FROST Research Updates

Elizabeth Crites
University of Edinburgh

Chelsea Komlo
University of Waterloo
Zcash Foundation

August 7, 2022

What is FROST?

- **<u>F</u>**lexible **<u>R</u>**ound-**<u>O</u>**ptimized **<u>S</u>**chnorr **<u>T</u>**hreshold **<u>S</u>**ignatures [Komlo & Goldberg, 2020]

eprint.iacr.org/2020/852

What is FROST?

- <u>**F**</u>lexible <u>**R**</u>ound-<u>**O**</u>ptimized <u>**S**</u>chnorr <u>**T**</u>hreshold <u>**S**</u>ignatures [Komlo & Goldberg, 2020]
  1. PedPop: Distributed Key Generation (DKG)

eprint.iacr.org/2020/852

What is FROST?

- **F**lexible **R**ound-**O**ptimized **S**chnorr **T**hreshold **S**ignatures [Komlo & Goldberg, 2020]
    1. PedPop: Distributed Key Generation (DKG)
    2. Two-round threshold signing that is concurrently secure

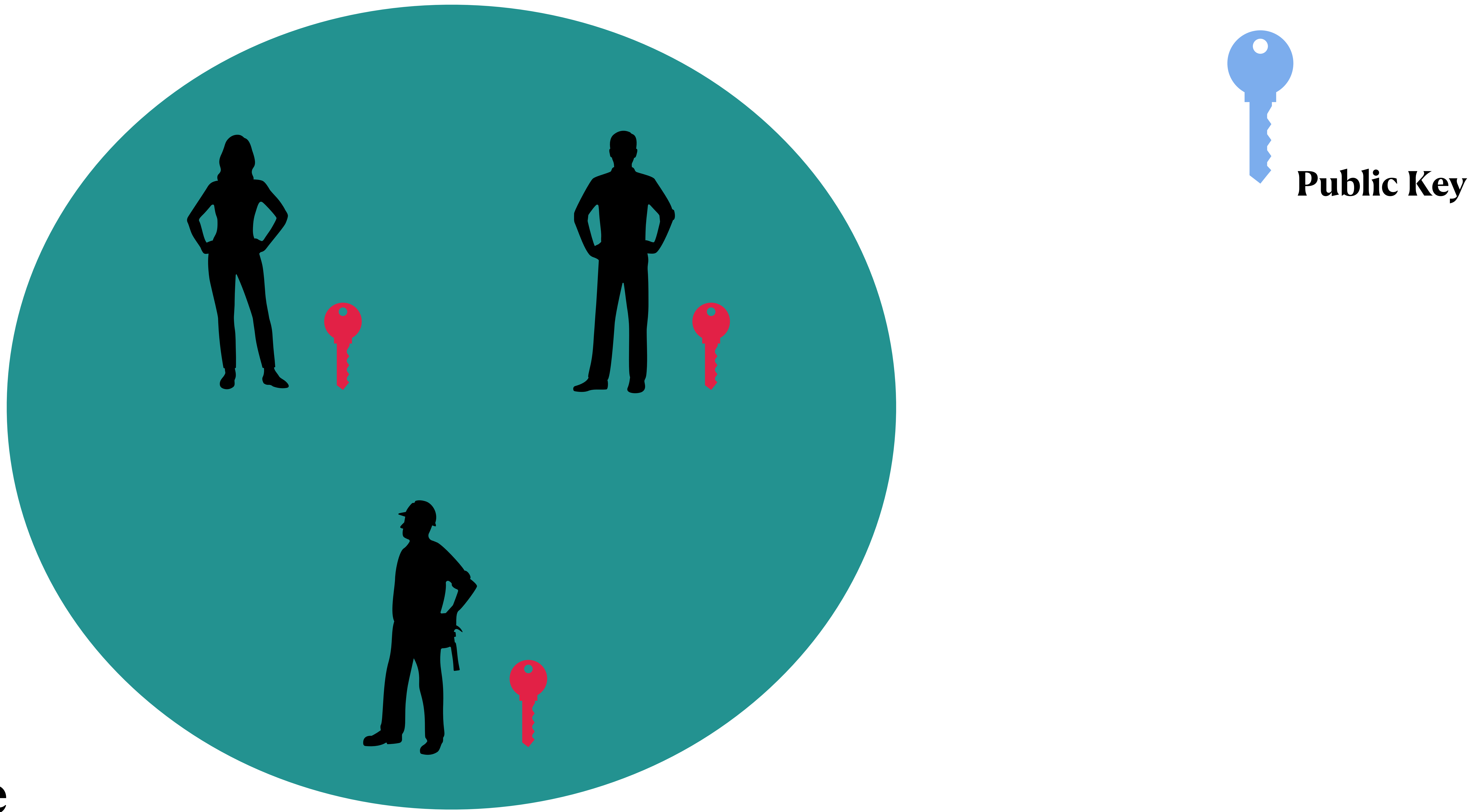
eprint.iacr.org/2020/852

What is FROST?

- **F**lexible **R**ound-**O**ptimized **S**chnorr **T**hreshold **S**ignatures [Komlo & Goldberg, 2020]
    1. PedPop: Distributed Key Generation (DKG)
    2. Two-round threshold signing that is concurrently secure
- Designed to solve needs in the Zcash ecosystem; now adopted as an industry standard

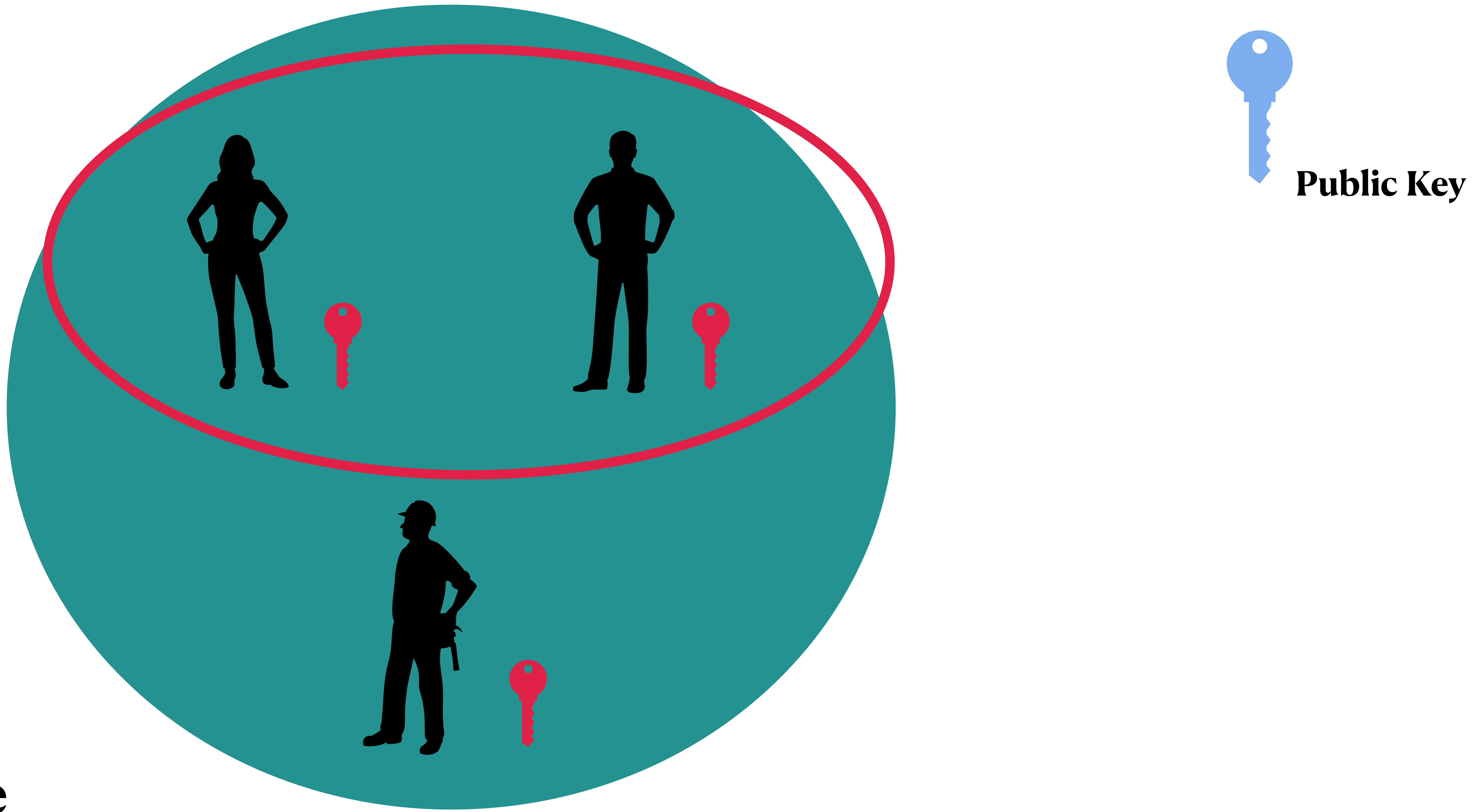
eprint.iacr.org/2020/852

What are Threshold Signatures?



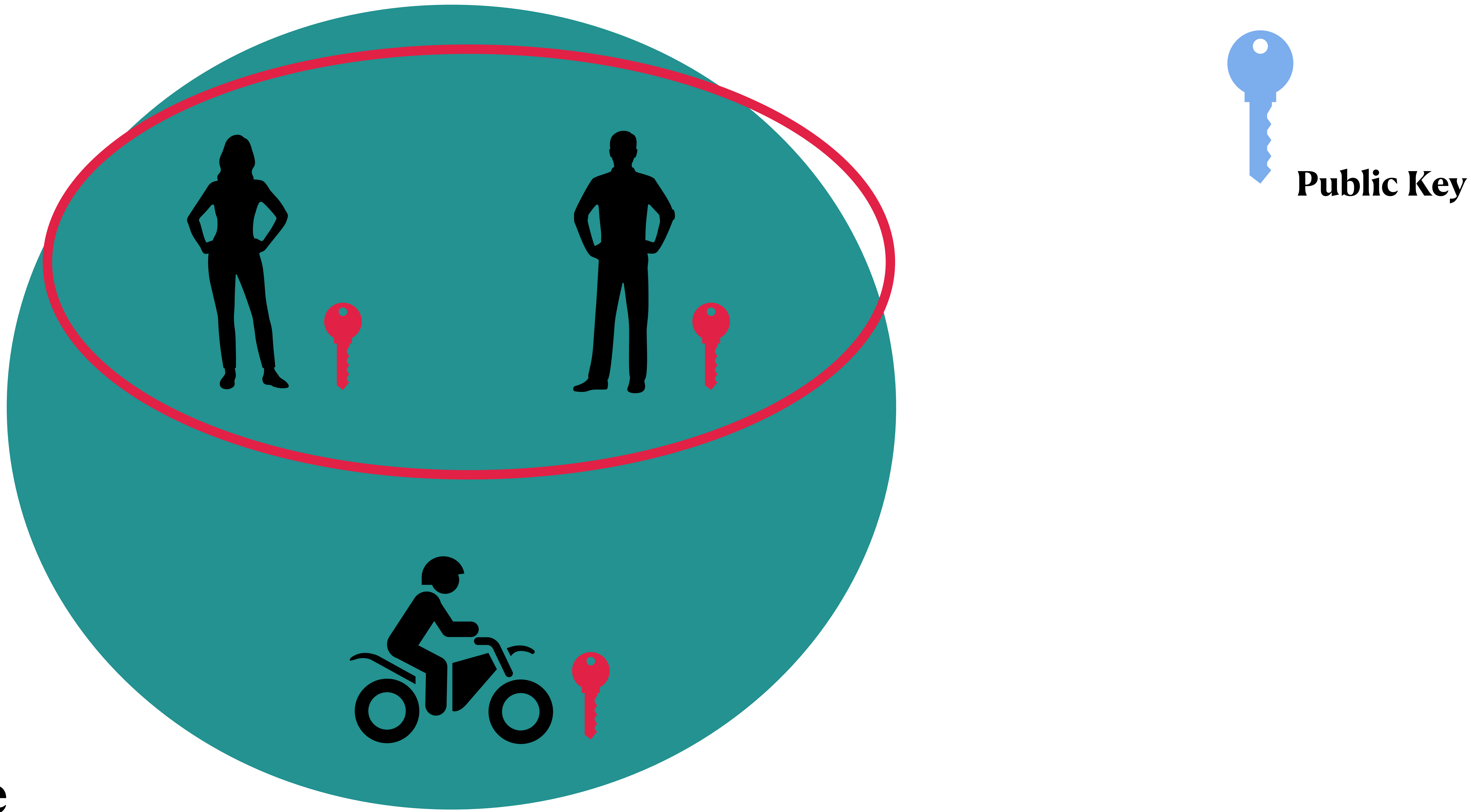
(2,3) Example

What are Threshold Signatures?



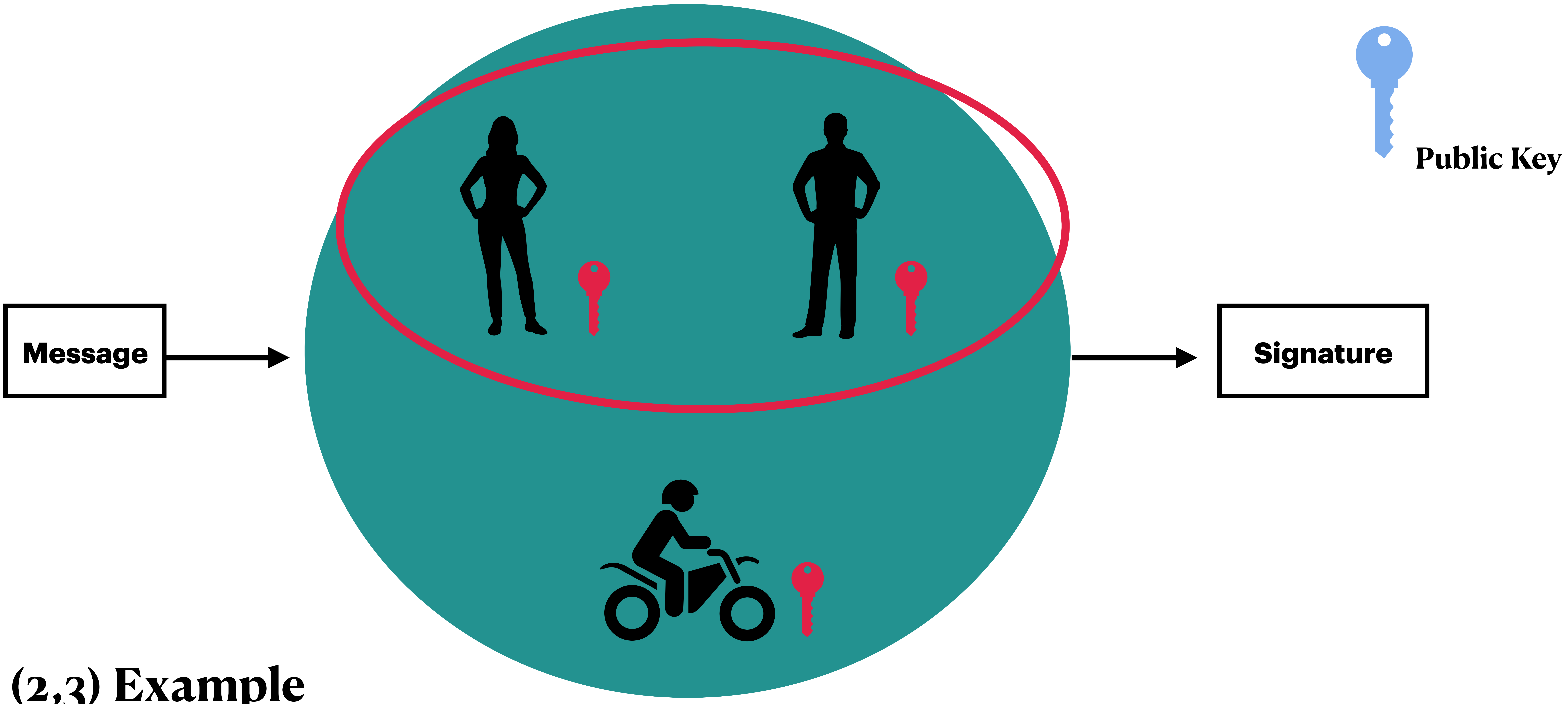
(2,3) Example

What are Threshold Signatures?



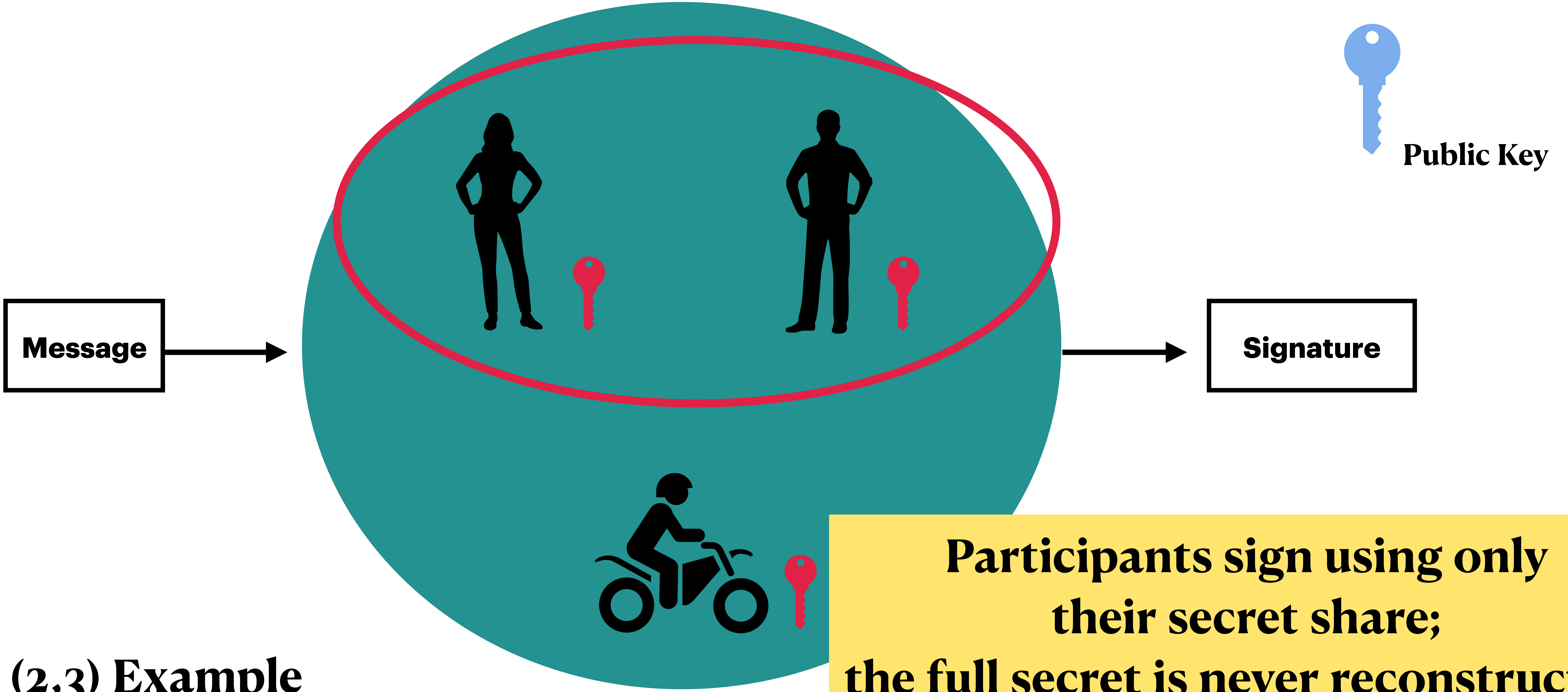
(2,3) Example

What are Threshold Signatures?



