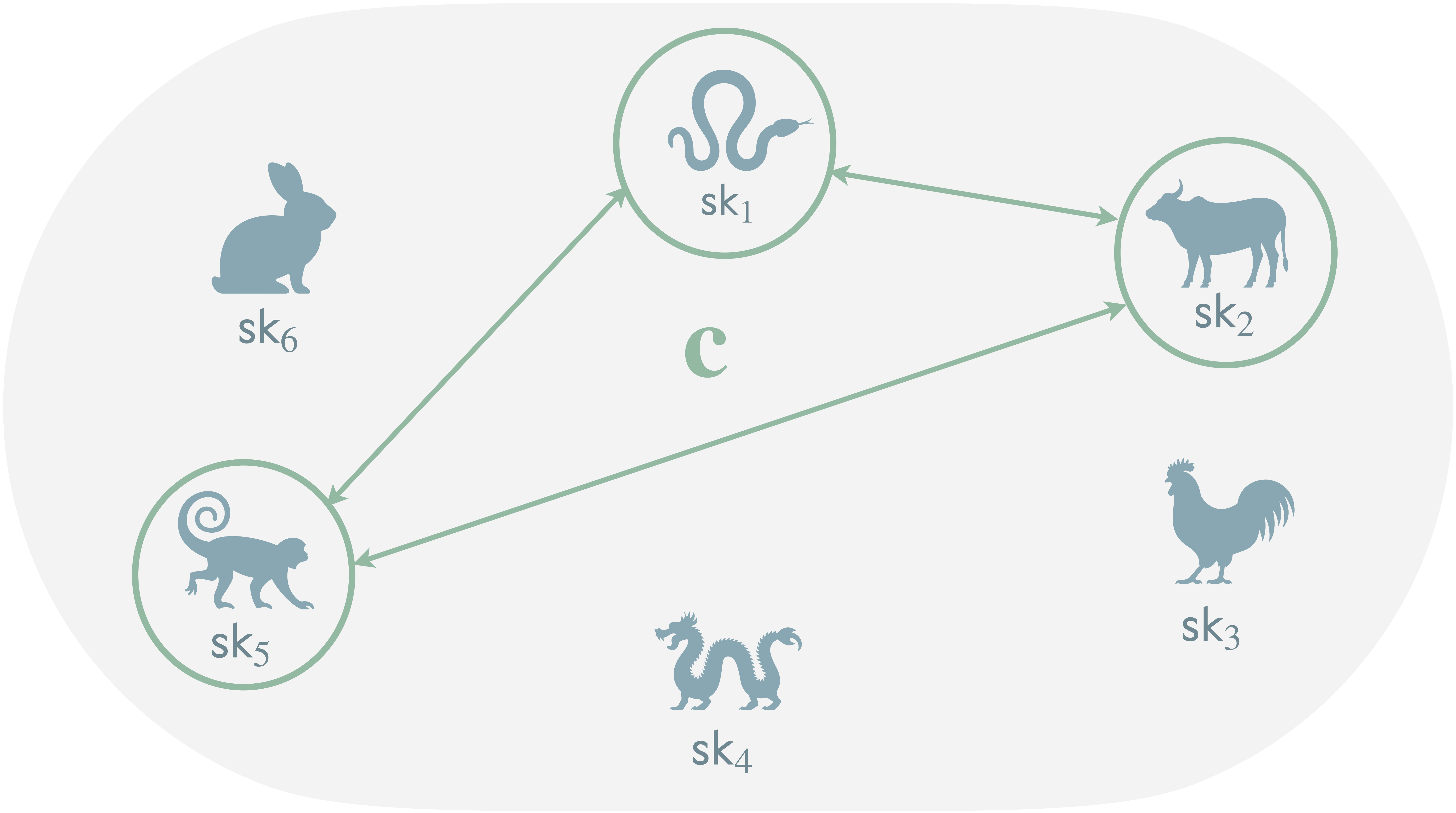
$$\mathsf{sk} = L_1 \mathsf{sk}_1 + L_2 \mathsf{sk}_2 + L_5 \mathsf{sk}_5$$

Verify

- $\mathbf{w}^* = A \cdot \mathbf{z} - \mathbf{c} \cdot \mathsf{vk}$
- Assert $\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}^*)$
- Assert $\mathbf{z}$ short

