Flood and Submerse:

Distributed Key Generation and Robust Threshold Signature from Lattices

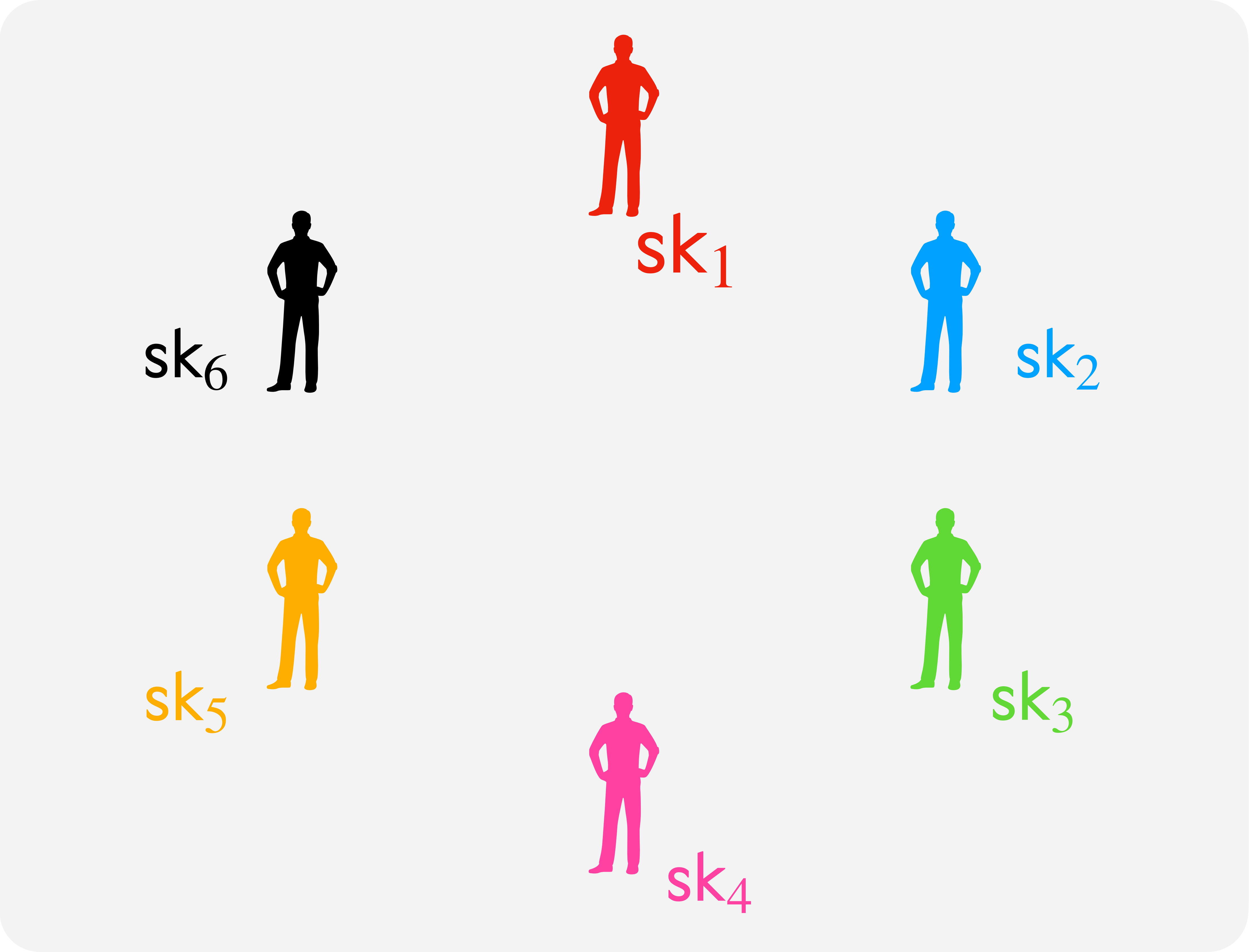
Thomas Espitau
Guilhem Niot
Thomas Prest

PQ SHIELD

1. Background

(T-out-of-N) threshold signatures
What are they?

An interactive protocol to distribute signature generation.

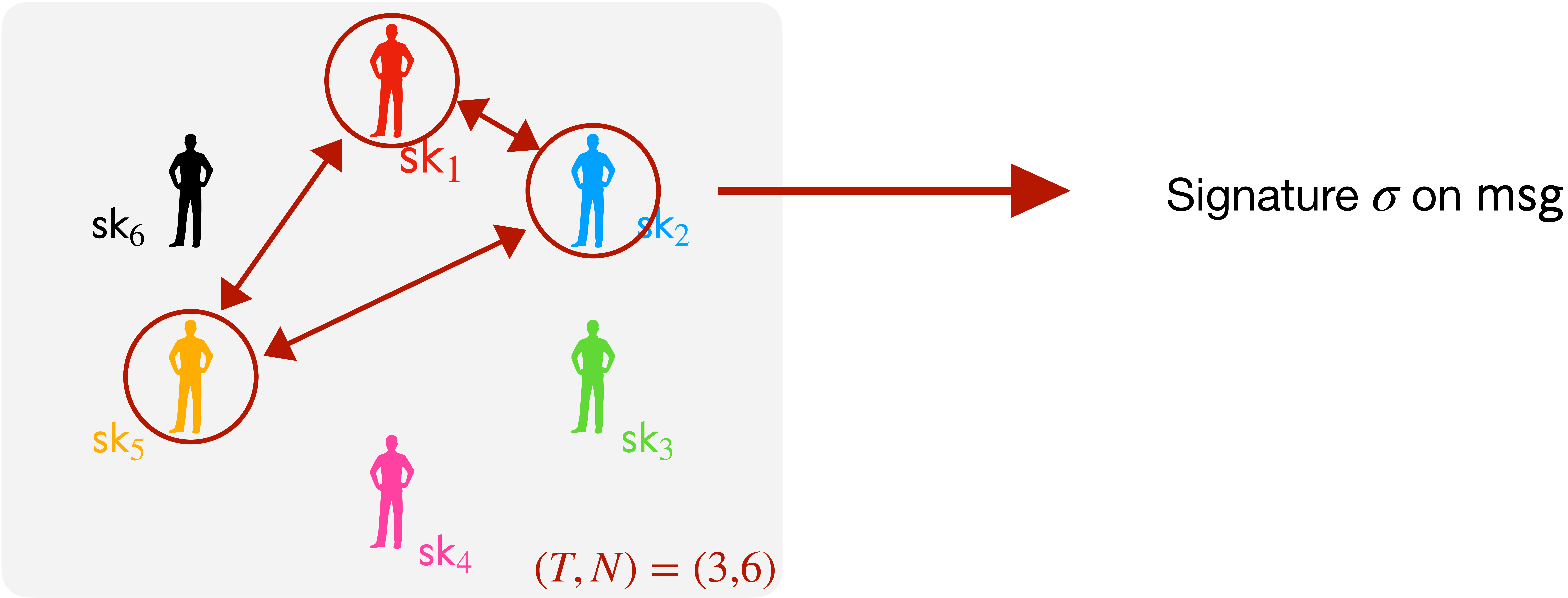


- 1 verification key vk
- 1 partial signing key sk_i per party
- Given at least T-out-of-N partial signing keys, we can sign.

(T-out-of-N) threshold signatures

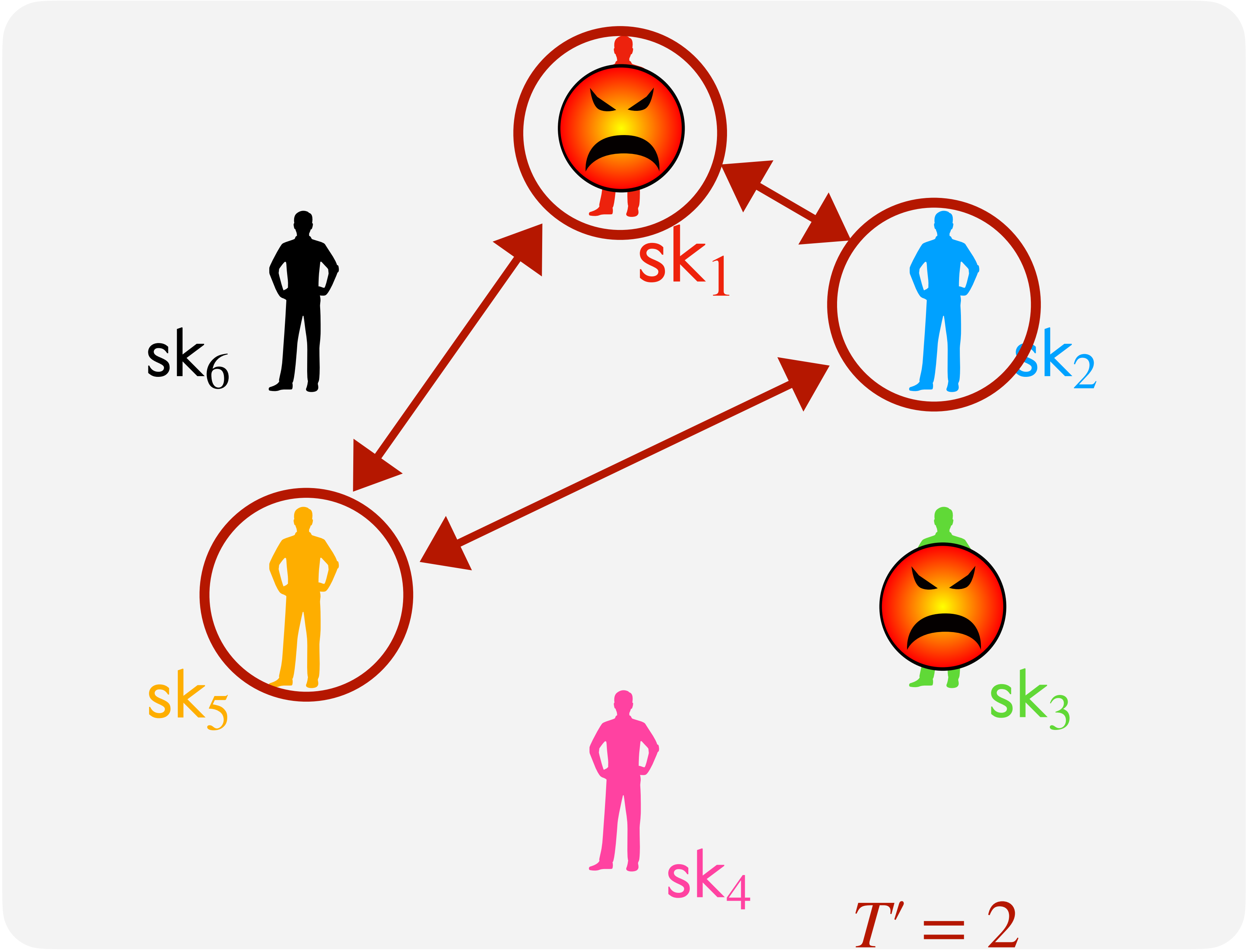
What are they?

