Unmasking TRaccoon:
A Lattice-Based Threshold Signature with An Efficient Identifiable Abort Protocol

Rafael del Pino
PQShield

Shuichi Katsumata
PQShield & AIST

Guilhem Niot
PQShield & Univ Rennes,CNRS,IRISA

Michael Reichle
ETH Zurich

Kaoru Takemure
PQShield & AIST

This presentation is based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO)

Our Identifiable Abort Protocol

Main Contribution: TRaccoon with Identifiable Abort Protocol

- Our interactive IA protocol is a simple add-on to TRaccoon
- Communication cost in IA protocol is $60 + 6.4 \cdot T$ KB per a signer

Side Contributions:

- The first game-based definition of TS with an *interactive* IA protocol
- The first formal security analysis of a variant of LaBRADOR with ZK

Background

T-out-of-N Threshold Signature

Verification key vk

$\Updownarrow$

Signing key sk

Key Generation

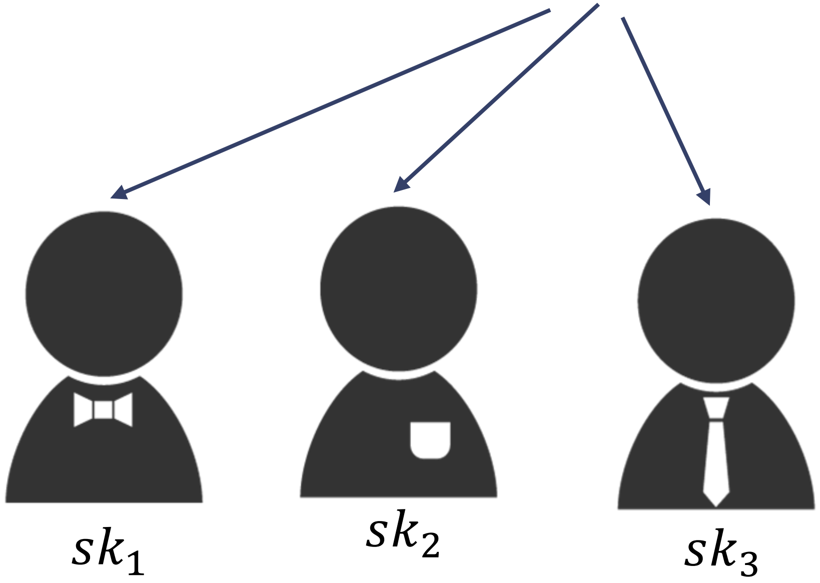
※2-out-of-3

T-out-of-N Threshold Signature

Verification key vk

⇕

Signing key $\underline{sk}$



※2-out-of-3

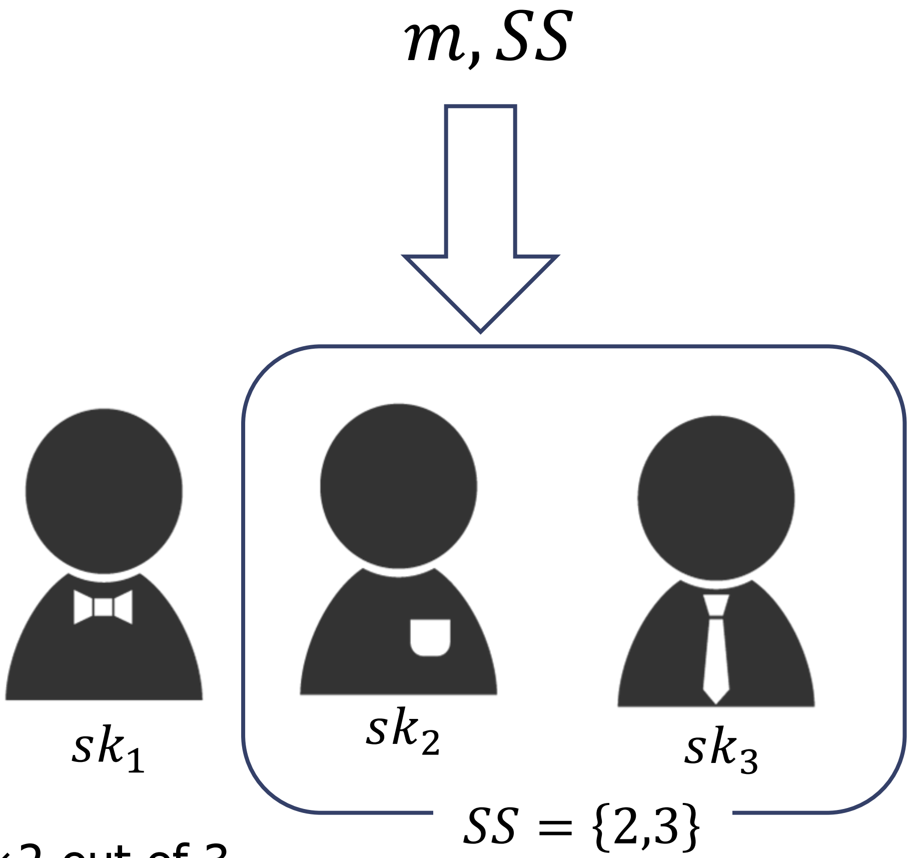
Key Generation

- T or more key shares reconstruct sk
- No signer knows sk
- Less than T key shares leak no information about sk

※We assume that a trusted party executes distributed key generation as well as [BCK+22,dPKM+24] etc.