1 - Agree on challenge $\mathbf{c}$

$$\mathsf{sk} = L_1\mathsf{sk}_1 + L_2\mathsf{sk}_2 + L_5\mathsf{sk}_5$$



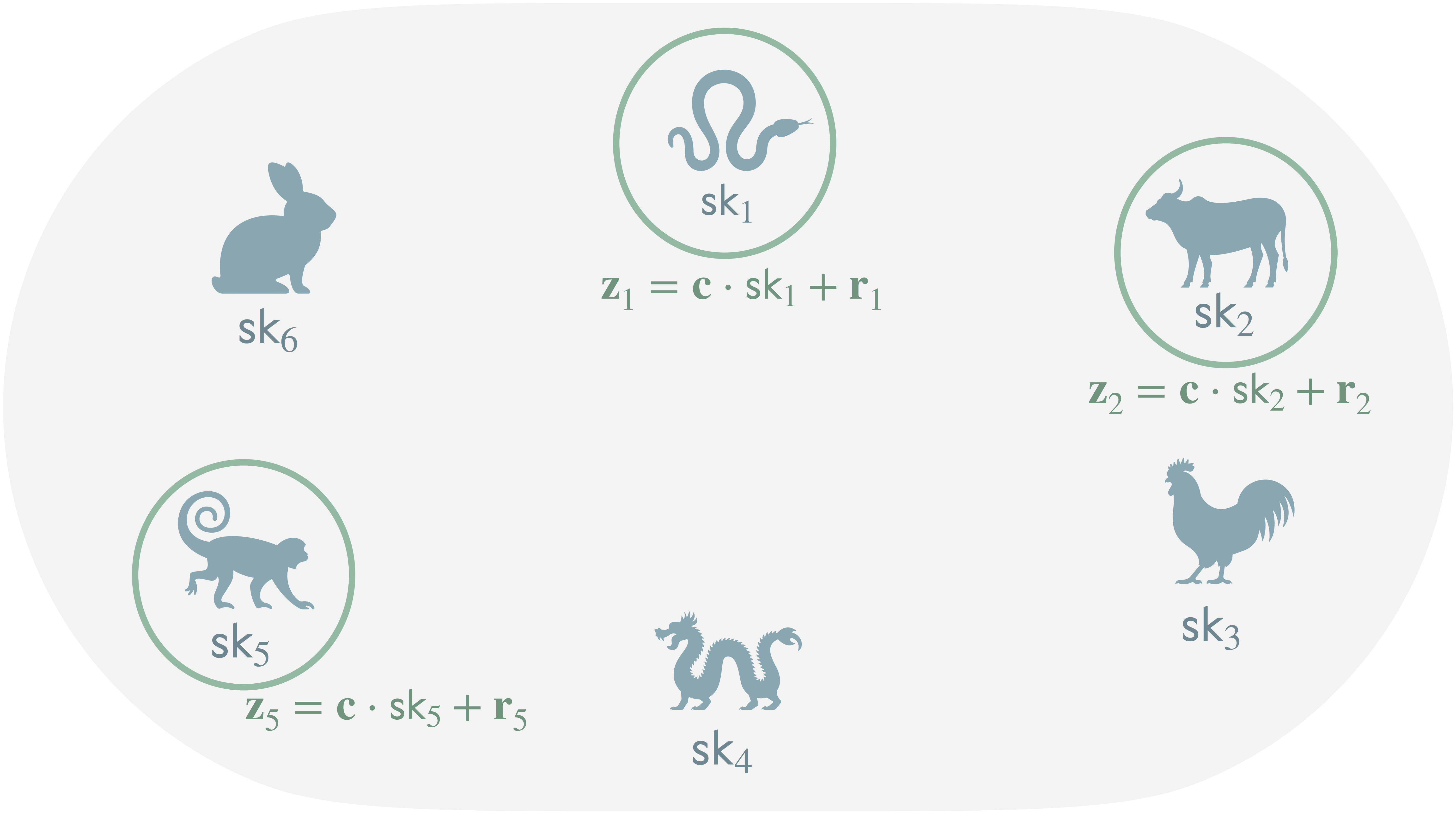
Verify

- $\mathbf{w}^* = A \cdot \mathbf{z} - \mathbf{c} \cdot \mathsf{vk}$
- Assert $\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}^*)$
- Assert $\mathbf{z}$ short

1 - Agree on challenge $\mathbf{c}$

2 - Compute the partial signature

$$\mathsf{sk} = L_1\mathsf{sk}_1 + L_2\mathsf{sk}_2 + L_5\mathsf{sk}_5$$



Verify

- $\mathbf{w}^* = A \cdot \mathbf{z} - \mathbf{c} \cdot \mathsf{vk}$
- Assert $\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}^*)$
- Assert $\mathbf{z}$ short

1 - Agree on challenge $\mathbf{c}$

2 - Compute the partial signature

3 -Combine

$$\mathsf{sk} = L_1\mathsf{sk}_1 + L_2\mathsf{sk}_2 + L_5\mathsf{sk}_5$$

$$\mathbf{z}_5 = \mathbf{c} \cdot \mathsf{sk}_5 + \mathbf{r}_5$$
$$\mathbf{z}_2 = \mathbf{c} \cdot \mathsf{sk}_2 + \mathbf{r}_2$$
$$+ \quad \mathbf{z}_1 = \mathbf{c} \cdot \mathsf{sk}_1 + \mathbf{r}_1$$