(2,3) Example

What are Threshold Signatures?

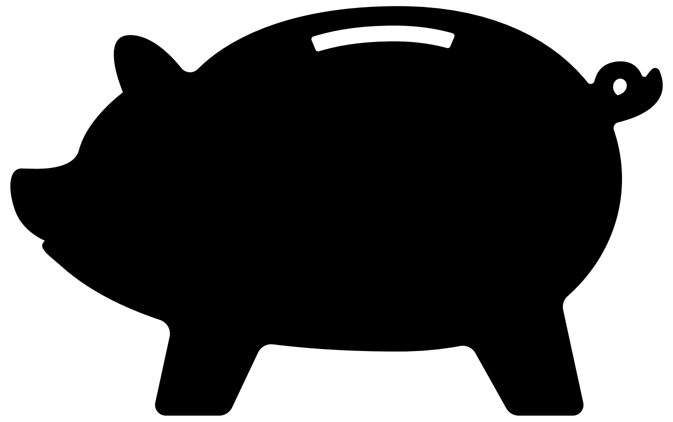


(2,3) Example

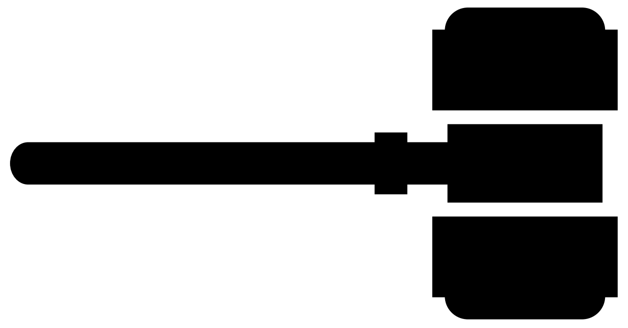
Participants sign using only their secret share; the full secret is never reconstructed!

Uses for Threshold Signatures

Banks and Exchanges



Cryptocurrency Wallets



Trust Authorities (CAs)

Where was FROST Last Year?

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Where is FROST *Now?*

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Where is FROST *Now?*

5+ Implementations

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Where is FROST *Now?*

5+ Implementations

Optimized FROST2

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Where is FROST *Now?*

5+ Implementations

Optimized FROST2

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Additional Security Proofs

Where is FROST *Now?*

5+ Implementations

Optimized FROST2

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Additional Security Proofs

IETF Draft Almost Done

Where is FROST *Now?*

5+ Implementations

Optimized FROST2

Robust FROST (ROAST)

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Additional Security Proofs

IETF Draft Almost Done

Where is FROST *Now?*

5+ Implementations

Robust FROST (ROAST)

Optimized FROST2

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Additional Security Proofs

IETF Draft Almost Done

Threshold BLS & PedPop

Where is FROST *Now?*

- 5+ Implementations
- Optimized FROST2
- Robust FROST (ROAST)
- Paper Published (FROST1)
- Presented to NIST
- IETF Draft Started
- Unlinkable FROST
- Additional Security Proofs
- IETF Draft Almost Done
- Threshold BLS & PedPop

Where is FROST *Now?*

5+ Implementations

Optimized FROST2

Practical use!

Robust FROST (ROAST)

Paper Published (FROST1)

Presented to NIST

IETF Draft Started

Unlinkable FROST

Additional Security Proofs

IETF Draft Almost Done

Threshold BLS & PedPop

How to Prove Schnorr Assuming Schnorr: Security of Multi- and Threshold Signatures

Elizabeth Crites, Chelsea Komlo, Mary Maller

eprint.iacr.org/2021/1375

(Single-Party) Schnorr Signature Scheme

(Single-Party) Schnorr Signature Scheme

To generate a key pair:

$$sk \xleftarrow{\$} \mathbb{F}\ ;\ \ PK \leftarrow g^{sk}$$

(Single-Party) Schnorr Signature Scheme

To generate a key pair:

$$sk \xleftarrow{\$} \mathbb{F} \,; \;\; PK \leftarrow g^{sk}$$

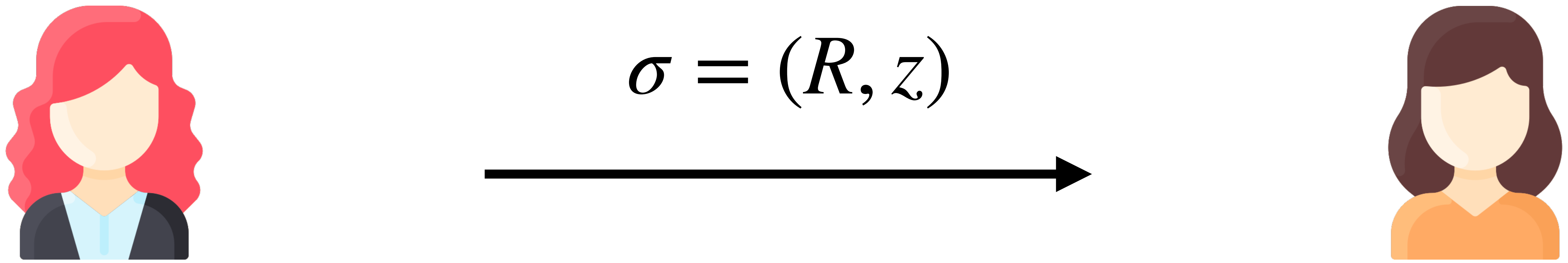
To sign a message m:

$$r \xleftarrow{\$} \mathbb{F} \,; \;\; R \leftarrow g^{r}$$

$$c \leftarrow H(PK, m, R)$$

$$z \leftarrow r + c\,sk$$

(Single-Party) Schnorr Signature Scheme



To generate a key pair:

$$sk \xleftarrow{\$} \mathbb{F}\ ;\ \ PK \leftarrow g^{sk}$$

To sign a message m:

$$r \xleftarrow{\$} \mathbb{F}\ ;\ \ R \leftarrow g^{r}$$

$$c \leftarrow H(PK, m, R)$$

$$z \leftarrow r + c\,sk$$

(Single-Party) Schnorr Signature Scheme

$$\xrightarrow{\sigma = (R, z)}$$

To generate a key pair:

$$sk \xleftarrow{\$} \mathbb{F}\ ;\ \ PK \leftarrow g^{sk}$$

To sign a message m:

$$r \xleftarrow{\$} \mathbb{F}\ ;\ \ R \leftarrow g^{r}$$

$$c \leftarrow H(PK, m, R)$$

$$z \leftarrow r + csk$$

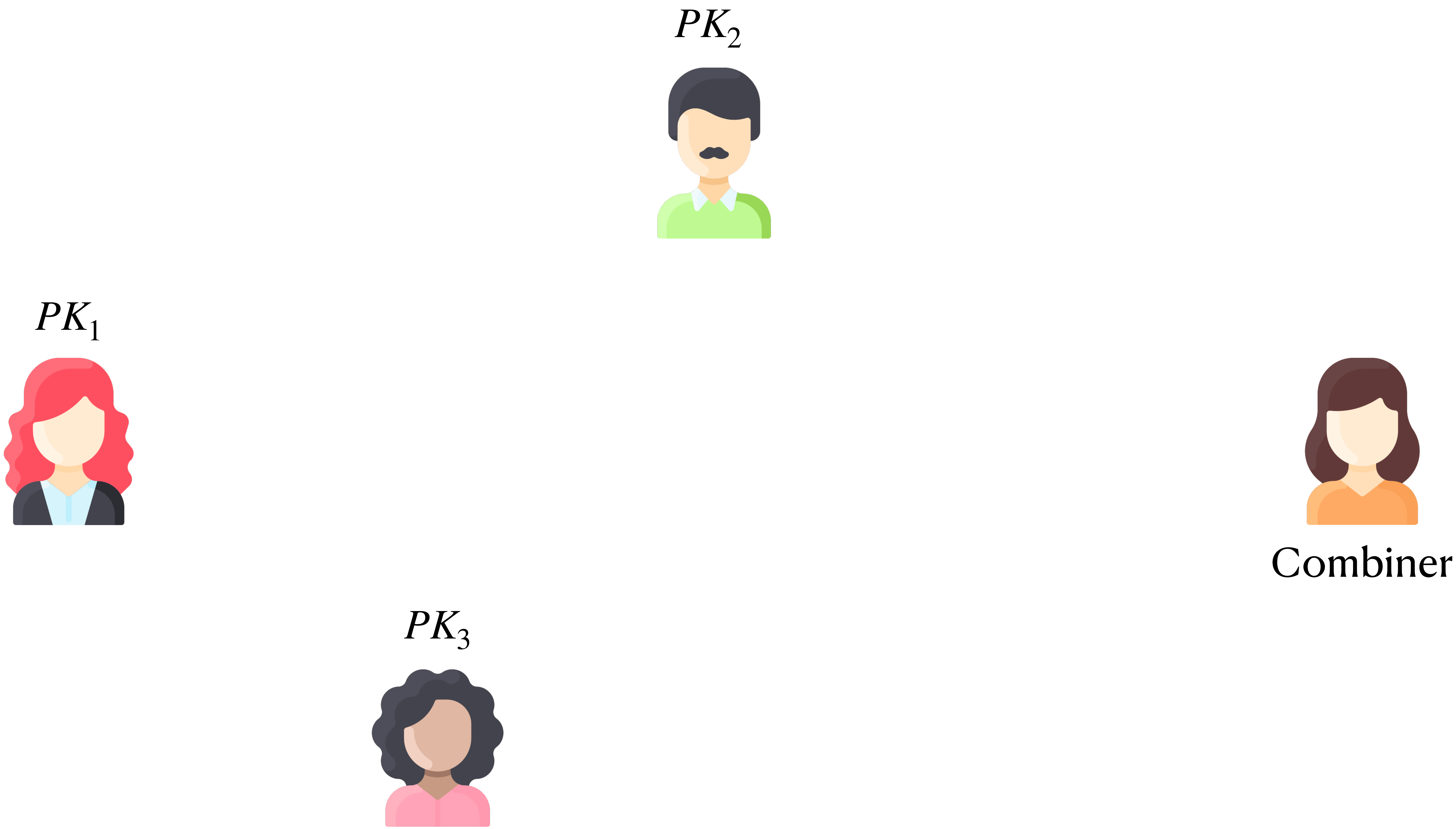
To verify (PK, σ, m):

$$c \leftarrow H(PK, m, R)$$

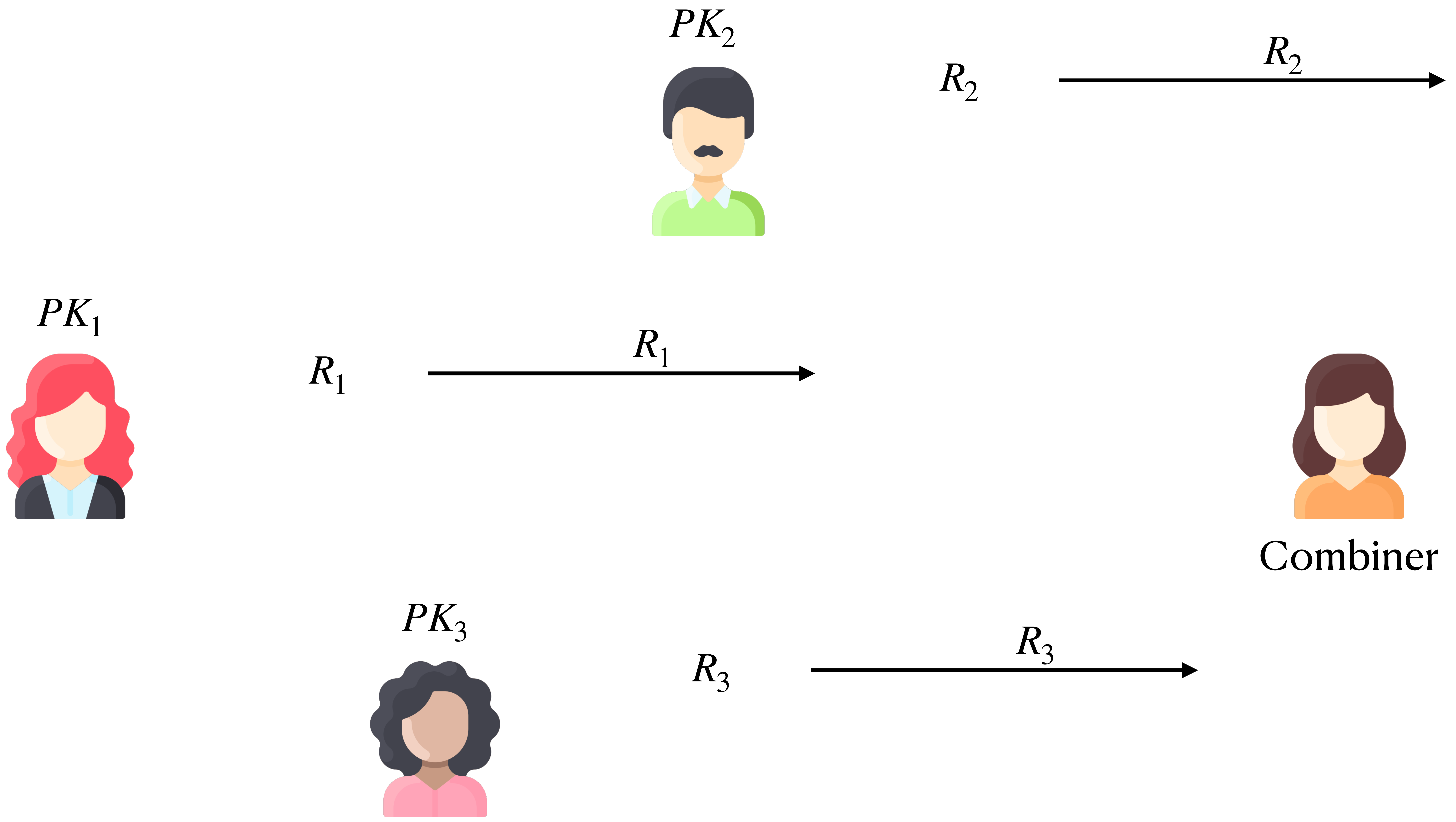
$$R \cdot PK^{c} \stackrel{?}{=} g^{z}$$

output accept/reject

Attempt: Multi-Party Schnorr



Attempt: Multi-Party Schnorr



Attempt: Multi-Party Schnorr

PK_2

R_2

$z_2 \leftarrow r_2 + csk_2$

$\xrightarrow{R_2}$

$\xleftarrow{c}$

w/ proof of possession of sk_j

PK_1

R_1

$z_1 \leftarrow r_1 + csk_1$

$\xrightarrow{R_1}$

$\xleftarrow{c}$

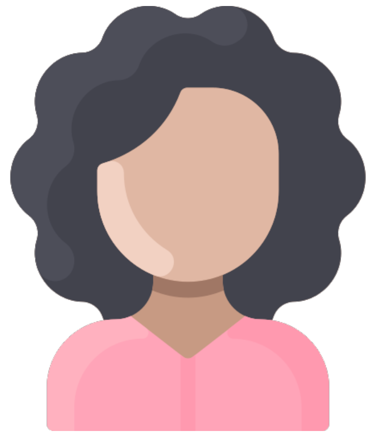
$\tilde{PK} = PK_1 PK_2 PK_3$

$\tilde{R} = R_1 R_2 R_3$

$c \leftarrow H(\tilde{PK}, m, \tilde{R})$

Combiner

PK_3



R_3

$z_3 \leftarrow r_3 + csk_3$

$\xrightarrow{R_3}$

$\xleftarrow{c}$

Attempt: Multi-Party Schnorr

PK_2

R_2

$z_2 \leftarrow r_2 + csk_2$

$\xrightarrow{R_2}$

$\xleftarrow{c}$

$\xrightarrow{z_2}$

PK_1

R_1

$z_1 \leftarrow r_1 + csk_1$

$\xrightarrow{R_1}$

$\xleftarrow{c}$

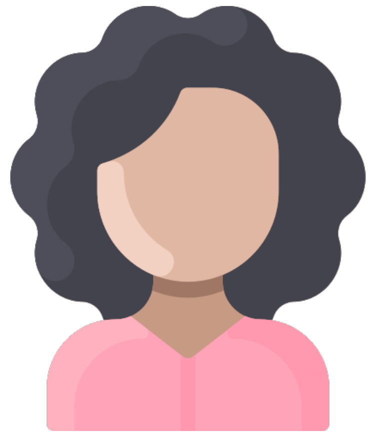
$\xrightarrow{z_1}$

Combiner

w/ proof of possession of sk_j

$$\tilde{PK} = PK_1 PK_2 PK_3$$
$$\tilde{R} = R_1 R_2 R_3$$
$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$
$$z \leftarrow z_1 + z_2 + z_3$$
$$\sigma = (\tilde{R}, z)$$

PK_3



R_3

$z_3 \leftarrow r_3 + csk_3$

$\xrightarrow{R_3}$

$\xleftarrow{c}$

$\xrightarrow{z_3}$

Attempt: Multi-Party Schnorr

PK_2

R_2

$z_2 \leftarrow r_2 + csk_2$

$\xrightarrow{R_2}$

$\xleftarrow{c}$

$\xrightarrow{z_2}$

w/ proof of possession of sk_j

PK_1

R_1

$z_1 \leftarrow r_1 + csk_1$

$\xrightarrow{R_1}$

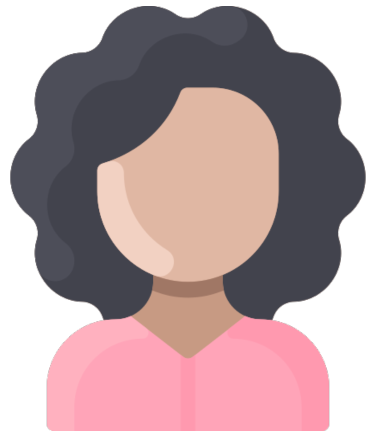
$\xleftarrow{c}$

$\xrightarrow{z_1}$

Combiner

$$\tilde{PK} = PK_1 PK_2 PK_3$$
$$\tilde{R} = R_1 R_2 R_3$$
$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$
$$z \leftarrow z_1 + z_2 + z_3$$
$$\sigma = (\tilde{R}, z)$$