T-out-of-N Threshold Signature

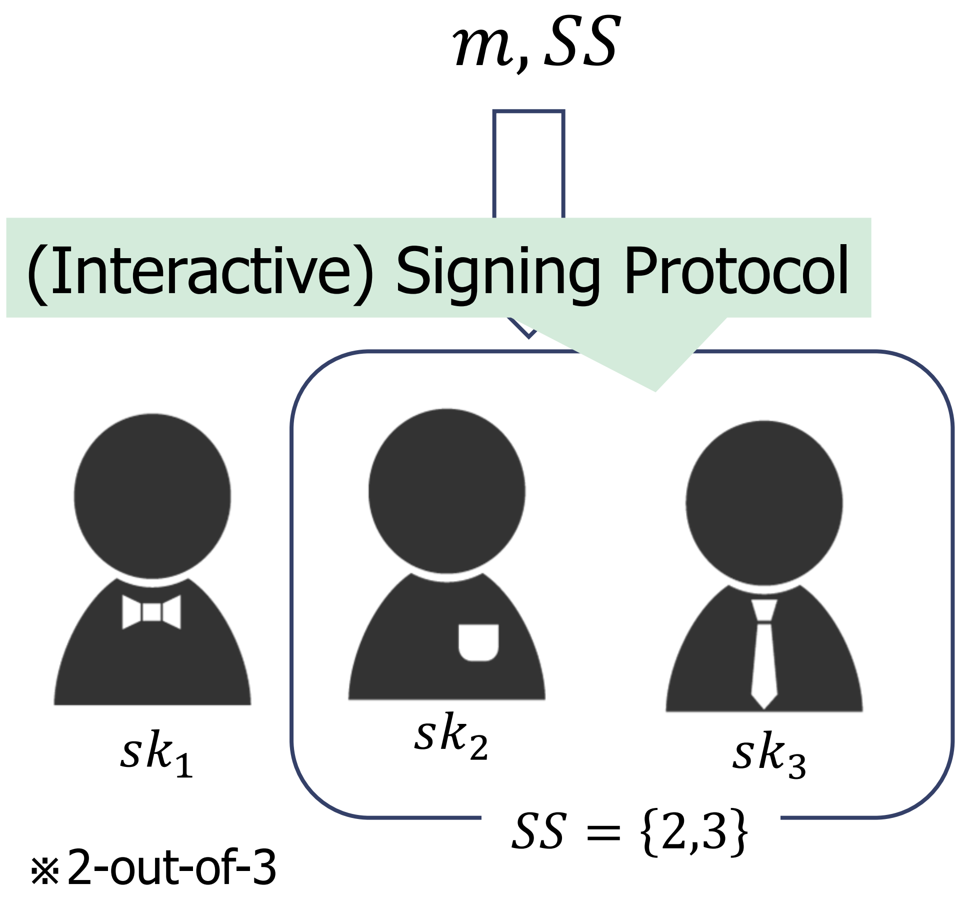
Signing Protocol



※2-out-of-3

T-out-of-N Threshold Signature

Signing Protocol

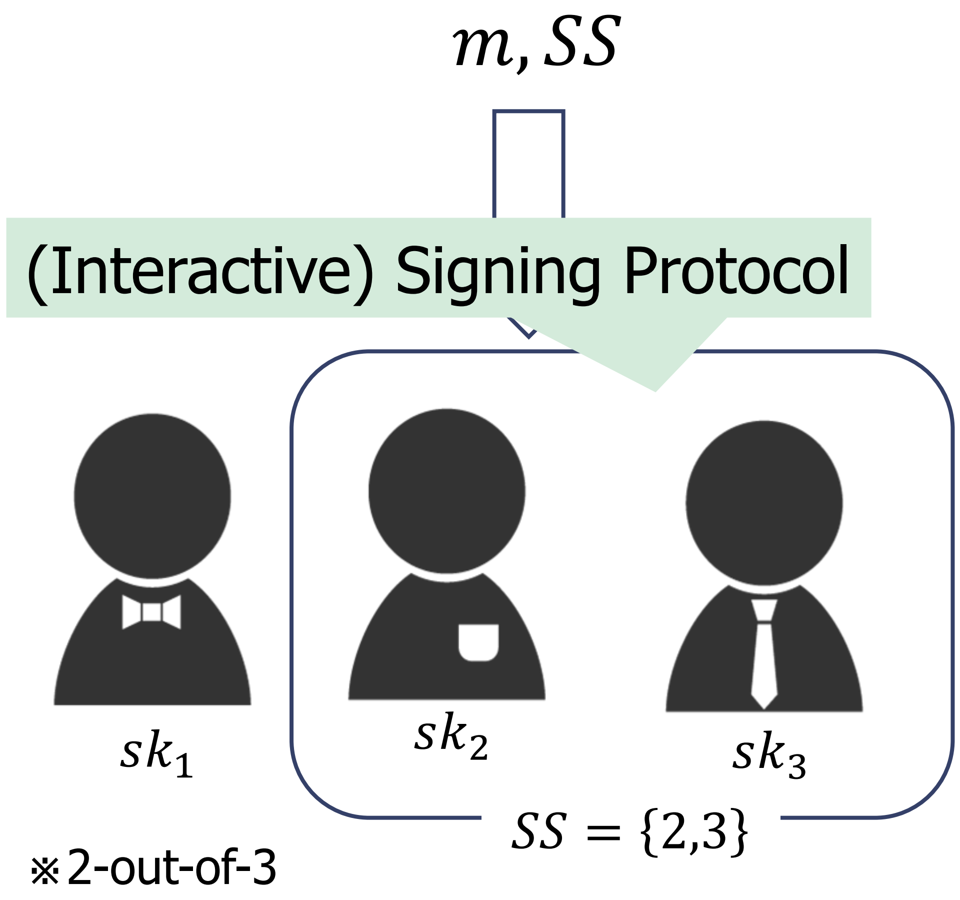


General Procedure:

1. One decides message m and signer set SS
2. Users in SS execute signing protocol

T-out-of-N Threshold Signature

Signing Protocol



General Procedure:

1. One decides message m and signer set SS
2. Users in SS execute signing protocol

⇒ Signature σ

PQ Threshold Signature Schemes

Early Schemes: [BKP13], [BGG+18], [ASY22], [GKS23]
$\Rightarrow$The use of heavy tools, e.g., FHE and HTDC

Recent Schemes: [dPKM+24], [EKT24], [KRT24], [CATZ24], [BKL+25], etc
$\Rightarrow$No use of such heavy tools

PQ Threshold Signature Schemes

Early Schemes: [BKP13], [BGG+18], [ASY22], [GKS23]
⇒The use of heavy tools, e.g., FHE and HTDC

Recent Schemes: [dPKM+24], [EKT24], [KRT24], [CATZ24], [BKL+25], etc
⇒No use of such heavy tools

<u>TRaccoon [dPKM+24]:</u>

- Three-round signing protocol
- Efficient sig size compared with early schemes

PQ Threshold Signature Schemes

Early Schemes: [BKP13], [BGG+18], [ASY22], [GKS23]
⇒The use of heavy tools, e.g., FHE and HTDC

Recent Schemes: [dPKM+24], [EKT24], [KRT24], [CATZ24], [BKL+25], etc
⇒No use of such heavy tools