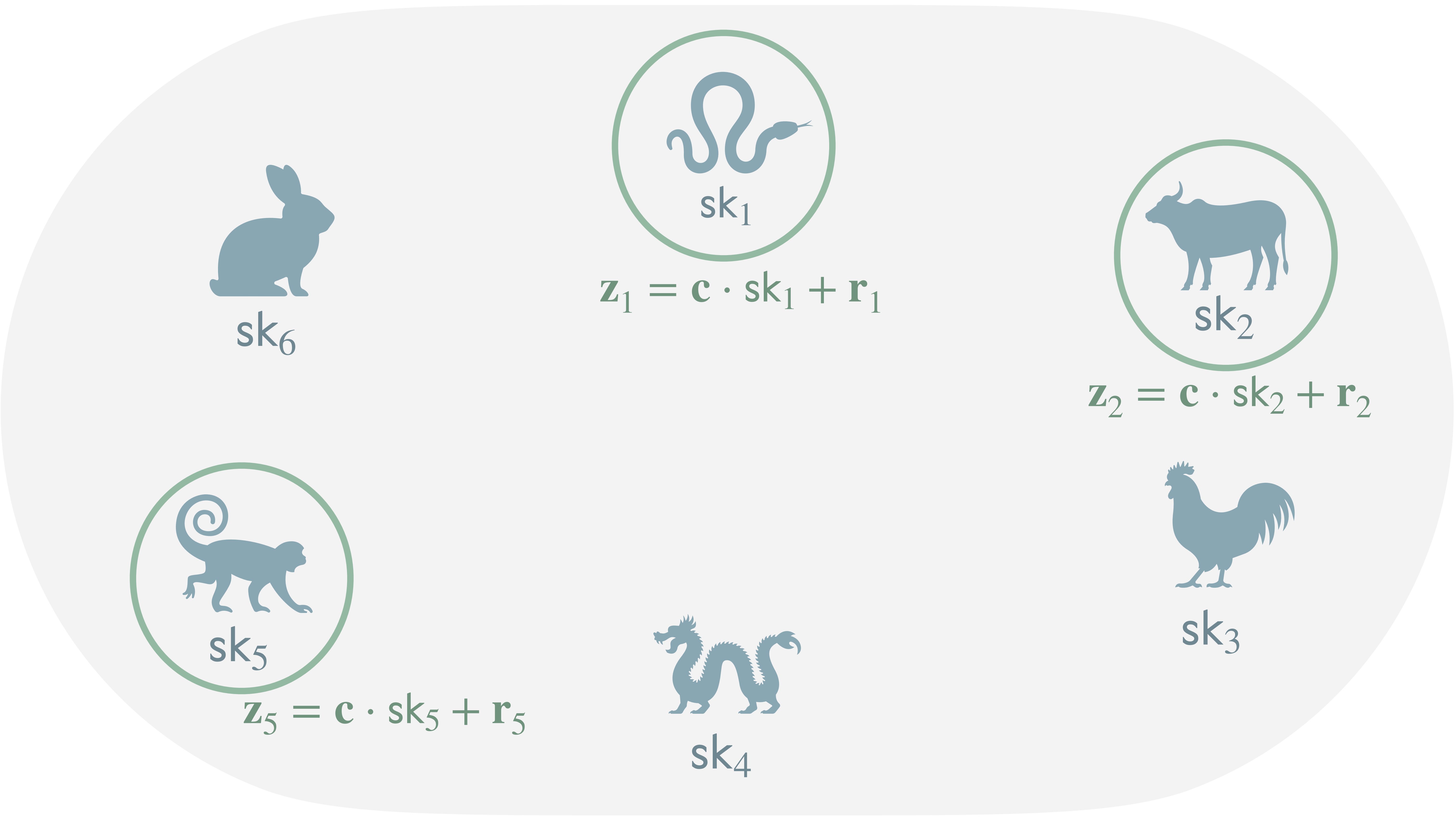
Combine

Output $(\mathbf{c}, L_1\mathbf{z}_1 + L_2\mathbf{z}_2 + L_5\mathbf{z}_5)$

Verify

- $\mathbf{w}^* = A \cdot \mathbf{z} - \mathbf{c} \cdot \mathsf{vk}$
- Assert $\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}^*)$
- Assert $\mathbf{z}$ short

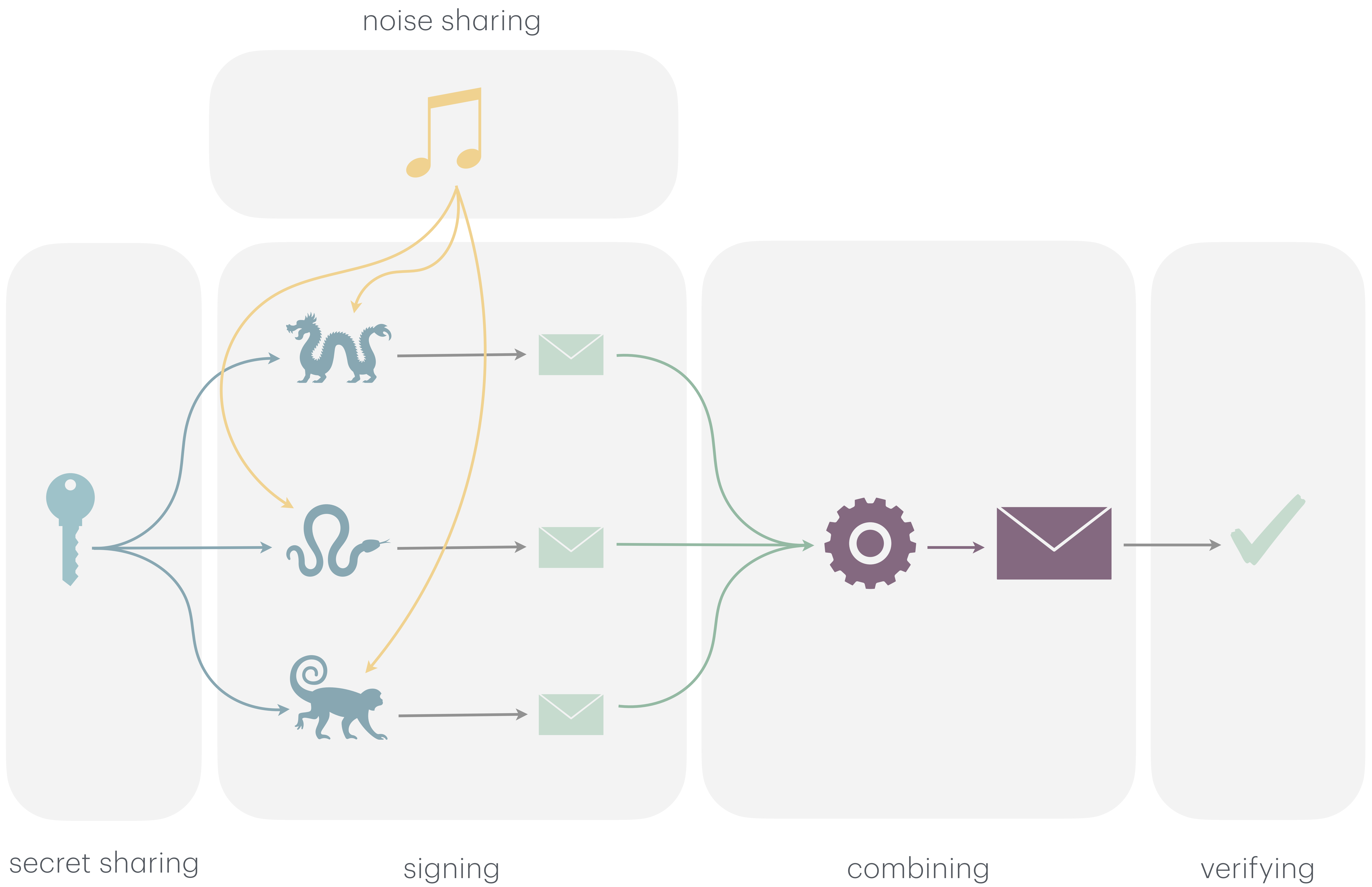
and now what?