An interactive protocol to distribute trust.



Security properties

- **Correctness:** Given at least T-out-of-N partial signing keys, we can sign.
- **Unforgeability:** The signature scheme remains unforgeable even if up to $T' < T$ parties are corrupted. Often $T' = T - 1$.



It's not possible to forge a new signature, even by taking part in the signing protocol.

An active field of research

- Aggregating hash-based signatures: [KCLM22]
- Sequential TS scheme based on isogenies: [DM20]
- Lattice-based threshold signatures:
  - 2-round TS via FHE: [BGG+18], [ASY22], [GKS23]
  - TS with noise flooding: 3-round [dPKM+23], 2-round [EKT24], [BKLM+24]

An active field of research

- Aggregating hash-based signatures: [KCLM22]
- Sequential TS scheme based on isogenies: [DM20]
- Lattice-based threshold signatures:
  - 2-round TS via FHE: [BGG+18], [ASY22], [GKS23]
  - TS with noise flooding: 3-round [dPKM+23], 2-round [EKT24], [BKLM+24]

Open problems for lattice-based schemes

- **Distributed Key Generation** (DKG)
- **Robustness**: Guarantee valid signature in the presence of malicious signers

Our techniques for DKG + robust signing are quite generic:

- in our paper, applied to Plover [EENP+24]: hash-and-sign scheme
- can be applied to all **3-round [dPKM+23]**, 2-round [EKT24], [BKLM+24]

Threshold Raccoon [dPKM+23]

Lyubashevsky's signature scheme (without aborts)

$$\mathsf{vk} = \boxed{\mathrm{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathscr{R}_q^k \qquad \mathsf{sk} = \boxed{\mathbf{s}} \in \mathscr{R}_q^\ell \text{ short}$$

$\mathbf{r} \leftarrow \chi$

$\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$ $\xrightarrow{\mathbf{w}}$

$\longleftarrow$ $c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) \in \mathscr{R}_q$ "small"

$\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$ $\longrightarrow$ Accept if

- $\mathbf{z}$ is short
- $\mathbf{A} \cdot \mathbf{z} = c \cdot \mathbf{t} + \mathbf{w}$

Secure if $\|\mathbf{r}\| = \mathcal{O}(\sqrt{Q_s} \cdot \|c\|)$

Threshold Raccoon [dPKM+23]

Threshold signature: use (T, N)-Shamir sharing on secret

$$\mathsf{sk} = \mathbf{s} \in \mathcal{R}_q^{\ell} \quad \text{short}$$

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

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

For any set S of T shares, reconstruct $\mathbf{s}$:

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

Lagrange coefficient

Threshold Raccoon [dPKM+23]

Threshold signature: use (T, N)-Shamir sharing on secret

sk = $\boxed{\mathbf{s}} \in \mathcal{R}_q^{\ell}$ short

For any set S of T shares, reconstruct $\mathbf{s}$:

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

$\mathbf{r} \leftarrow \chi$

$\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$

$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$

$\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$

Threshold Raccoon [dPKM+23]

Threshold signature: use (T, N)-Shamir sharing on secret

sk = $\boxed{\mathbf{s}} \in \mathcal{R}_q^{\ell}$ short

For any set S of T shares, reconstruct $\mathbf{s}$:

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

$\mathbf{r} \leftarrow \chi$

$\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$

$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$

$\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$

$\mathbf{r_i} \leftarrow \chi$

$\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r_i}$

$\xrightarrow{\mathsf{cmt}_i = H(\mathbf{w_i})}$

$\xleftarrow[(\mathsf{cmt}_j)_{j \in S}]{}$

$\xrightarrow{\mathbf{w}_i}$

$\xleftarrow[(\mathbf{w}_j)_{j \in S}]{}$

Threshold Raccoon [dPKM+23]

Threshold signature: use (T,N)-Shamir sharing on secret

sk = $\boxed{\mathbf{s}} \in \mathcal{R}_q^{\ell}$ short

For any set S of T shares, reconstruct $\mathbf{s}$:

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

$\mathbf{r} \leftarrow \chi$

$\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$

$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$

$\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$

$\mathbf{r_i} \leftarrow \chi$

$\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r_i}$

$\xrightarrow{\mathsf{cmt_i} = H(\mathbf{w_i})}$

$\xleftarrow[(\mathsf{cmt}_j)_{j \in S}]{}$

$\xrightarrow{\mathbf{w}_i}$

$\xleftarrow[(\mathbf{w}_j)_{j \in S}]{}$

$\mathbf{w} = \sum_{j \in S} \mathbf{w_j}$

$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$

$[\![\mathbf{z}]\!]_i = c \cdot L_{S,i} \cdot [\![\mathbf{s}]\!]_i + \mathbf{r}_i + \Delta_i \xrightarrow{\mathbf{z}_i}$

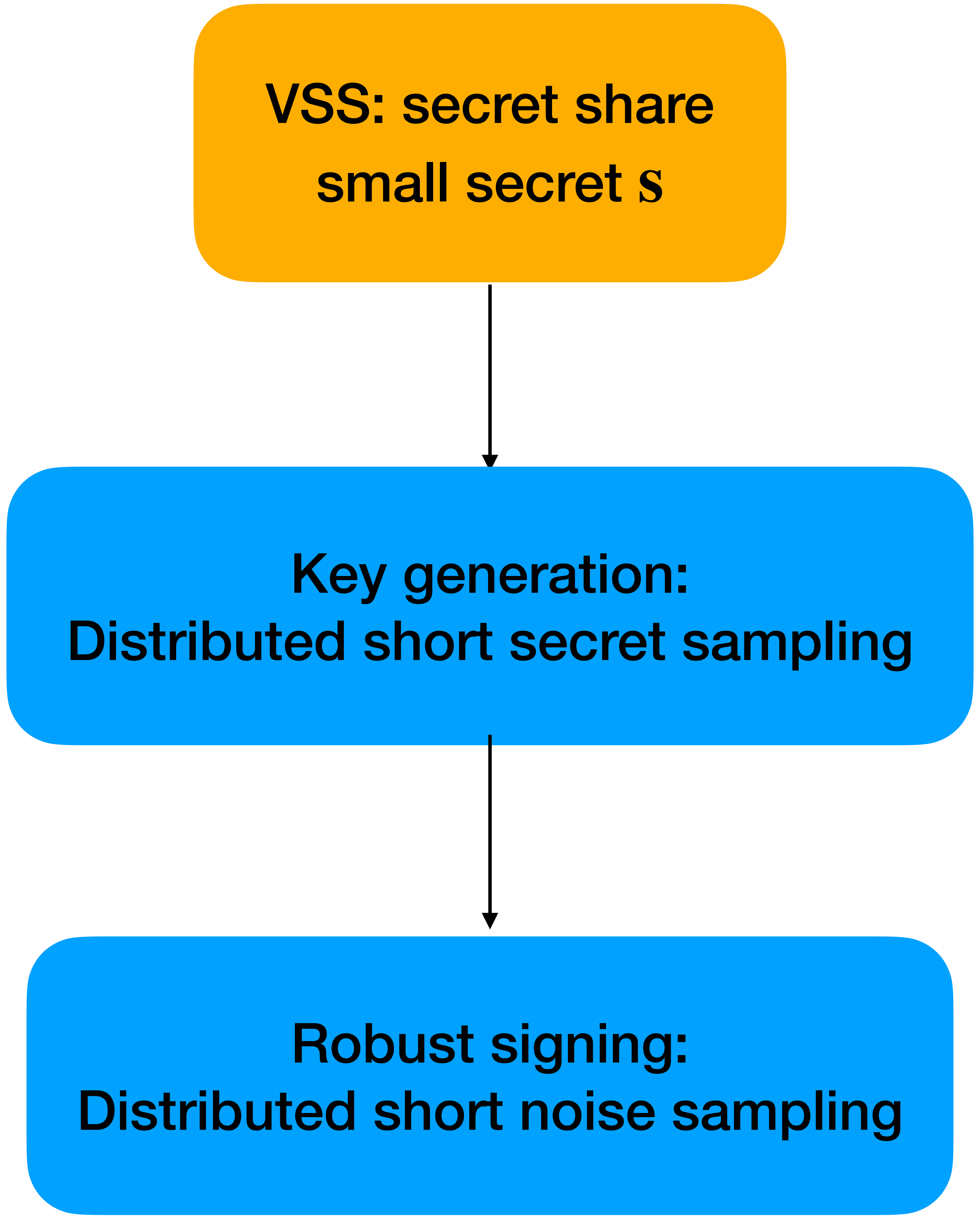
Share of 0

Accept if

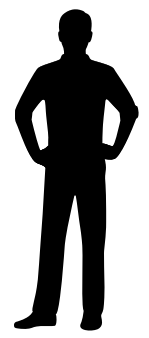
- $\mathbf{z} = \sum_{j \in S} [\![\mathbf{z}]\!]_j = c \cdot \mathbf{s} + \sum_{j \in S} \mathbf{r}_j$ is short
- $\mathbf{A} \cdot \mathbf{z} = c \cdot \mathbf{t} + \mathbf{w}$

