Universally Composable Interactive and Ordered Multisignatures

Carsten Baum

DTU

Bernardo David

IT UNIVERSITY OF CPH

Elena Pagnin

CHALMERS
UNIVERSITY OF TECHNOLOGY

Akira Takahashi

J.P.Morgan
AI Research &
Algocrypt COE

In a Nutshell

- Interactive Multi-Signature (IMS): Allows n parties to generated a compact signature on the same message given interaction, e.g., preprocessing
- Ordered Multi-Signature (OMS): Order of signing attempts matters
- Previous OMS: Based on bilinear pairings
- Our results
  - First Schnorr-like OMS with a single-round online phase
  - New UC definitions for IMS
  - New game-based and UC definitions for OMS with preprocessing
  - Equivalence between game-based and UC definitions for IMS/OMS
- Applications to Sequential Communication Delay [BDPT24] (used for VDFs and TLPs) and routing [BGOY07]

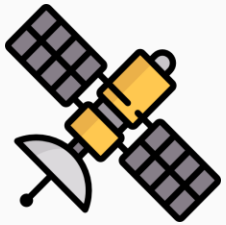
Motivating Application

Sequential Communication Delay Protocol [BDPT24]

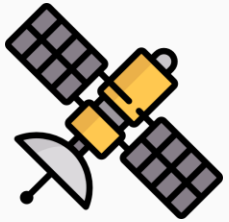
- Each satellite has sk_i
- Due to the speed of light, communication delay between two satellites is precisely lower bounded by their relative distance
- OMS guarantees m has incurred a certain minimum delay due to being transmitted from satellite 1 to n
- Proof of SCD helps us construct VDF

Verifier

$m, \sigma, L = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$



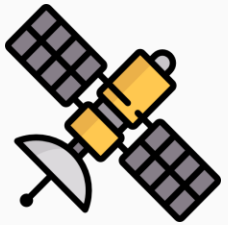
m, σ_1



m, σ_2

$\ldots \quad \ldots$

m, σ_{n-1}



$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m, L)$ $\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m, L)$ $\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m, L)$