PK_3



R_3

$z_3 \leftarrow r_3 + csk_3$

$\xrightarrow{R_3}$

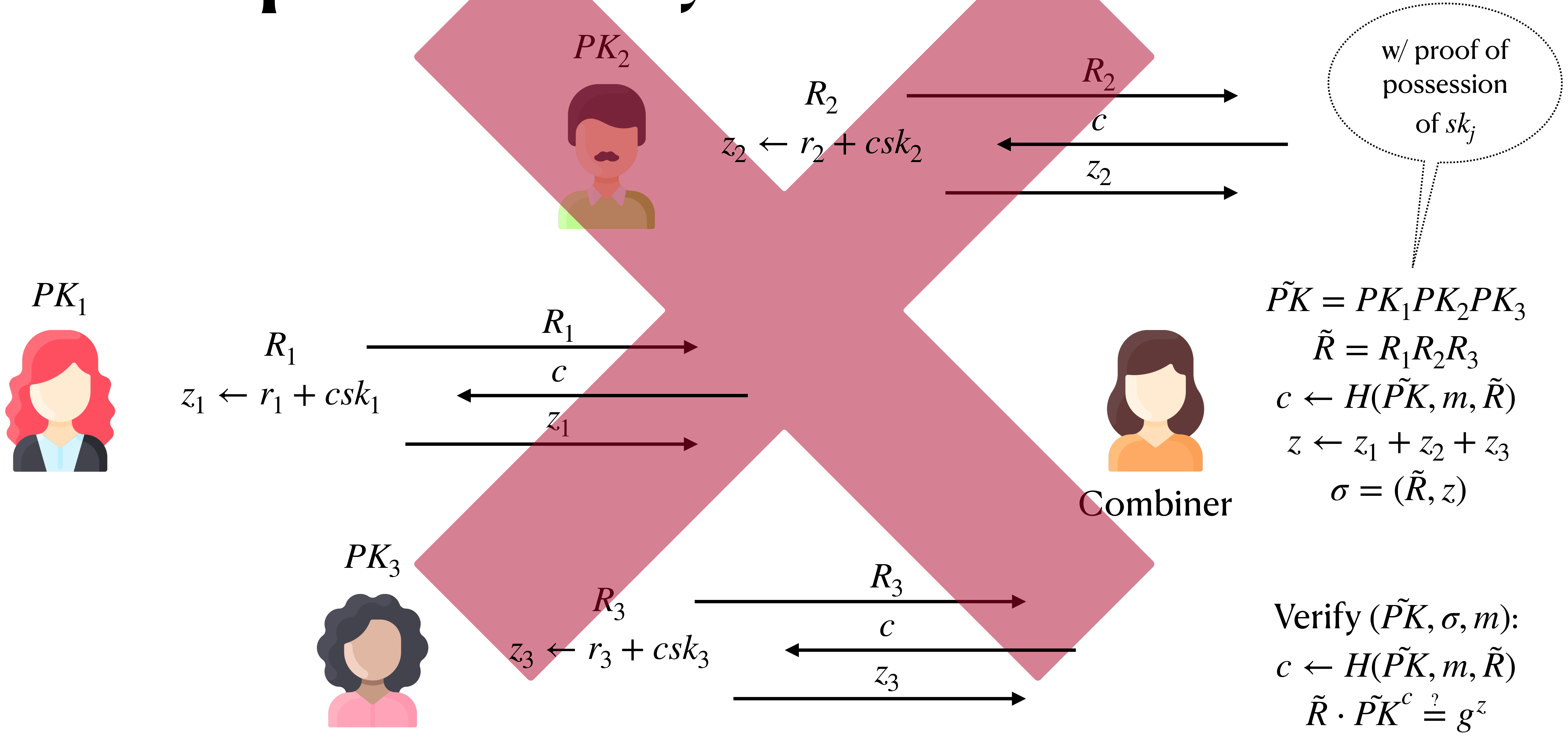
$\xleftarrow{c}$

$\xrightarrow{z_3}$

Verify $(\tilde{PK}, \sigma, m)$:

$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$
$$\tilde{R} \cdot \tilde{PK}^c \stackrel{?}{=} g^z$$

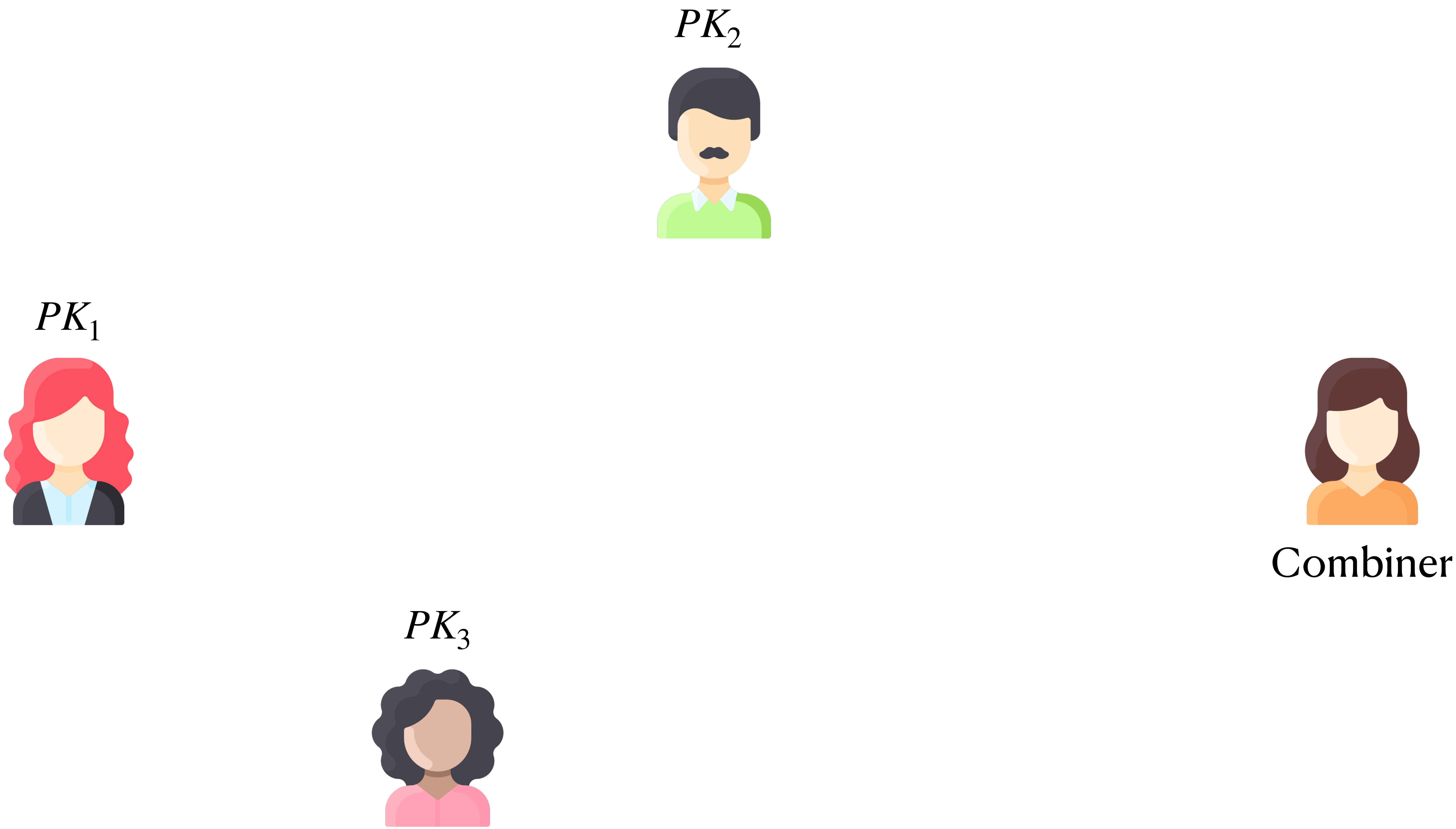
Attempt: Multi-Party Schnorr



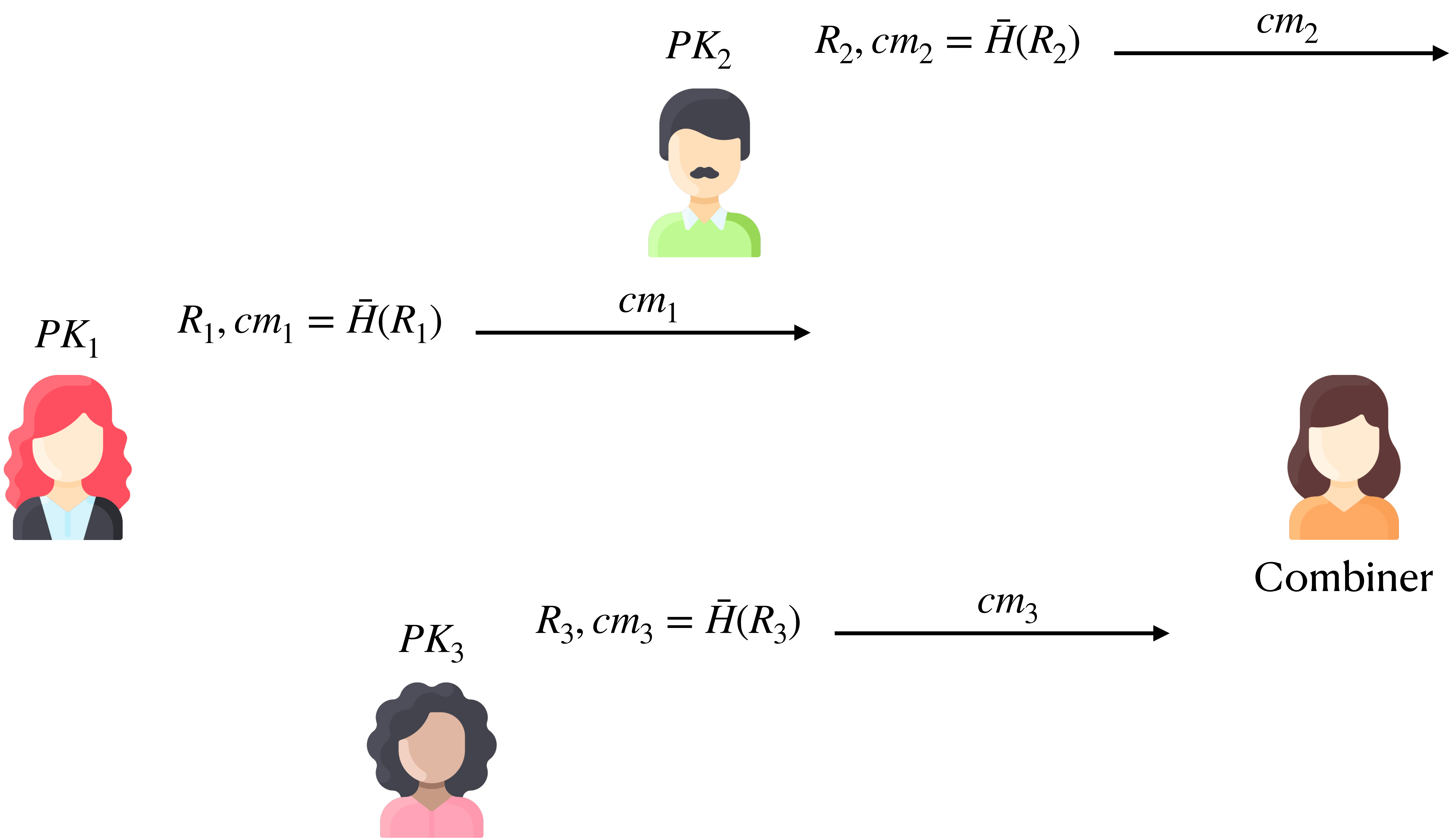
ROS Attack

- ROS problem originally stated in [Schnorr91]
- Drijvers et al. [DEFKLNS19] show how to break unforgeability
- confirmed polynomial-time attack by Benhamouda et al. [BLOR20]
- concurrent attack:
    - adversary opens multiple signing sessions at once
    - sees honest nonces first and makes its nonce a function of them
    - forges signature