$$\mathbf{z} = L_1\mathbf{z}_1 + L_2\mathbf{z}_2 + L_5\mathbf{z}_5 = \mathbf{c} \cdot (L_1\mathsf{sk}_1 + L_2\mathsf{sk}_2 + L_5\mathsf{sk}_5) + (L_1\mathbf{r}_1 + L_2\mathbf{r}_2 + L_5\mathbf{r}_5)$$

Secret sharing of the secret

Secret sharing of the noise


noise sharing
secret sharing
signing
combining
verifying

Going further… the big picture

Keygen() → sk, vk
vk $= \mathbf{A} \cdot$ sk, for short sk
sk$_1$
sk$_2$
sk$_3$
Sign
Sample a short $\mathbf{r}_1$
$\mathbf{w}_1 = A \cdot \mathbf{r}_1$
$\mathbf{c} = \mathrm{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
$\mathbf{z}_1 = \mathbf{c} \cdot \mathrm{sk}_1 + \mathbf{r}_1$
Output $\mathbf{c}, \mathbf{z}_1$
Sign
Sample a short $\mathbf{r}_2$
$\mathbf{w}_2 = A \cdot \mathbf{r}_2$
$\mathbf{c} = \mathrm{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
$\mathbf{z}_2 = \mathbf{c} \cdot \mathrm{sk}_2 + \mathbf{r}_2$
Output $\mathbf{c}, \mathbf{z}_2$
Sign
Sample a short $\mathbf{r}_3$
$\mathbf{w}_3 = A \cdot \mathbf{r}_3$
$\mathbf{c} = \mathrm{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
$\mathbf{z}_3 = \mathbf{c} \cdot \mathrm{sk}_3 + \mathbf{r}_3$
Output $\mathbf{c}, \mathbf{z}_3$
PartialVerify
$\mathbf{w}* = A \cdot \mathbf{z}_1 - c \cdot \mathrm{vk}_1$
Assert $\mathbf{c} = \mathrm{Hash}(m, \mathbf{w}*)$
Assert $\mathbf{z}_1$ short
Combine
Output $(\mathbf{c}, \mathbf{z}_1 + \mathbf{z}_2 + \mathbf{z}_3)$
PartialVerify
$\mathbf{w}* = A \cdot \mathbf{z}_3 - c \cdot \mathrm{vk}_3$
Assert $\mathbf{c} = \mathrm{Hash}(m, \mathbf{w}*)$
Assert $\mathbf{z}_3$ short
Verify
$\mathbf{w}* = A \cdot \mathbf{z} - c \cdot \mathrm{vk}$
Assert $\mathbf{c} = \mathrm{Hash}(m, \mathbf{w}*)$
Assert $\mathbf{z}$ short

What is happening here?

share of signature = signature of share

required assumption on dkg/key sharing

$\Rightarrow$ all users have generated and distributed/exchanged keys securely.

$\Rightarrow$ their secret keys and their reconstruction coefficients are short

Achievable by *ramp secret sharing / distributed secret sharing* techniques

$$
\begin{tikzcd}
pk \arrow[dd, bend right=60, dotted, "\texttt{Verify}"'] & & \llbracket pk \rrbracket = (pk_1, \dots, pk_N) \arrow[dd, bend left=60, dashed, "\texttt{Verify}"] \\
s_k \arrow[u, "As + e"] \arrow[rr, bend left, "\texttt{Share}"] \arrow[d, "\texttt{Sign}"] & & \llbracket s \rrbracket = (s_1, \dots, s_N) \arrow[ll, bend left, "\texttt{Reconstruct}"] \arrow[u, "As_i + e_i"'] \arrow[d, "\texttt{Sign}"'] \\
(\mathbf{z}, \mathbf{y}) = \texttt{Sign}(sk, m) \arrow[rr, bend left, "\texttt{Share}"] & & \llbracket (\mathbf{z}, \mathbf{y}) \rrbracket = (\texttt{Sign}(s_i, m))_i \arrow[ll, bend left, "\texttt{Reconstruct}"]
\end{tikzcd}
$$

Short look at the norm verification

$$\text{sig} = (c, \mathbf{z}) \text{ where } \mathbf{z} = \sum_i \mathbf{z}_i$$

$$\|\mathbf{z}\|^2 = \sum_i \|\mathbf{z}_i\|^2 + \sum_{i \neq j} \langle \mathbf{z}_i, \mathbf{z}_j \rangle$$

$\leq B_{part}^2$ thanks to PartialVerify

(≈ fully honest case)

Can *not* assume $\mathbf{z}_i$'s are Gaussians (=honest).