2. Achieving additional threshold properties with Verifiable Secret Sharing

Achieving additional threshold properties with Verifiable Secret Sharing

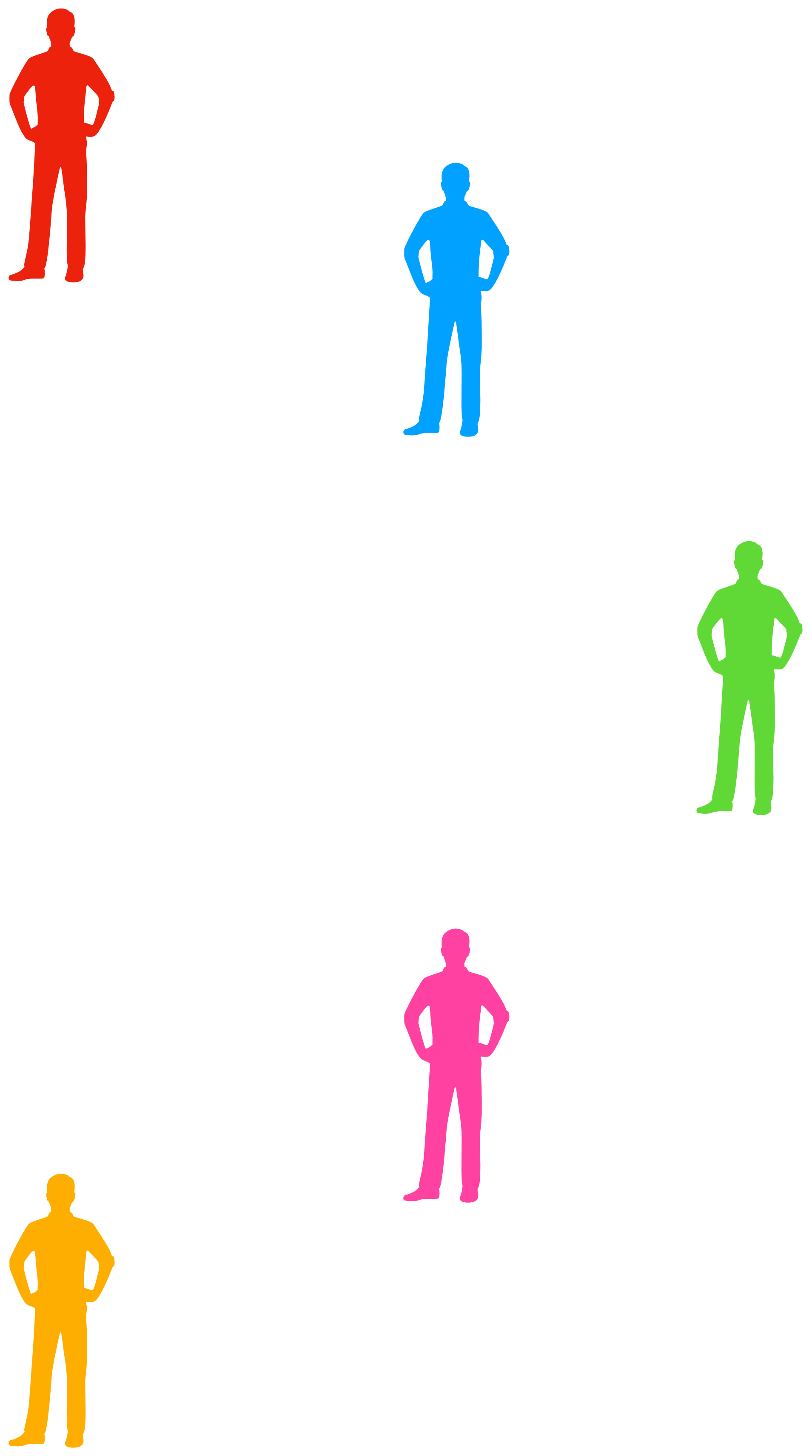


Verifiable Secret Sharing (VSS)



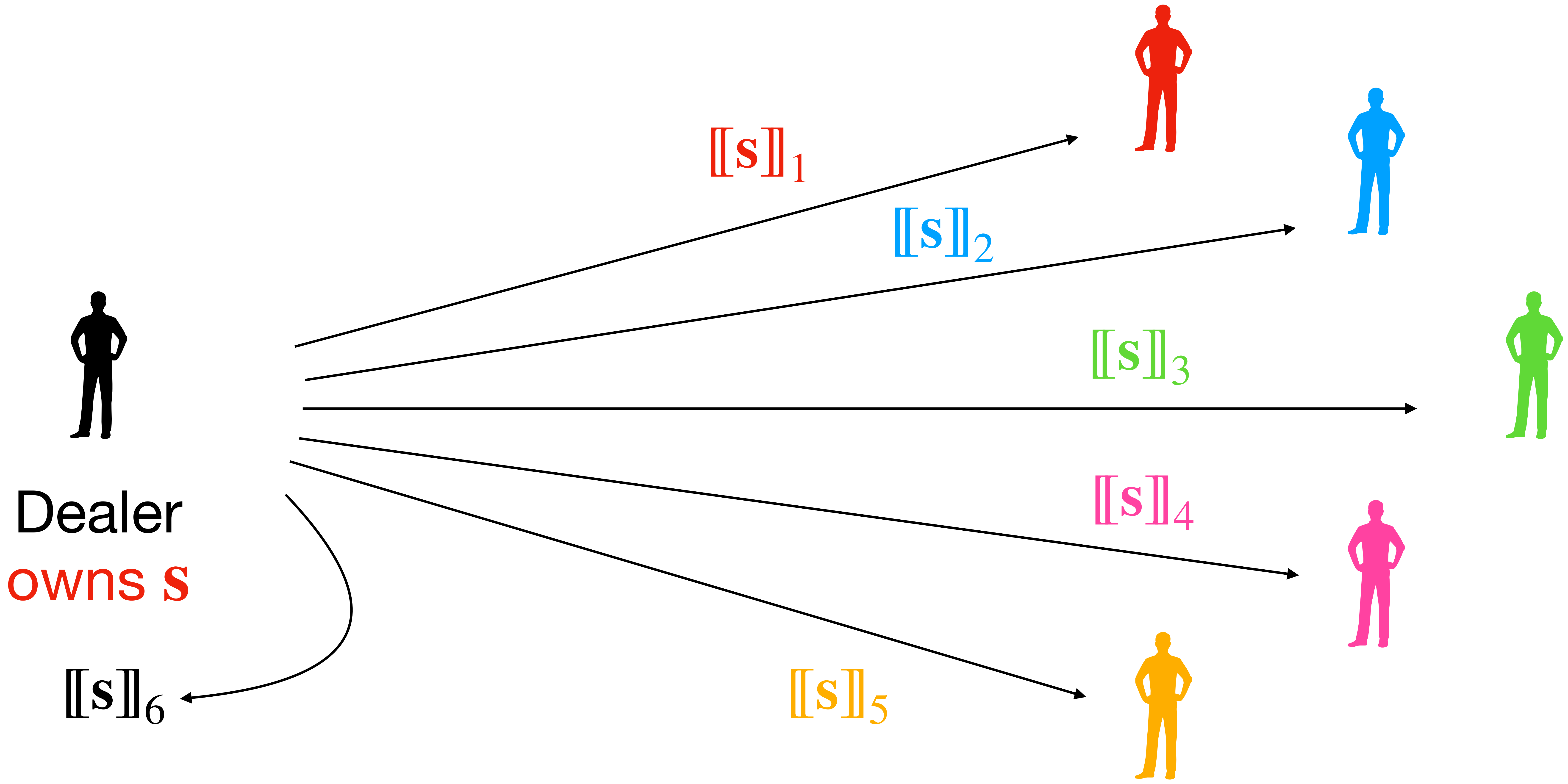
Dealer

owns s



Verifiable Secret Sharing (VSS)