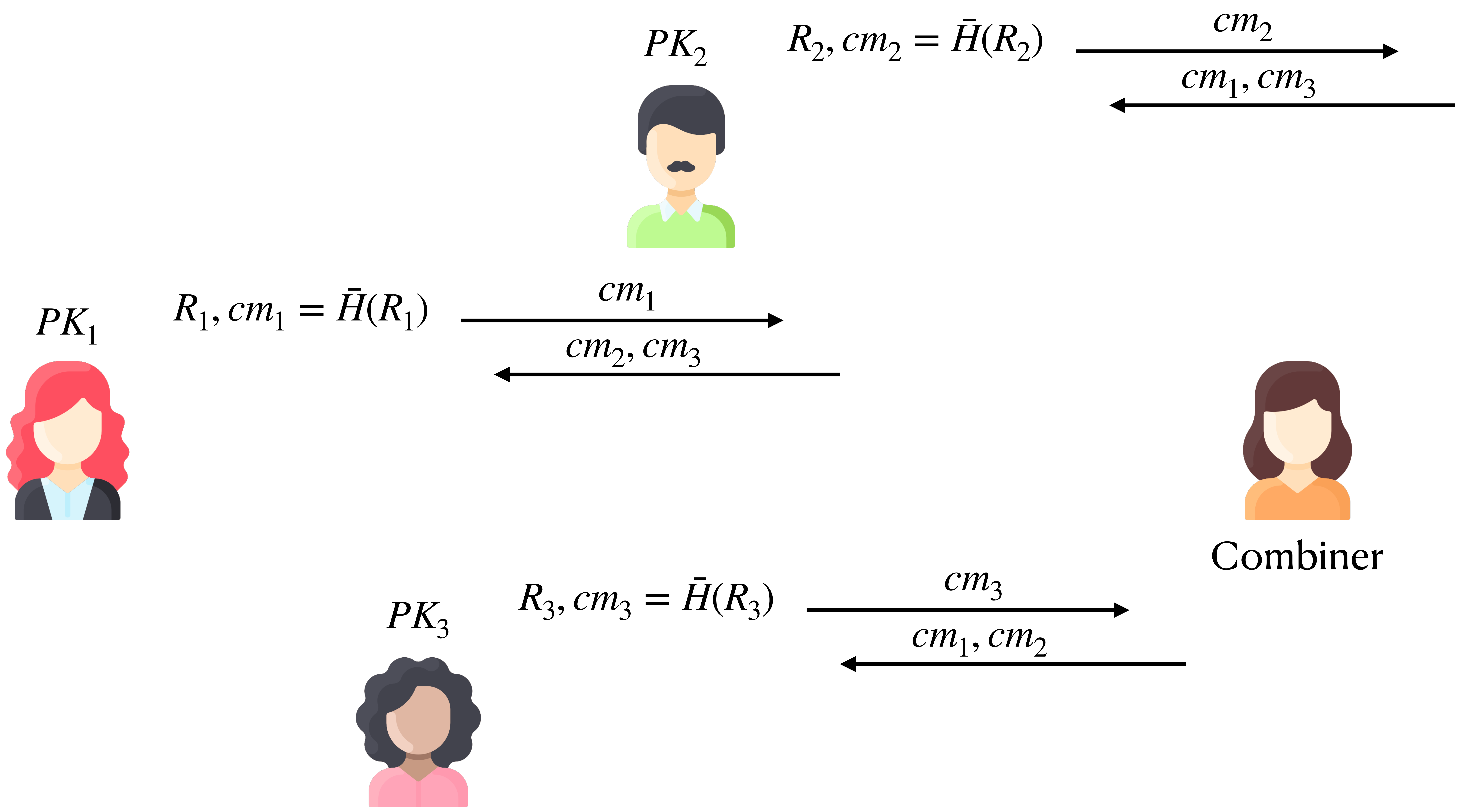
Fix #1: 3 Rounds



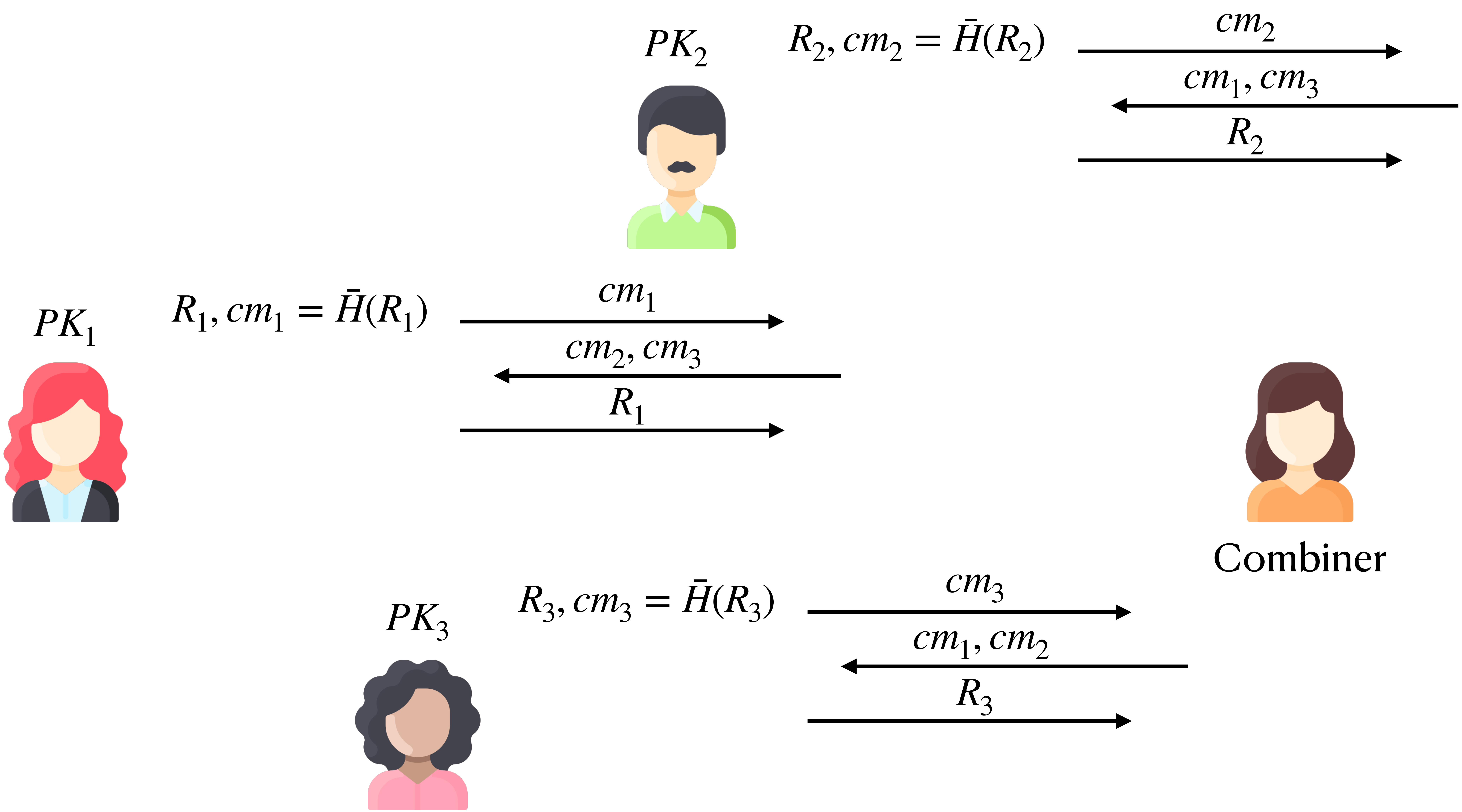
Fix #1: 3 Rounds



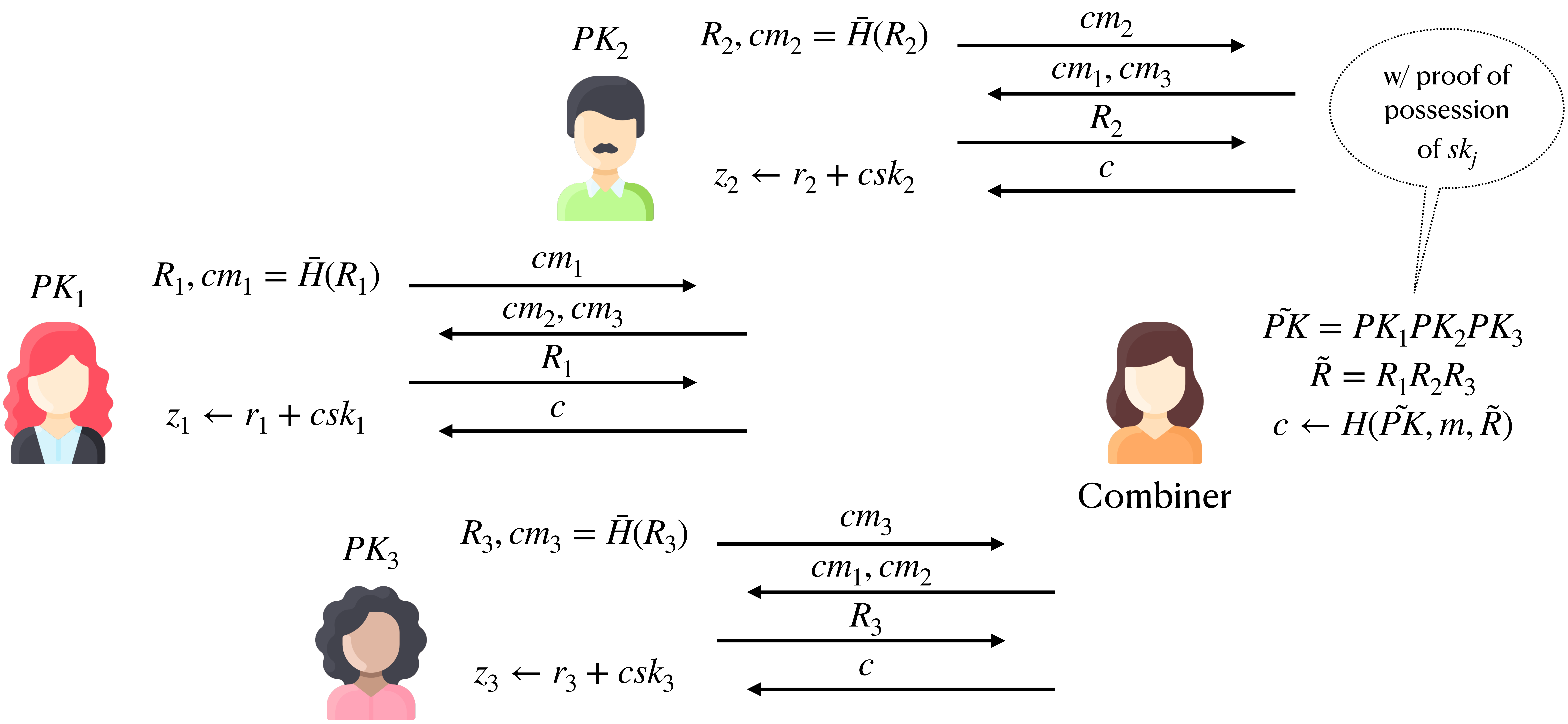
Fix #1: 3 Rounds



Fix #1: 3 Rounds



Fix #1: 3 Rounds



Fix #1: 3 Rounds

PK_2

$R_2, cm_2 = \bar{H}(R_2)$ $\xrightarrow{cm_2}$

$\xleftarrow{cm_1, cm_3}$

$\xrightarrow{R_2}$

$z_2 \leftarrow r_2 + csk_2$ $\xleftarrow{c}$

$\xrightarrow{z_2}$

PK_1

$R_1, cm_1 = \bar{H}(R_1)$ $\xrightarrow{cm_1}$

$\xleftarrow{cm_2, cm_3}$

$\xrightarrow{R_1}$

$z_1 \leftarrow r_1 + csk_1$ $\xleftarrow{c}$

$\xrightarrow{z_1}$

PK_3

$R_3, cm_3 = \bar{H}(R_3)$ $\xrightarrow{cm_3}$

$\xleftarrow{cm_1, cm_2}$

$\xrightarrow{R_3}$

$z_3 \leftarrow r_3 + csk_3$ $\xleftarrow{c}$

$\xrightarrow{z_3}$

Combiner

w/ proof of possession of sk_j

$\tilde{PK} = PK_1 PK_2 PK_3$

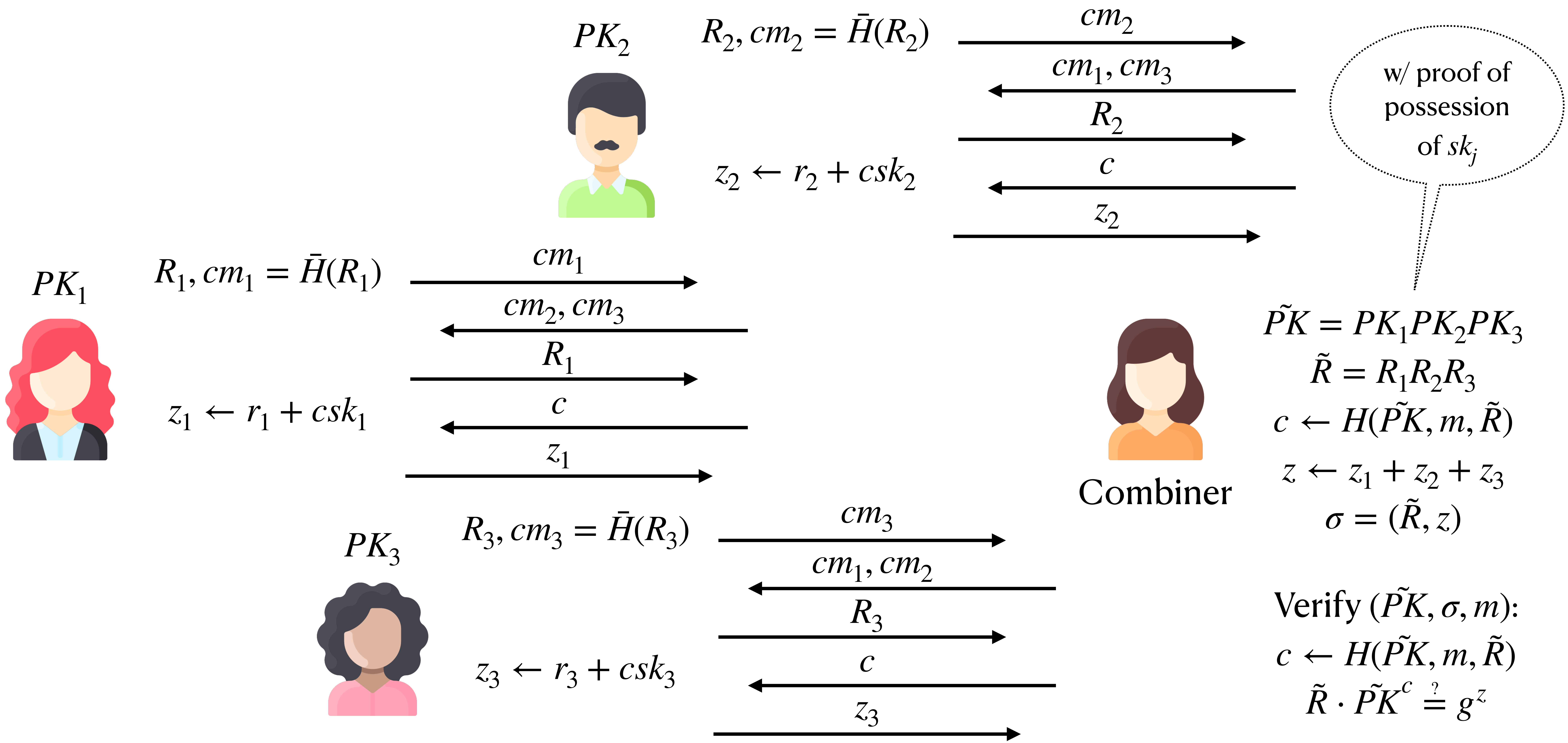
$\tilde{R} = R_1 R_2 R_3$

$c \leftarrow H(\tilde{PK}, m, \tilde{R})$

$z \leftarrow z_1 + z_2 + z_3$

$\sigma = (\tilde{R}, z)$

Fix #1: 3 Rounds

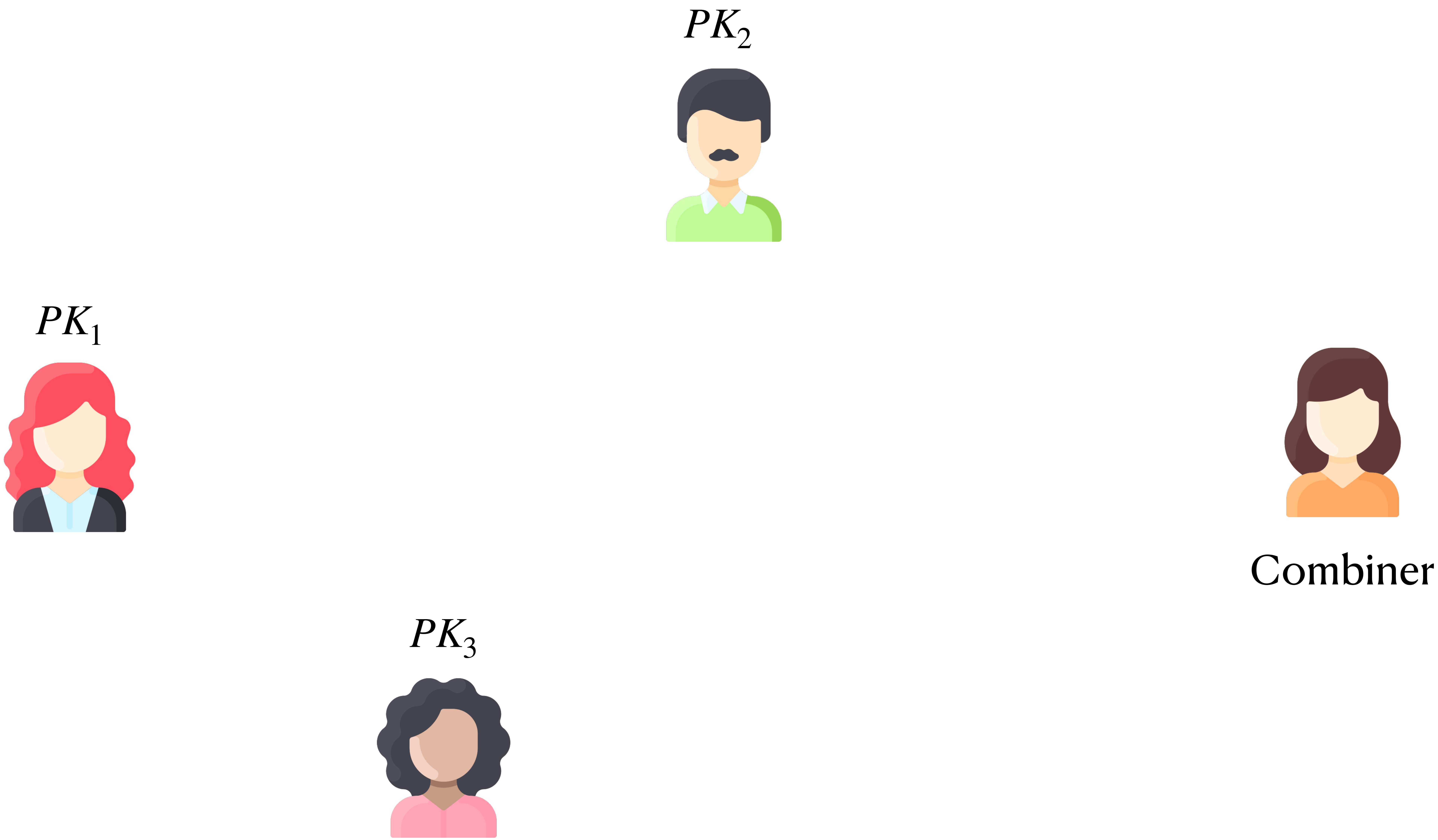


"How to Prove Schnorr Assuming Schnorr"

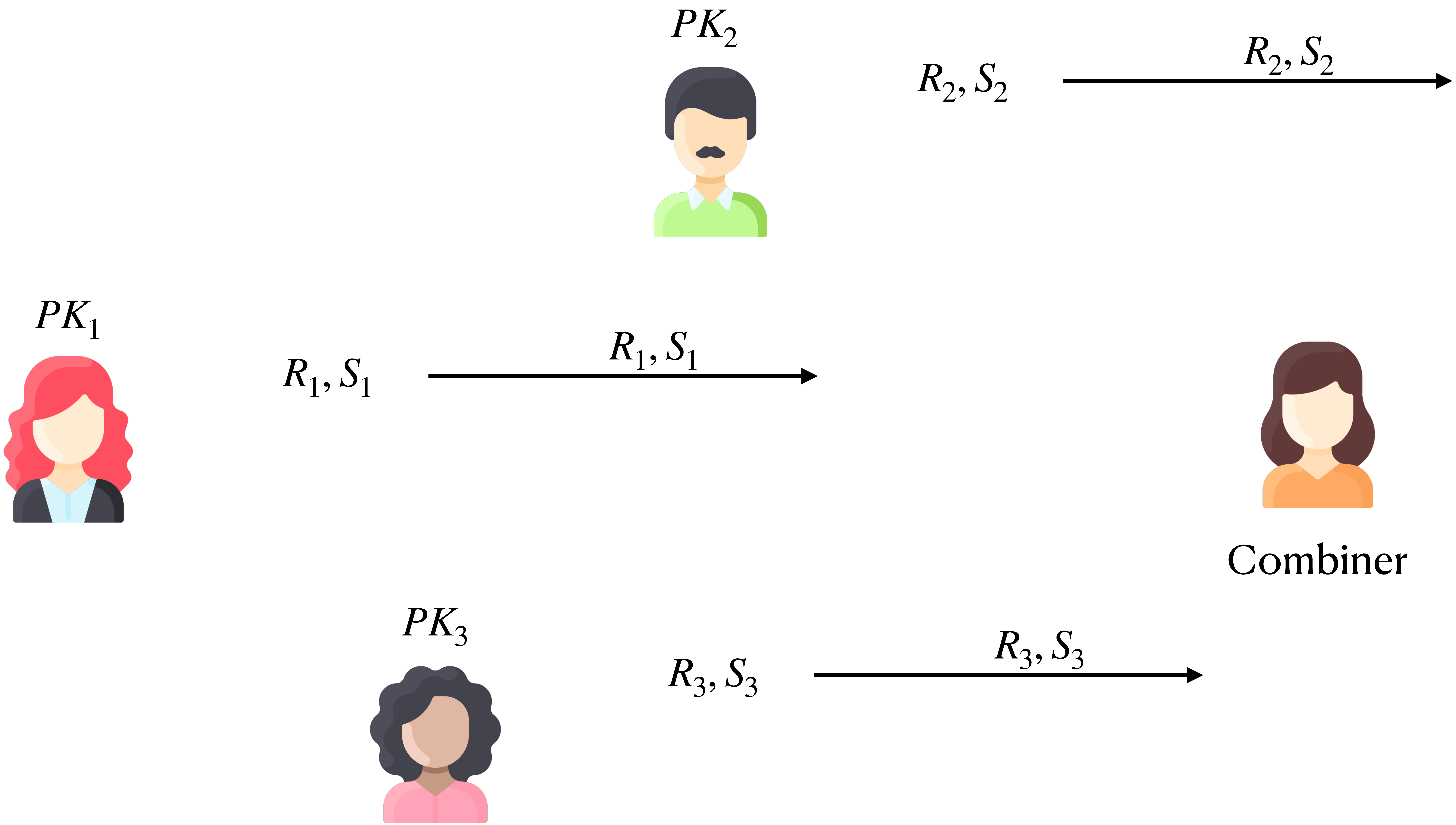
Elizabeth Crites, Chelsea Komlo, Mary Maller

- Our Contributions:
  - 3-round (n,n) multisignature SimpleMuSig (with PoP of keys)
  - 3-round (t,n) threshold signature SimpleTSig (with PedPoP)

Fix #2: 2 Nonces (“FROST”)



Fix #2: 2 Nonces (“FROST”)



Fix #2: 2 Nonces ("FROST")

PK_2

R_2, S_2 $\xrightarrow{R_2, S_2}$

$z_2 \leftarrow r_2 + a_2 s_2 + c sk_2$ $\xleftarrow{c}$

PK_1

R_1, S_1 $\xrightarrow{R_1, S_1}$

$z_1 \leftarrow r_1 + a_1 s_1 + c sk_1$ $\xleftarrow{c}$

PK_3

R_3, S_3 $\xrightarrow{R_3, S_3}$

$z_3 \leftarrow r_3 + a_3 s_3 + c sk_3$ $\xleftarrow{c}$

w/ proof of possession of sk_j

$$\tilde{PK} = PK_1 PK_2 PK_3$$
$$a_j = \tilde{H}(j, \tilde{PK}, m, \{R_j, S_j\}_1^3)$$
$$\tilde{R} = \Pi_1^3 R_j S_j^{a_j}$$
$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$