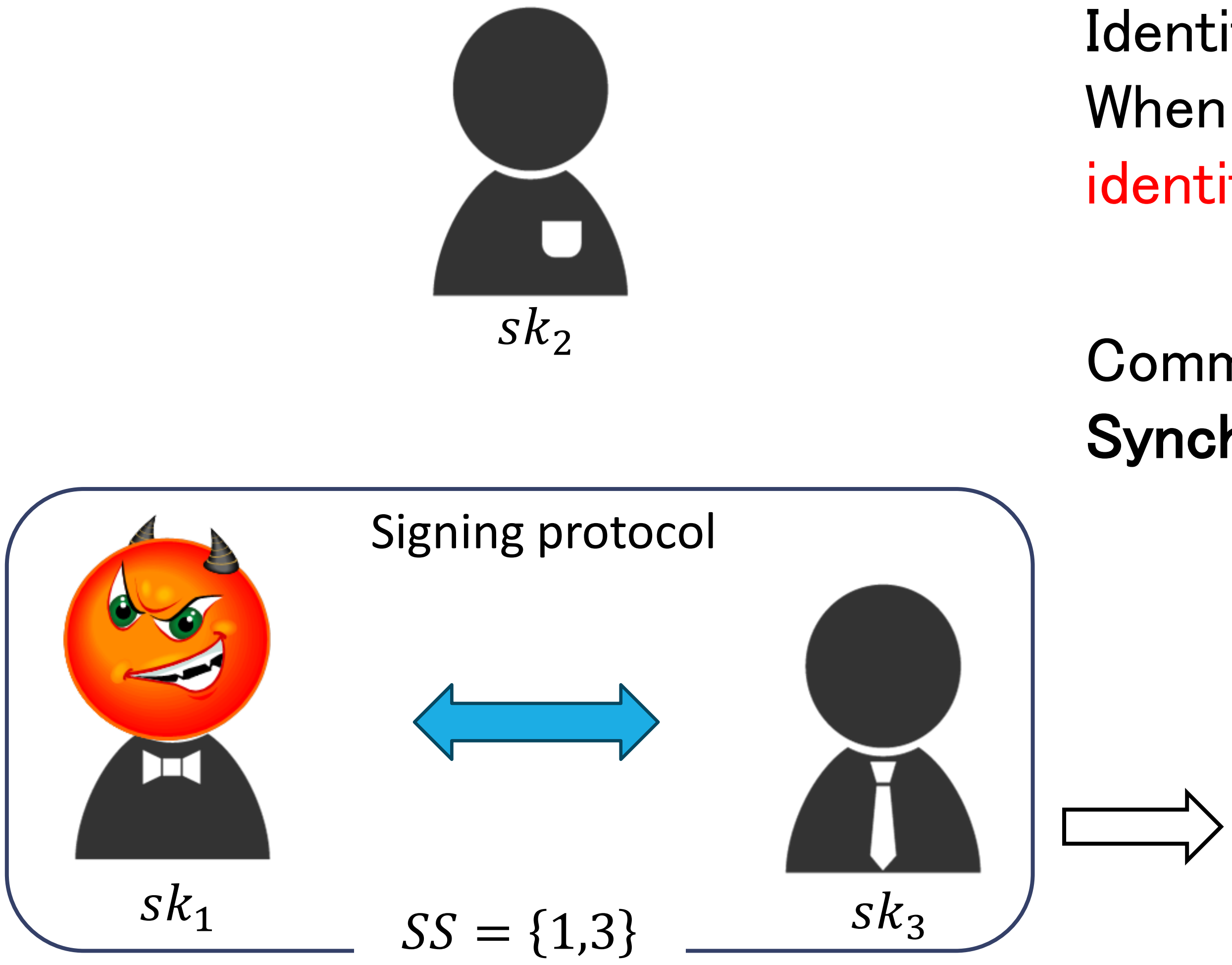
One drawback: No Availability

Malicious signer can arbitrarily cause the signing protocol to fail

<u>TRaccoon [dPKM+24]:</u>

- Three-round signing protocol
- Efficient sig size compared with early schemes

Availability for TS: Identifiable Abort



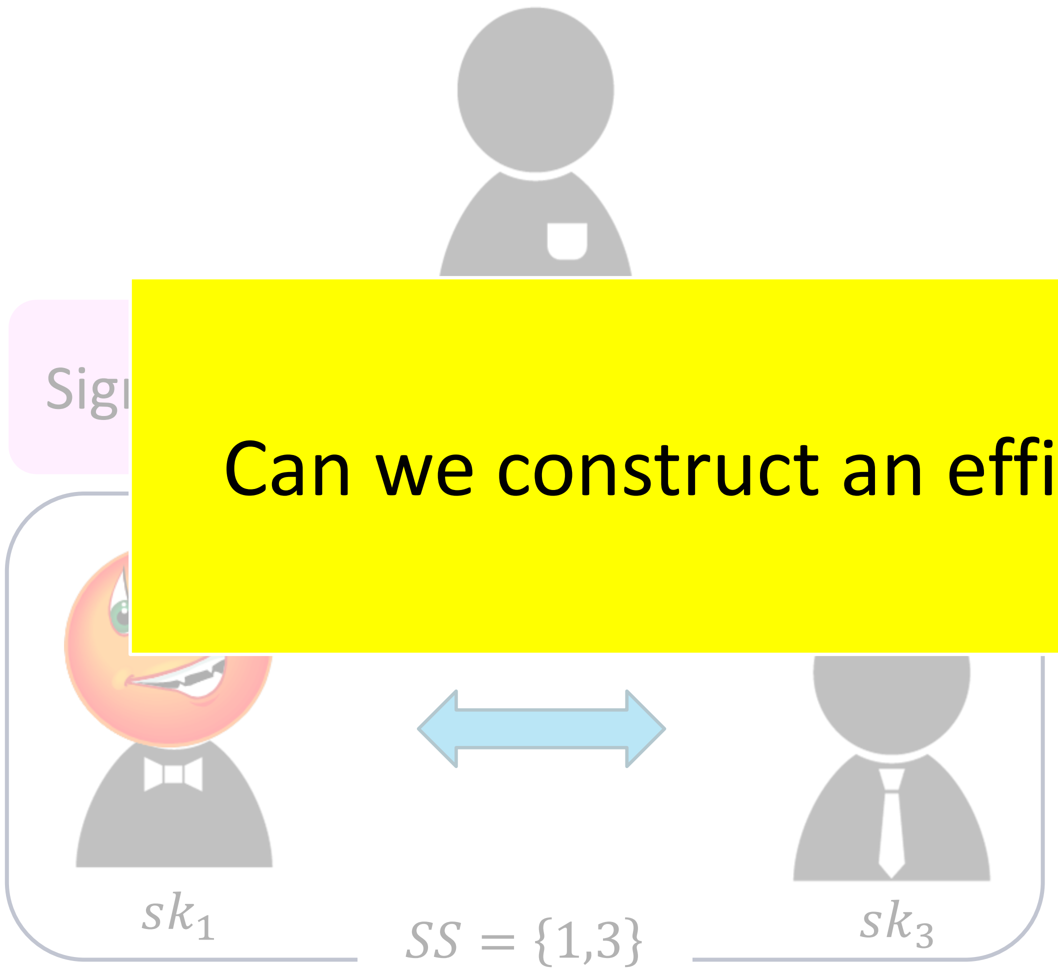
Identifiable Abort:
When the signing protocol fails, honest signers identify misbehaving signers.

Communication Channel:
Synchronous authenticated Broadcast

Valid σ
or
Signer 1 is misbehavior

Availability for TS: Identifiable Abort

Identifiable Abort:
When the signing protocol fails, honest signers identify misbehaving signers.



Can we construct an efficient IA protocol for TRaccoon?

Valid σ
or
Signer 1 is misbehavior

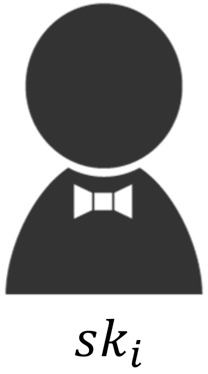
TRaccoon

Threshold Raccoon [dPKM+24]

vk: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{t} = \boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e}$ where short vectors $(\boldsymbol{s}, \boldsymbol{e}) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$

sk_i: $\boldsymbol{s}_i$ is a secret share of $\boldsymbol{s}$, $\left(seed_{i,j}, seed_{j,i}\right)_{j\in[N]}$ are pair-wise seeds.

Lattice variant of Sparkle[CKM23]



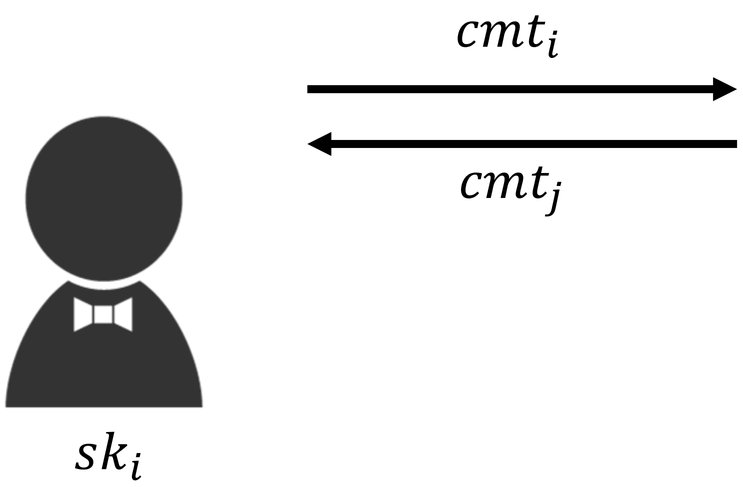
Threshold Raccoon [dPKM+24]

vk: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{t} = \boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e}$ where short vectors $(\boldsymbol{s},\boldsymbol{e}) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$
sk_i: $\boldsymbol{s}_i$ is a secret share of $\boldsymbol{s}$, $\left(seed_{i,j}, seed_{j,i}\right)_{j\in[N]}$ are pair-wise seeds.