(Interactive) Multi-Signature

sk_2

Signer

sk_1

Signer

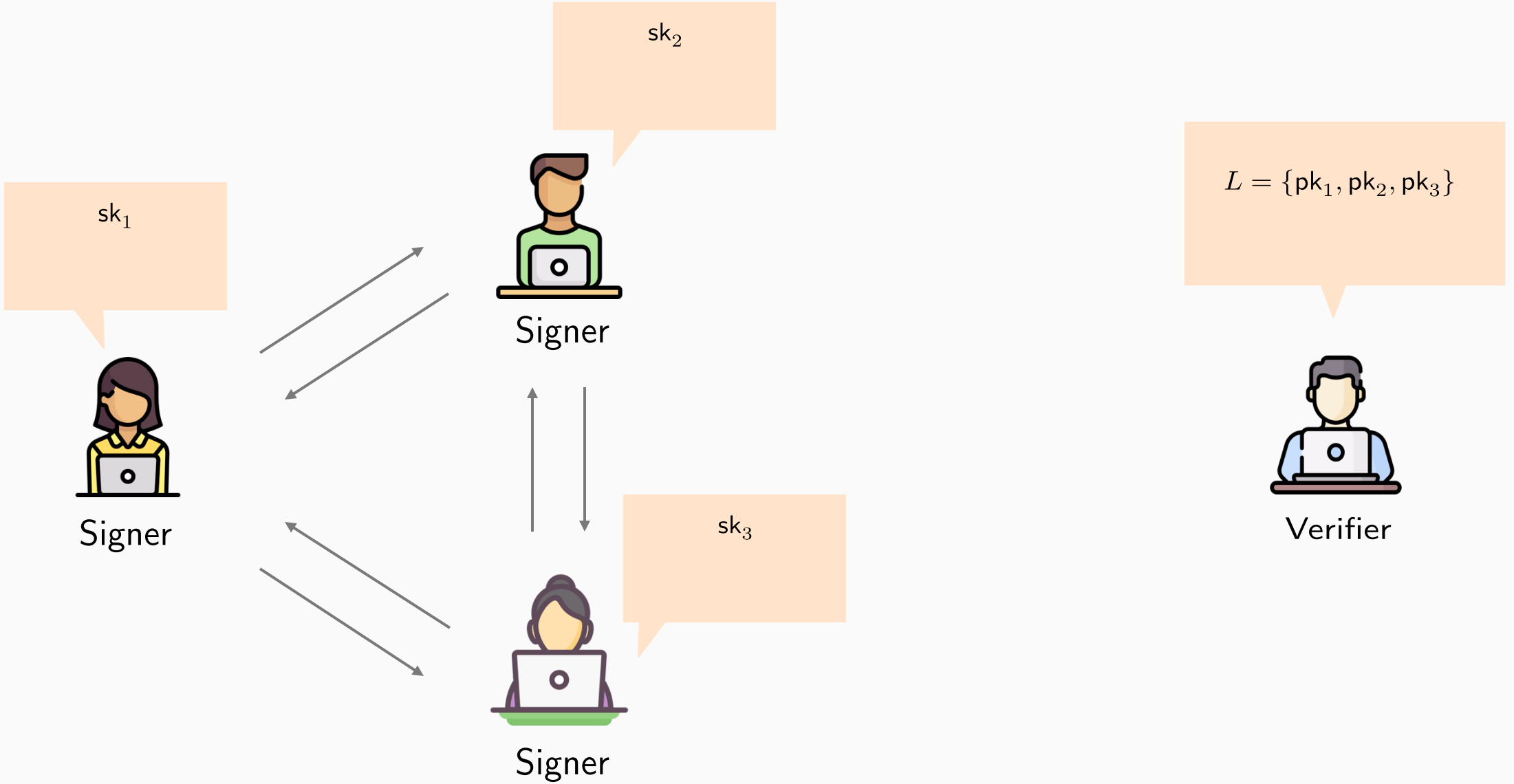
$L = \{\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3\}$

