From Theory to Practice to Theory: Lessons Learned in Multi-Party Schnorr Signatures

Elizabeth Crites

University of Edinburgh

Chelsea Komlo

University of Waterloo
Zcash Foundation, Dfns

Tim Ruffing

Blockstream

Mar. 29, 2023

Why Multi-Party Schnorr Signatures? Why now?

"Efficient Identification and Signatures for Smart Cards," Schnorr, 1990.

Multi-Party Schnorr (Stinson & Strobl), 2001.

3-Round Multi-Party Schnorr (MuSig), 2018.

2-Round Multi-Party Schnorr (MuSig2, DWMS, FROST), 2020.

Bitcoin moves to Schnorr (BIP340), 2021.

NIST initial draft call for threshold signatures, 2023.

Patent on Schnorr signatures, filed 1991.

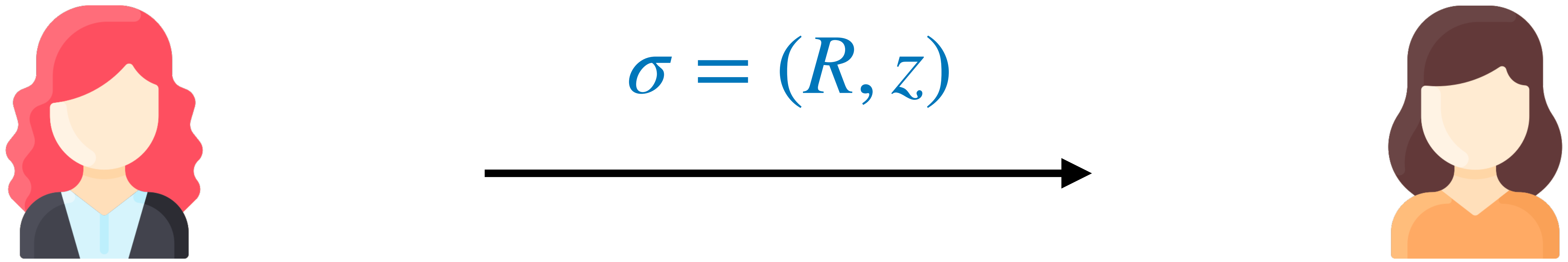
Patent on Schnorr signatures, expired 2008.

FROST2, Improved proof for FROST, 2021.

FROST CFRG draft, 2021.

NIST standardizes single-party EdDSA, 2022.

(Single-Party) Schnorr Signature Scheme [Sch90]

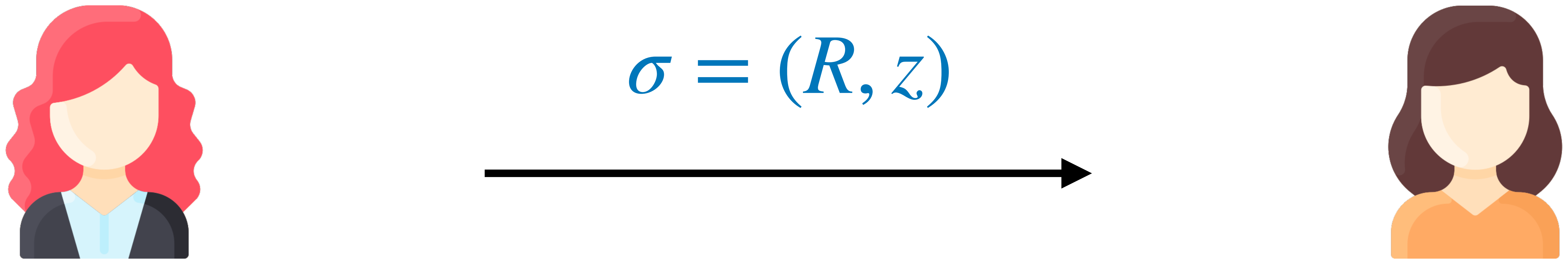


$\sigma = (R, z)$

To generate a key pair:

$$PK \leftarrow g^{sk}$$

(Single-Party) Schnorr Signature Scheme [Sch90]



$\sigma = (R, z)$

To generate a key pair:

$$PK \leftarrow g^{sk}$$

To sign a message m:

$$R \leftarrow g^{r}$$
$$c \leftarrow H(PK, m, R)$$
$$z \leftarrow r + c \cdot sk$$

(Single-Party) Schnorr Signature Scheme

[Sch90]

$\sigma = (R, z)$

To generate a key pair:

$$PK \leftarrow g^{sk}$$

To sign a message m:

$$R \leftarrow g^{r}$$

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

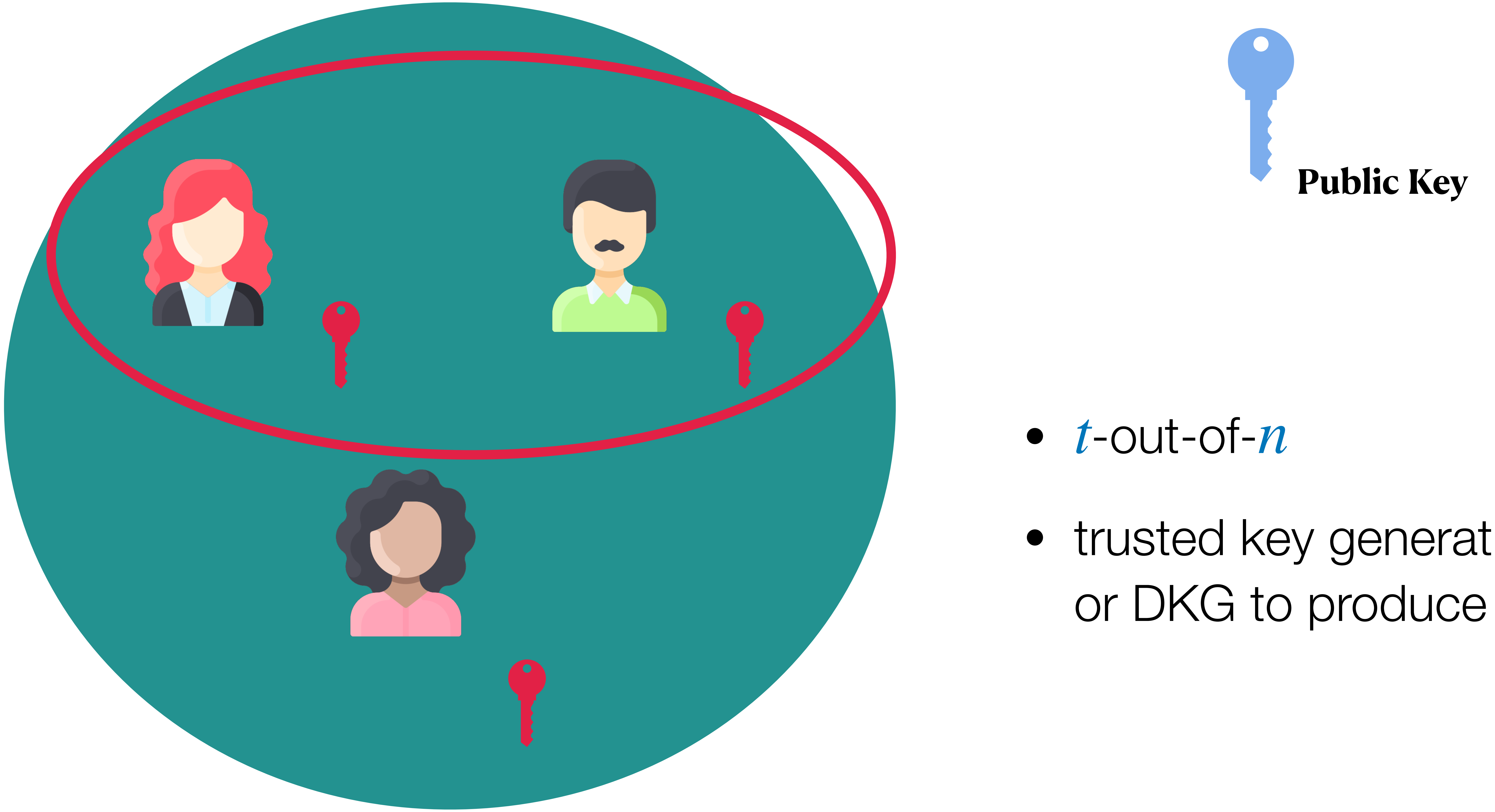
$$z \leftarrow r + c \cdot sk$$

Verify:

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

$$R \cdot PK^{c} = g^{z}$$ ✓

What are Threshold Signatures? [D87, DF89]



- t-out-of-n
- trusted key generation or DKG to produce PK

(2,3) Example

What are Multi-Signatures? [IN83, BN06]

- n-out-of-n
- key aggregation to produce PK
- n signers can be spontaneous

(3,3) Example

Multi-Party Schnorr Signatures

How to share sk ?