Lattice variant of Sparkle[CKM23]

Round 1:

1. Sample short vectors $(\boldsymbol{r}_i, \boldsymbol{e}'_i) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$
2. $\boldsymbol{w}_i \leftarrow \boldsymbol{A}\cdot\boldsymbol{r}_i + \boldsymbol{e}'_i$
3. Broadcast $cmt_i \leftarrow H(\boldsymbol{w}_i)$



Threshold Raccoon [dPKM+24]

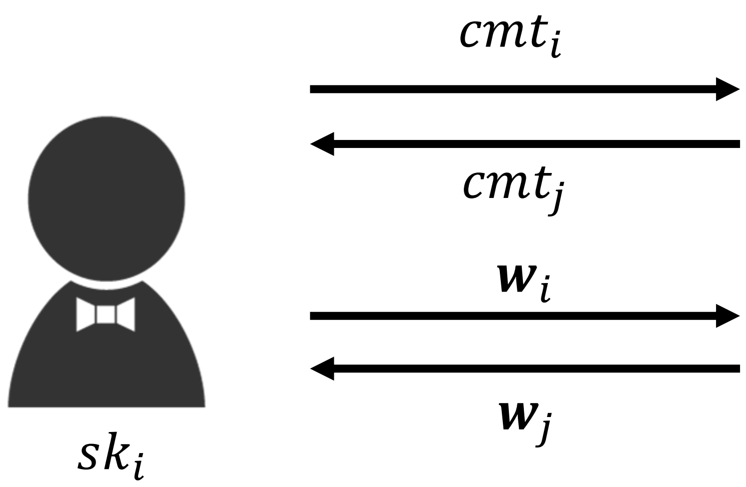
vk: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{t} = \boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e}$ where short vectors $(\boldsymbol{s}, \boldsymbol{e}) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$

sk_i: $\boldsymbol{s}_i$ is a secret share of $\boldsymbol{s}$, $\left(seed_{i,j}, seed_{j,i}\right)_{j\in[N]}$ are pair-wise seeds.

Lattice variant of Sparkle[CKM23]

Round 1:
1. Sample short vectors $(\boldsymbol{r}_i, \boldsymbol{e}'_i) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$
2. $\boldsymbol{w}_i \leftarrow \boldsymbol{A}\cdot\boldsymbol{r}_i + \boldsymbol{e}'_i$
3. Broadcast $cmt_i \leftarrow H(\boldsymbol{w}_i)$

Round 2:
1. Broadcast $\boldsymbol{w}_i$



Threshold Raccoon [dPKM+24]

vk: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{t} = \boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e}$ where short vectors $(\boldsymbol{s}, \boldsymbol{e}) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$

sk_i: $\boldsymbol{s}_i$ is a secret share of $\boldsymbol{s}$, $\left(seed_{i,j}, seed_{j,i}\right)_{j\in[N]}$ are pair-wise seeds.

Lattice variant of Sparkle[CKM23]

Round 1:

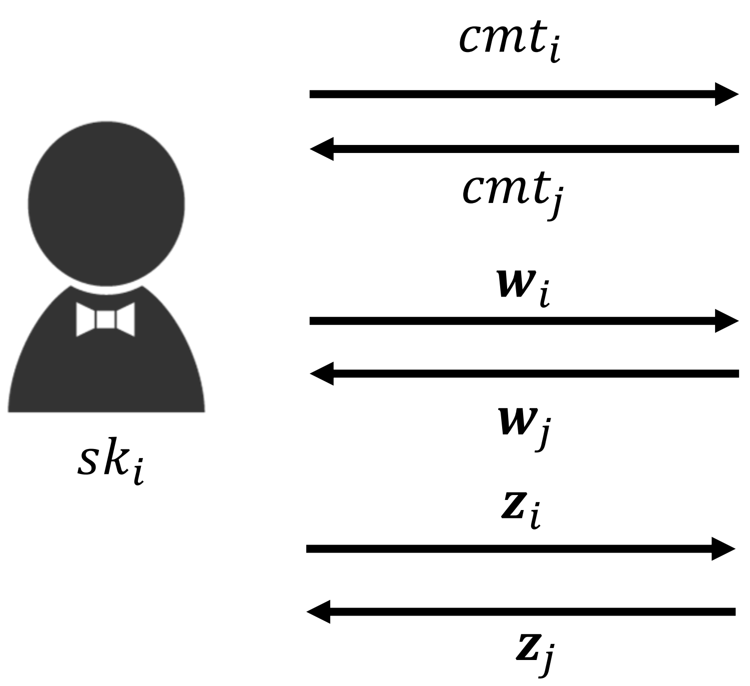
1. Sample short vectors $(\boldsymbol{r}_i, \boldsymbol{e}'_i) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$
2. $\boldsymbol{w}_i \leftarrow \boldsymbol{A}\cdot\boldsymbol{r}_i + \boldsymbol{e}'_i$
3. Broadcast $cmt_i \leftarrow H(\boldsymbol{w}_i)$

Round 2:

1. Broadcast $\boldsymbol{w}_i$

Round 3:

1. Check $cmt_j = H(\boldsymbol{w}_j)$
2. $\boldsymbol{w} \leftarrow \sum_j \boldsymbol{w}_j$
3. $c \leftarrow H_c(vk, m, \boldsymbol{w})$
4. Broadcast $\boldsymbol{z}_i \leftarrow c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$



Threshold Raccoon [dPKM+24]

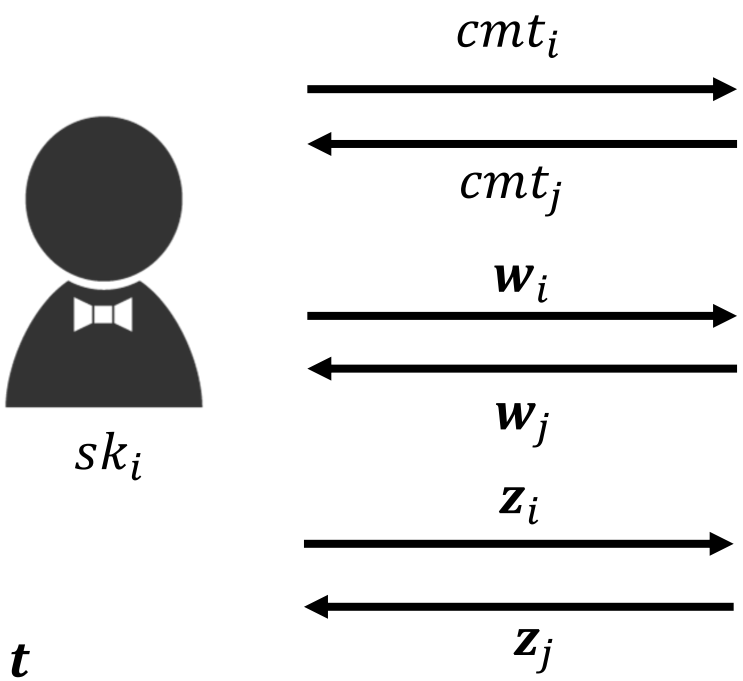
vk: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{t} = \boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e}$ where short vectors $(\boldsymbol{s}, \boldsymbol{e}) \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$
sk_i: $\boldsymbol{s}_i$ is a secret share of $\boldsymbol{s}$, $\left(seed_{i,j}, seed_{j,i}\right)_{j\in[N]}$ are pair-wise seeds.

Lattice variant of Sparkle[CKM23]

Round 1:
1. Sample short vectors $(\boldsymbol{r}_i, \boldsymbol{e}_i') \in \mathcal{R}_q^{\ell} \times \mathcal{R}_q^{k}$
2. $\boldsymbol{w}_i \leftarrow \boldsymbol{A}\cdot\boldsymbol{r}_i + \boldsymbol{e}_i'$
3. Broadcast $cmt_i \leftarrow H(\boldsymbol{w}_i)$

Round 2:
1. Broadcast $\boldsymbol{w}_i$

Round 3:
1. Check $cmt_j = H(\boldsymbol{w}_j)$
2. $\boldsymbol{w} \leftarrow \sum_j \boldsymbol{w}_j$
3. $c \leftarrow H_c(vk, m, \boldsymbol{w})$
4. Broadcast $\boldsymbol{z}_i \leftarrow c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$



Resulting signature: $(c, \boldsymbol{z}, \boldsymbol{h})$ where $\boldsymbol{z} = \sum_i \boldsymbol{z}_i$, $\boldsymbol{h} = \boldsymbol{w} - \boldsymbol{A}\cdot\boldsymbol{z} + c\cdot\boldsymbol{t}$
Verification: $c = H_c(vk, m, \boldsymbol{A}\cdot\boldsymbol{z} - c\cdot\boldsymbol{t} + \boldsymbol{h})$

Threshold Raccoon [dPKM+24]

vk:
sk_i:

Rou

Rou

Rou

Important difference from Sparkle:

Masking Term: $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$ such that $\sum_i \Delta_i = 0$

where $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z), ctnt_z = SS||m||(cmt_i, w_i)_{i \in SS}$.