Verifier

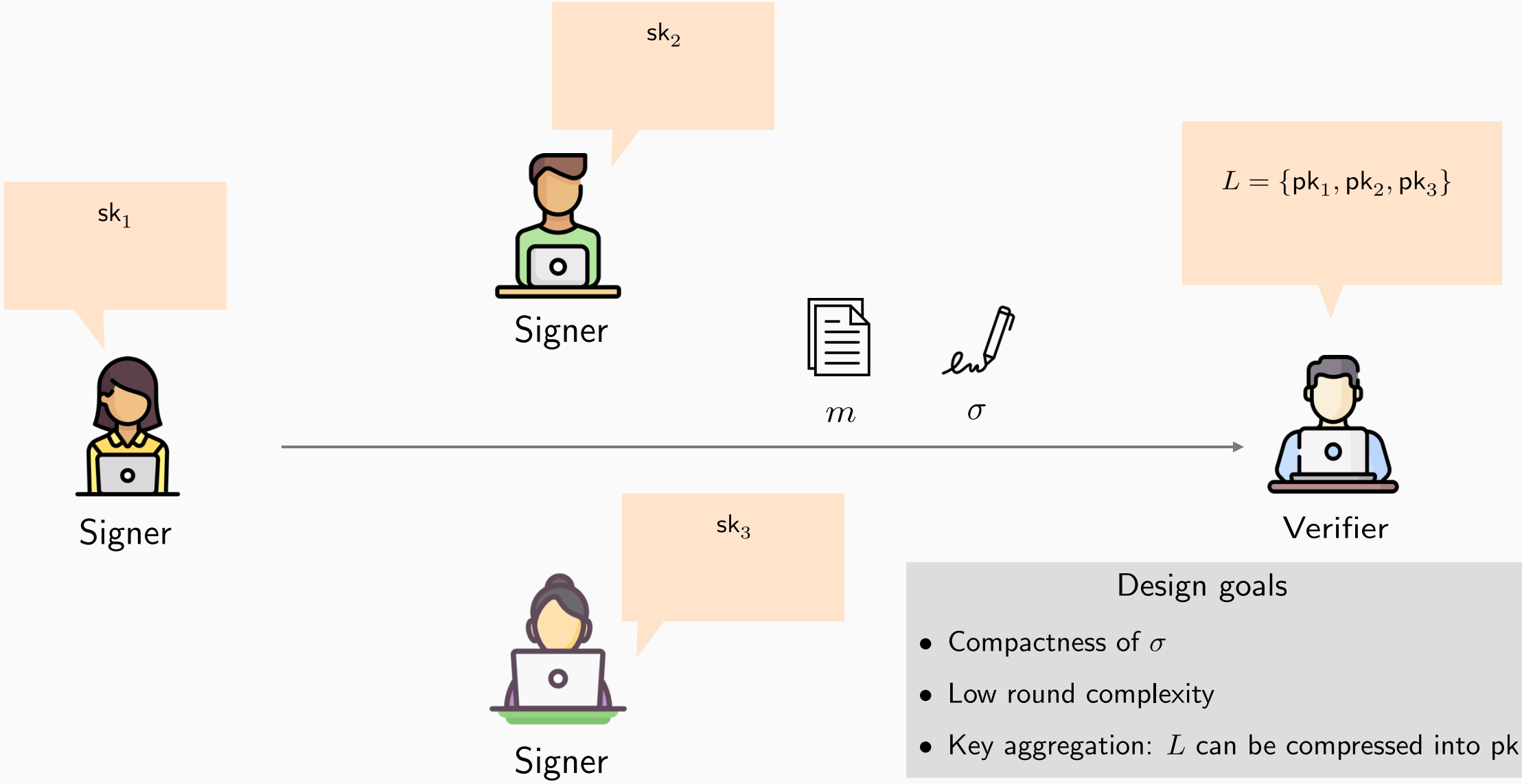
sk_3

Signer

(Interactive) Multi-Signature



(Interactive) Multi-Signature



Design goals

- Compactness of σ
- Low round complexity
- Key aggregation: L can be compressed into pk

Security Notion of Multi-Signature

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$

MS-UF-CMA game

Attacker wins if:

- (σ^*, m^*, L^*) is valid
- $\mathsf{pk}_1 \in L^*$
- (m^*, L^*) hasn't been queried

- Attacker can **concurrently** launch multiple sessions
- Sufficient to consider "all-but-one" corruptions

$\mathsf{Signer}(\mathsf{sk}_1, \mathsf{sid}, m, L)$

$\mathsf{Attacker}(\mathsf{pk}_1)$

Security Notion of Multi-Signature

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$

$m \qquad L$

σ

$\mathsf{Signer}(\mathsf{sk}_1, \mathsf{sid}, m, L)$ → $\mathsf{Attacker}(\mathsf{pk}_1)$

MS-UF-CMA game

Attacker wins if:

- (σ^*, m^*, L^*) is valid
- $\mathsf{pk}_1 \in L^*$
- (m^*, L^*) hasn't been queried

- Attacker can **concurrently** launch multiple sessions
- Sufficient to consider "all-but-one" corruptions

Security Notion of Multi-Signature

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$

MS-UF-CMA game

Attacker wins if:

- (σ^*, m^*, L^*) is valid
- $\mathsf{pk}_1 \in L^*$
- (m^*, L^*) hasn't been queried

- Attacker can **concurrently** launch multiple sessions
- Sufficient to consider "all-but-one" corruptions

$m \quad L$

$\vdots$

σ

$\mathsf{Signer}(\mathsf{sk}_1, \mathsf{sid}, m, L)$

$\mathsf{Attacker}(\mathsf{pk}_1)$

$\sigma^* \quad m^* \quad L^*$

Verifier

Ordered Multi-Signature [BGOY07]

- MS with stronger guarantee, with an **ordered** set L
- Application to routing path verification, SCD, VDF, etc.
- Parties in $L = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ must sign in the specified order
- Naive way: signer i signs $(m, \sigma_1, \ldots, \sigma_{i-1}) \rightsquigarrow$ not compact!
- The only known construction is from pairing [BGOY07]
- Can we instantiate efficient OMS from standard prime order groups?