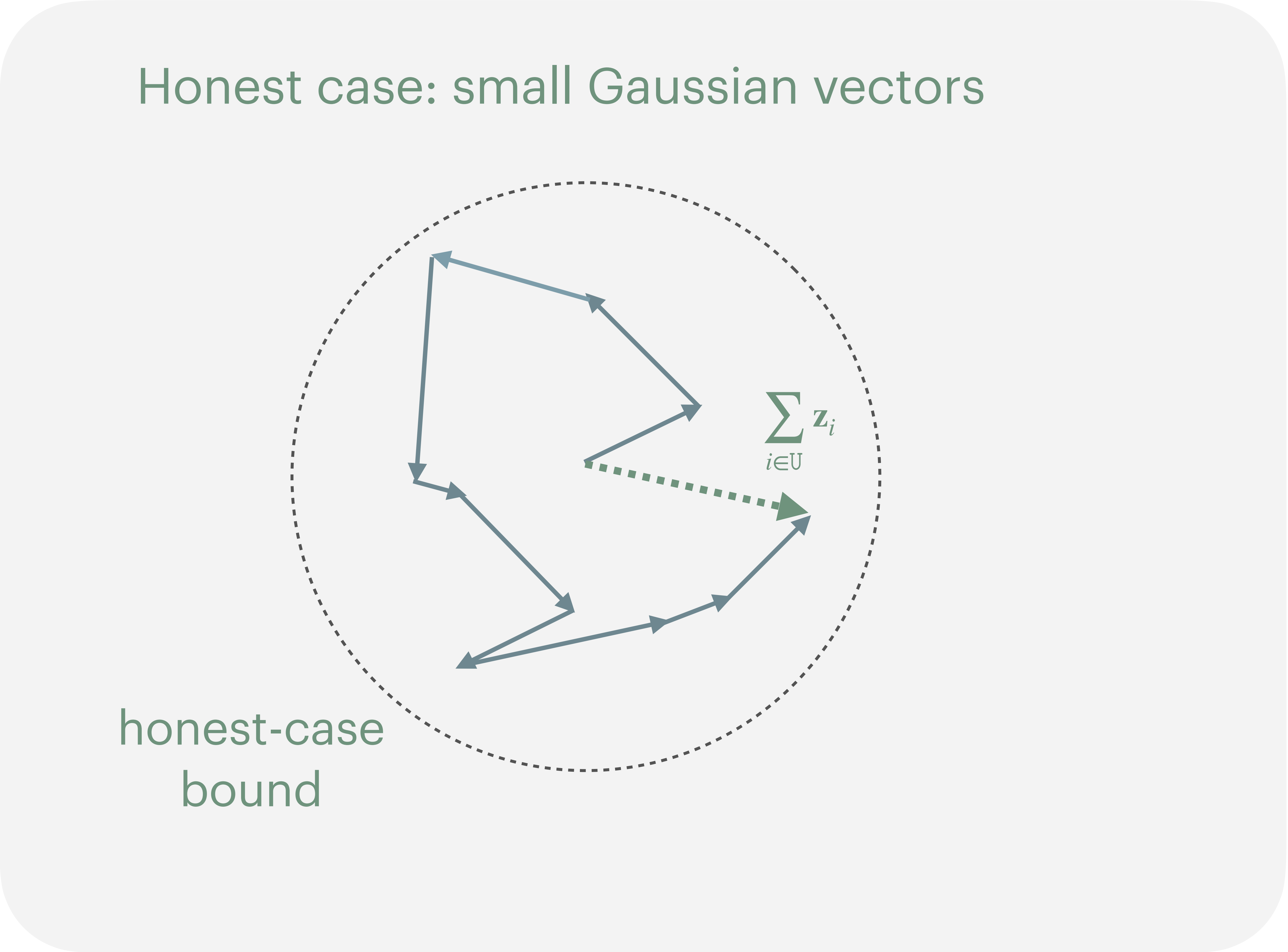
This corresponds to malicious users could

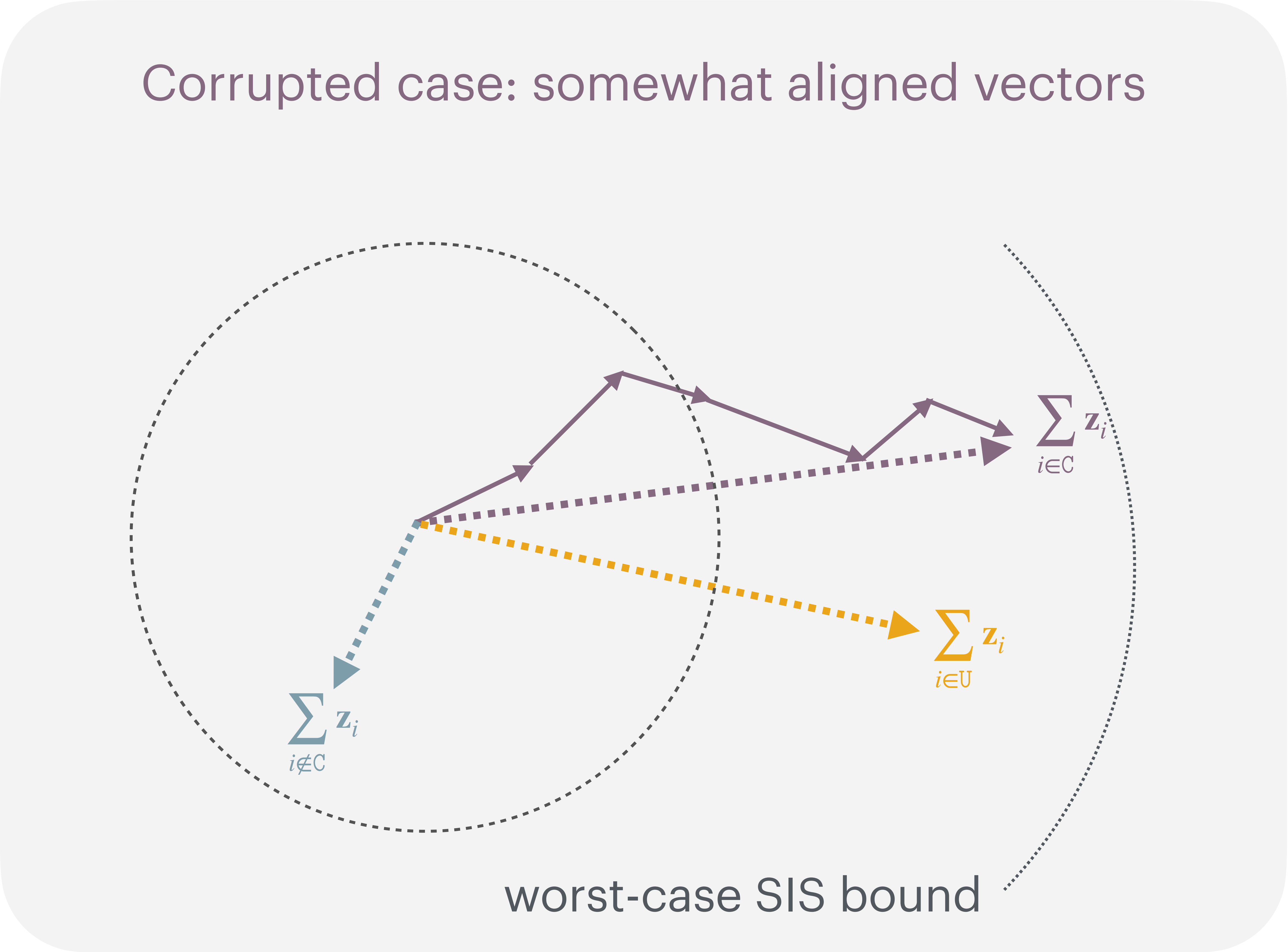
align their $\mathbf{z}_i$'s.