Combiner

Fix #2: 2 Nonces (“FROST”)

w/ proof of possession of sk_j

PK_2

R_2, S_2 $\xrightarrow{R_2, S_2}$

$z_2 \leftarrow r_2 + a_2 s_2 + c sk_2$ $\xleftarrow{c}$

$\xrightarrow{z_2}$

PK_1

R_1, S_1 $\xrightarrow{R_1, S_1}$

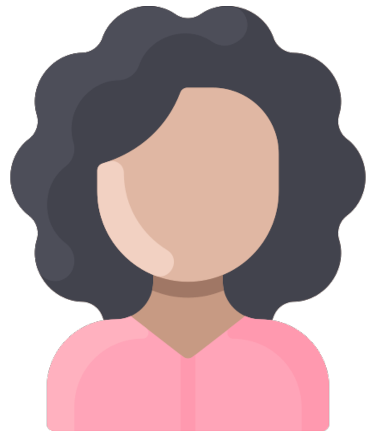
$z_1 \leftarrow r_1 + a_1 s_1 + c sk_1$ $\xleftarrow{c}$

$\xrightarrow{z_1}$

Combiner

$$\tilde{PK} = PK_1 PK_2 PK_3$$
$$a_j = \tilde{H}(j, \tilde{PK}, m, \{R_j, S_j\}_1^3)$$
$$\tilde{R} = \Pi_1^3 R_j S_j^{a_j}$$
$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$
$$z \leftarrow z_1 + z_2 + z_3$$
$$\sigma = (\tilde{R}, z)$$

PK_3



R_3, S_3 $\xrightarrow{R_3, S_3}$

$z_3 \leftarrow r_3 + a_3 s_3 + c sk_3$ $\xleftarrow{c}$

$\xrightarrow{z_3}$

Fix #2: 2 Nonces ("FROST")

PK_2

R_2, S_2 $\xrightarrow{R_2, S_2}$

$z_2 \leftarrow r_2 + a_2 s_2 + c sk_2$ $\xleftarrow{c}$

$\xrightarrow{z_2}$

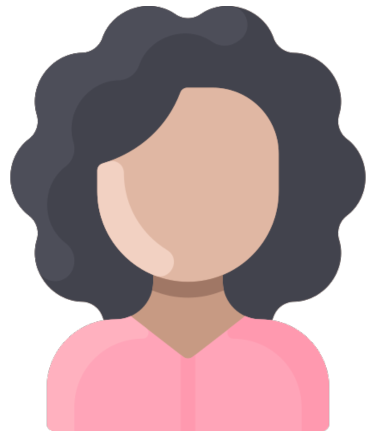
PK_1

R_1, S_1 $\xrightarrow{R_1, S_1}$

$z_1 \leftarrow r_1 + a_1 s_1 + c sk_1$ $\xleftarrow{c}$

$\xrightarrow{z_1}$

PK_3



R_3, S_3 $\xrightarrow{R_3, S_3}$

$z_3 \leftarrow r_3 + a_3 s_3 + c sk_3$ $\xleftarrow{c}$

$\xrightarrow{z_3}$

Combiner

w/ proof of possession of sk_j

$$\tilde{PK} = PK_1 PK_2 PK_3$$
$$a_j = \tilde{H}(j, \tilde{PK}, m, \{R_j, S_j\}_1^3)$$
$$\tilde{R} = \Pi_1^3 R_j S_j^{a_j}$$
$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$
$$z \leftarrow z_1 + z_2 + z_3$$
$$\sigma = (\tilde{R}, z)$$

Verify $(\tilde{PK}, \sigma, m)$:

$$c \leftarrow H(\tilde{PK}, m, \tilde{R})$$
$$\tilde{R} \cdot \tilde{PK}^c \stackrel{?}{=} g^z$$

Fix #3: 2 Nonces ("FROST2")

PK_2

R_2, S_2

$z_2 \leftarrow r_2 + as_2 + csk_2$

$\xrightarrow{R_2, S_2}$

$\xleftarrow{c}$

$\xrightarrow{z_2}$

w/ proof of possession of sk_j

PK_1

R_1, S_1

$z_1 \leftarrow r_1 + as_1 + csk_1$

$\xrightarrow{R_1, S_1}$

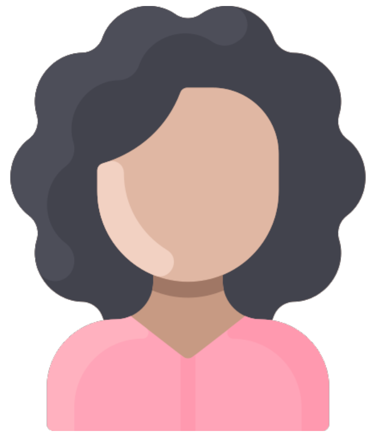
$\xleftarrow{c}$

$\xrightarrow{z_1}$

Combiner

$$\widetilde{PK} = PK_1 PK_2 PK_3$$
$$a = \tilde{H}(\widetilde{PK}, m, \{R_j, S_j\}_1^3)$$
$$\tilde{R} = \Pi_1^3 R_j S_j^a$$
$$c \leftarrow H(\widetilde{PK}, m, \tilde{R})$$
$$z \leftarrow z_1 + z_2 + z_3$$
$$\sigma = (\tilde{R}, z)$$

PK_3



R_3, S_3

$z_3 \leftarrow r_3 + as_3 + csk_3$

$\xrightarrow{R_3, S_3}$

$\xleftarrow{c}$

$\xrightarrow{z_3}$

Verify $(\widetilde{PK}, \sigma, m)$:

$$c \leftarrow H(\widetilde{PK}, m, \tilde{R})$$
$$\tilde{R} \cdot \widetilde{PK}^c \stackrel{?}{=} g^z$$

“How to Prove Schnorr Assuming Schnorr”

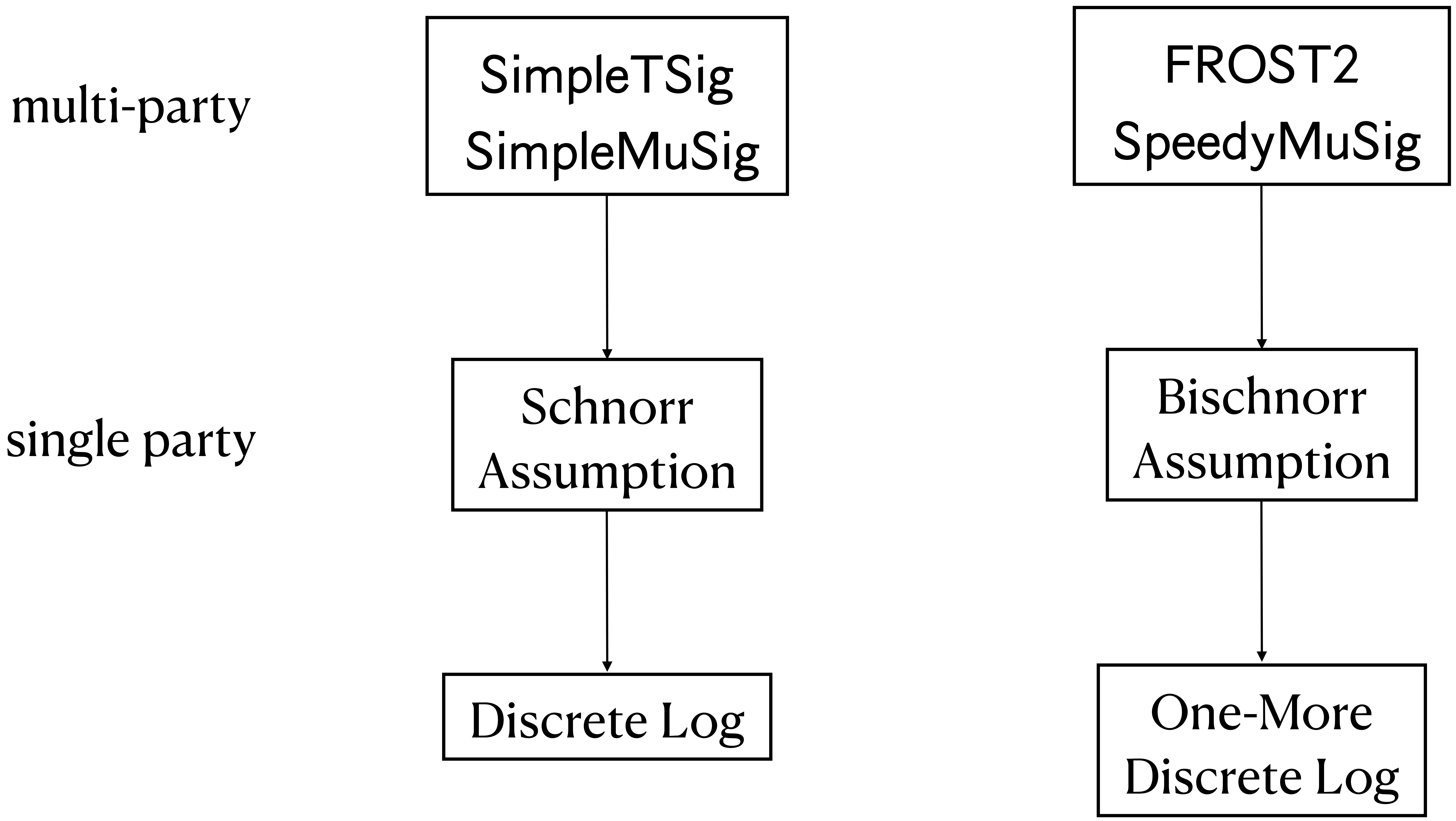
Elizabeth Crites, Chelsea Komlo, Mary Maller

- Our Contributions:
  - 3-round (n,n) multisignature SimpleMuSig (with PoP of keys)
  - 3-round (t,n) threshold signature SimpleTSig (with PedPoP)
  - optimized 2-round (n,n) multisignature SpeedyMuSig (with PoP of keys)
  - optimized 2-round (t,n) threshold signature FROST2 (with PedPoP)
    - reduces exponentiations from at least t to one
  - new proving framework

Proving the Security of Multi-Party Schnorr

- Security reductions for multi-party signatures have two moving parts:
  1. Simulating honest users interacting with the adversary
  2. Extracting a solution to some hard problem from the adversary's responses
- Idea: Separate the two parts for a more modular reduction

Proving the Security of Multi-Party Schnorr



“Stronger Security for Non-Interactive Threshold Signatures”

Mihir Bellare, Stefano Tessaro, Chenzhi Zhu

- Merged with our paper for CRYPTO 2022

eprint.iacr.org/2022/833

“Stronger Security for Non-Interactive Threshold Signatures”

Mihir Bellare, Stefano Tessaro, Chenzhi Zhu

- Merged with our paper for CRYPTO 2022
- Hierarchy of notions of unforgeability for threshold signatures

eprint.iacr.org/2022/833

“Stronger Security for Non-Interactive Threshold Signatures”

Mihir Bellare, Stefano Tessaro, Chenzhi Zhu

- Merged with our paper for CRYPTO 2022
- Hierarchy of notions of unforgeability for threshold signatures
- Independent proofs for unforgeability of FROST1 and FROST2 in ROM/OMDL

eprint.iacr.org/2022/833

“Stronger Security for Non-Interactive Threshold Signatures”

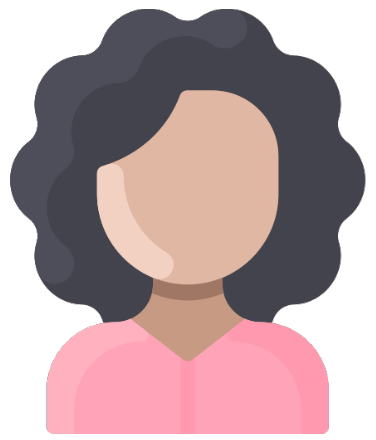
Mihir Bellare, Stefano Tessaro, Chenzhi Zhu

- Merged with our paper for CRYPTO 2022
- Hierarchy of notions of unforgeability for threshold signatures
- Independent proofs for unforgeability of FROST1 and FROST2 in ROM/OMDL
- Finds FROST2 to be malleable with respect to the signing set

eprint.iacr.org/2022/833

- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ

PK_3, PK_4



Corrupt

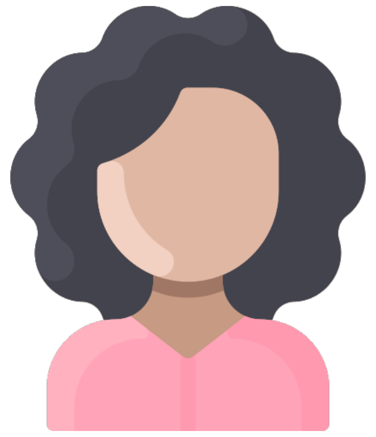
Honest

PK_2

PK_1

- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ

PK_3, PK_4



Corrupt

PK_2

Honest

$R_2 = g^{r_2}, S_2 = g^{s_2}$ $\xrightarrow{R_2, S_2}$

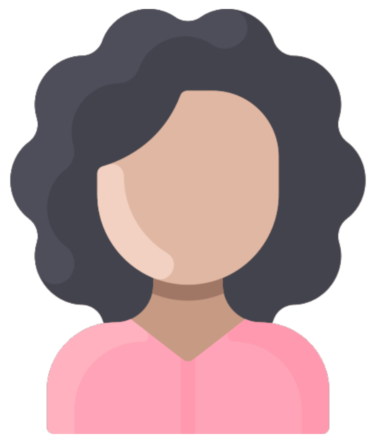
Begin signing protocol with signers (1, 2, 3)

PK_1

$R_1 = g^{r_1}, S_1 = g^{s_1}$ $\xrightarrow{R_1, S_1}$

- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ

PK_3, PK_4



Corrupt

Honest

PK_2

$R_2 = g^{r_2}, S_2 = g^{s_2}$ $\xrightarrow{R_2, S_2}$

PK_1

$R_1 = g^{r_1}, S_1 = g^{s_1}$ $\xrightarrow{R_1, S_1}$

$$R_3 = (R_1)^{\delta-1} \cdot (R_2)^{-1}$$

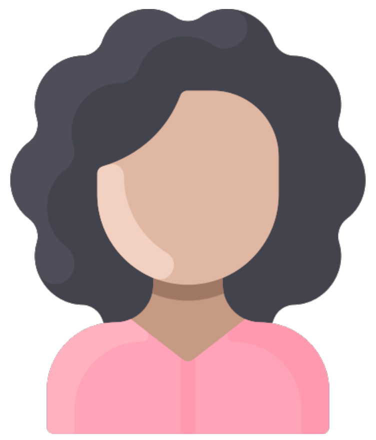
$$S_3 = (S_1)^{\delta-1} \cdot (S_2)^{-1}$$

$\xleftarrow{R_3, S_3}$

$$R = \prod_{i=1}^{3} R_i \cdot (S_i)^a$$

- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ

PK_3, PK_4



Corrupt

PK_2

Honest

$R_2 = g^{r_2}, S_2 = g^{s_2}$ $\xrightarrow{R_2, S_2}$

PK_1

$R_1 = g^{r_1}, S_1 = g^{s_1}$ $\xrightarrow{R_1, S_1}$

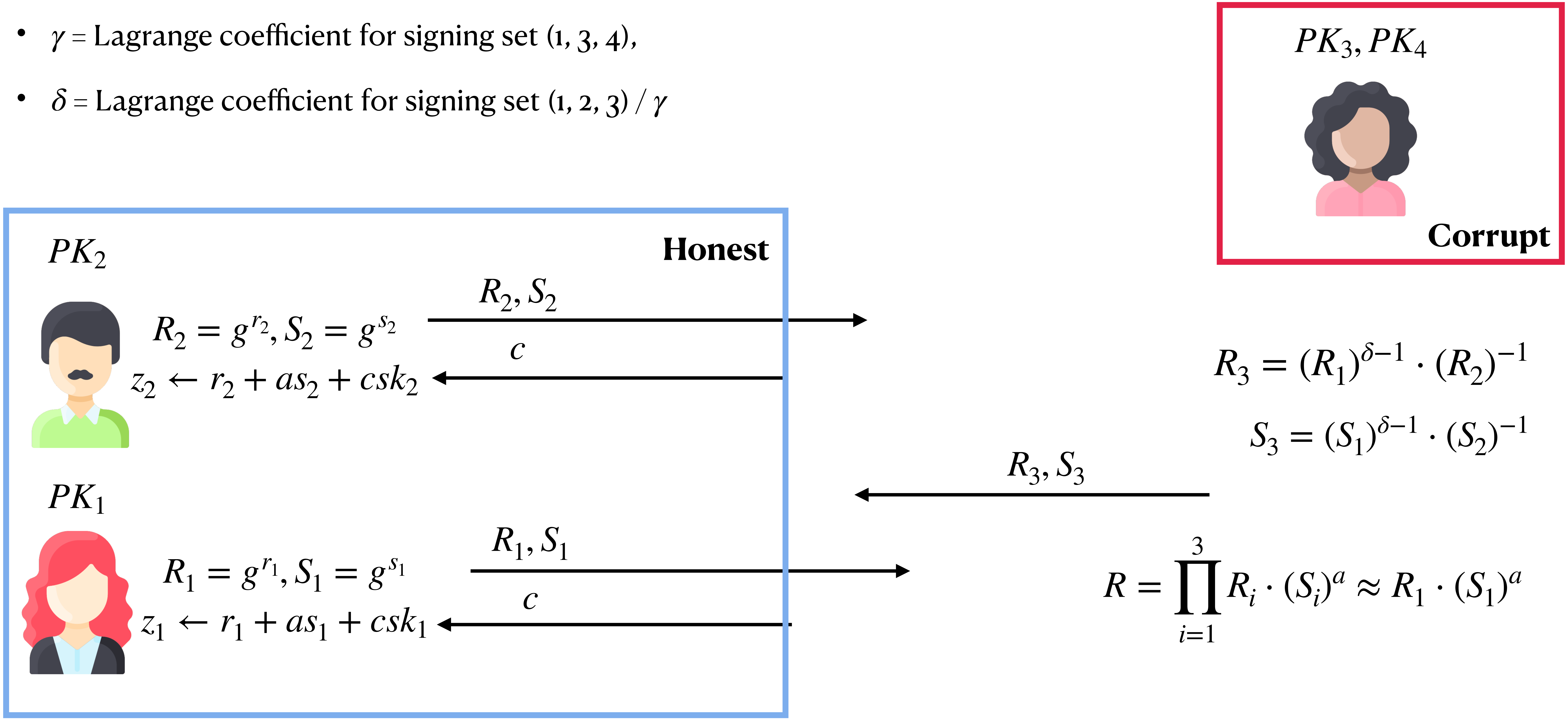
$$R_3 = (R_1)^{\delta-1} \cdot (R_2)^{-1}$$