Verifier

$m, \sigma, L = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m, L)$

m, σ_1

$\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m, L)$

m, σ_2

$\cdots \quad \cdots$

m, σ_{n-1}

$\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m, L)$

Security Goal of OMS

- Let $H = (\mathsf{pk}_{i_1}, \ldots, \mathsf{pk}_{i_h}) \subseteq L$ be a sequence of honest keys
- Adversary cannot shuffle the order of signing attempts made by parties in H
- For all $j, k \in [1, h]$ such that $j < k$, the owner of pk_{i_j} must have signed **before** pk_{i_k}
- "All-but-one" corruption model is not enough!
- We only consider static corruption

Verifier

$m, \sigma, L = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

$\ldots \quad \ldots$

m, σ_{i_j}

$\mathsf{Sign}_{\mathsf{sk}_{i_j}}(\sigma_{i_j-1}, m, L)$

$\ldots$

$\ldots \quad \ldots$

$\mathsf{Sign}_{\mathsf{sk}_{i_k}}(\sigma_{i_k-1}, m, L)$

$\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m, L)$

Next Step: Adapting MuSig2

Offline Phase

1: $r_{i,1}, r_{i,2} \leftarrow\$ \mathbb{Z}_q$
2: $R_{i,1} = g^{r_{i,1}}; R_{i,2} = g^{r_{i,2}}$
3: Broadcast $R_{i,1}, R_{i,2}$
4: Receive $R_{j,1}, R_{j,2}$ for $j \neq i$
5: $R_1 = \prod_{j=1}^{n} R_{j,1}$
6: $R_2 = \prod_{j=1}^{n} R_{j,2}$

Online Phase

1: Receive $(R', \tilde{z})$ and m from party $i-1$
2: $v = H(m, L, (R_{j,1}, R_{j,2})_{j=1}^{n})$
3: $R = R_1 \cdot R_2^v$
4: Check $R = R'$
5: $c = H(R, m, L)$
6: $z_i = c \cdot \mathsf{sk}_i + r_{i,1} + v \cdot r_{i,2}$
7: $z := \tilde{z} + z_i$
8: Send $\sigma = (R, z)$ and m to party $i+1$

1: Parse $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) := L$
2: $\mathsf{pk} = \prod_{j=1}^{n} \mathsf{pk}_j$
3: $c = H(R, m, L)$
4: Check $R = g^z \mathsf{pk}^c$

$\mathsf{Signer}_{\mathsf{sk}_i}$

m, σ

$\ldots \quad \ldots$

m, σ

$\mathsf{Verifier}(\sigma, m, L)$

Next Step: Adapting MuSig2

Offline Phase

1: $r_{i,1}, r_{i,2} \leftarrow\$ \mathbb{Z}_q$
2: $R_{i,1} = g^{r_{i,1}}; R_{i,2} = g^{r_{i,2}}$
3: Broadcast $R_{i,1}, R_{i,2}$
4: Receive $R_{j,1}, R_{j,2}$ for $j \neq i$
5: $R_1 = \prod_{j=1}^{n} R_{j,1}$
6: $R_2 = \prod_{j=1}^{n} R_{j,2}$