1) Send individual shares

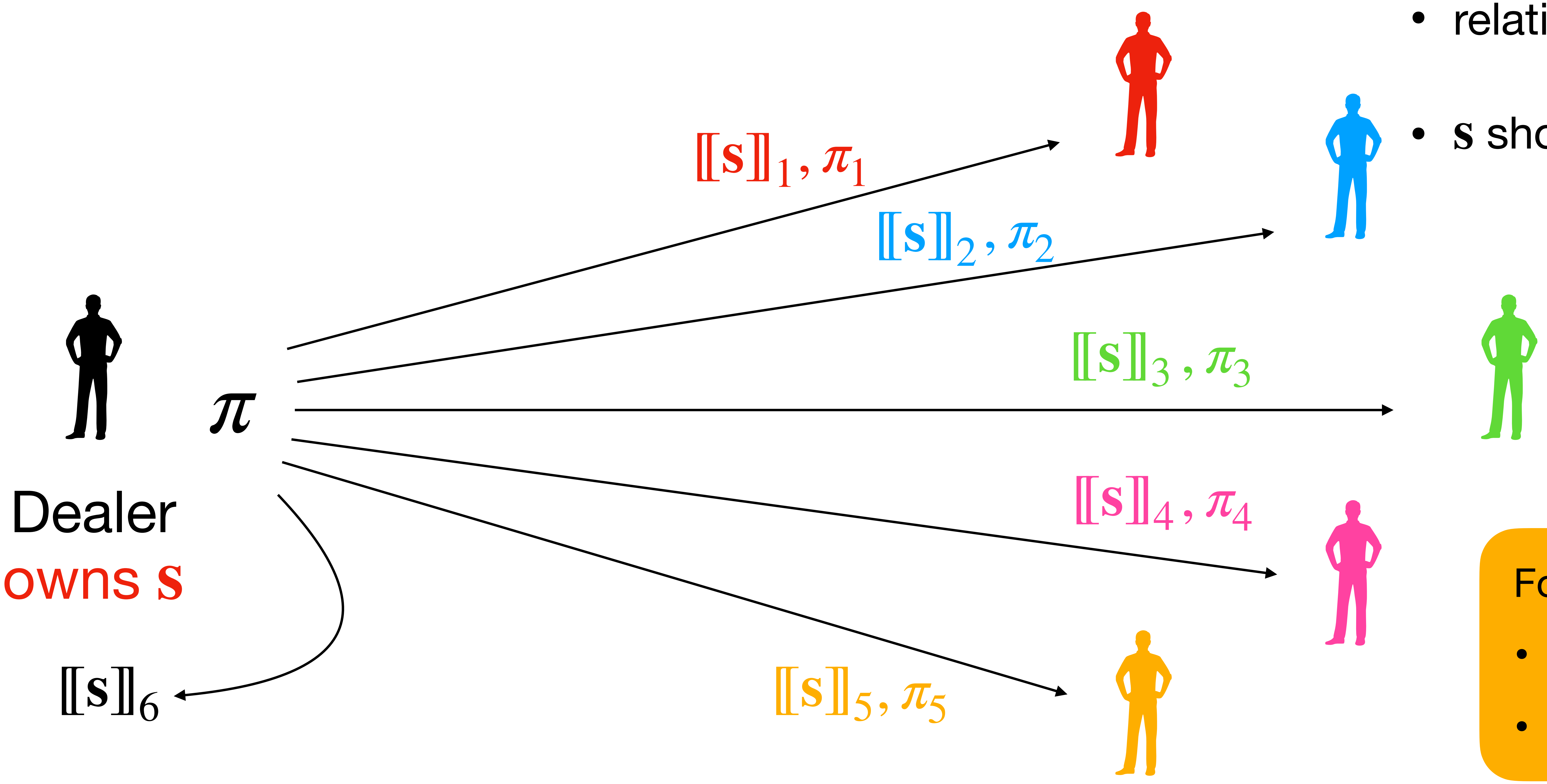


Verifiable Secret Sharing (VSS)

1) Send individual shares

2) Prove correct sharing, i.e.

- relation $\mathbf{s} = \sum_{i \in S} L_{S,i} \cdot [\![\mathbf{s}]\!]_i$ for $|S| = T$
- $\mathbf{s}$ short



Formally,

- VSS . Share$(\mathbf{s}) \to (\pi, ([\![\mathbf{s}]\!]_i, \pi_i)_{1 \le i \le N})$
- VSS . Verify$(i, [\![\mathbf{s}]\!]_i, \pi, \pi_i) \to \mathbf{ok}\,|\,\mathbf{fail}$

Distributed Key Generation (DKG) from VSS

$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathcal{R}_q^k$$

$$\mathsf{sk} = \boxed{\mathbf{s}} \in \mathcal{R}_q^\ell \quad \text{short}$$

Distributed Key Generation (DKG) from VSS

$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathcal{R}_q^k$$

$$\mathsf{sk} = \boxed{\mathbf{s}} \in \mathcal{R}_q^\ell \quad \text{short}$$

1. Construct and share secret key $\mathbf{s}$

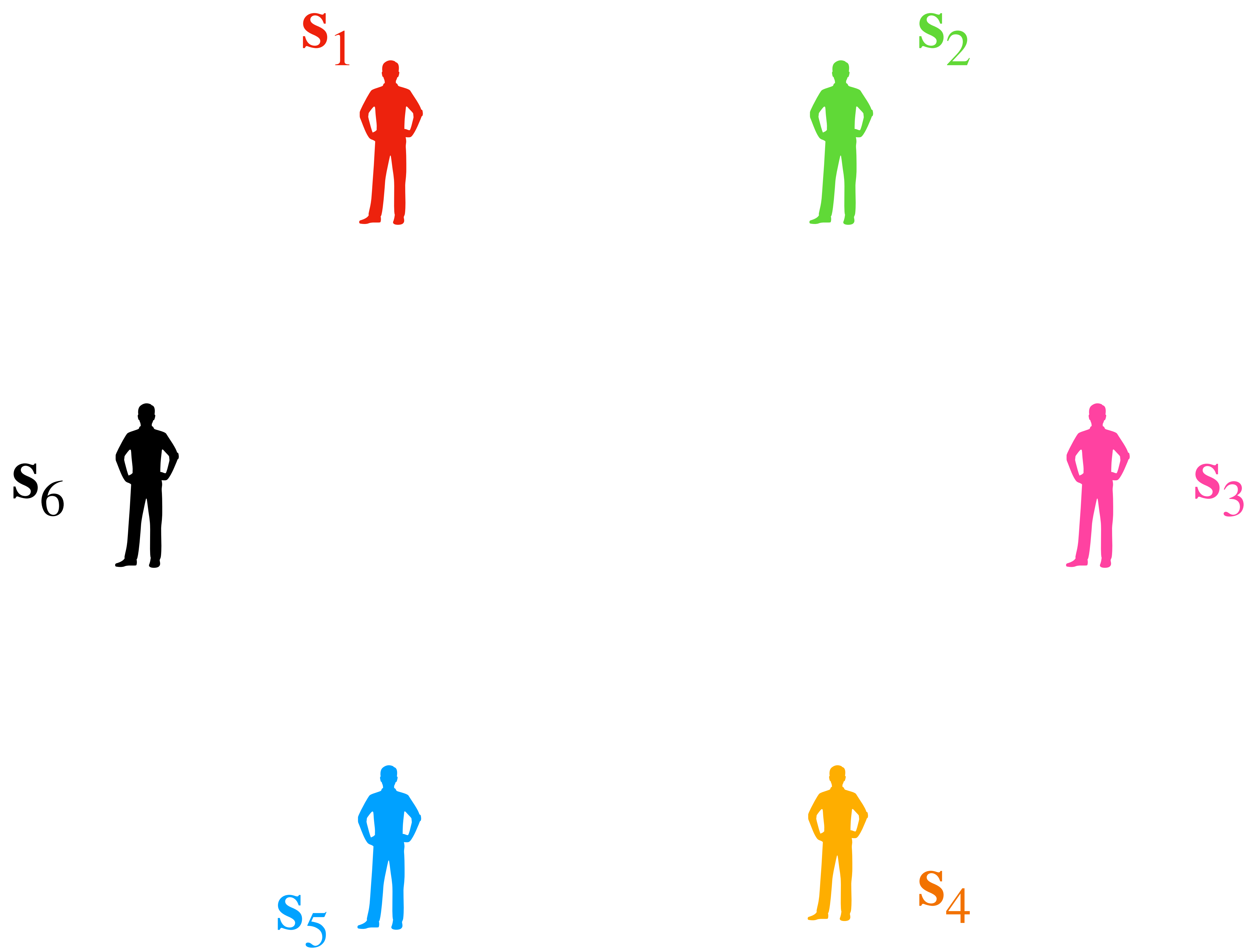
Distributed Key Generation (DKG) from VSS

$$\text{vk} = \boxed{\mathbf{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathscr{R}_q^k$$