How to share r ?

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

$$sig = (R, z)$$

What do we want?

- output signature that verifies like standard Schnorr signature
    - public key looks like standard Schnorr signature public key
- few (2-3) rounds
    - Stinson & Strobl 2001 uses DKG for signing
- reasonable security assumptions
- concurrent security

	Scheme	Assumptions	Signing Rounds
Multi-sigs	MuSig [MPSW18, BDN18] SimpleMuSig [BDN18, CKM21]	DL+ROM	3
	MuSig2 [NRS21] DWMS [AB21] SpeedyMuSig [CKM21]	OMDL+ROM	2
Threshold	Lindell22 Sparkle [CKM23]	Schnorr DL+ROM	3
	FROST [KG20, BCKMTZ22] FROST2 [CKM21]	OMDL+ROM	2

One-More Discrete Log (OMDL):

- stronger assumption

+ partially non-interactive schemes

	Scheme	Assumptions	Signing Rounds
Multi-sigs	MuSig [MPSW18, BDN18] SimpleMuSig [BDN18, CKM21]	DL+ROM	3
	MuSig2 [NRS21] DWMS [AB21] SpeedyMuSig [CKM21]	OMDL+ROM	2
Threshold	Lindell22 Sparkle [CKM23]	Schnorr DL+ROM	3
	FROST [KG20, BCKMTZ22] FROST2 [CKM21]	OMDL+ROM	2

All are concurrently secure ✓

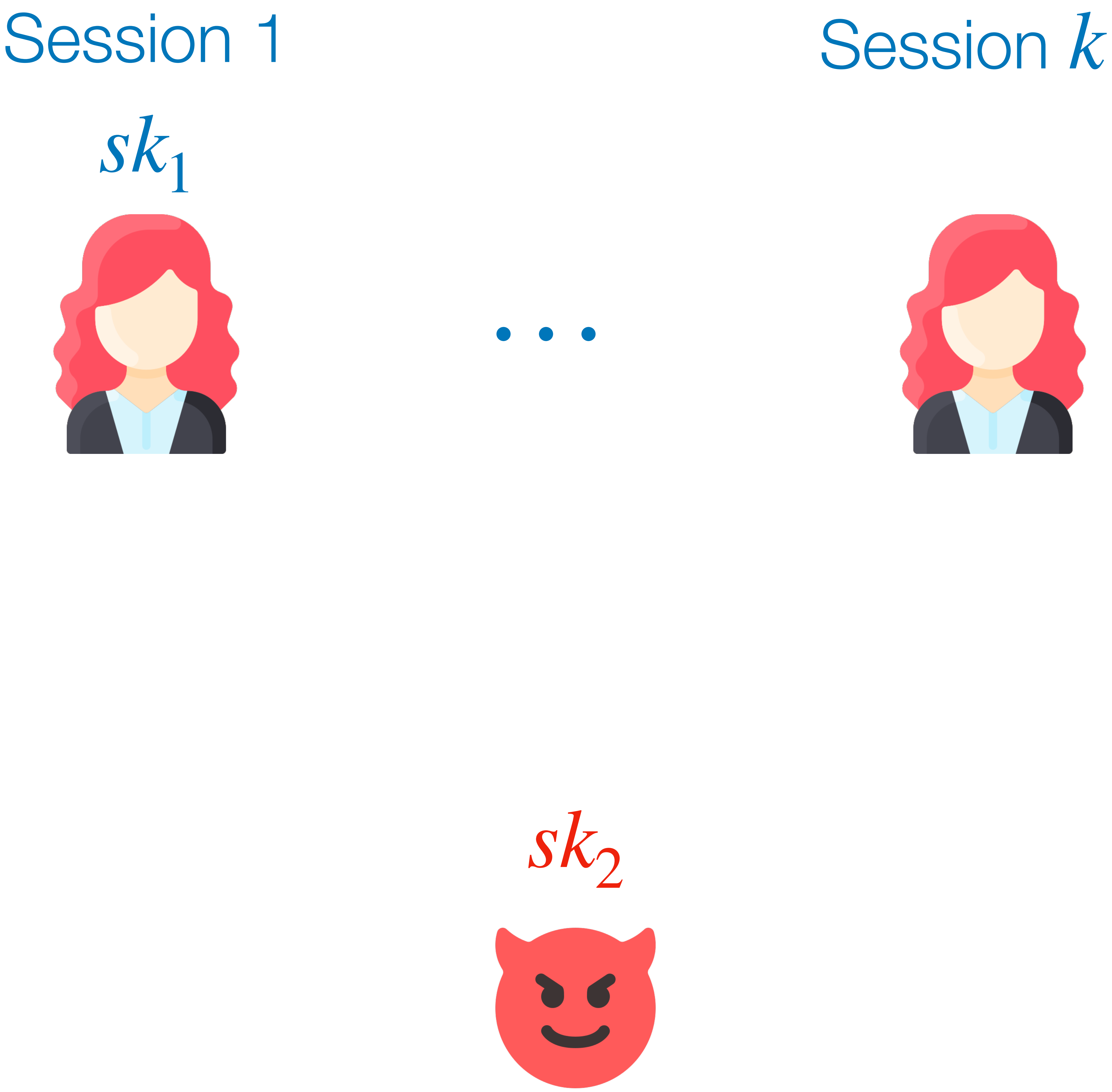
One-More Discrete Log (OMDL):

- stronger assumption

+ partially non-interactive schemes

Concurrent Security: ROS Attacks

[NKDM03, DEFKLNS19, BLLOR21]