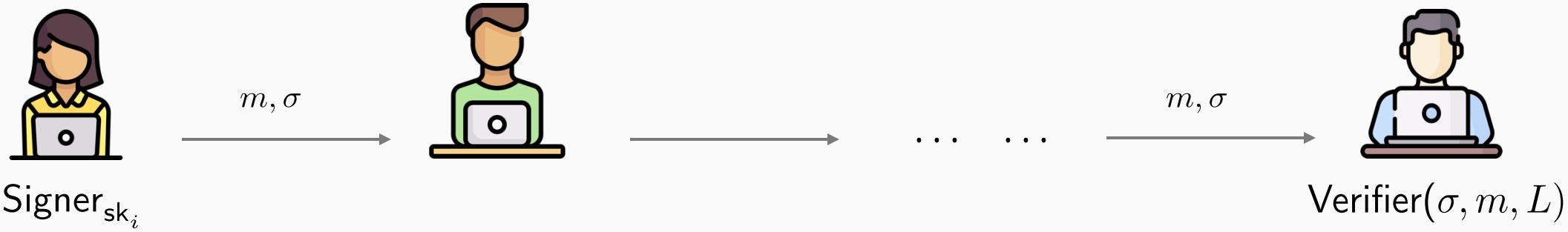
Online Phase

1: Receive $(R', \tilde{z})$ and m from party $i-1$
2: $v = H(m, L, (R_{j,1}, R_{j,2})_{j=1}^{n})$
3: $R = R_1 \cdot R_2^v$
4: Check $R = R'$
5: $c = H(R, m, L)$
6: $z_i = c \cdot \mathsf{sk}_i + r_{i,1} + v \cdot r_{i,2}$
7: $z := \tilde{z} + z_i$
8: Send $\sigma = (R, z)$ and m to party $i+1$

Attack!

- Party i may agree to sign before party $j < i$ contributes
- Need to validate **so-far** aggregation

1: Parse $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) := L$
2: $\mathsf{pk} = \prod_{j=1}^{n} \mathsf{pk}_j$
3: $c = H(R, m, L)$
4: Check $R = g^z \mathsf{pk}^c$



Secure Construction: Ordered MuSig2

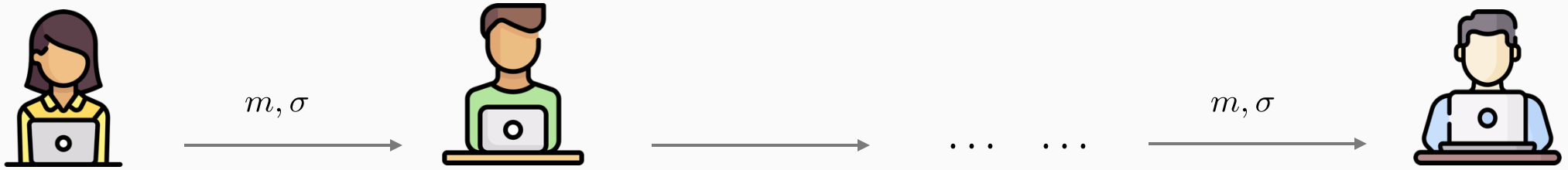
Offline Phase

1: $r_{i,1}, r_{i,2} \leftarrow\$ \mathbb{Z}_q$
2: $R_{i,1} = g^{r_{i,1}}; R_{i,2} = g^{r_{i,2}}$
3: Broadcast $R_{i,1}, R_{i,2}$
4: Receive $R_{j,1}, R_{j,2}$ for $j \neq i$
5: $R_1 = \prod_{j=1}^{n} R_{j,1}$
6: $R_2 = \prod_{j=1}^{n} R_{j,2}$
7: $\tilde{R}_1 = \prod_{j=1}^{i-1} R_{j,1}$
8: $\tilde{R}_2 = \prod_{j=1}^{i-1} R_{j,2}$
9: $\tilde{\mathsf{pk}} = \prod_{j=1}^{i-1} \mathsf{pk}_j$