$$S_3 = (S_1)^{\delta-1} \cdot (S_2)^{-1}$$

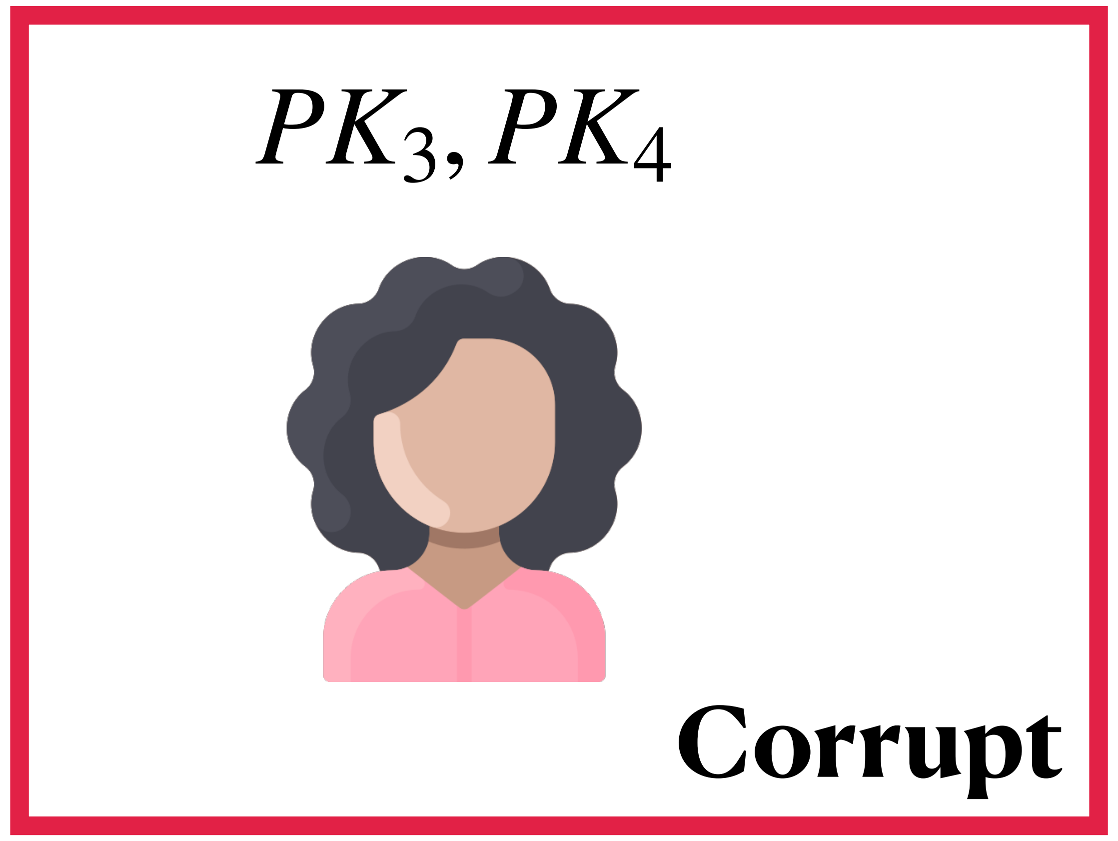
$\xleftarrow{R_3, S_3}$

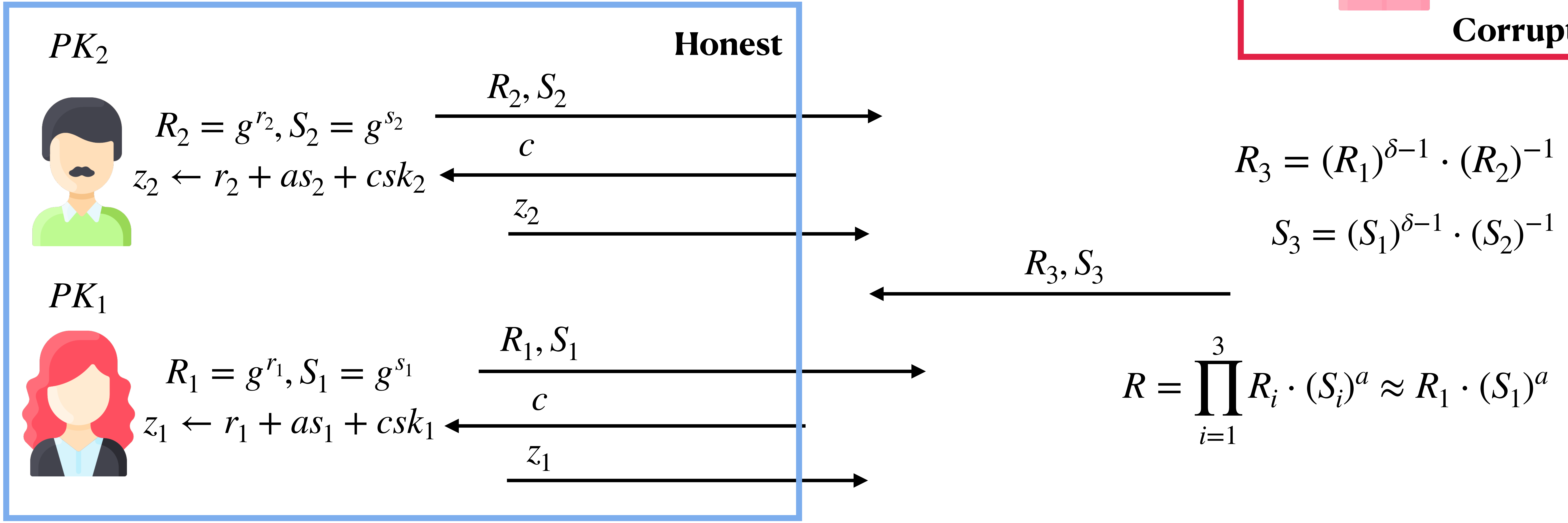
$$R = \prod_{i=1}^{3} R_i \cdot (S_i)^a \approx \boxed{R_1 \cdot (S_1)^a}$$

- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ

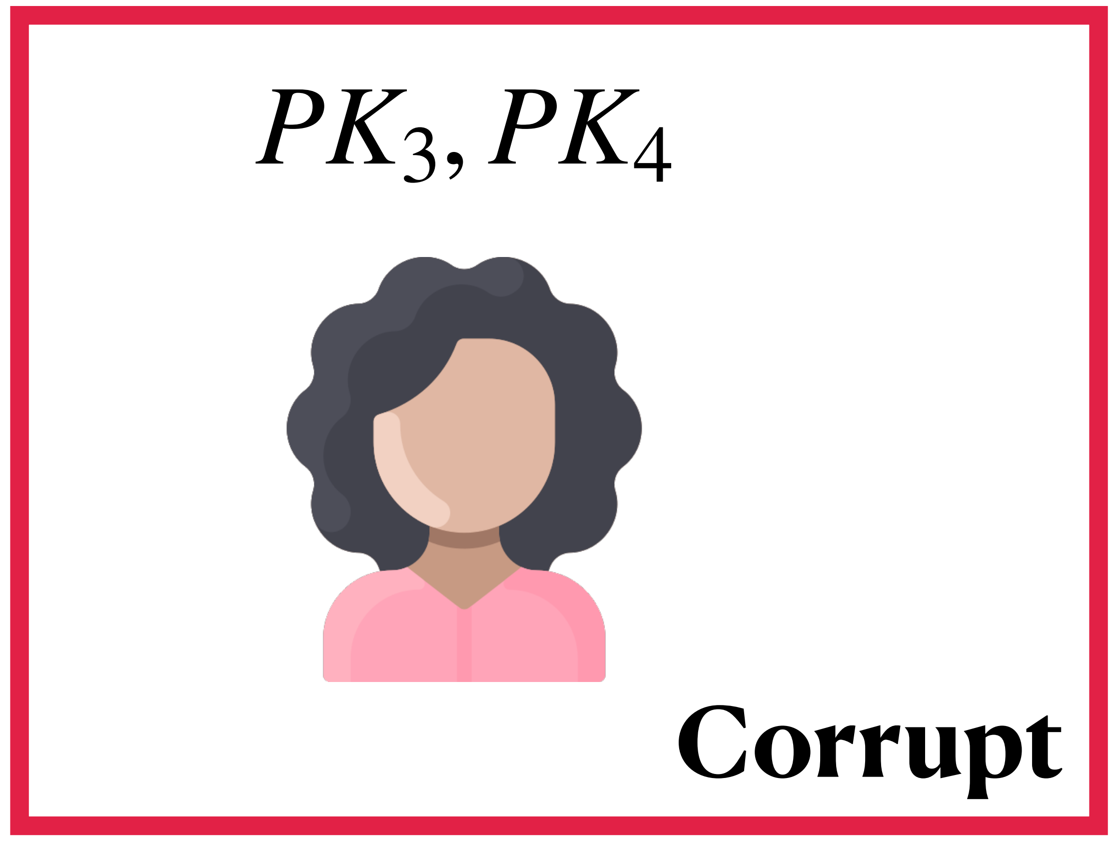


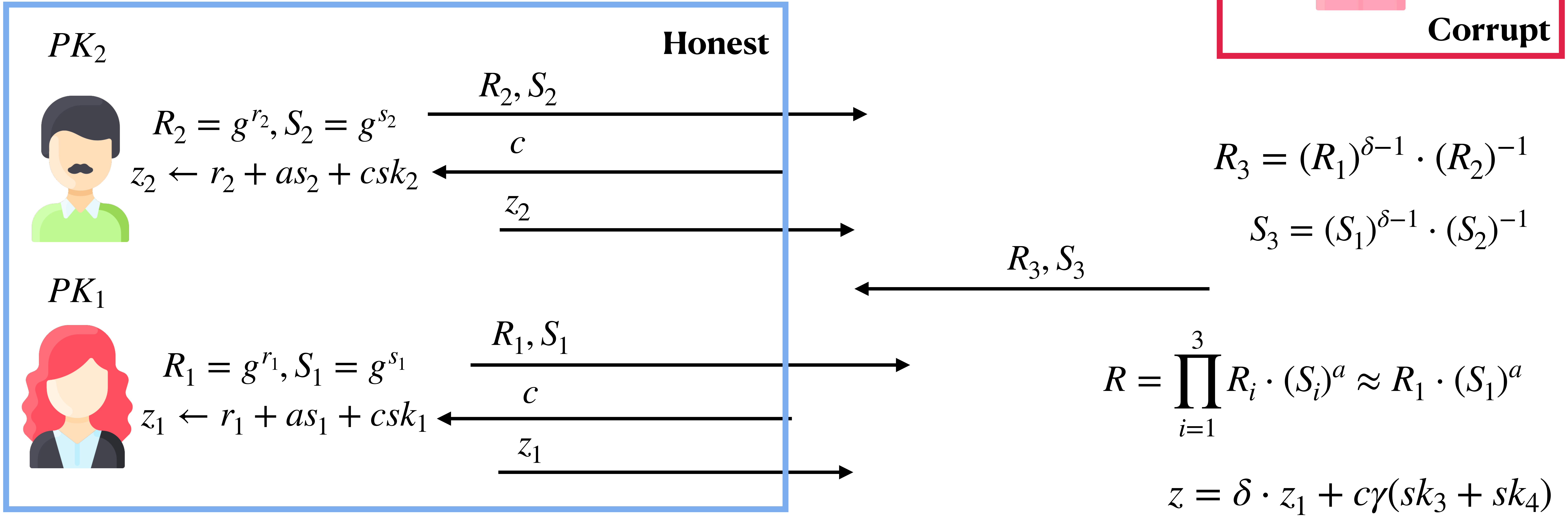
- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ



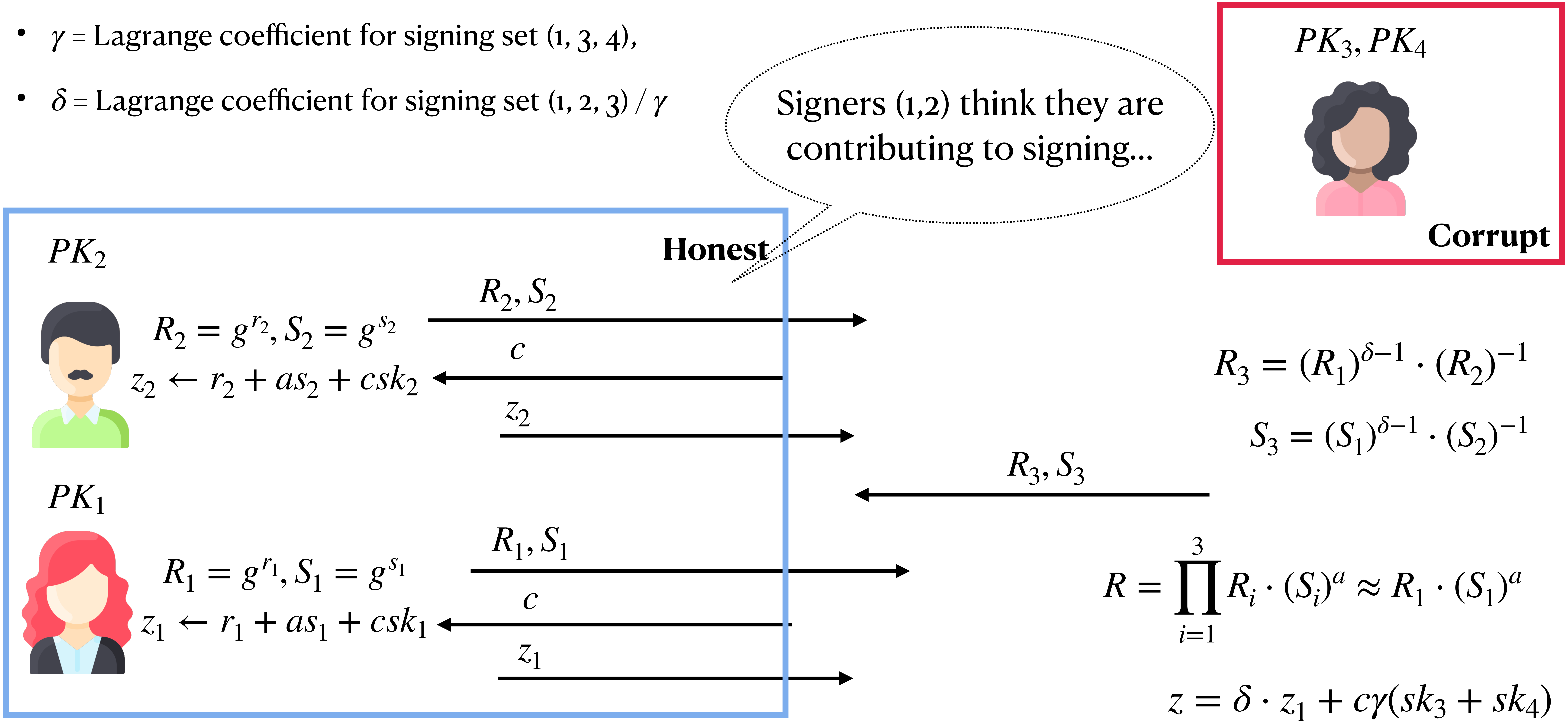


- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ

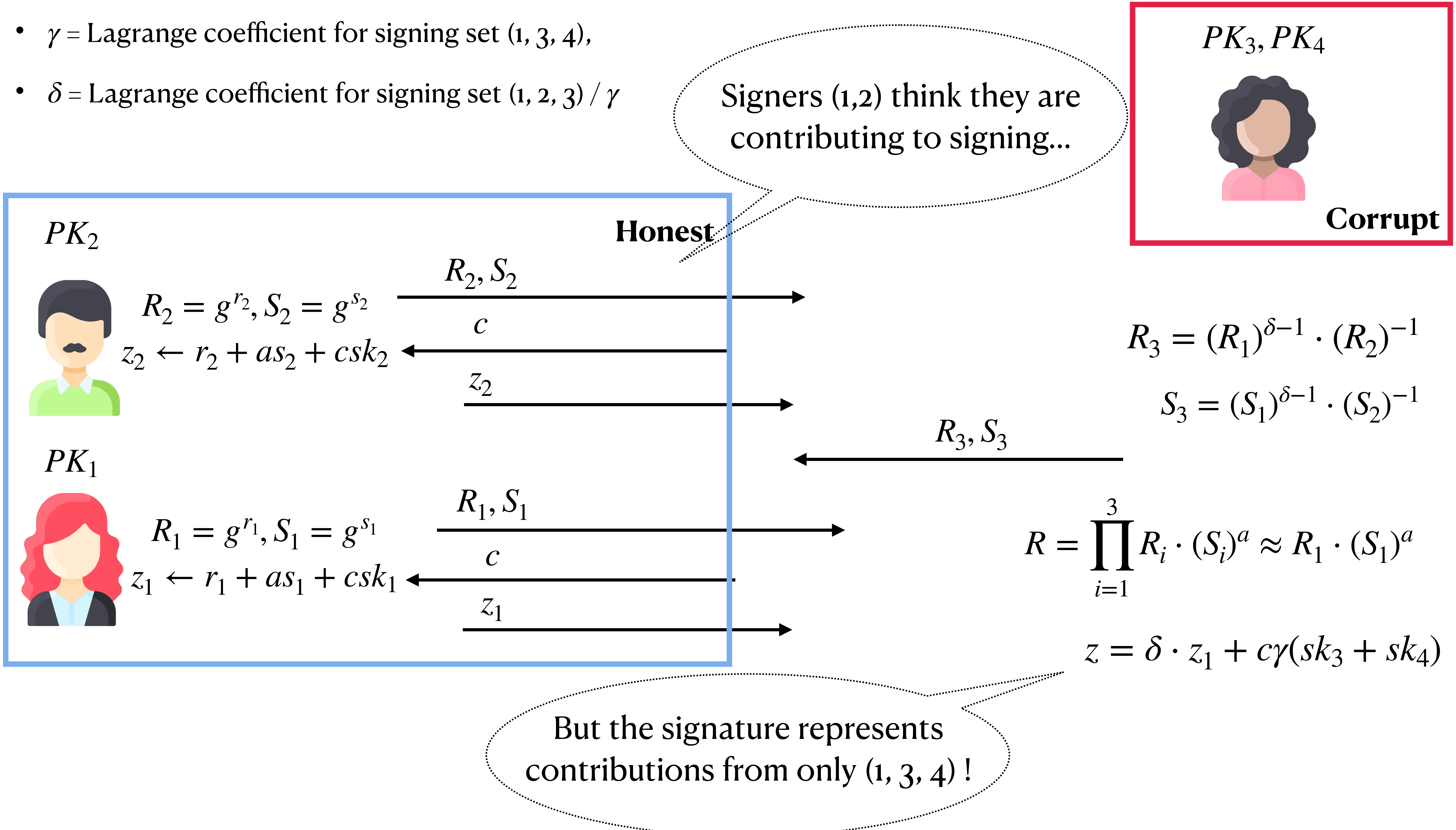




- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ



- γ = Lagrange coefficient for signing set (1, 3, 4),
- δ = Lagrange coefficient for signing set (1, 2, 3) / γ



Signing Set Malleability

- This is a new notion for threshold signatures - is it actually needed in practice?