This is a crucial component to prevent lattice-specific attacks.

2. $\boldsymbol{w} \leftarrow \sum_j \boldsymbol{w}_j$

3. $c \leftarrow H_c(vk, m, \boldsymbol{w})$

4. Broadcast $\boldsymbol{z}_i \leftarrow c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

sk_1 $\boldsymbol{w}_j$ $\boldsymbol{z}_i$ $\boldsymbol{z}_j$

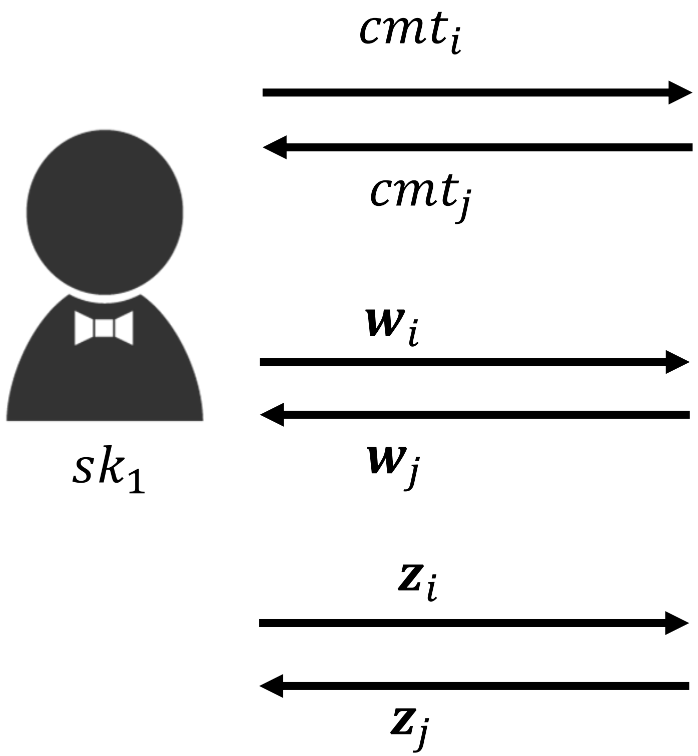
Resulting signature: $(c, \boldsymbol{z}, \boldsymbol{h})$ where $\boldsymbol{z} = \sum_i \boldsymbol{z}_i$, $\boldsymbol{h} = \boldsymbol{w} - \boldsymbol{A} \cdot \boldsymbol{z} + c \cdot \boldsymbol{t}$

Verification: $c = H_c(vk, m, \boldsymbol{A} \cdot \boldsymbol{z} - c \cdot \boldsymbol{t} + \boldsymbol{h})$

Our Approach

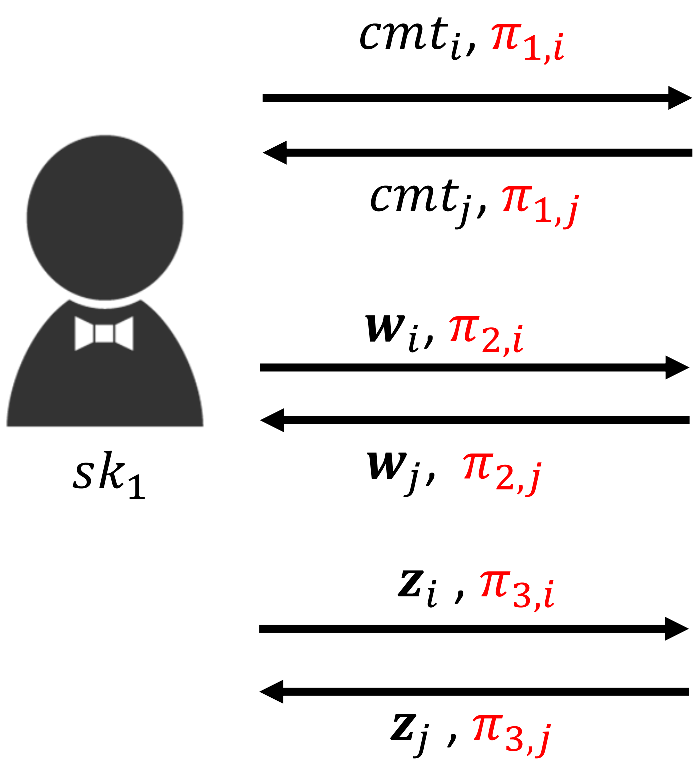
Straightforward Approach

All signers prove that they honestly executed the signing protocol for each round.



Straightforward Approach

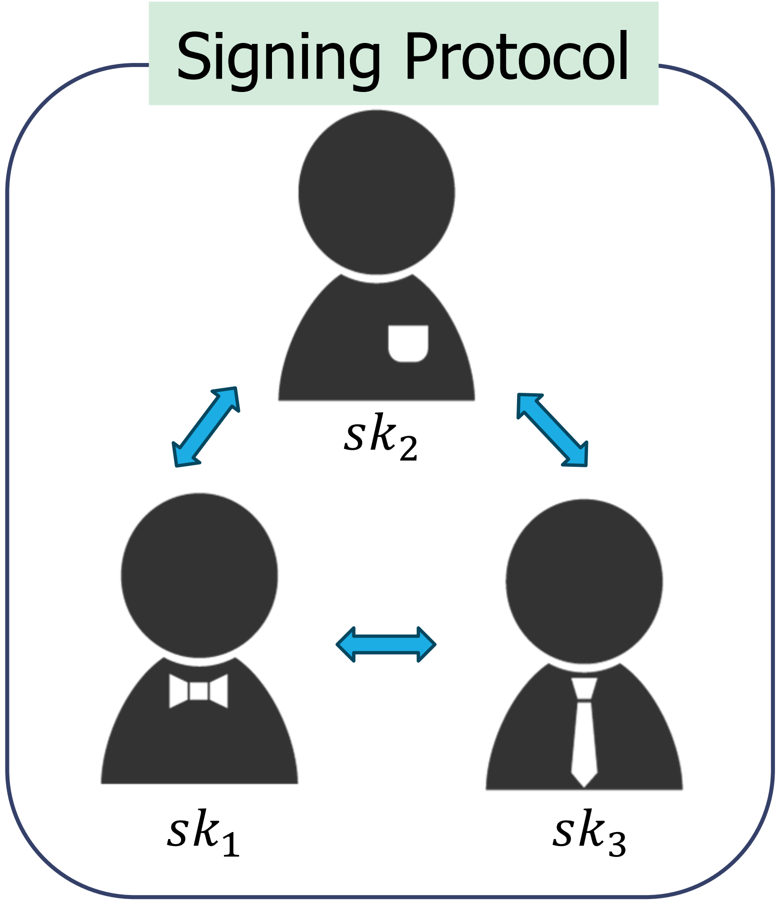
All signers prove that they honestly executed the signing protocol for each round.