Session 1

sk_1

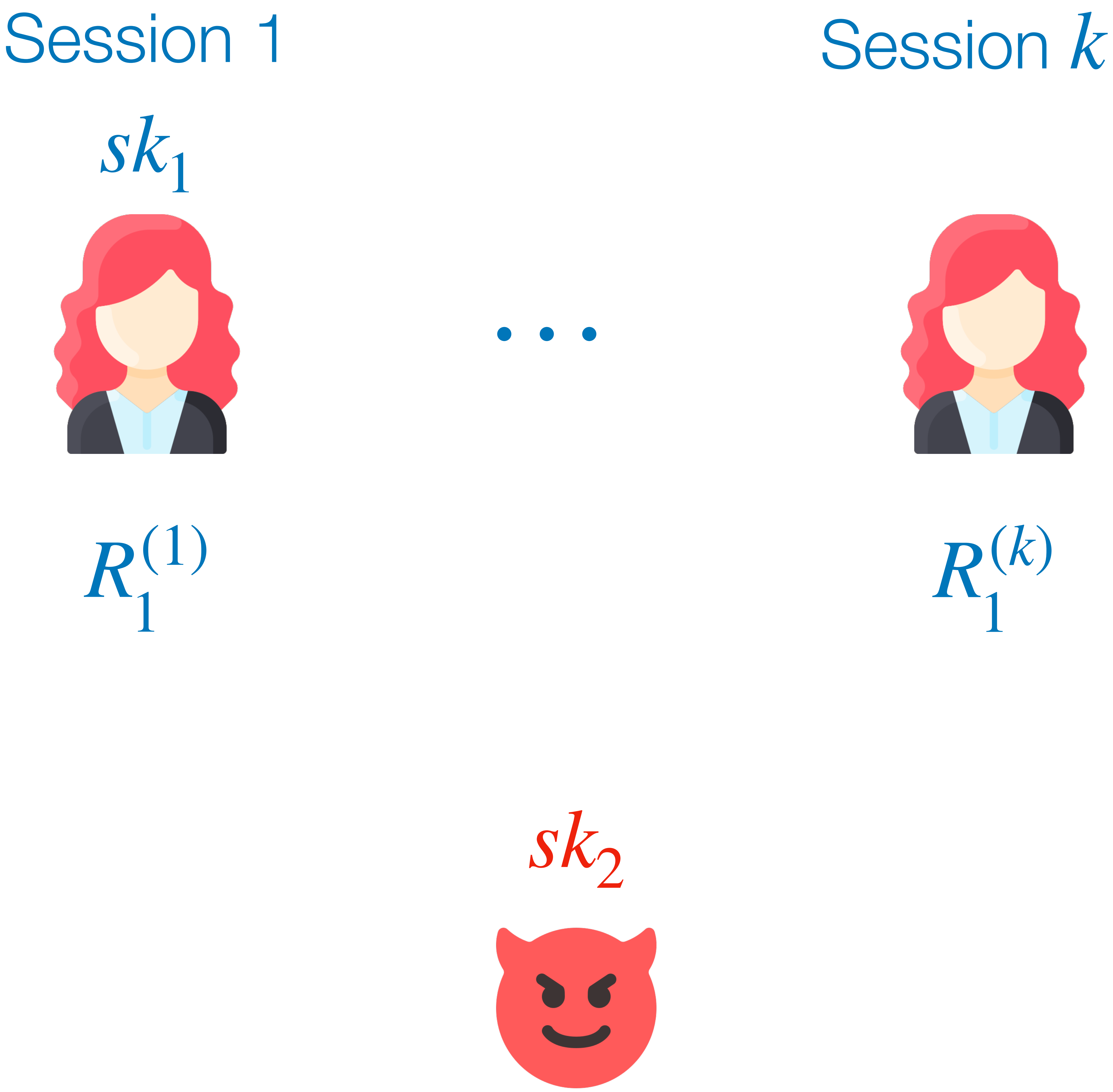
$\dots$

Session k

sk_2

Concurrent Security: ROS Attacks

[NKDM03, DEFKLNS19, BLLOR21]



Session 1

sk_1

$\cdots$

$R_1^{(1)}$

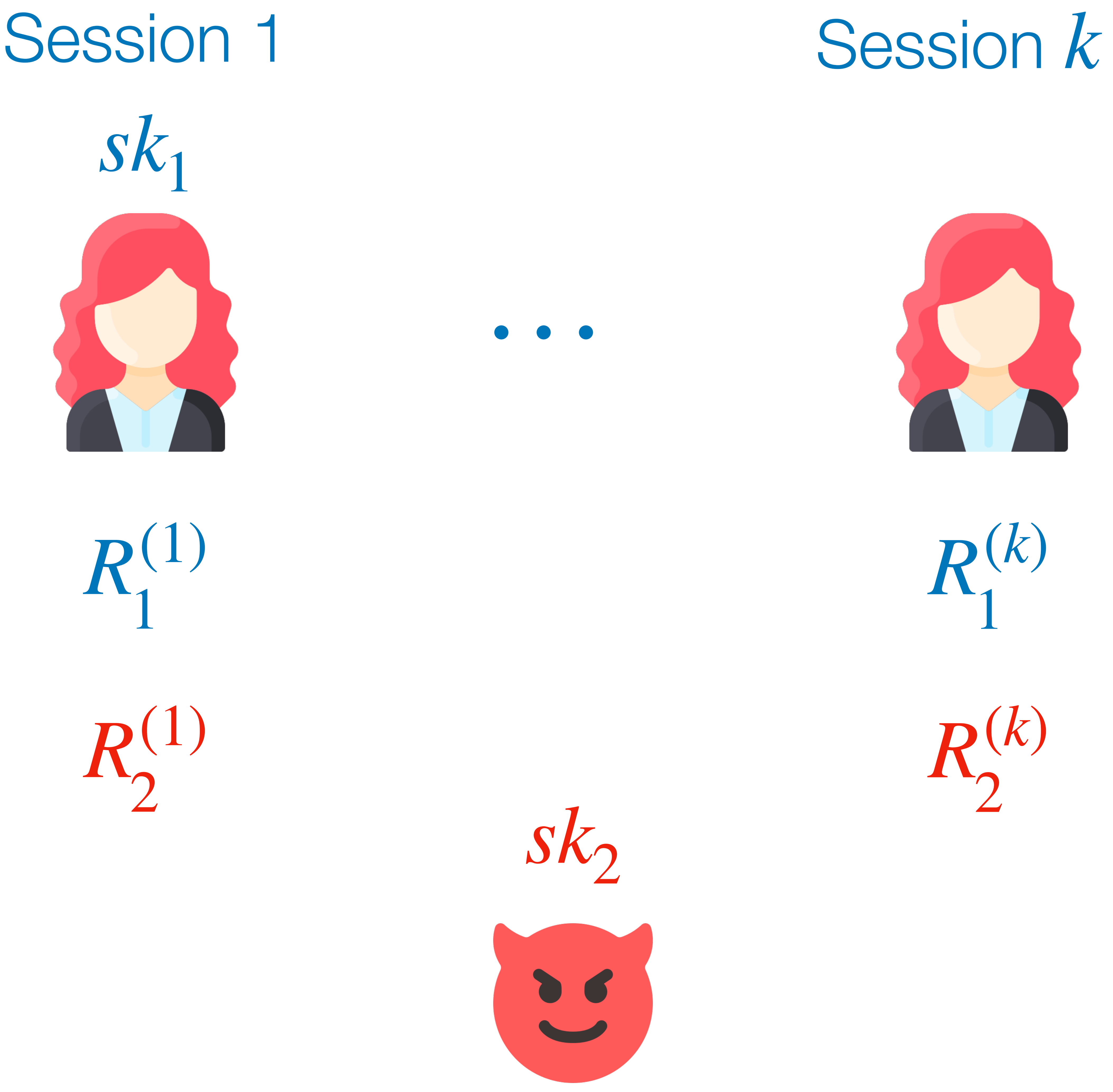
Session k

$R_1^{(k)}$

sk_2

Concurrent Security: ROS Attacks

[NKDM03, DEFKLNS19, BLLOR21]



Session 1

sk_1

$\cdots$

Session k

$R_1^{(1)}$

$R_2^{(1)}$

$R_1^{(k)}$

$R_2^{(k)}$

sk_2

Concurrent Security: ROS Attacks

[NKDM03, DEFKLNS19, BLLOR21]