$$\text{sk} = \boxed{\mathbf{s}} \in \mathscr{R}_q^\ell \quad \text{short}$$

1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$

   1.a) Sample small secrets $\mathbf{s}_i$



Distributed Key Generation (DKG) from VSS

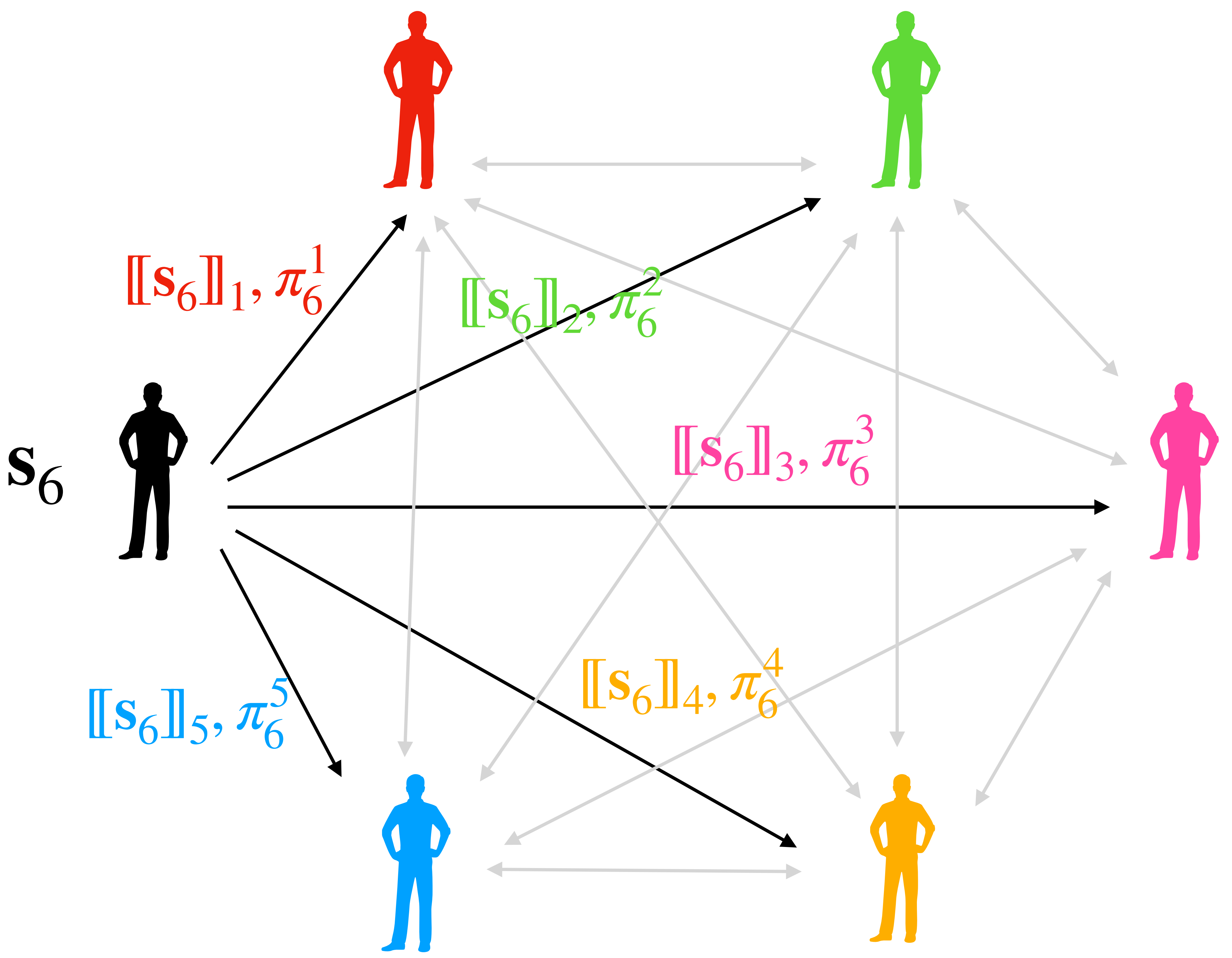
$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\begin{bmatrix} \mathbf{A}' & \mathbf{I} \end{bmatrix}}_{\mathbf{A}} \cdot \mathbf{s} \in \mathscr{R}_q^k$$

$$\mathsf{sk} = \mathbf{s} \in \mathscr{R}_q^\ell \quad \text{short}$$

1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$

   1.a) Sample small secrets $\mathbf{s}_i$

   1.b) Send shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$



Distributed Key Generation (DKG) from VSS

$$\mathrm{vk} = \begin{bmatrix} \mathrm{t} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}' & \mathbf{I} \end{bmatrix}}_{\mathbf{A}} \cdot \begin{bmatrix} \mathbf{s} \end{bmatrix} \in \mathcal{R}_q^k$$

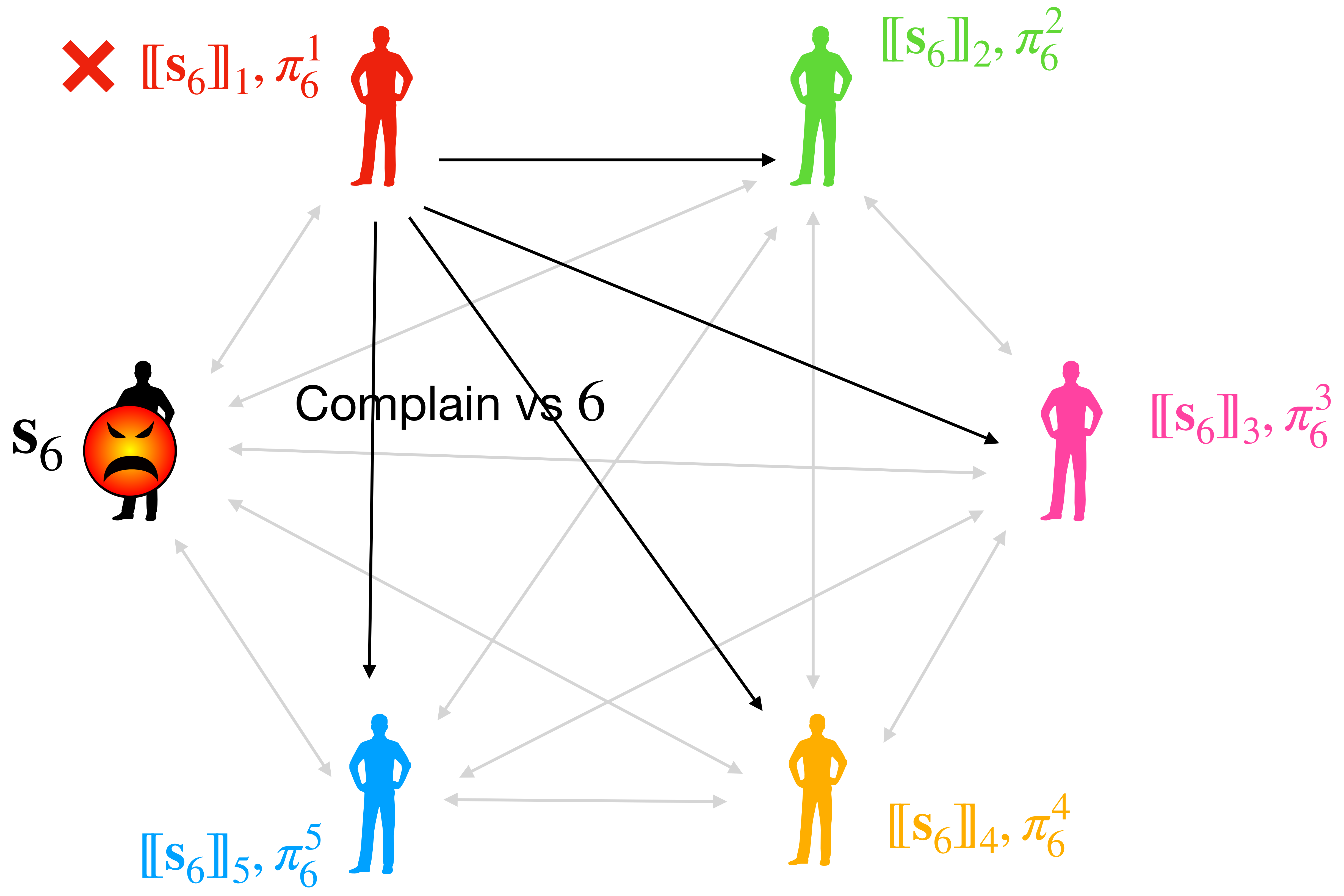
$$\mathrm{sk} = \begin{bmatrix} \mathbf{s} \end{bmatrix} \in \mathcal{R}_q^\ell \quad \text{short}$$

1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$

   1.a) Sample small secrets $\mathbf{s}_i$

   1.b) Send shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$

   1.c) Verify shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$ and complain