Increase communication cost during signing protocol 😞

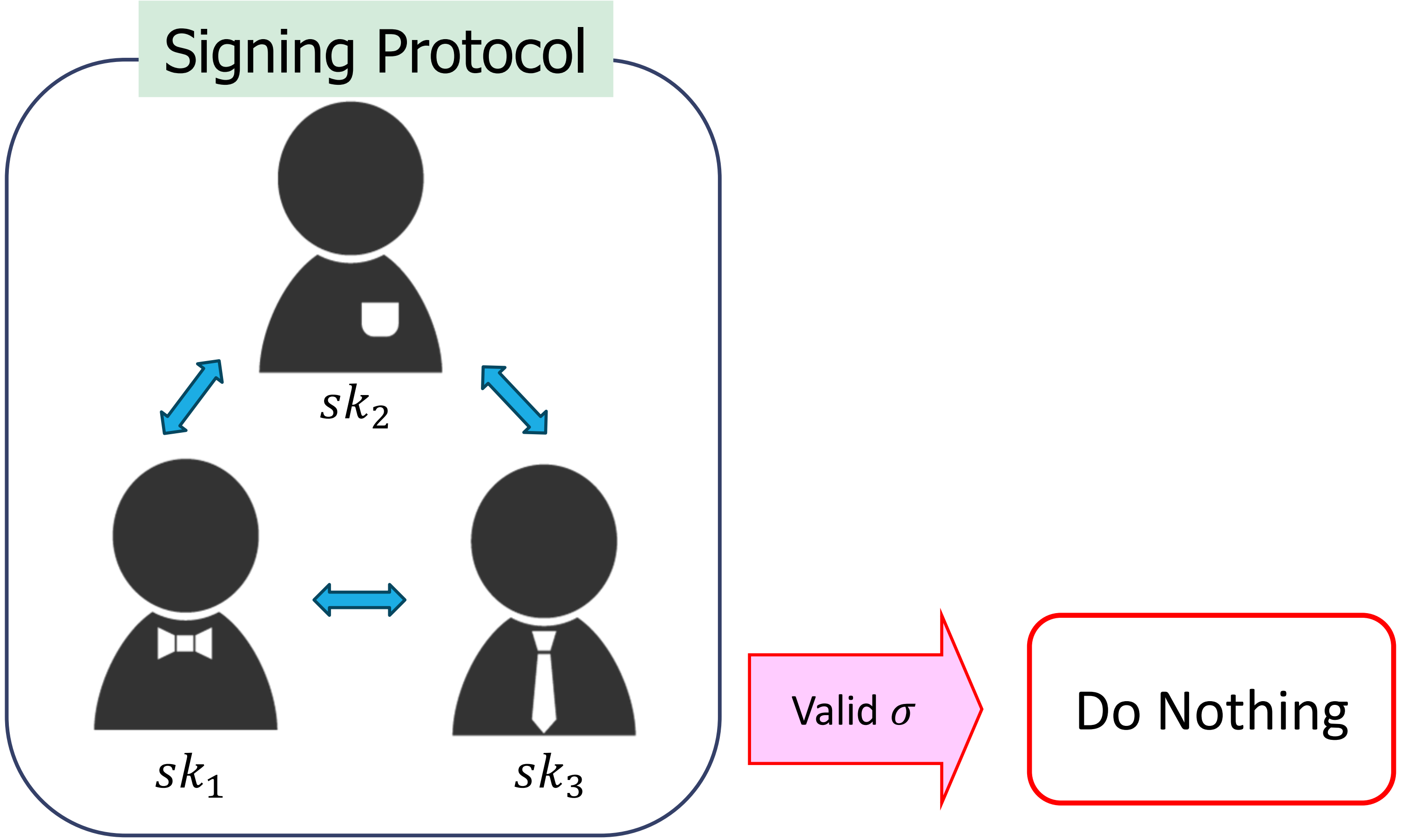
Deferred IA Protocol

Delay the identification of misbehaving signers until the protocol aborts.



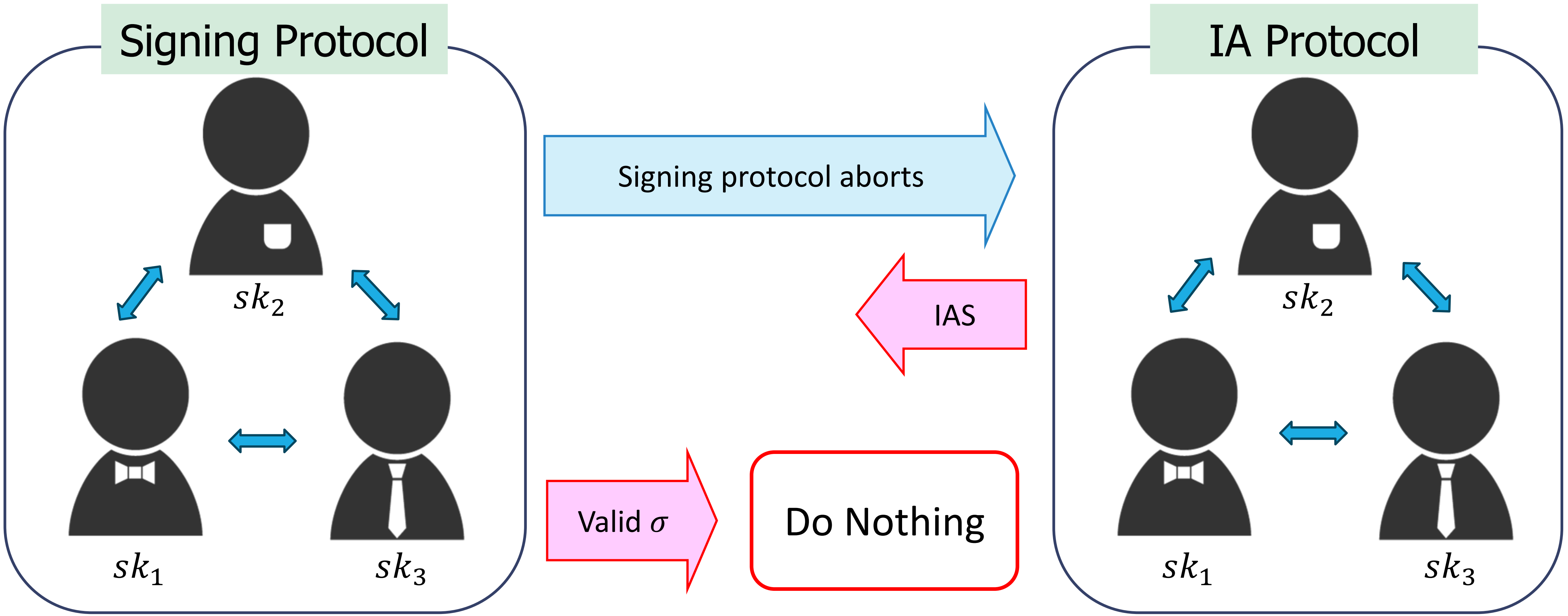
Deferred IA Protocol

Delay the identification of misbehaving signers until the protocol aborts.