(what SIS bound must cover)

How malicious users can break it

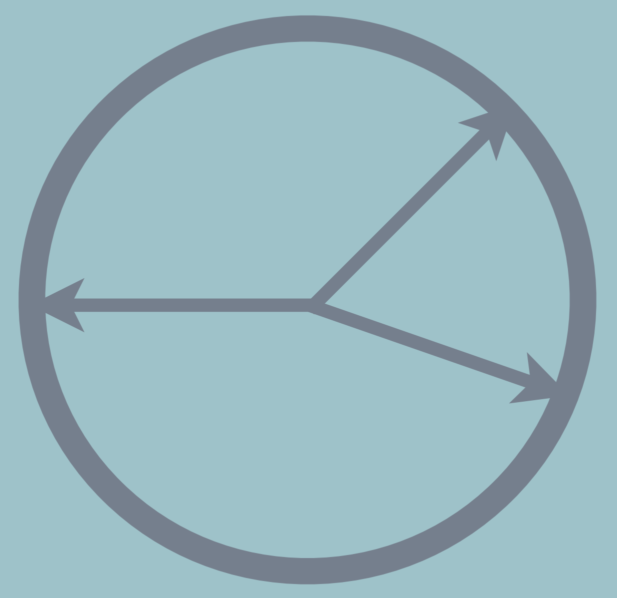
idea: a subset C of malicious users *collude* >> can pass PartialVerify, but fail the global one





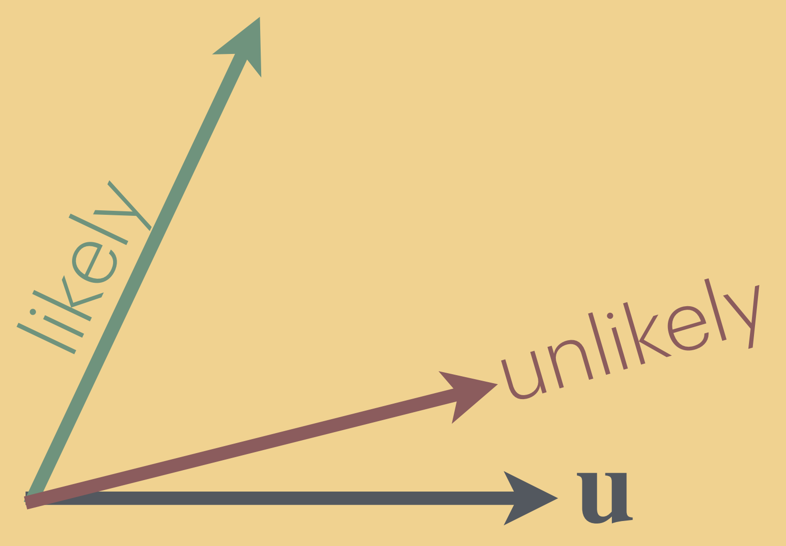
What can we say about honest vectors?

Gaussian vectors are in a narrow *spherical crust.*



$\Rightarrow \|\mathbf{z}\| \in \sigma\sqrt{n} \cdot [1-\delta, 1+\delta]$ with overwhelming probability.

direction of a Gaussian is uniformly distributed.



$\Rightarrow \mathbb{P}_{\mathbf{z} \leftarrow \mathscr{D}_\sigma}\left[\langle \mathbf{z}, \mathbf{u} \rangle \geq \sigma\sqrt{\ell}\right] \leq 2^{-\Omega(\ell)}$