Online Phase

1: Receive $(R', \tilde{z})$ and m from party $i-1$
2: $v = H(m, L, (R_{j,1}, R_{j,2})_{j=1}^{n})$
3: $R = R_1 \cdot R_2^v$
4: Check $R = R'$
5: $\tilde{R} = \tilde{R}_1 \cdot \tilde{R}_2^v$
6: $c = H(R, m, L)$
7: Check $g^{\tilde{z}} = \tilde{R} \cdot \tilde{\mathsf{pk}}^c$
8: $z_i = c \cdot \mathsf{sk}_i + r_{i,1} + v \cdot r_{i,2}$
9: $z := \tilde{z} + z_i$
10: Send $\sigma = (R, z)$ and m to party $i+1$

1: Parse $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) := L$
2: $\mathsf{pk} = \prod_{j=1}^{n} \mathsf{pk}_j$
3: $c = H(R, m, L)$
4: Check $R = g^z \mathsf{pk}^c$



$\mathsf{Signer}_{\mathsf{sk}_i}$ $\mathsf{Verifier}(\sigma, m, L)$

Secure Construction: Ordered MuSig2

Offline Phase

1: $r_{i,1}, r_{i,2} \leftarrow\$ \mathbb{Z}_q$
2: $R_{i,1} = g^{r_{i,1}}; R_{i,2} = g^{r_{i,2}}$
3: Broadcast $R_{i,1}, R_{i,2}$
4: Receive $R_{j,1}, R_{j,2}$ for $j \neq i$
5: $R_1 = \prod_{j=1}^{n} R_{j,1}$
6: $R_2 = \prod_{j=1}^{n} R_{j,2}$
7: $\tilde{R}_1 = \prod_{j=1}^{i-1} R_{j,1}$
8: $\tilde{R}_2 = \prod_{j=1}^{i-1} R_{j,2}$
9: $\tilde{\mathsf{pk}} = \prod_{j=1}^{i-1} \mathsf{pk}_j$

Online Phase

1: Receive $(R', \tilde{z})$ and m from party $i-1$
2: $v = H(m, L, (R_{j,1}, R_{j,2})_{j=1}^{n})$
3: $R = R_1 \cdot R_2^v$
4: Check $R = R'$
5: $\tilde{R} = \tilde{R}_1 \cdot \tilde{R}_2^v$
6: $c = H(R, m, L)$
7: Check $g^{\tilde{z}} = \tilde{R} \cdot \tilde{\mathsf{pk}}^c$
8: $z_i = c \cdot \mathsf{sk}_i + r_{i,1} + v \cdot r_{i,2}$
9: $z := \tilde{z} + z_i$
10: Send $\sigma = (R, z)$ and m to party $i+1$

Security

- Party i makes sure $j < i$ contributed
- Since $\tilde{R}$ is not hashed, the proof gets quite involved
- Still secure under the AOMDL assumption

1: Parse $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) := L$
2: $\mathsf{pk} = \prod_{j=1}^{n} \mathsf{pk}_j$
3: $c = H(R, m, L)$
4: Check $R = g^z \mathsf{pk}^c$

UC security notions for IMS and OMS

- We define ideal functionalities for IMS and OMS adapting the standard digital signature ideal functionality of [Canetti04]
- As in [Canetti04] we show equivalence to game-based security definitions
- Special care must be taken in modelling the interaction/preprocessing (and signing order) in the ideal functionality and correctness in the game [CDLLR24]
- A known caveat:
  - As in [Canetti04], our functionalities allow a Signer to request signatures, which are provided by the ideal adversary and returned to the Signer
  - A concurrent work [CDLLR24] observes that the adversary can refuse to provide signatures and essentially halt the ideal functionality of [Canetti04]
  - This issue is also present in our ideal functionalities but has no impact in our applications, as the adversary could simply crash one of the signers

Wrapping up

- We constructed an efficient and compact Schnorr-based OMS
- Defined UC security notion for IMS
- Defined both game-based and UC security notions for preprocessing OMS
- Proved equivalence between game-based and UC definitions for IMS and OMS
- Open questions:
  - Security under adaptive corruptions?
  - Post-quantum instantiation e.g. lattices