Signing Set Malleability

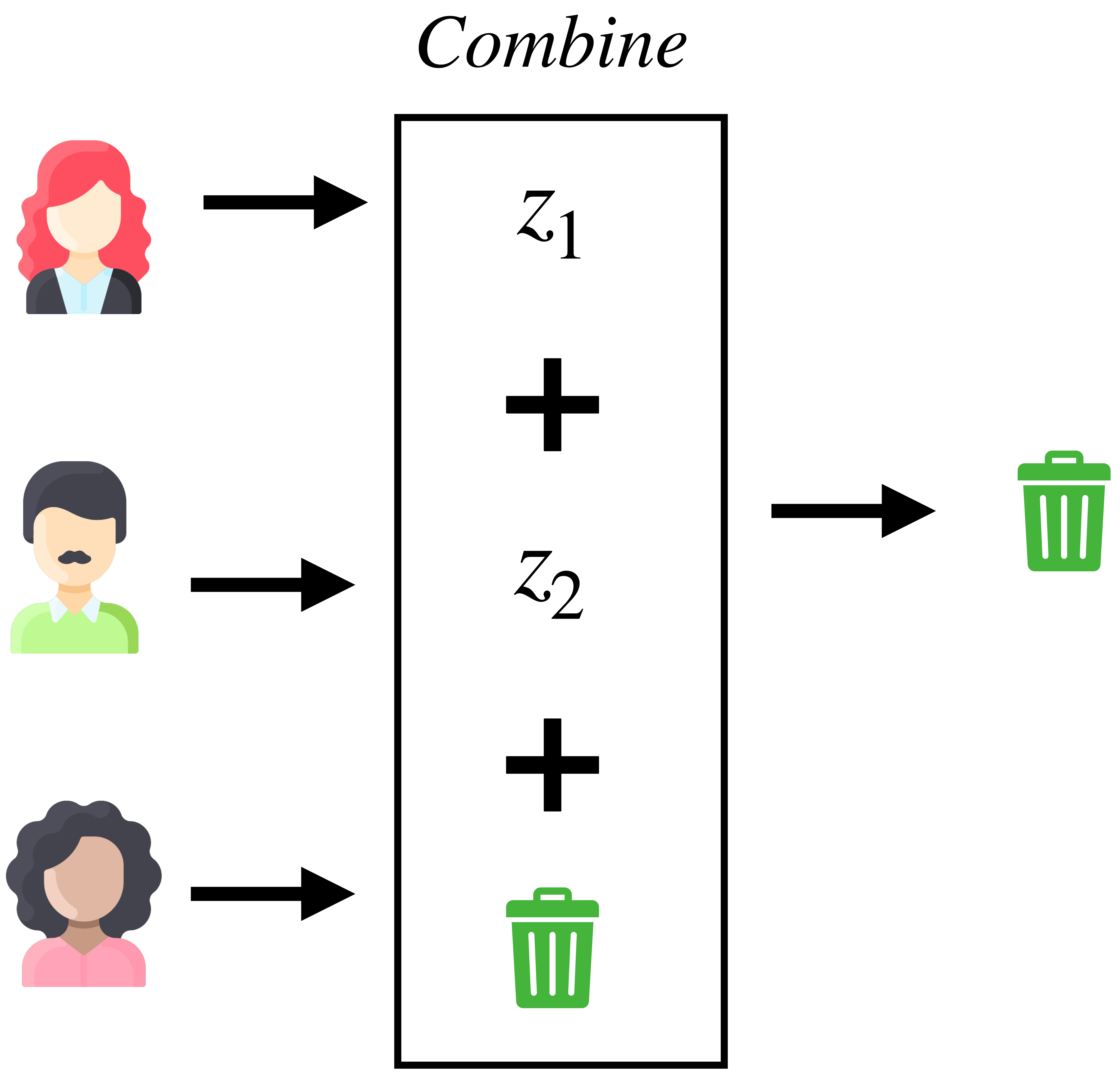
- This is a new notion for threshold signatures - is it actually needed in practice?
- IETF draft is now back to FROST1.

Signing Set Malleability

- This is a new notion for threshold signatures - is it actually needed in practice?
- IETF draft is now back to FROST1.
- FROST2 may be of interest for performance critical settings.

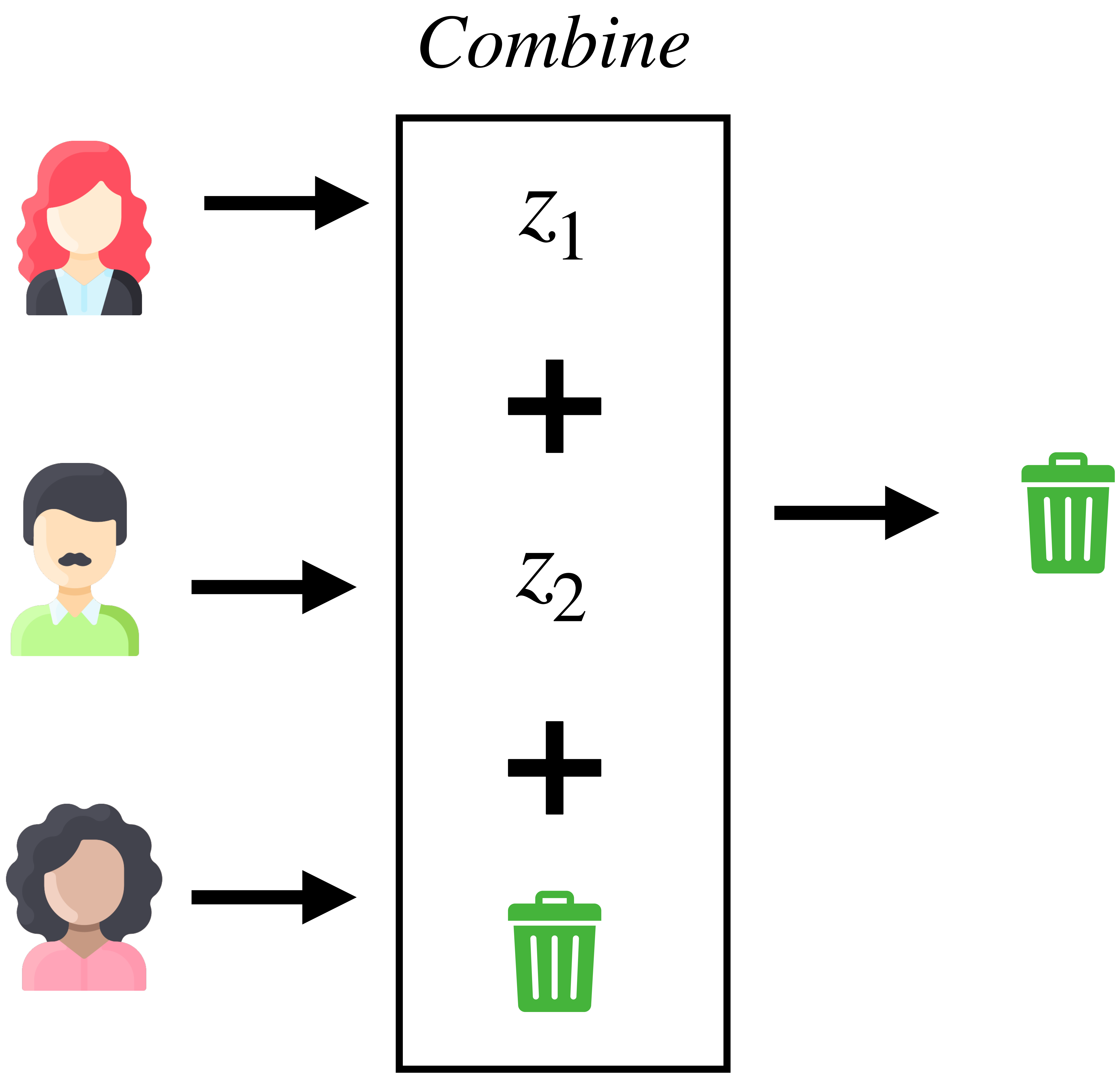
FROST & Robustness

- **Robustness**: the protocol succeeds so long as at least t players participate honestly.



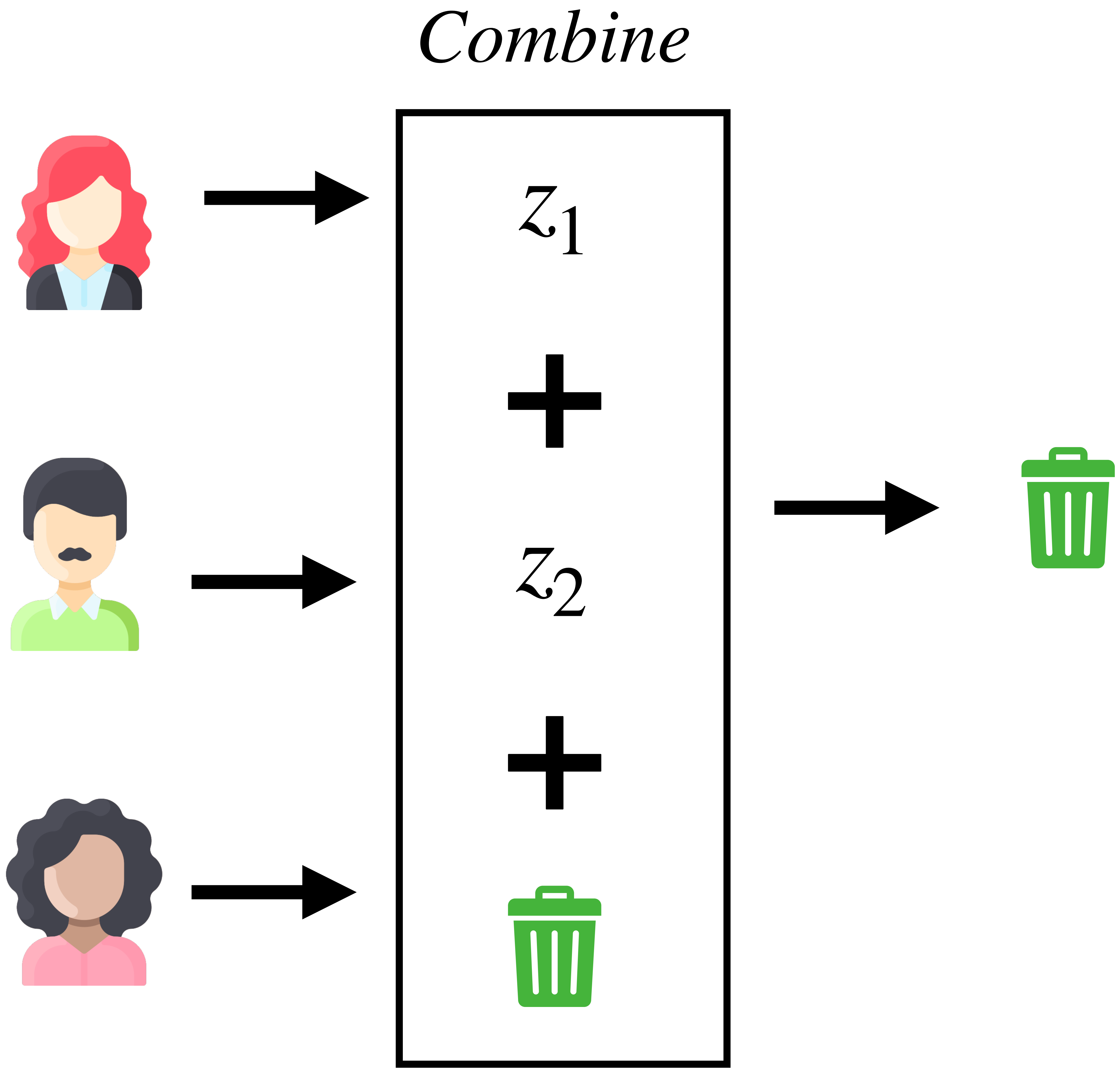
FROST & Robustness

- **Robustness**: the protocol succeeds so long as at least t players participate honestly.
- FROST is **not** robust.



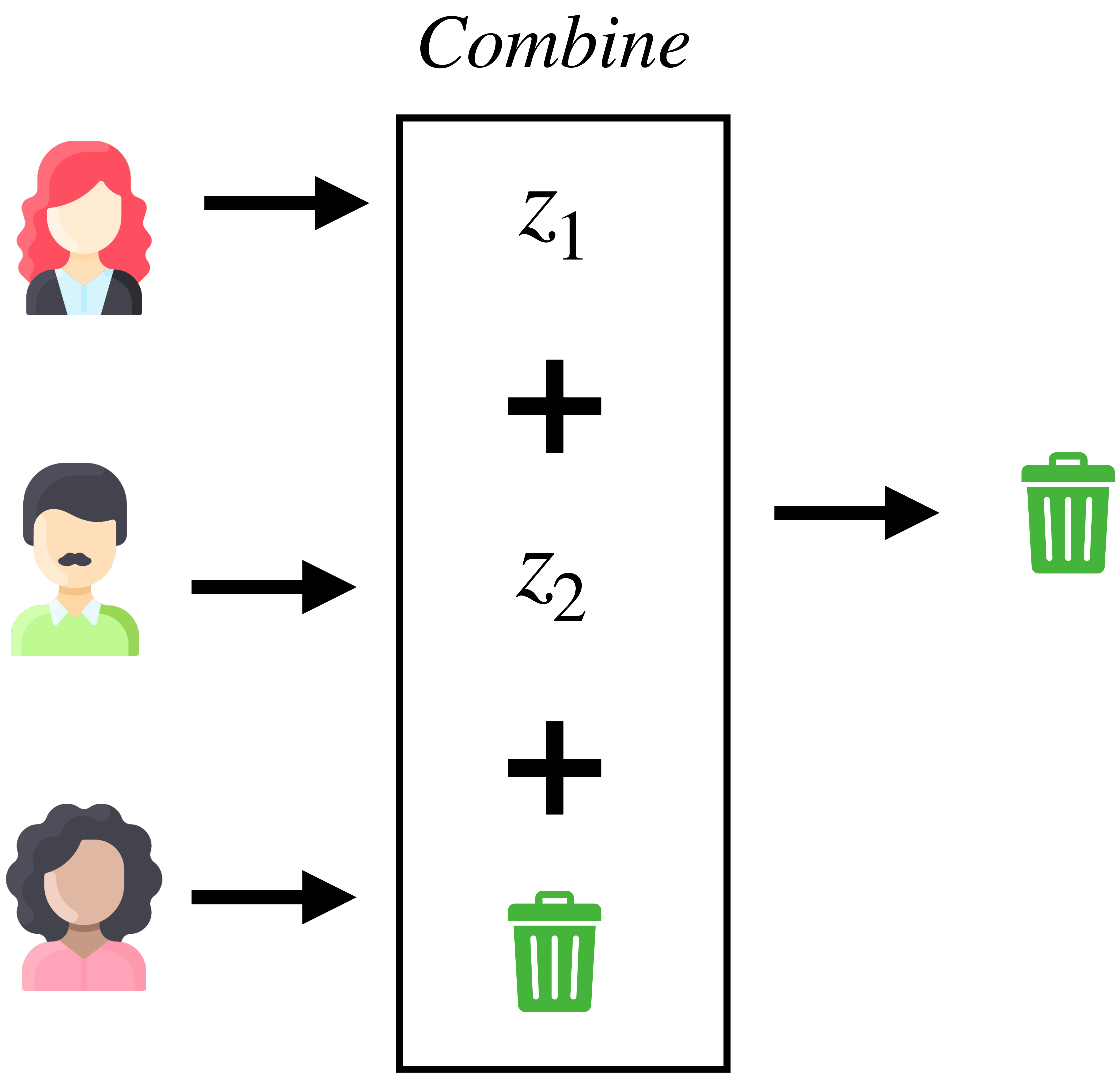
FROST & Robustness

- **Robustness**: the protocol succeeds so long as at least t players participate honestly.
- FROST is **not** robust.
- Example: n=3, t=2



FROST & Robustness

- **Robustness**: the protocol succeeds so long as at least t players participate honestly.
- FROST is **not** robust.
- Example: n=3, t=2
- If even one FROST signer issues garbage, the resulting signature is garbage and the protocol must be re-run (even if more than two signed).