for any unit vector $\mathbf{u}$

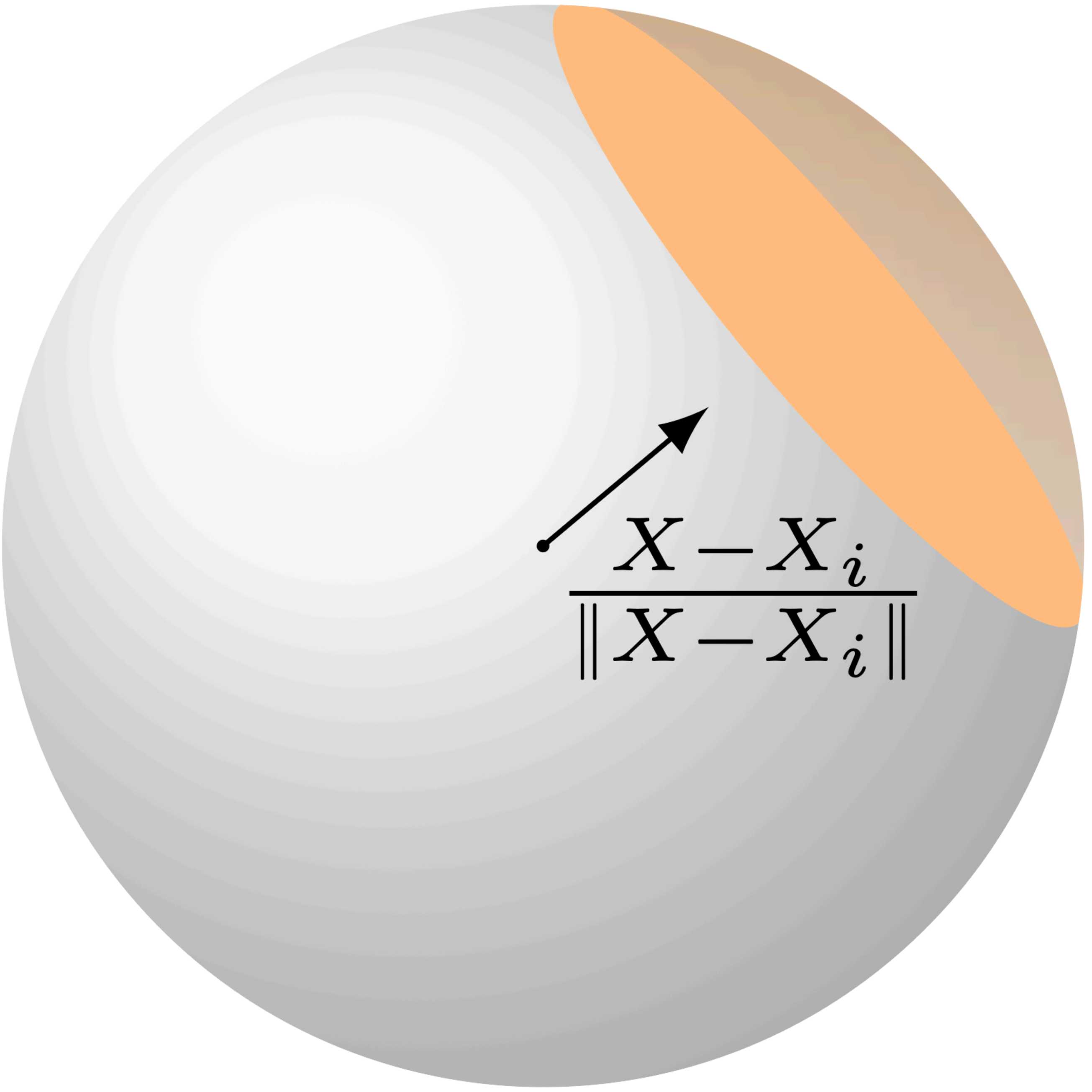
honest signatures : $\|\mathbf{z}_i\| \approx O(\sigma\sqrt{n})$ and $\left\| \sum_{\mathrm{U}} \mathbf{z}_i \right\| \approx O(\sigma\sqrt{Nn})$

vector *correlating too much* with the final signature is likely to be *corrupted*

The D* test $N\sigma\sqrt{n} \cdot (1+o(1)) \longrightarrow N\sigma\sqrt{\rho n} \cdot (1+o(1))$

D* identifier

1. traitors $= \varnothing$
2. **For** $i \in$ U
   - **If** $\|\mathbf{z}_i\| > (1+\delta)\sigma\sqrt{n}$, **put user** i **in** traitors
   - **Else if** $\dfrac{\langle \mathbf{z}_i, \mathbf{z} - \mathbf{z}_i \rangle}{\|\mathbf{z} - \mathbf{z}_i\|} > \sigma\sqrt{\ell}$, **put user** i **in** traitors
3. **Return** traitors.



get 10-20 bits of security — *for free*

Beyond Raccoon : back to rejection sampling

signatures are too big for passing MLDSA verify —> add *rejection*

motto : "reject until the distribution of signatures is small enough "

no free lunch: the further away my starting distribution is, the more reject I have to do.