Session 1

sk_1

$R_1^{(1)}$

$R_2^{(1)}$

$\dots$

Session k

$R_1^{(k)}$

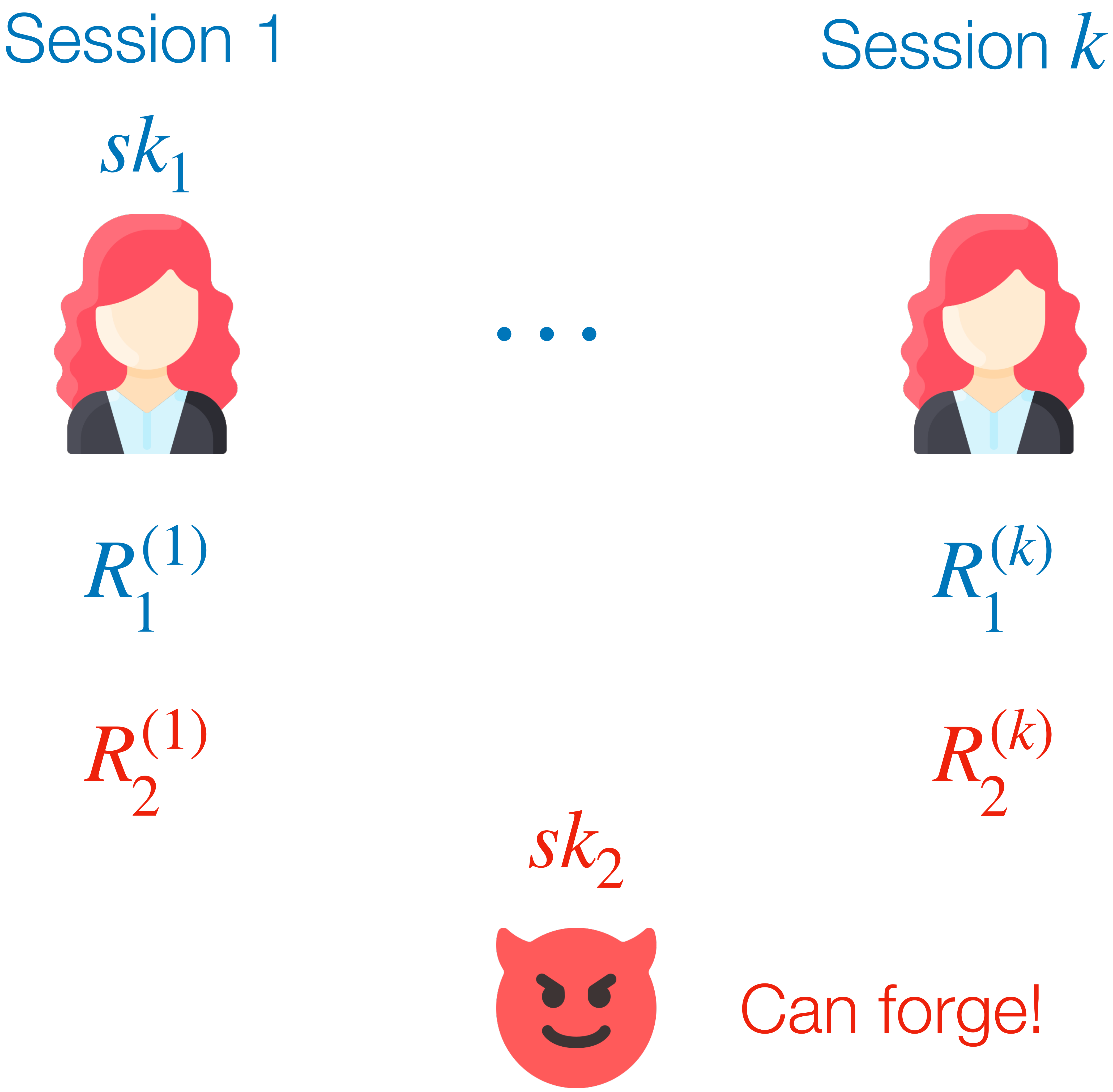
$R_2^{(k)}$

sk_2

Can forge!

Concurrent Security: ROS Attacks

[NKDM03, DEFKLNS19, BLLOR21]



Session 1

Session k

sk_1

$\cdots$

$R_1^{(1)}$

$R_2^{(1)}$

$R_1^{(k)}$

$R_2^{(k)}$

sk_2

Can forge!

Affected:

- multi-signatures
- threshold signatures
- blind signatures

Solution: Force adversary to commit to its nonces…

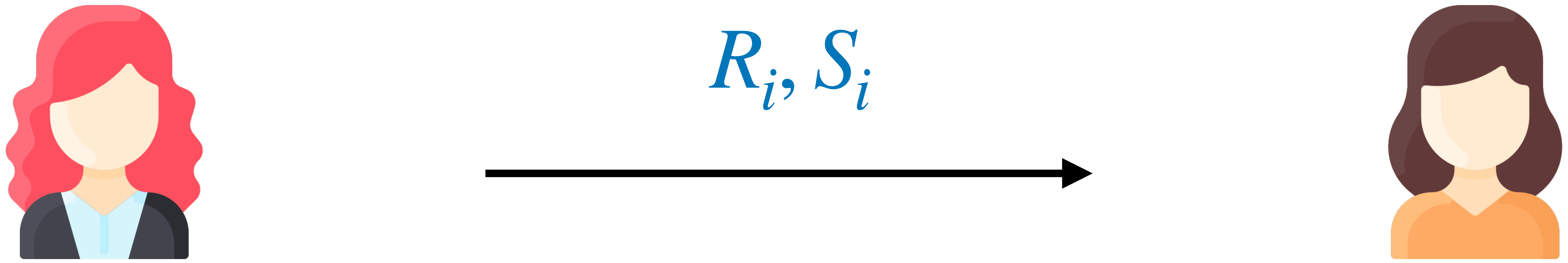
MuSig2 / SpeedyMuSig / FROST/2

Key Generation:

$(sk_i, PK_i), PK$

Combine / Verify:

MuSig2 / SpeedyMuSig / FROST/2


R_i, S_i

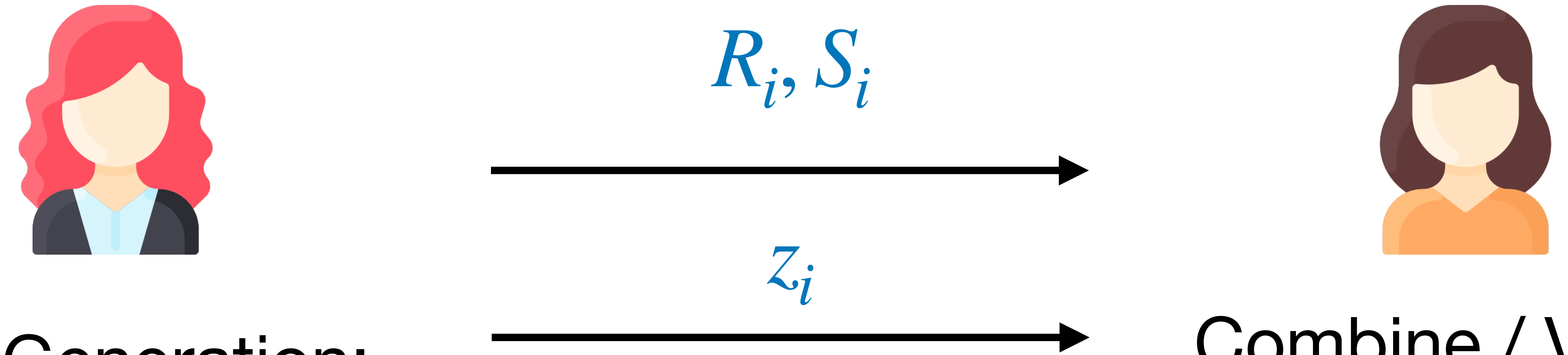
Key Generation:

$(sk_i, PK_i), PK$

Combine / Verify:

Round 1: Output $R_i \leftarrow g^{r_i}, S_i \leftarrow g^{s_i}$

MuSig2 / SpeedyMuSig / FROST/2



Key Generation:

$(sk_i, PK_i), PK$

Combine / Verify:

Round 1: Output $R_i \leftarrow g^{r_i},\ S_i \leftarrow g^{s_i}$

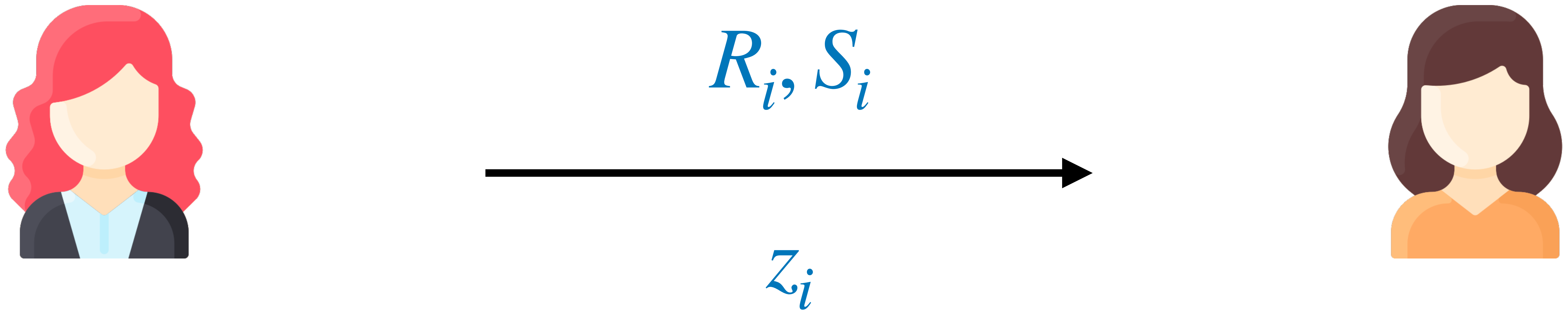
Round 2: $a \leftarrow H'(PK, m, \{R_i, S_i\}_{i=1}^{n})$

$R = \Pi_{i=1}^{n} R_i S_i^a$

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

Output $z_i \leftarrow r_i + as_i + csk_i$

MuSig2 / SpeedyMuSig / FROST/2



R_i, S_i

z_i

Key Generation:

$(sk_i, PK_i), PK$

Round 1: Output $R_i \leftarrow g^{r_i}, \ S_i \leftarrow g^{s_i}$

Round 2: $a \leftarrow H'(PK, m, \{R_i, S_i\}_{i=1}^{n})$

$R = \Pi_{i=1}^{n} R_i S_i^a$

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

Output $z_i \leftarrow r_i + as_i + csk_i$

Combine / Verify:

$z = \Sigma_{i=1}^{n} z_i$

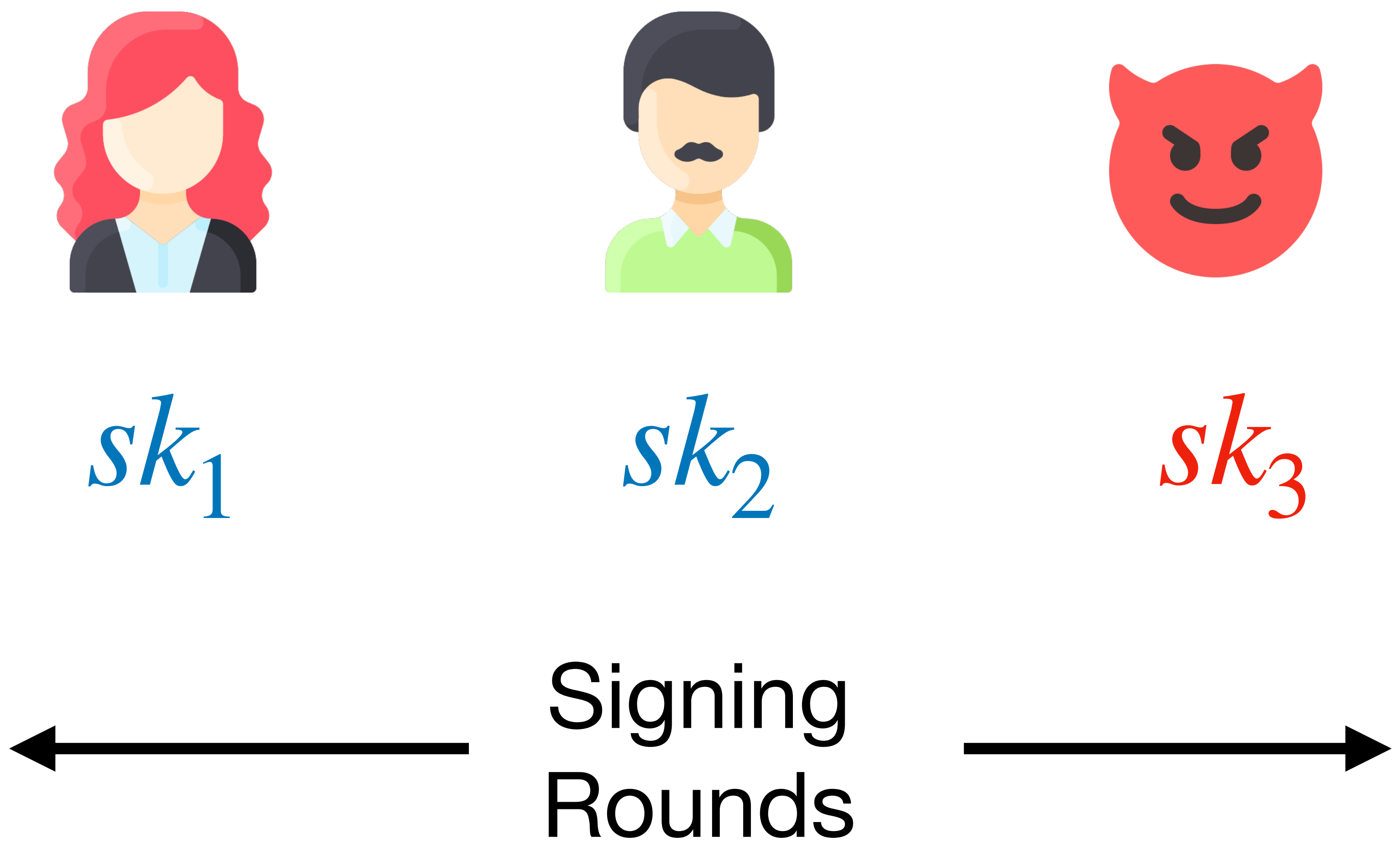
$sig = (R, z)$

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

$R \cdot PK^c = g^z$ ✓

Adaptive Security

Static Corruption

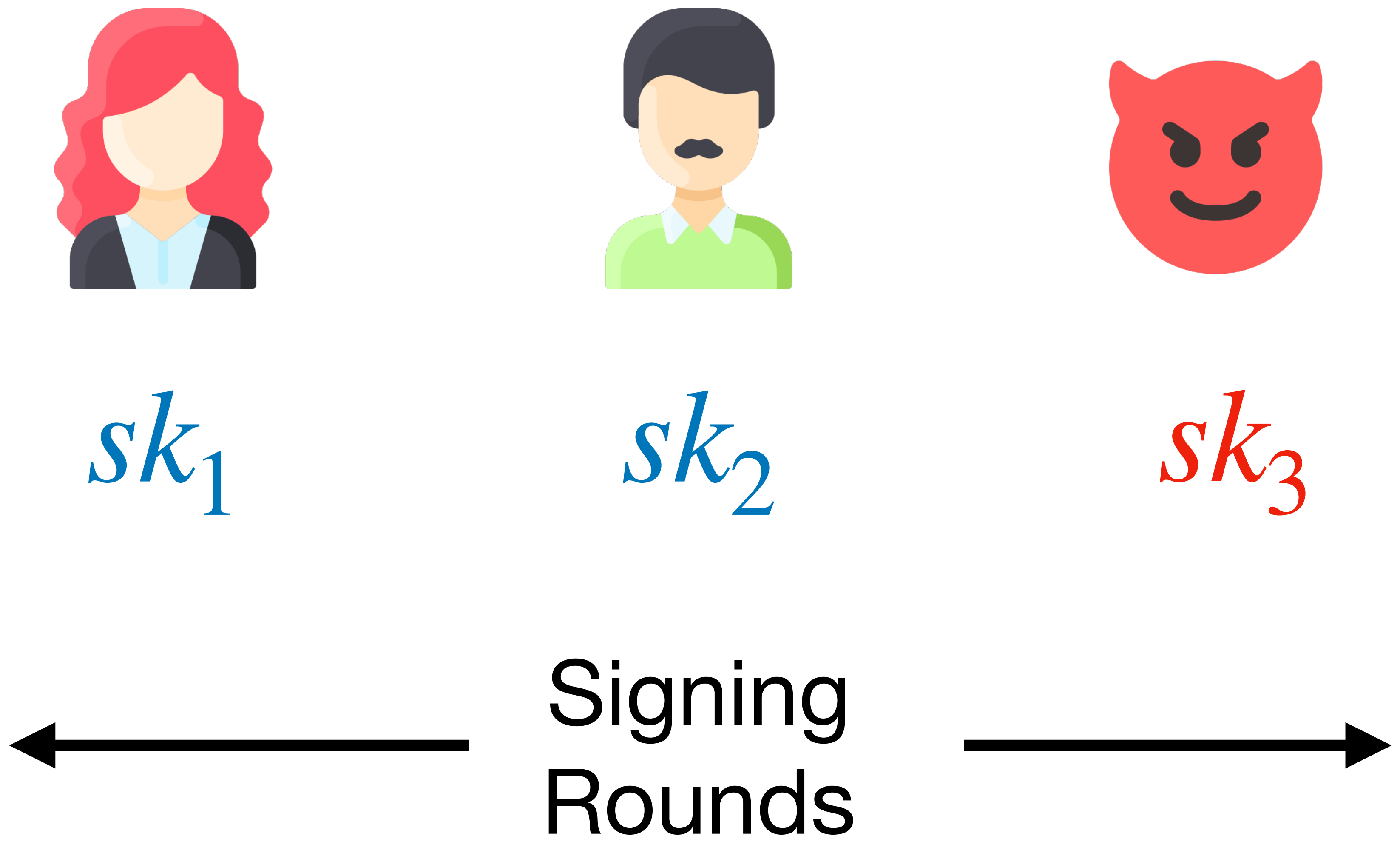


sk_1 sk_2 sk_3

Signing Rounds

Adaptive Security

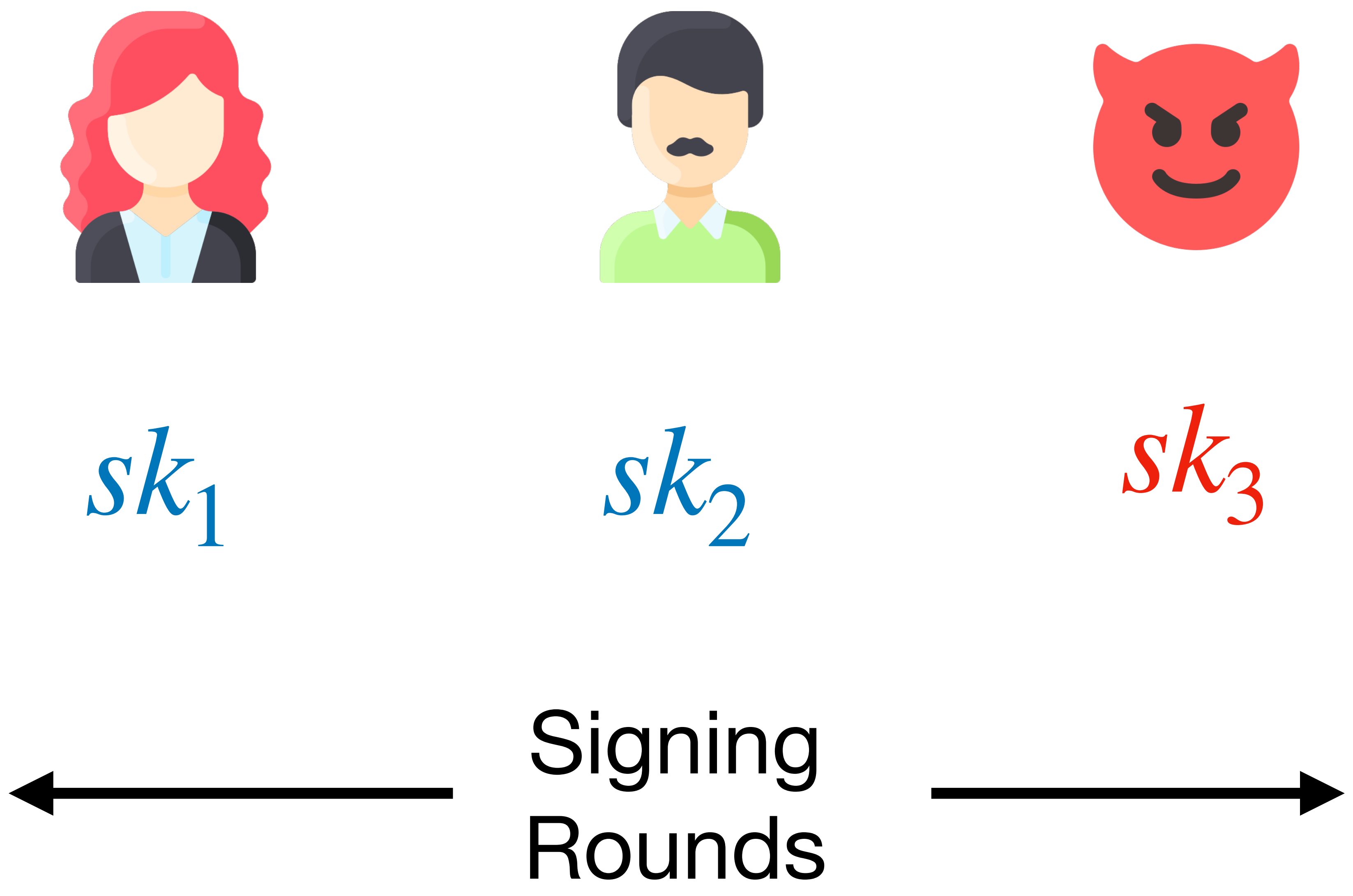
Static Corruption



sk_1 sk_2 sk_3