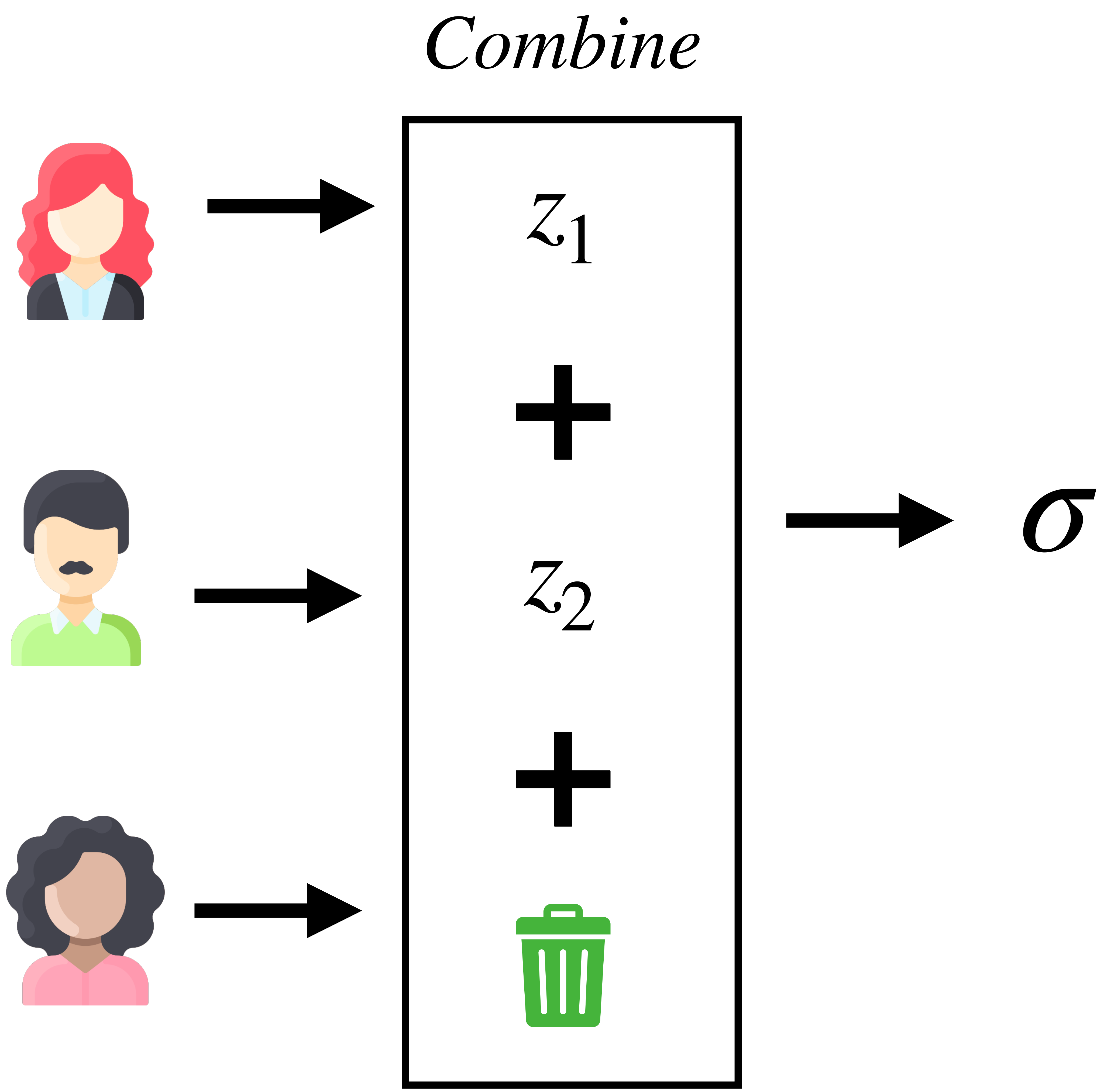


"ROAST: Robust Asynchronous Schnorr Threshold Signatures"

Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, Dominique Schröder

- ROAST defines a wrapper protocol to make FROST robust.

eprint.iacr.org/2022/550

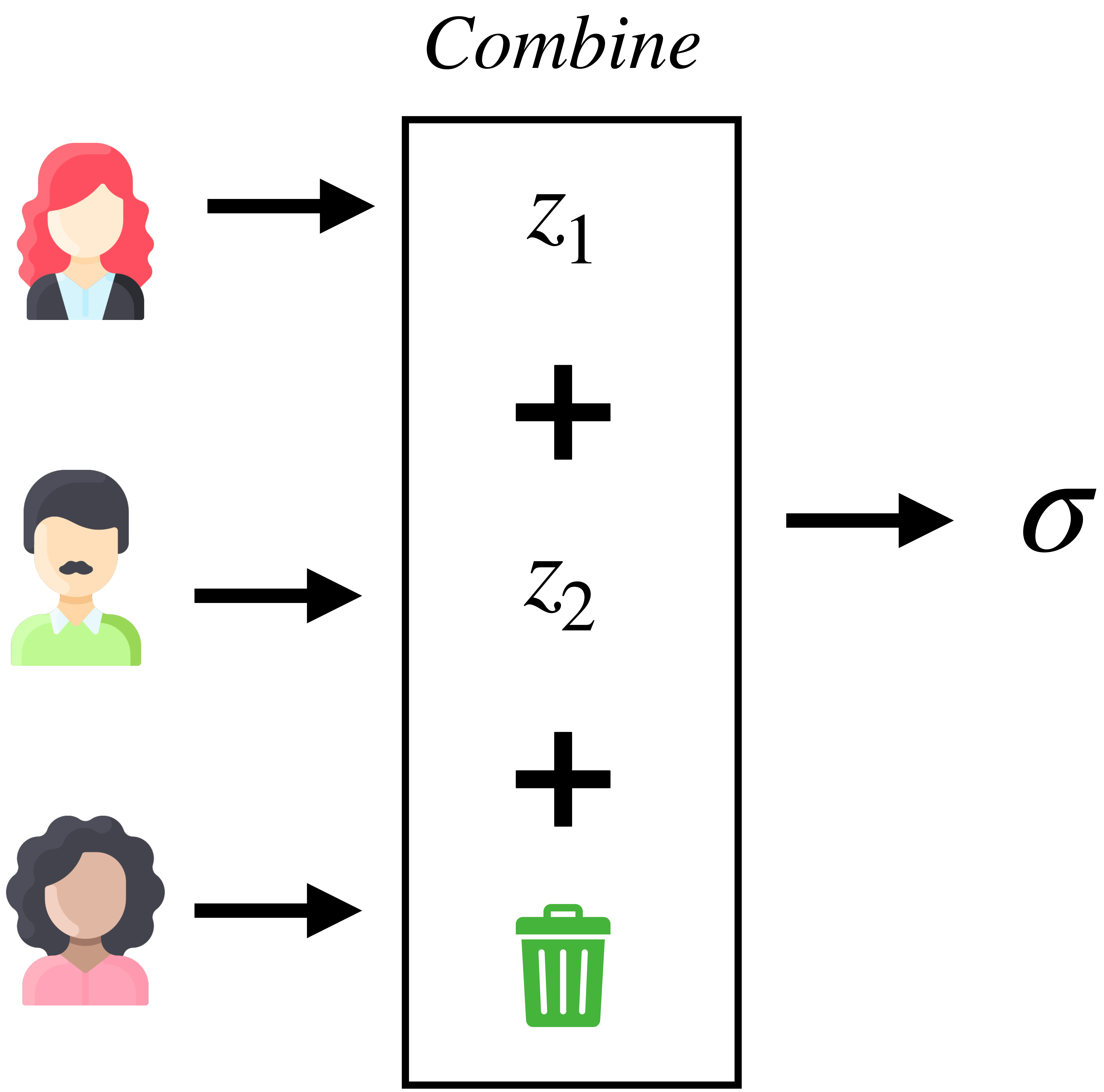


“ROAST: Robust Asynchronous Schnorr Threshold Signatures”

Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, Dominique Schröder

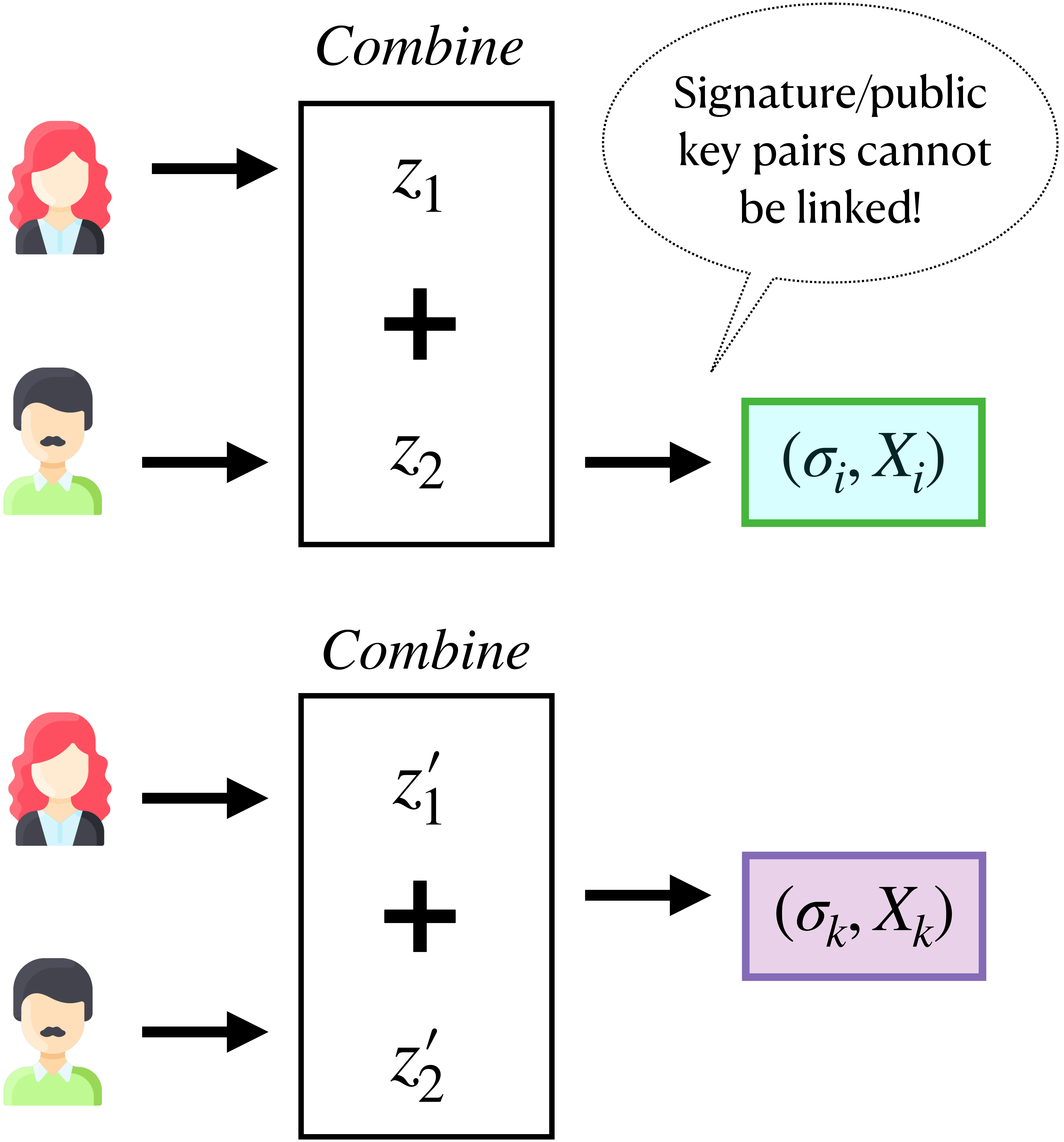
- ROAST defines a wrapper protocol to make FROST robust.
- Improves on the trivial solution of maintaining $\binom{n}{t}$ concurrent sessions to $n - t + 1$

eprint.iacr.org/2022/550



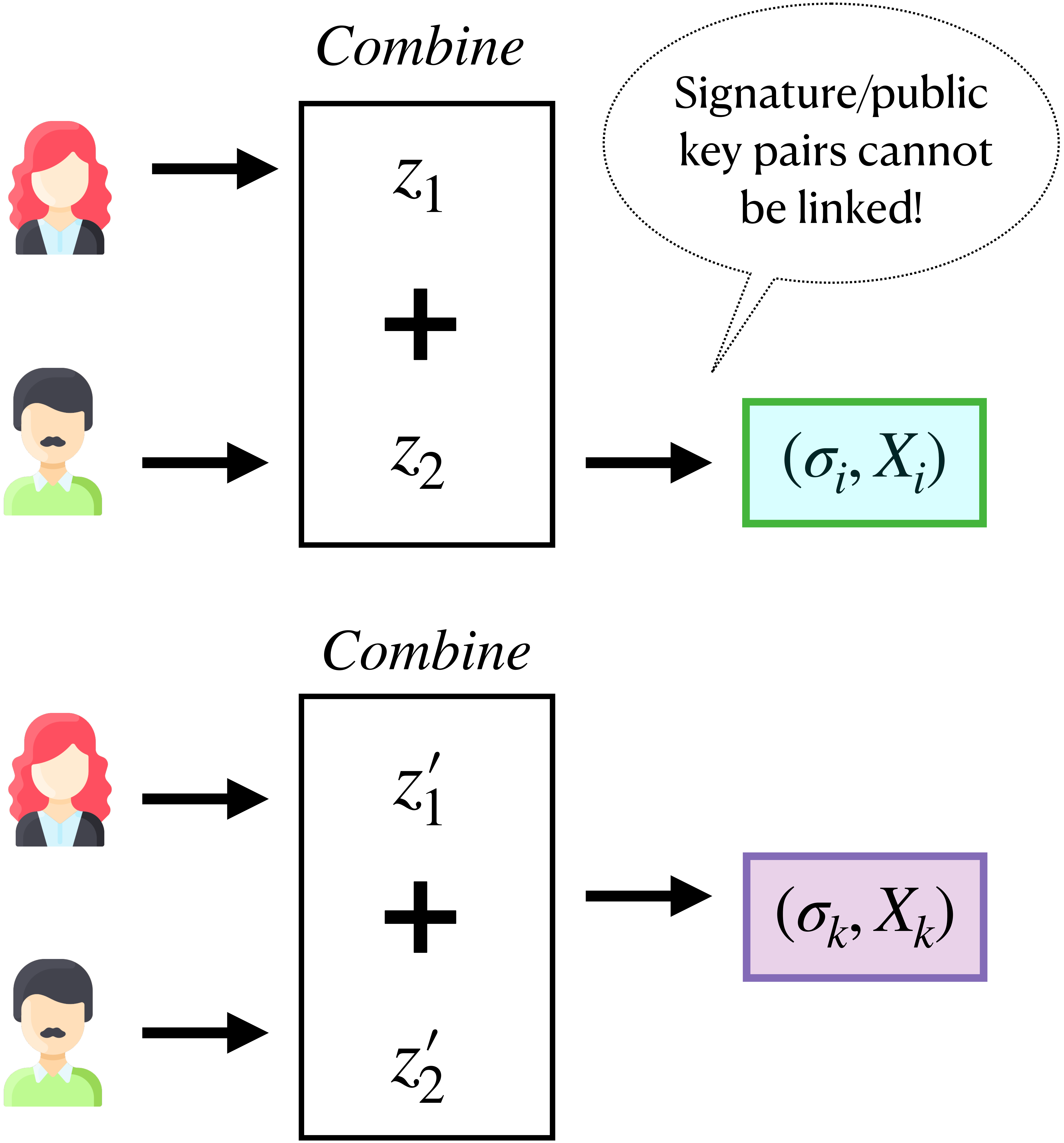
What's Next: Unlinkable FROST

- This is an ongoing effort, we have a candidate scheme



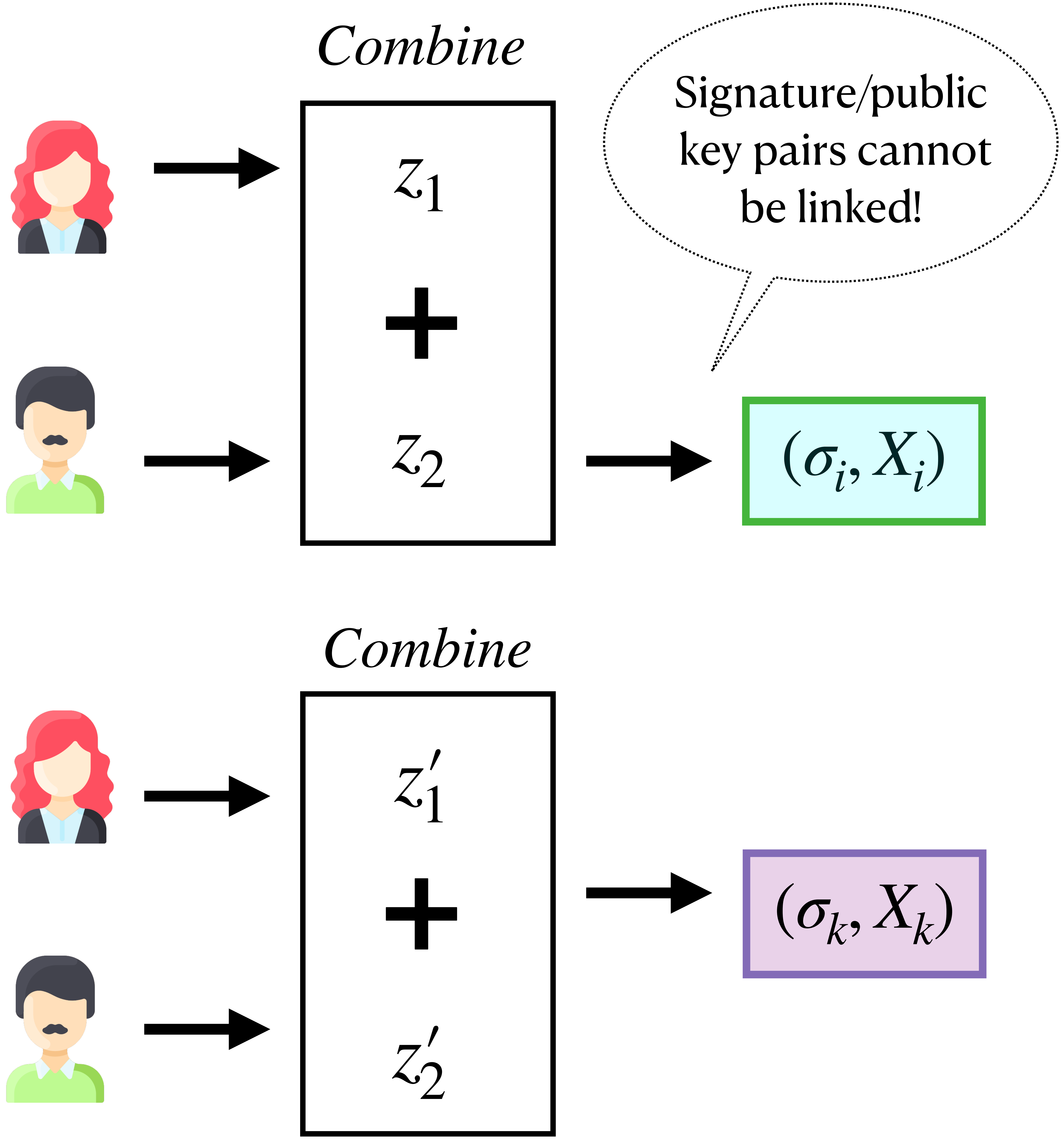
What's Next: Unlinkable FROST

- This is an ongoing effort, we have a candidate scheme
- We are investigating the various trust and privacy tradeoffs



What's Next: Unlinkable FROST

- This is an ongoing effort, we have a candidate scheme
- We are investigating the various trust and privacy tradeoffs
- If you are interested in this work or have use cases, come talk to me!



What do you need?

- FROST extensions for different privacy/security use cases?

What do you need?

- FROST extensions for different privacy/security use cases?
- Beyond multi signatures? We do other things too :)

What do you need?

- FROST extensions for different privacy/security use cases?
- Beyond multi signatures? We do other things too :)
- We are always looking for research projects, so please come talk to us!

What do you need?

- FROST extensions for different privacy/security use cases?
- Beyond multi signatures? We do other things too :)
- We are always looking for research projects, so please come talk to us!

Thank you! ❄