Beyond Raccoon : back to rejection sampling

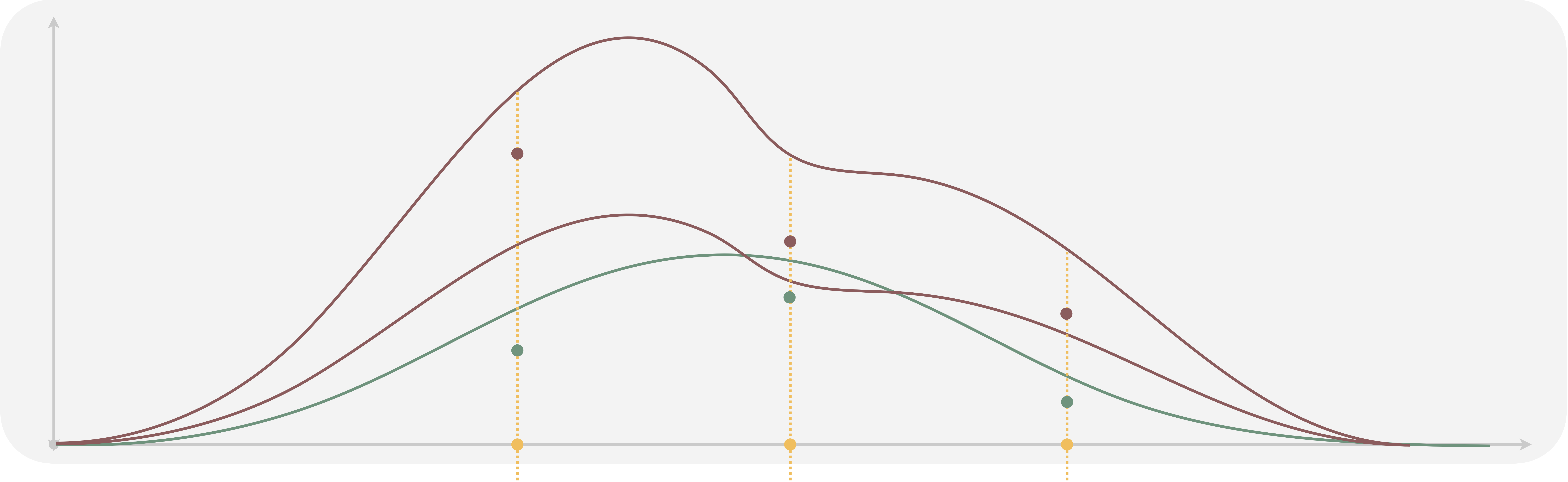
Rejection sampling

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

Start from $\mathbf{v}$, add mask $\mathbf{r} \sim \chi_{\mathbf{r}}$, targeting $\chi_{\mathbf{z}}$

$$\mathsf{Rej}(\mathbf{v}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M) \sim \big(\chi_{\mathbf{z}} \,|\, \mathscr{B}(1/M)\big)$$

$\rightarrow$ distribution of $\mathbf{z}$ is *independent* of the secret value $\mathbf{v}$




Keygen() → sk, vk
$\mathsf{vk} = \mathbf{A} \cdot \mathsf{sk}$, for short sk
sk_1
sk_2
sk_3
Sign
Sample a short $\mathbf{r}_1$
$\mathbf{w}_1 = A \cdot \mathbf{r}_1$
$\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
Rejection sampling
$\mathbf{z}_1 = \mathsf{Rej}(c \cdot \mathsf{sk}_1, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_1)$
Sign
Sample a short $\mathbf{r}_2$
$\mathbf{w}_2 = A \cdot \mathbf{r}_2$
$\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
Rejection sampling
$\mathbf{z}_2 = \mathsf{Rej}(c \cdot \mathsf{sk}_2, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_2)$
Sign
Sample a short $\mathbf{r}_3$
$\mathbf{w}_3 = A \cdot \mathbf{r}_3$
$\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
Rejection sampling
$\mathbf{z}_3 = \mathsf{Rej}(c \cdot \mathsf{sk}_3, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_3)$
Combine
Output $(\mathbf{c}, \mathbf{z}_1 + \mathbf{z}_2 + \mathbf{z}_3)$


Keygen() $\rightarrow$ sk, vk
$\mathsf{vk} = \mathbf{A} \cdot \mathsf{sk}$, for short sk
sk_1
sk_2
sk_3
Sign
Sample a short $\mathbf{r}_1$
$\mathbf{w}_1 = A \cdot \mathbf{r}_1$
$\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
Rejection sampling
$\mathbf{z}_1 = \mathsf{Rej}(c \cdot \mathsf{sk}_1, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_1)$
Sign
Sample a short $\mathbf{r}_2$
$\mathbf{w}_2 = A \cdot \mathbf{r}_2$
$\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
Rejection sampling
$\mathbf{z}_2 = \mathsf{Rej}(c \cdot \mathsf{sk}_2, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_2)$
Sign
Sample a short $\mathbf{r}_3$
$\mathbf{w}_3 = A \cdot \mathbf{r}_3$
$\mathbf{c} = \mathsf{Hash}(m, \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3)$
Rejection sampling
$\mathbf{z}_3 = \mathsf{Rej}(c \cdot \mathsf{sk}_3, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}, M; \mathbf{r}_3)$
Combine
if $\mathbf{z}_1 + \mathbf{z}_2 + \mathbf{z}_3$ too large, reject
Output $(\mathbf{c}, \mathbf{z}_1 + \mathbf{z}_2 + \mathbf{z}_3)$

Beyond Raccoon : back to rejection sampling

signatures are too big for passing MLDSA verify —> add *rejection*

⇒ Needs to carefully control the parameters of the rejection

✔ Works !

✗ Can't scale beyond 6 users with the current technology

Quantitatively : ThMLDSA

Needs a few more tricks to get under MLDSA verification

- Unbalanced rejection sampling on hyperballs
- Parallel repetitions

scales … quite badly
But … is *compatible* with the standard ML-DSA !

- 1st round can be done offline (independent of the message)
- An existing MLDSA key can be shared (amounts to sample a sharing of zero)
- *Compatible* but not *indistinguishable* : the distribution of signature is not the same same as the original MLDSA

Communication cost for Th-MLDSA at N parties with threshold T

N \ T	**2**	**3**	**4**	**5**	**6**
2	10.5 kB				
3	15.8 kB	21.0 kB			
4	15.8 kB	36.8 kB	42.0 kB		
5	15.8 kB	73.5 kB	157.4 kB	84.0 kB	
6	21.0 kB	99.8 kB	388.4 kB	524.8 kB	194.2 kB

Timing for Th-MLDSA on a MacBook M3

(T, N)	**KeyGen** (ms)	**Sign+Combine** (ms)	**Verify** (ms)
Threshold ML-DSA			
(3,3)	0.3669	0.6810	0.0306
(2,4)	0.1709	0.4570	
(3,4)	0.2062	1.0961	
(4,4)	0.1655	1.3672	
(3,5)	0.2870	2.2263	
(4,5)	0.2940	4.9832	
(5,5)	0.1956	2.8453	
(4,6)	0.5016	12.1949	
(6,6)	0.2181	7.6784	