Signing Rounds

Adaptive Corruption

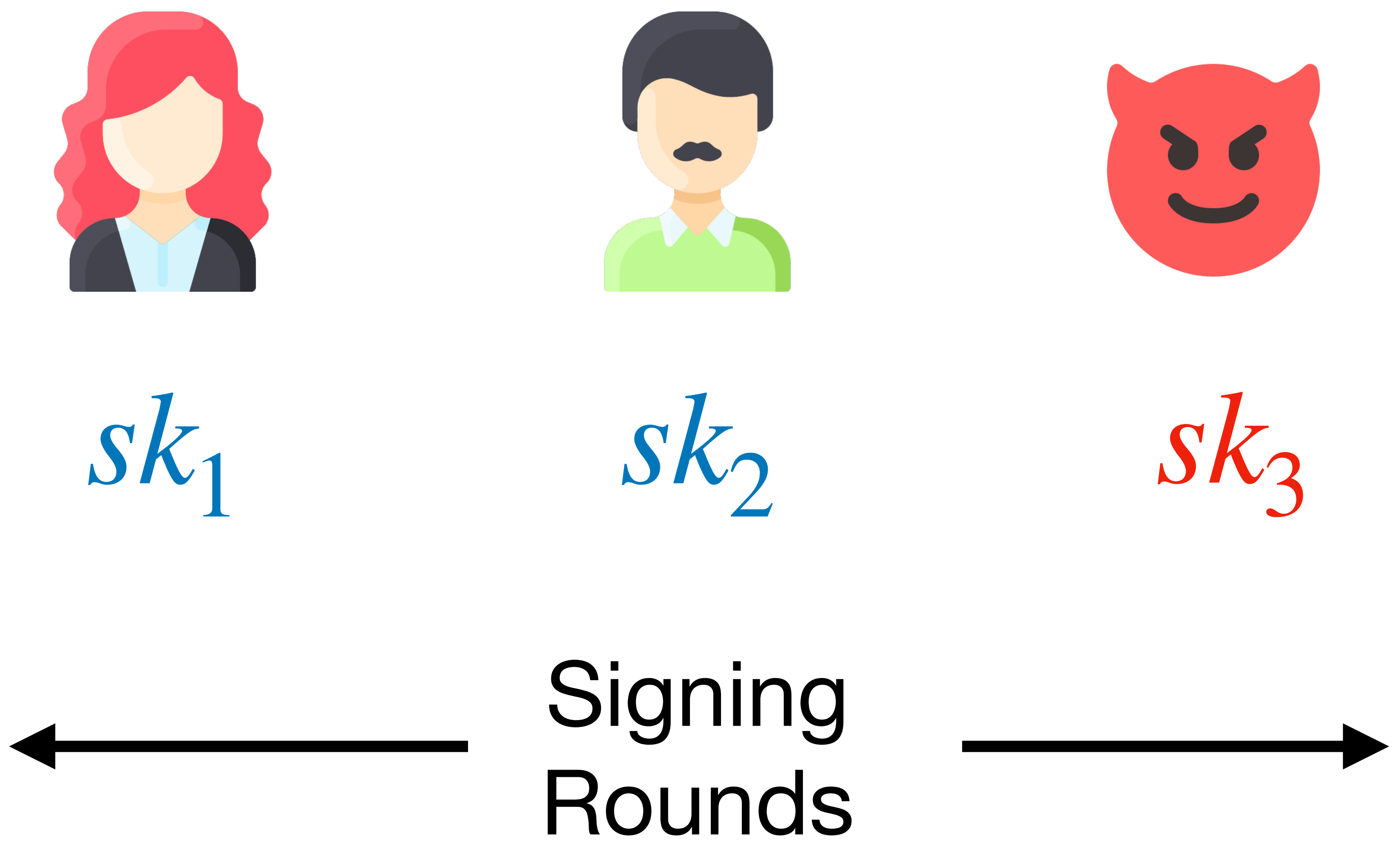


sk_1 sk_2 sk_3

Signing Rounds

Adaptive Security

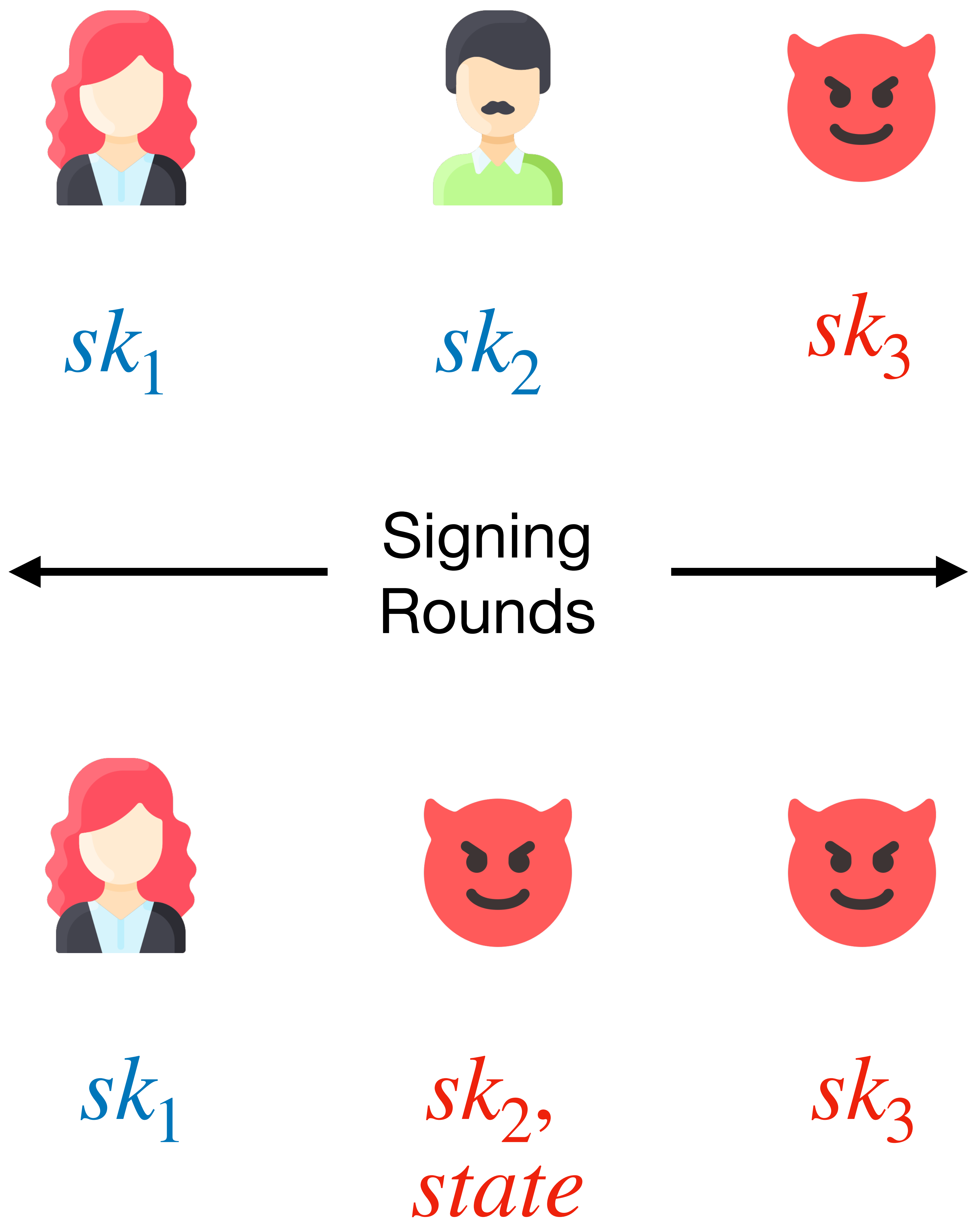
Static Corruption



sk_1 sk_2 sk_3

Signing Rounds

Adaptive Corruption



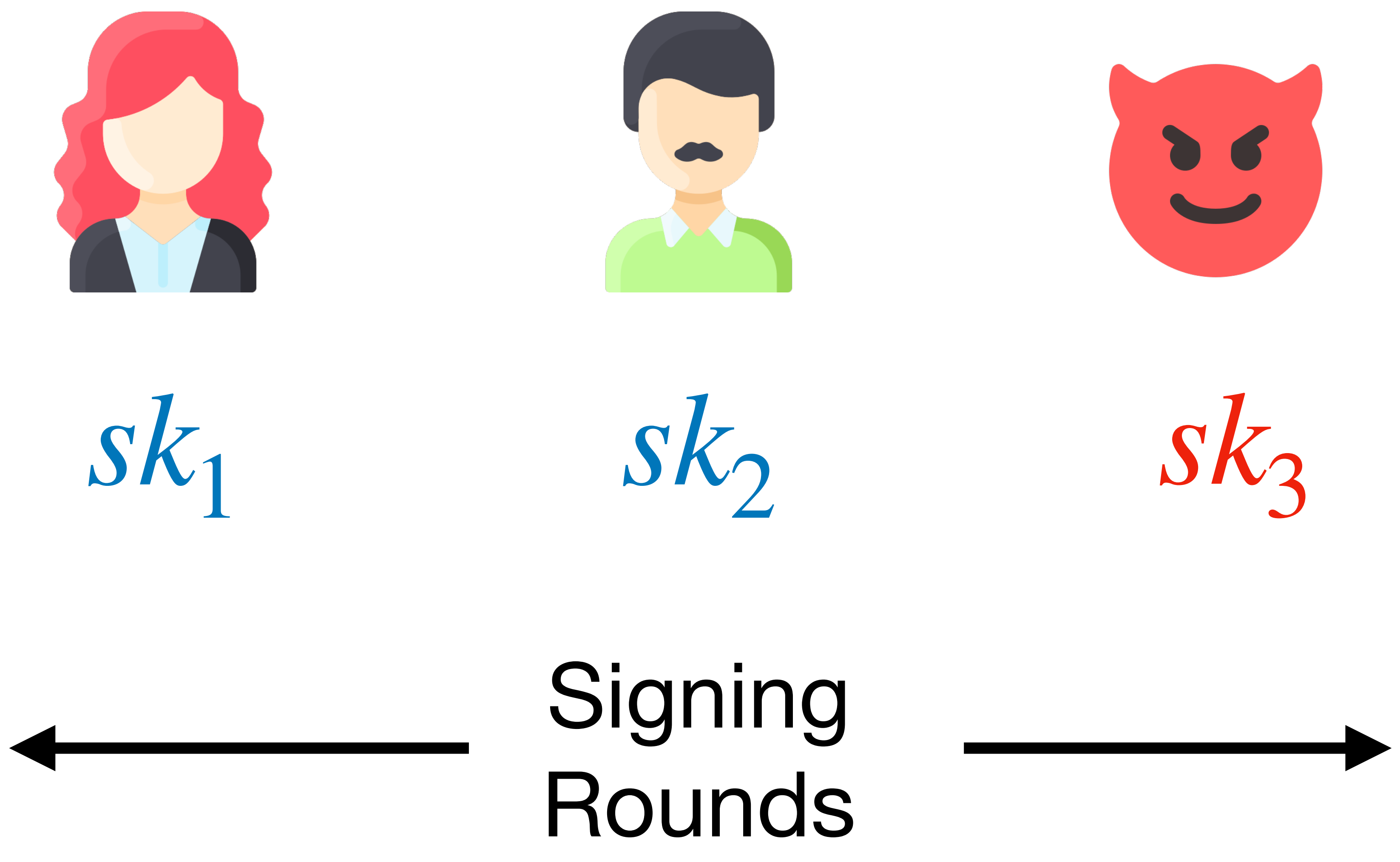
sk_1 sk_2 sk_3

Signing Rounds

sk_1 $sk_2, state$ sk_3

Adaptive Security