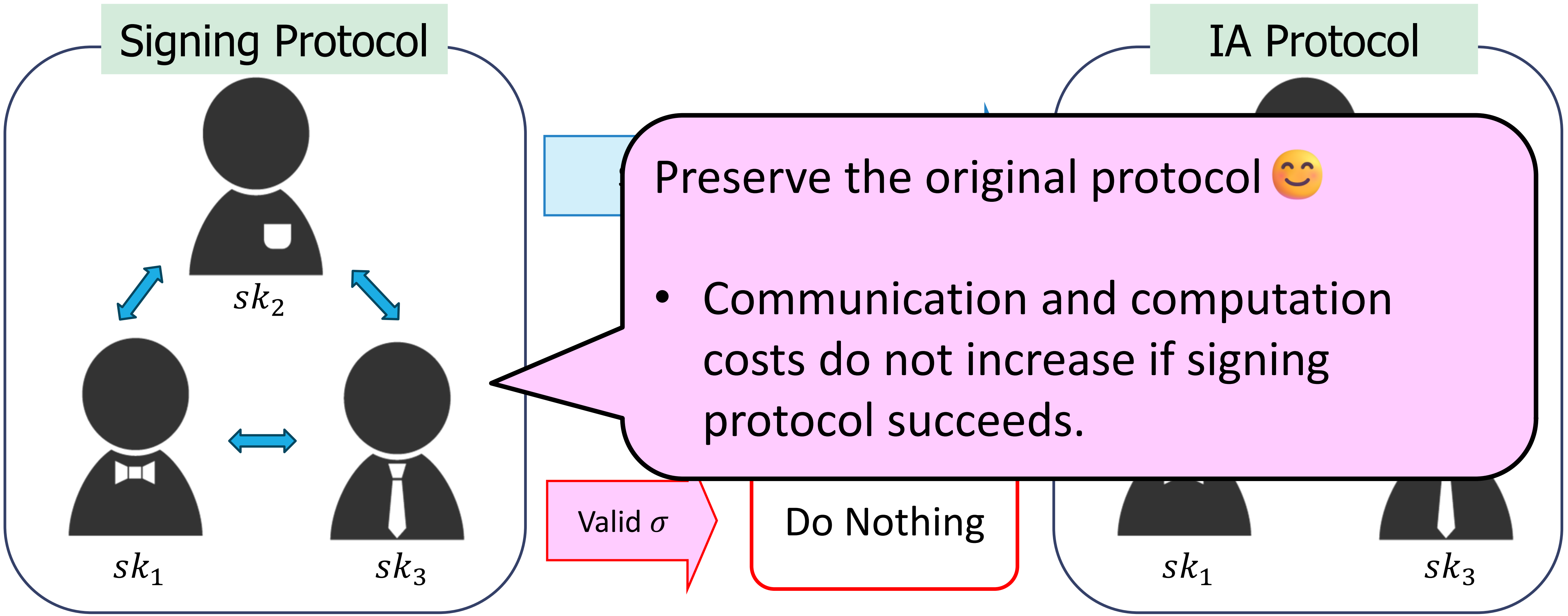
Deferred IA Protocol

Delay the identification of misbehaving signers until the protocol aborts.



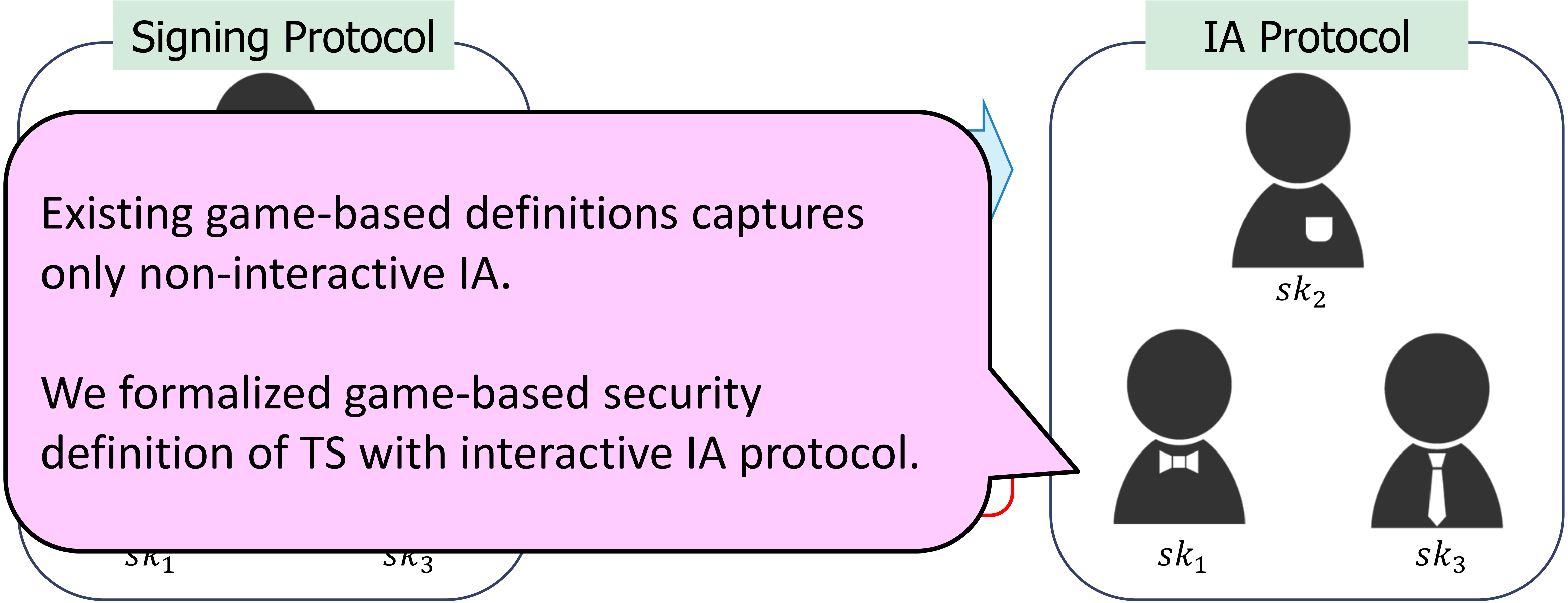
Deferred IA Protocol

Delay the identification of misbehaving signers until the protocol aborts.



Deferred IA Protocol

Delay the identification of misbehaving signers until the protocol aborts.



Existing game-based definitions captures only non-interactive IA.

We formalized game-based security definition of TS with interactive IA protocol.

Relations to be Proven via NIZK

Our IA protocol follows the approach using NIZK.

Relations to be proven:

(1) $\boldsymbol{r}_i$ is short

(2) $\boldsymbol{z}_i = c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

(3) $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$

(4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$

Relations to be Proven via NIZK

Our IA protocol follows the approach using NIZK.

Relations to be proven:

(1) $\boldsymbol{r}_i$ is short

(2) $\boldsymbol{z}_i = c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

(3) $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$

(1)–(3): Algebraic

(4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$

(4): Non-Algebraic

Relations to be Proven via NIZK

Our IA protocol follows the approach using NIZK.

Relations to be proven:

(1) $\boldsymbol{r}_i$ is short

(2) $\boldsymbol{z}_i = c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

(3) $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$

(4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$

Algebraic: (1)–(3)

Non-Algebraic: (4)

Proving "mixed" relations is impractical 😞
How can we avoid this?

Bypassing Non-Algebraic Relation

Why is (4) $\boldsymbol{m}_{i,j} = H_{msk}\left(seed_{i,j}, ctnt_z\right)$ required?

Bypassing Non-Algebraic Relation

Why is (4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$ required?

$\Rightarrow$ Ensure $\sum_i \Delta_i = 0$

Bypassing Non-Algebraic Relation

Why is (4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$ required?

⇒ Ensure $\sum_i \Delta_i = 0$

⇒ (★) Each pair of signers uses the same masks

Bypassing Non-Algebraic Relation

Why is (4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$ required?

$\Rightarrow$ Ensure $\sum_i \Delta_i = 0$

$\Rightarrow$ (★) Each pair of signers uses the same masks

Our observation:
As long as each pair uses the same $\boldsymbol{m}_{i,j}$ even though it is not honestly generated, $\sum_i \Delta_i = 0$ holds.

Bypassing Non-Algebraic Relation

Why is (4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$ required?

⇒ Ensure $\sum_i \Delta_i = 0$

⇒ (★) Each pair of signers uses the same masks

Our observation:
As long as each pair uses the same $\boldsymbol{m}_{i,j}$ even though it is not honestly generated, $\sum_i \Delta_i = 0$ holds.