Distributed Key Generation (DKG) from VSS

$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathscr{R}_q^k$$

$$\mathsf{sk} = \boxed{\mathbf{s}} \in \mathscr{R}_q^\ell \quad \text{short}$$

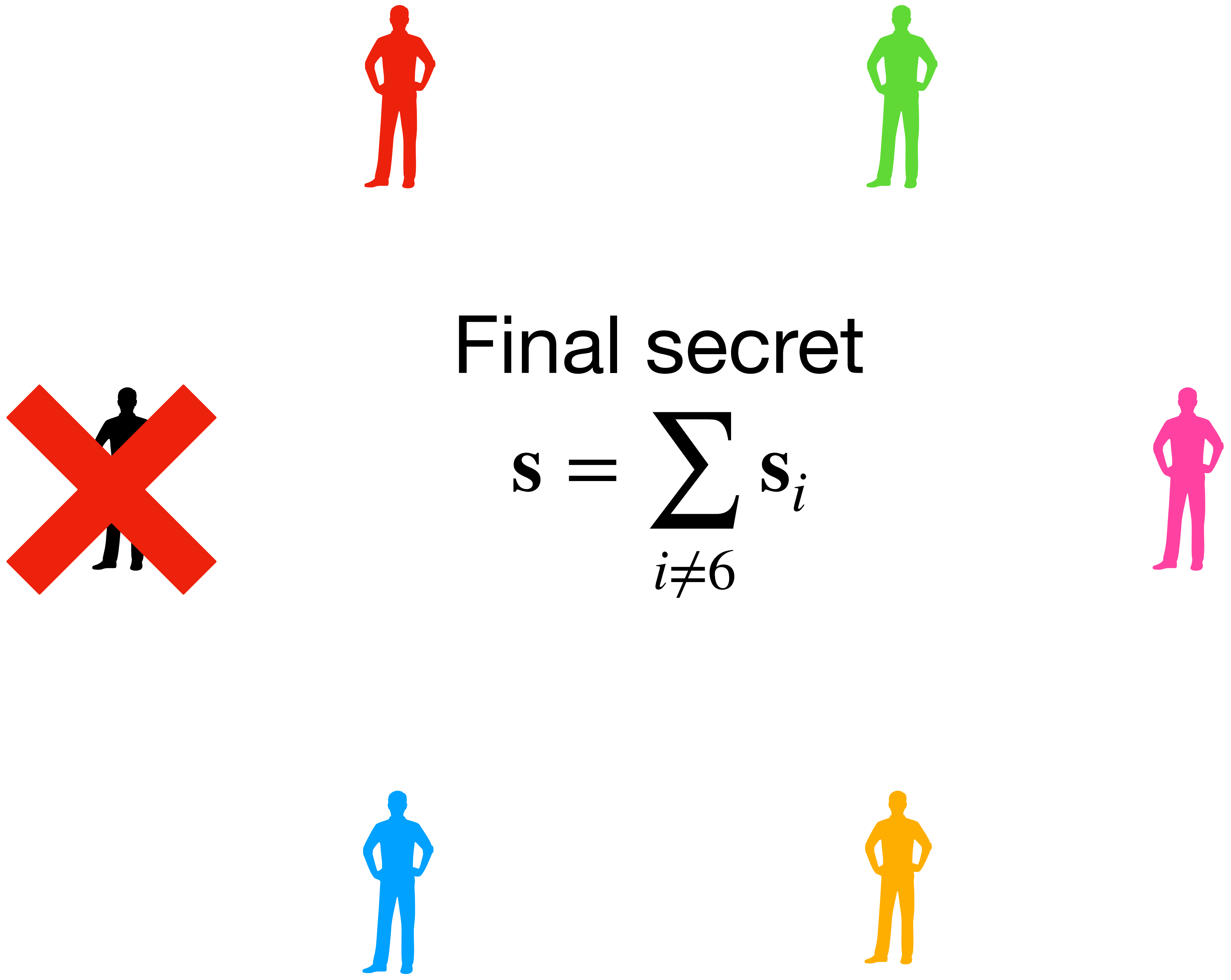
1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$

   1.a) Sample small secrets $\mathbf{s}_i$

   1.b) Send shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$

   1.c) Verify shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$ and complain

   1.d) Aggregate



Distributed Key Generation (DKG) from VSS

$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\begin{bmatrix} \mathbf{A}' & \mathbf{I} \end{bmatrix}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathcal{R}_q^k$$

$$\mathsf{sk} = \boxed{\mathbf{s}} \in \mathcal{R}_q^\ell \quad \text{short}$$

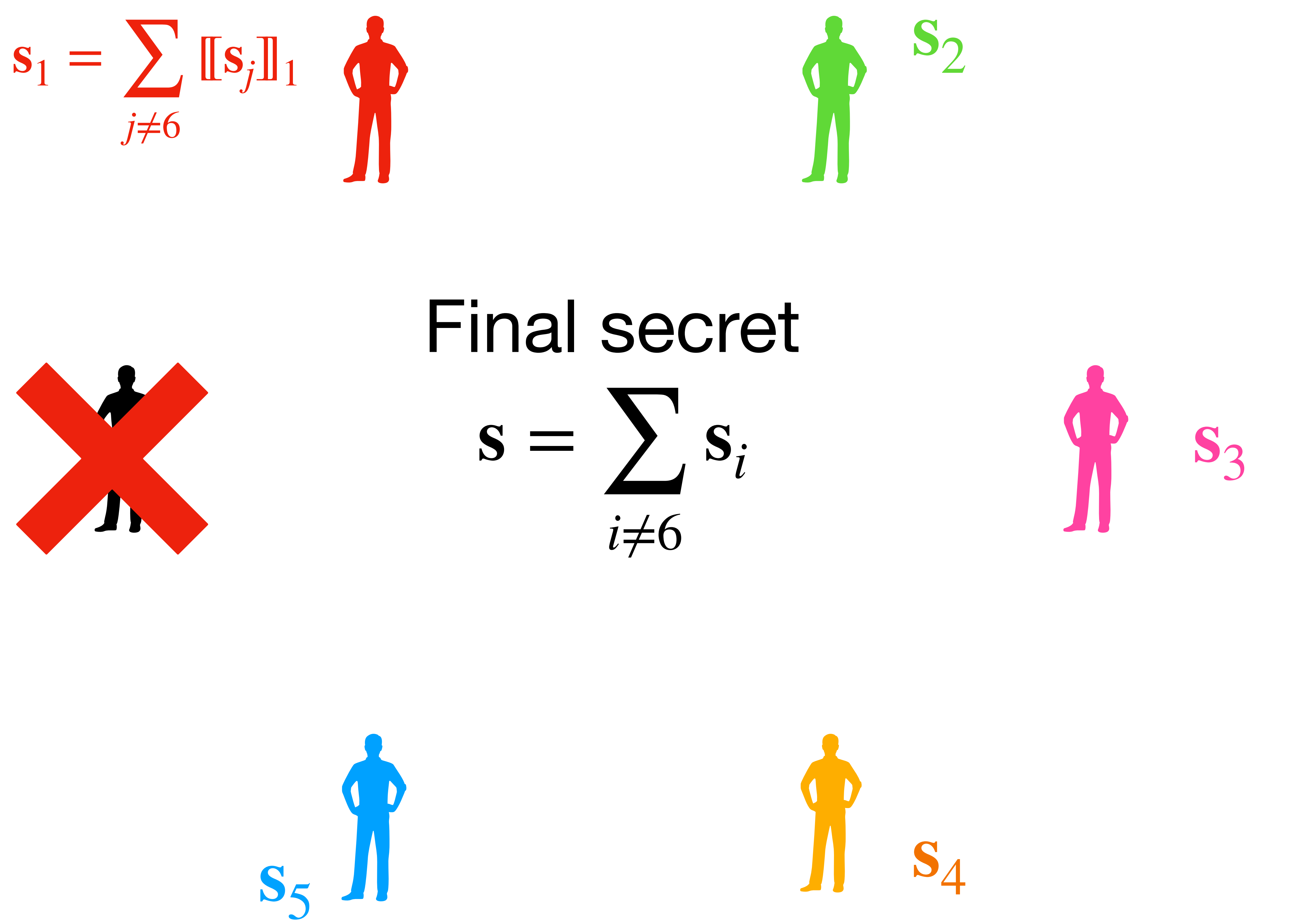
1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$

1.a) Sample small secrets $\mathbf{s}_i$

1.b) Send shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$

1.c) Verify shares $([\![\mathbf{s}_i]\!]_j, \pi_i^j)_j$ and complain

1.d) Aggregate



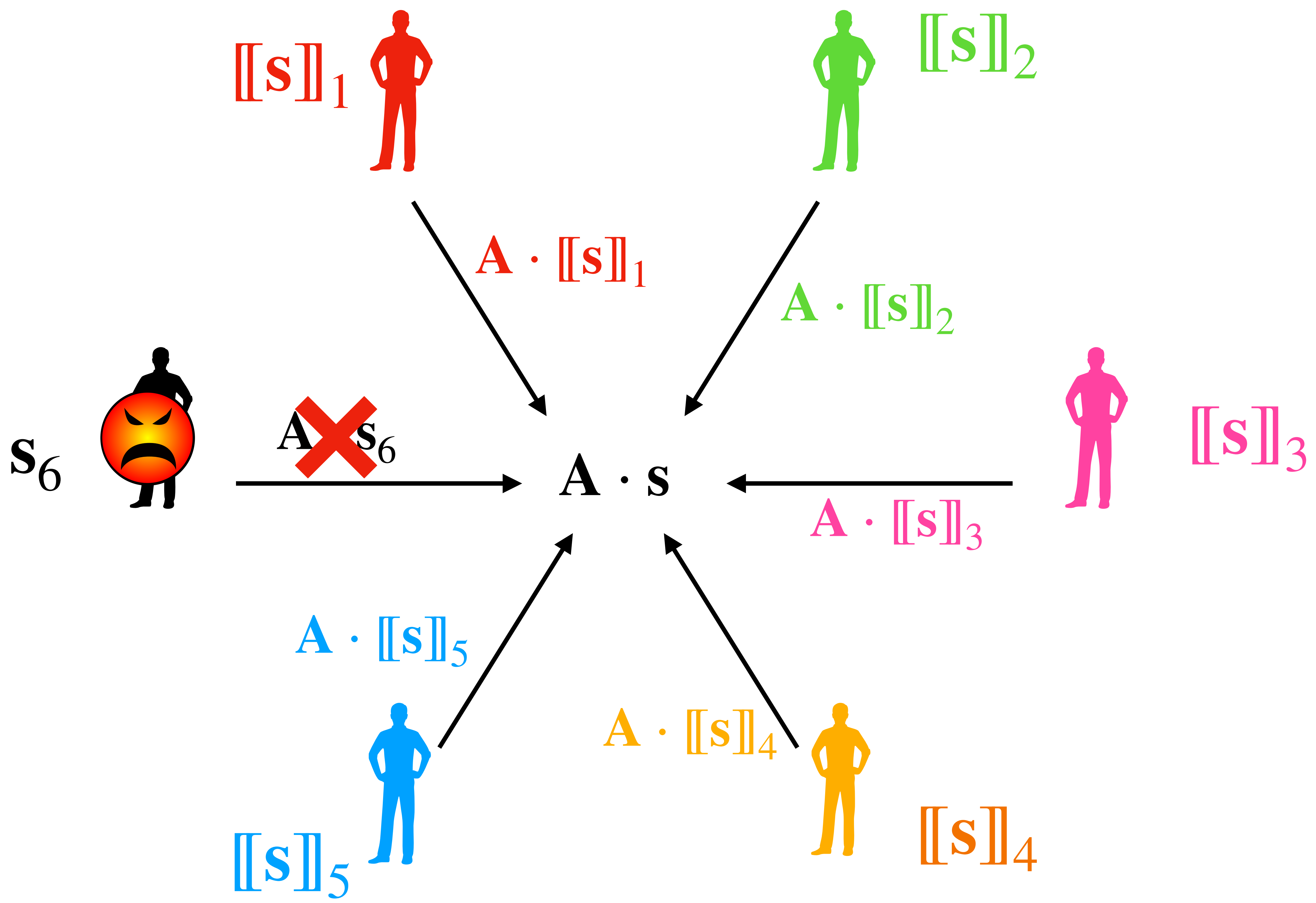
Distributed Key Generation (DKG) from VSS