Idea: Ensure (★) outside of NIZK

How to Check (★)

Com: Lattice-based commitment scheme

1. For $j \in SS \setminus \{i\}$, compute $D_{i,j}^{(i)} \leftarrow Com(\boldsymbol{m}_{i,j}; \delta_{i,j})$ and $D_{j,i}^{(i)} \leftarrow Com(\boldsymbol{m}_{j,i}; \delta_{j,i})$ where $\delta_{i,j} = H_{rnd}(seed_{i,j}, ctnt_z)$, $\delta_{j,i} = H_{rnd}(seed_{j,i}, ctnt_z)$.
   Broadcast $\left(D_{i,j}^{(i)}, D_{j,i}^{(i)}\right)_{j \in SS \setminus \{i\}}$

Deterministic

How to Check (★)

Com: Lattice-based commitment scheme

1. For $j \in SS \setminus \{i\}$, compute $D_{i,j}^{(i)} \leftarrow Com(\boldsymbol{m}_{i,j}; \delta_{i,j})$ and $D_{j,i}^{(i)} \leftarrow Com(\boldsymbol{m}_{j,i}; \delta_{j,i})$ where $\delta_{i,j} = H_{rnd}(seed_{i,j}, ctnt_z)$, $\delta_{j,i} = H_{rnd}(seed_{j,i}, ctnt_z)$.
   Broadcast $\left(D_{i,j}^{(i)}, D_{j,i}^{(i)}\right)_{j \in SS \setminus \{i\}}$

2. Broadcast $\left(seed_{i,j}^{(i)}, seed_{j,i}^{(i)}\right)$ for j s.t. $D_{i,j}^{(i)} \neq D_{i,j}^{(j)}$ or $D_{j,i}^{(i)} \neq D_{j,i}^{(j)}$

Deterministic

Inconsistent Mask

How to Check (★)

Com: Lattice-based commitment scheme

1. For $j \in SS \setminus \{i\}$, compute $D_{i,j}^{(i)} \leftarrow Com(\boldsymbol{m}_{i,j}; \delta_{i,j})$ and $D_{j,i}^{(i)} \leftarrow Com(\boldsymbol{m}_{j,i}; \delta_{j,i})$ where $\delta_{i,j} = H_{rnd}(seed_{i,j}, ctnt_z)$, $\delta_{j,i} = H_{rnd}(seed_{j,i}, ctnt_z)$.
   Broadcast $\left(D_{i,j}^{(i)}, D_{j,i}^{(i)}\right)_{j \in SS \setminus \{i\}}$

Deterministic

2. Broadcast $\left(seed_{i,j}^{(i)}, seed_{j,i}^{(i)}\right)$ for j s.t. $D_{i,j}^{(i)} \neq D_{i,j}^{(j)}$ or $D_{j,i}^{(i)} \neq D_{j,i}^{(j)}$

Inconsistent Mask

$H_{msk}\left(seed_{k,\ell}^{(k)}, ctnt_z\right)$ $H_{rnd}\left(seed_{k,\ell}^{(k)}, ctnt_z\right)$

3. Check $D_{k,\ell}^{(k)} = Com\left(\boldsymbol{m}_{k,\ell}^{(k)}; \delta_{k,\ell}^{(k)}\right)$ and $C_{i,j} = H_{seed}(seed_{k,\ell}^{(k)})$
   If not, k is misbehavior.

Generated in KeyGen

How to Check (★)

Com: Lattice-based commitment scheme

1. For $j \in SS \setminus \{i\}$, compute $D_{i,j}^{(i)} \leftarrow Com(\boldsymbol{m}_{i,j}; \delta_{i,j}$
   where $\delta_{i,j} = H_{rnd}(seed_{i,j}, ctnt_z)$, $\delta_{j,i} = H_{rnd}($
   Broadcast $\left(D_{i,j}^{(i)}, D_{j,i}^{(i)}\right)_{j \in SS \setminus \{i\}}$

Thanks to binding of commitment scheme, we can ensure that each pair uses the same masks!

2. Broadcast $\left(seed_{i,j}^{(i)}, seed_{j,i}^{(i)}\right)$ for j s.t. $D_{i,j}^{(i)} \neq D_{i,j}^{(j)}$ or $D_{j,i}^{(i)} \neq D_{j,i}^{(j)}$

Inconsistent Mask

$H_{msk}\left(seed_{k,\ell}^{(k)}, ctnt_z\right)$

$H_{rnd}\left(seed_{k,\ell}^{(k)}, ctnt_z\right)$

3. Check $D_{k,\ell}^{(k)} = Com\left(\boldsymbol{m}_{k,\ell}^{(k)}; \delta_{k,\ell}^{(k)}\right)$ and $C_{i,j} = H_{seed}(seed_{k,\ell}^{(k)})$
   If not, k is misbehavior.

Generated in KeyGen

How to Check (★)

heme

Revealing seeds does not harm the security because seeds for honest pairs are not revealed.

$\leftarrow Com(\boldsymbol{m}_{i,j}; \delta_{i,j}$

$t_z), \delta_{j,i} = H_{rnd}($

Thanks to binding of commitment scheme, we can ensure that each pair uses the same masks!

Inconsistent Mask

2. Broadcast $\left(seed_{i,j}^{(i)}, seed_{j,i}^{(i)}\right)$ for j s.t. $D_{i,j}^{(i)} \neq D_{i,j}^{(j)}$ or $D_{j,i}^{(i)} \neq D_{j,i}^{(j)}$

$H_{msk}\left(seed_{k,\ell}^{(k)}, ctnt_z\right)$

$H_{rnd}\left(seed_{k,\ell}^{(k)}, ctnt_z\right)$

3. Check $D_{k,\ell}^{(k)} = Com\left(\boldsymbol{m}_{k,\ell}^{(k)}; \delta_{k,\ell}^{(k)}\right)$ and $C_{i,j} = H_{seed}(seed_{k,\ell}^{(k)})$
   If not, k is misbehavior.

Generated in KeyGen

Eventual Relations to be Proven via NIZK

Our IA protocol follows the approach using NIZK.

Relations to be proven:

(1) $\boldsymbol{r}_i$ is short

(2) $\boldsymbol{z}_i = c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

(3) $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$

~~(4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$~~

(4)' $D_{i,j}^{(i)} = Com(\boldsymbol{m}_{i,j}; \delta_{i,j})$

Eventual Relations to be Proven via NIZK

Our IA protocol follows the approach using NIZK.

Relations to be proven:

(1) $\boldsymbol{r}_i$ is short

(2) $\boldsymbol{z}_i = c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

(3) $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$

~~(4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$~~

(4)' $D_{i,j}^{(i)} = Com(\boldsymbol{m}_{i,j}; \delta_{i,j})$

Algebraic !!

Eventual Relations to be Proven via NIZK

Our IA protocol follows the approach using NIZK.

Relations to be proven:

(1) $\boldsymbol{r}_i$ is short

(2) $\boldsymbol{z}_i = c \cdot L_{SS,i} \cdot \boldsymbol{s}_i + \boldsymbol{r}_i + \Delta_i$

(3) $\Delta_i = \sum_j (\boldsymbol{m}_{i,j} - \boldsymbol{m}_{j,i})$

~~(4) $\boldsymbol{m}_{i,j} = H_{msk}(seed_{i,j}, ctnt_z)$~~

(4)' $D_{i,j}^{(i)} = Com(\boldsymbol{m}_{i,j}; \delta_{i,j})$

Algebraic !!

<u>Lattice-based ZK-SNARK combining LNP[LNP22] + LaBRADOR[BS23]</u>
which is sketched in prior works [BS23,ADDG24].
We formally analyze security of this approach in a modular manner.

Performance

	$\lvert\sigma\rvert$	Com Cost in Signing	$\lvert SS\rvert$	Availability
Traccoon[dPKM+24]	12.7	28.2	T	-
Traccoon-IA	12.7	28.2	T	IA $60+6.4\cdot T$

Same cost in signing protocol

Simple add-on

Thank you for your attention!!

Future Works:

- Does our technique work on related lattice-based schemes using masking mechanism [EKT24], [KRT24], [BKL+25].
- Distributed Key Generation for our scheme

Independent and Concurrent Work:

[dPENP] Del Pino et al. "Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts"

- IA for a variant of TRaccoon based on new short secret sharing technique
- Non-interactive IA
- Efficient when the number of signers or corruption threshold is small