$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \boxed{\mathbf{s}} \in \mathcal{R}_q^k$$

$$\mathsf{sk} = \boxed{\mathbf{s}} \in \mathcal{R}_q^\ell \quad \text{short}$$

1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$

2. Compute vk $= \mathbf{A} \cdot \mathbf{s}$



Distributed Key Generation (DKG) from VSS

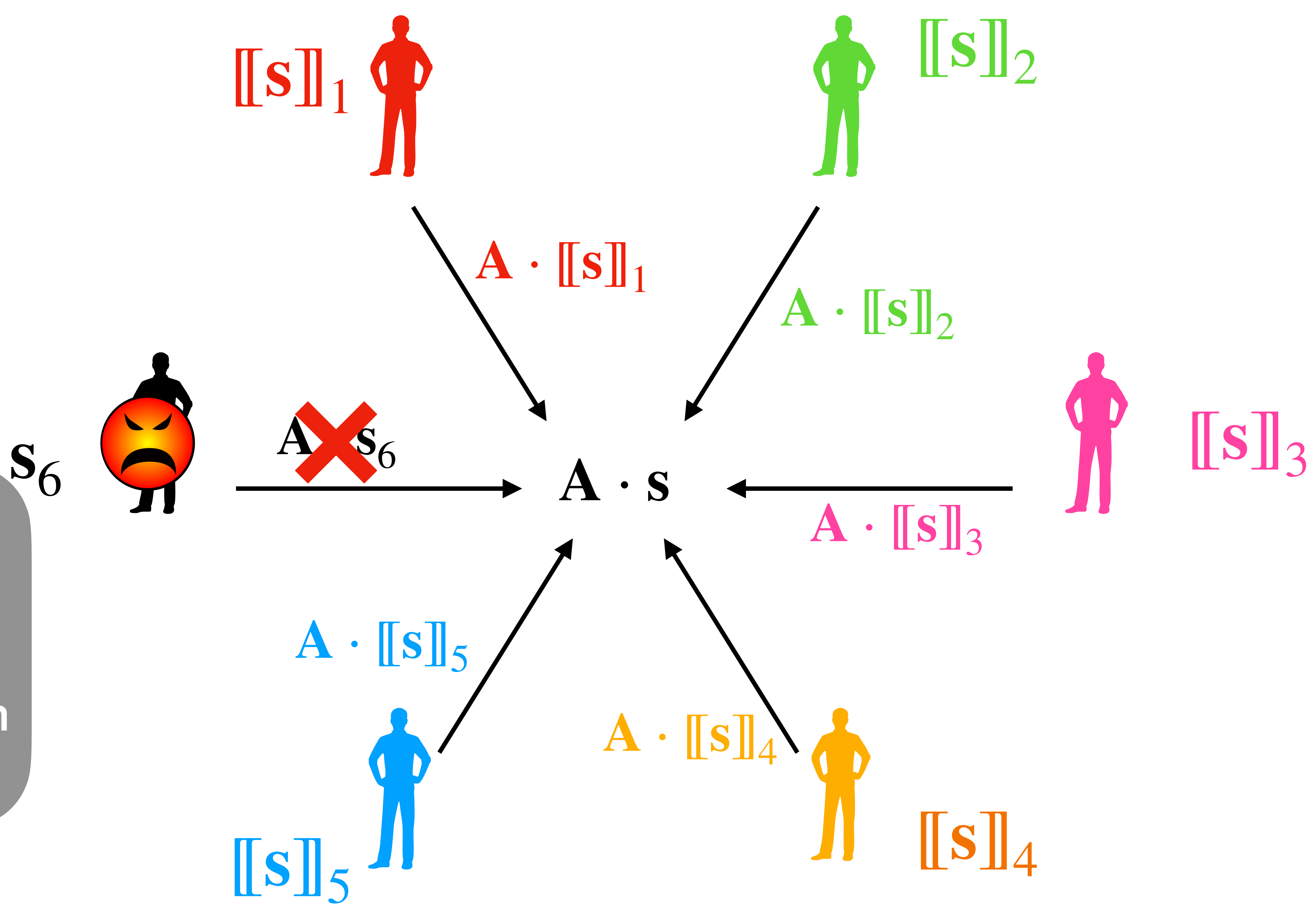
$$\mathsf{vk} = \boxed{\mathbf{t}} = \underbrace{\boxed{\mathbf{A}' \mid \mathbf{I}}}_{\mathbf{A}} \cdot \mathbf{s} \in \mathcal{R}_q^k$$

$$\mathsf{sk} = \mathbf{s} \in \mathcal{R}_q^\ell \quad \text{short}$$

Use Reed-Solomon error correction
to recover vk $= \mathbf{A} \cdot \mathbf{s}$

$\rightarrow$ can only support T'=T/3 corruption

1. Construct and share secret key $\mathbf{s} = \sum_i \mathbf{s}_i$
2. Compute vk $= \mathbf{A} \cdot \mathbf{s}$



Robust Signing with VSS

Threshold Raccoon

$\mathbf{r_i} \leftarrow \chi$

$\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r_i}$

$\xrightarrow{\mathsf{cmt_i} = H(\mathbf{w_i})}$

$\xleftarrow[(\mathsf{cmt}_j)_{j \in S}]{}$

$\xrightarrow{\mathbf{w}_i}$

$\xleftarrow[(\mathbf{w}_j)_{j \in S}]{}$

$$\mathbf{w} = \sum_{j \in S} \mathbf{w_j}$$

$$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$$

$$[\![\mathbf{z}]\!]_i = c \cdot L_{S,i} \cdot [\![\mathbf{s}]\!]_i + \mathbf{r}_i + \Delta_i \xrightarrow{\mathbf{z}_i}$$

Robust Signing with VSS

Threshold Raccoon

$\mathbf{r_i} \leftarrow \chi$

$\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r_i}$

$$\xrightarrow{\mathsf{cmt_i} = H(\mathbf{w_i})}$$
$$\xleftarrow[(\mathsf{cmt}_j)_{j \in S}]{}$$

$$\xrightarrow{\mathbf{w}_i}$$
$$\xleftarrow[(\mathbf{w}_j)_{j \in S}]{}$$

$\mathbf{w} = \sum_{j \in S} \mathbf{w_j}$

$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$

$[\![\mathbf{z}]\!]_i = c \cdot L_{S,i} \cdot [\![\mathbf{s}]\!]_i + \mathbf{r}_i + \Delta_i \xrightarrow{\mathbf{z}_i}$

Pelican

1) Use DKG to sample secret $\mathbf{r} = \sum_i \mathbf{r}_i$ and compute $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$: 3 rounds

Robust Signing with VSS

Threshold Raccoon

$\mathbf{r_i} \leftarrow \chi$

$\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r_i}$

$$\xrightarrow{\mathsf{cmt_i} = H(\mathbf{w_i})}$$
$$\xleftarrow[(\mathsf{cmt}_j)_{j \in S}]{}$$
$$\xrightarrow{\mathbf{w}_i}$$
$$\xleftarrow[(\mathbf{w}_j)_{j \in S}]{}$$

$\mathbf{w} = \sum_{j \in S} \mathbf{w_j}$

$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$

$[\![\mathbf{z}]\!]_i = c \cdot L_{S,i} \cdot [\![\mathbf{s}]\!]_i + \mathbf{r}_i + \Delta_i \xrightarrow{\mathbf{z}_i}$

Pelican

1) Use DKG to sample secret $\mathbf{r} = \sum_i \mathbf{r}_i$ and compute $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$: 3 rounds

2) Compute signature shares: 1 round

$$c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$$

$$[\![\mathbf{z}]\!]_i = c \cdot [\![\mathbf{s}]\!]_i + [\![\mathbf{r}]\!]_i$$

If corruption threshold $T' \leq T/3$, *Reed-Solomon error correction* guarantees signature output.

3. A practical VSS with approximate shortness proof

Prior work on VSS

- Classical setting (uniform secret)
  - BGW VSS [BGW88]: IT security
  - Pedersen VSS [Ped92]: relies on DL
  - [ABCP23] based on hash functions
- VSS with shortness proof [GHL21]: quite large and DL aggregation

Our VSS

How to prove shortness of a vector $\mathbf{s}$ without revealing it?

Use a random projection to a smaller space!

Our VSS

How to prove shortness of a vector $\mathbf{s}$ without revealing it?

Use a random projection to a smaller space!

Modular Johnson-Lindenstrauss lemma with offset [Ngu22]: Take a vector $\mathbf{y}$.

If a matrix $\mathbf{R}$ is sampled from a discrete distribution with coefficients ± 1 with proba $\frac{1}{4}$, and 0 with proba $\frac{1}{2}$.

Then, $\|\mathbf{R} \cdot \mathbf{s} + \mathbf{y} \bmod q\|_2$ is at least as large as $C \cdot \|\mathbf{s}\|_2$ for some $C = \omega(1)$.

Use small Gaussian noise keeping enough entropy in $\mathbf{s}$ instead of information theoretic.

Our VSS

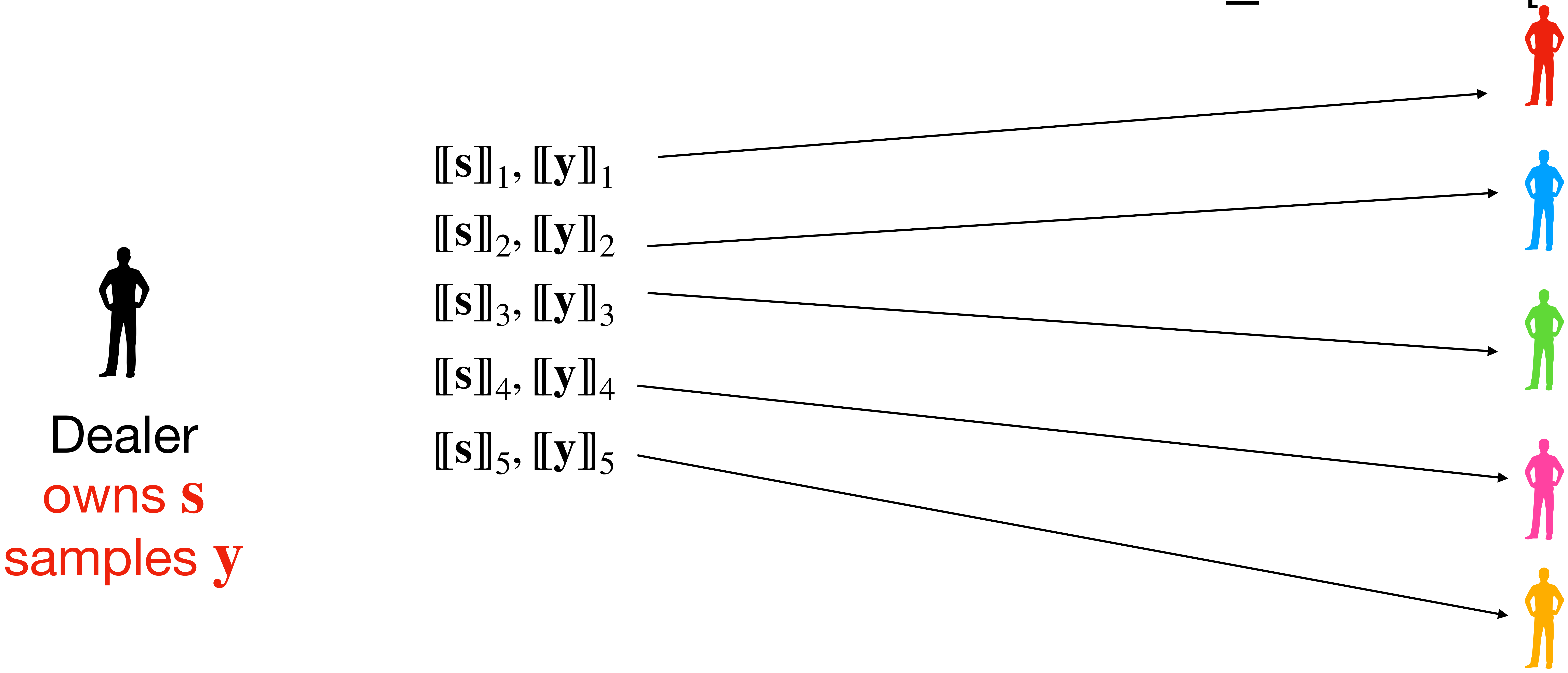
Johnson-Lindenstrauss only applies if $\mathbf{R}$ is sampled after $\mathbf{s}$ and $\mathbf{y}$.

$\longrightarrow$ Solution: hash-based verifiable randomness for $N \geq 2T'$ akin to [ABCP23].

Our VSS

Johnson-Lindenstrauss only applies if $\mathbf{R}$ is sampled after $\mathbf{s}$ and $\mathbf{y}$.

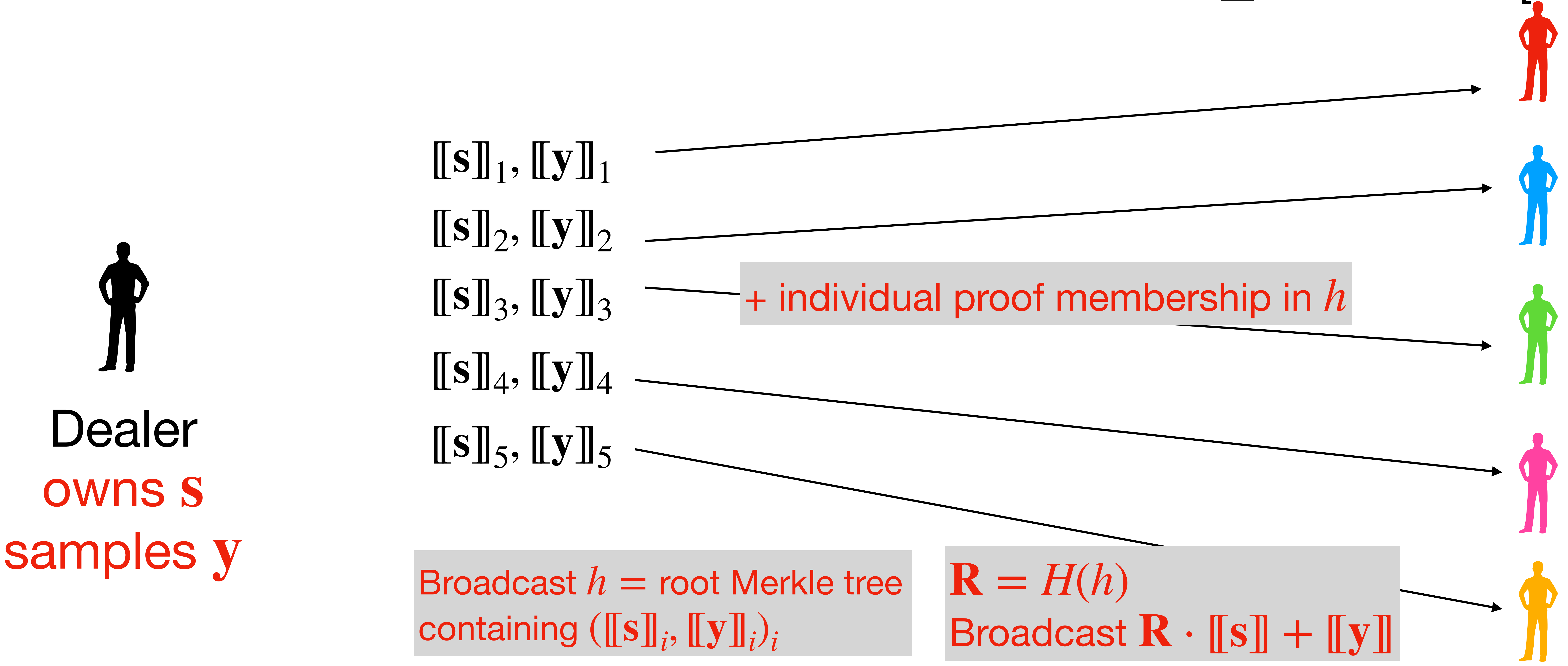
→ Solution: hash-based verifiable randomness for $N \geq 2T'$ akin to [ABCP23].



Our VSS

Johnson-Lindenstrauss only applies if $\mathbf{R}$ is sampled after $\mathbf{s}$ and $\mathbf{y}$.

→ Solution: hash-based verifiable randomness for $N \geq 2T'$ akin to [ABCP23].



Our VSS

- Our VSS reveals $\mathbf{R} \cdot \mathbf{s} + \mathbf{y}$ where $\mathbf{y}$ is Gaussian: smaller shortness gap compared to rejection sampling.
    - Not purely ZK: reduce security to Hint-MLWE

- **Approximation gap ~70**, vs $\gg 2500$ in [GHL21] using JL lemma

Conclusion

Conclusion

- Framework relying on VSS to achieve robust DKG and robust signature scheme with corruption threshold $T' = T/3$.

- **Pelican:** first lattice hash-and-sign threshold signature + DKG + robustness

Pelican = application to Plover, in this presentation applied to Raccoon

- Practical VSS scheme with approximate shortness proof: slack ~70

κ	**max T'**	**| vk |**	**| sig |**
128	16	12.8kB	12.3kB
196	1024	25.6kB	26.4kB

Proposed parameter sets for Pelican