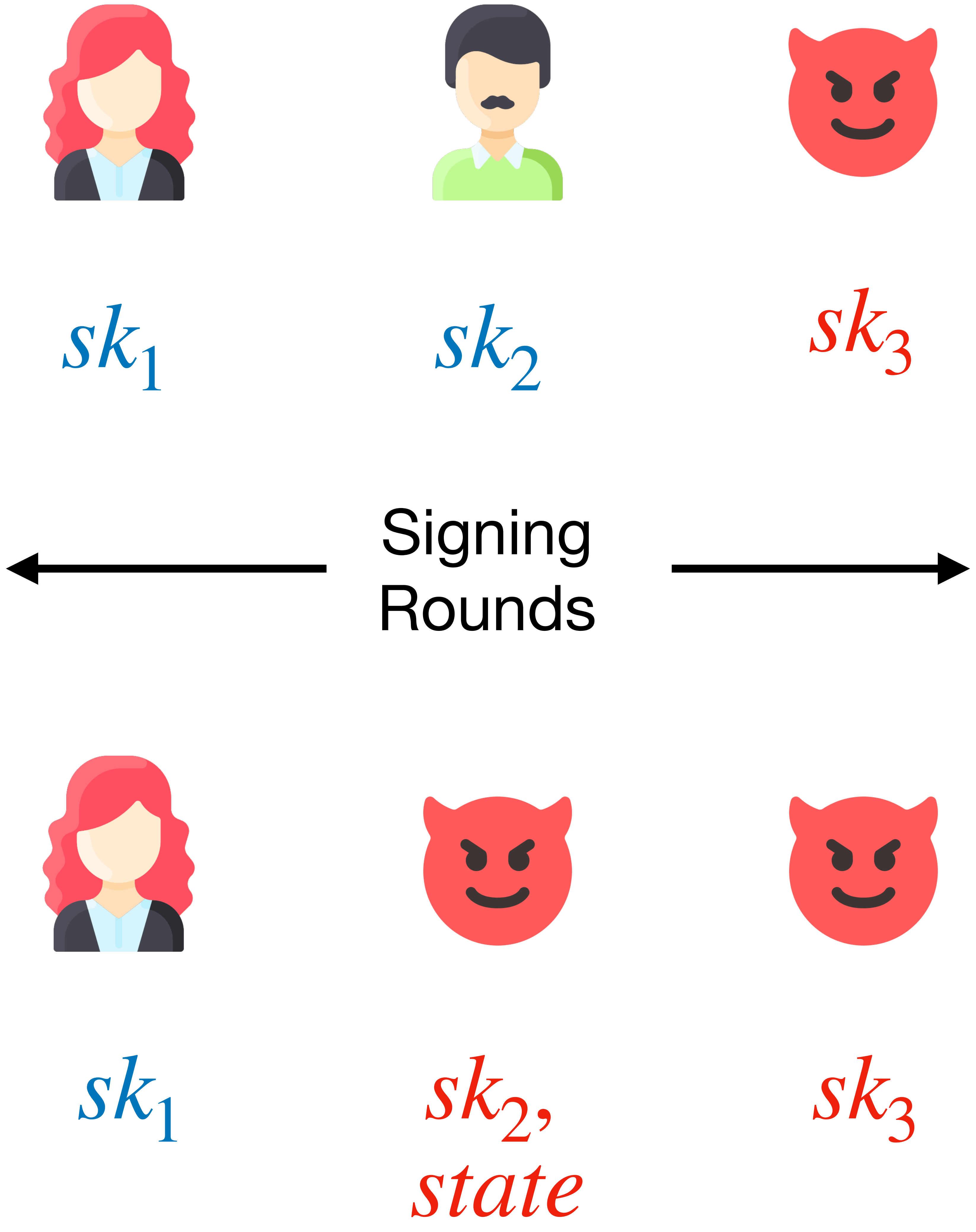
Static Corruption



sk_1 sk_2 sk_3

Signing Rounds

Adaptive Corruption



sk_1 sk_2 sk_3

Signing Rounds

sk_1 $sk_2,$ $state$ sk_3

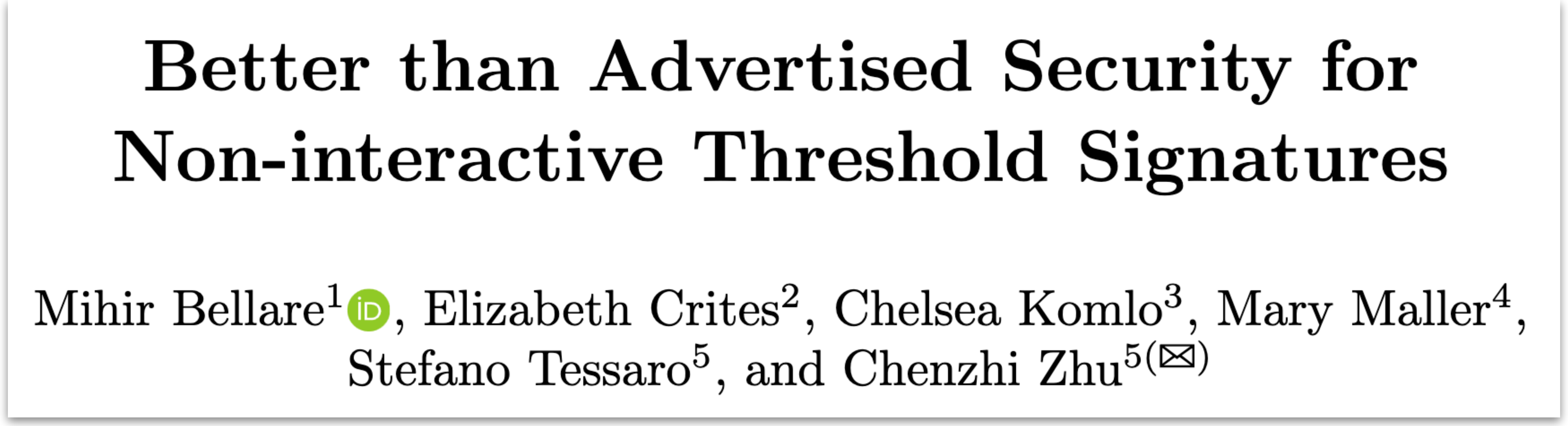
- Adaptive security of Sparkle [CKM23], FROST forthcoming

MuSig2: Simple Two-Round Schnorr Multi-Signatures

Jonas Nick[1], Tim Ruffing[1], and Yannick Seurin[2]

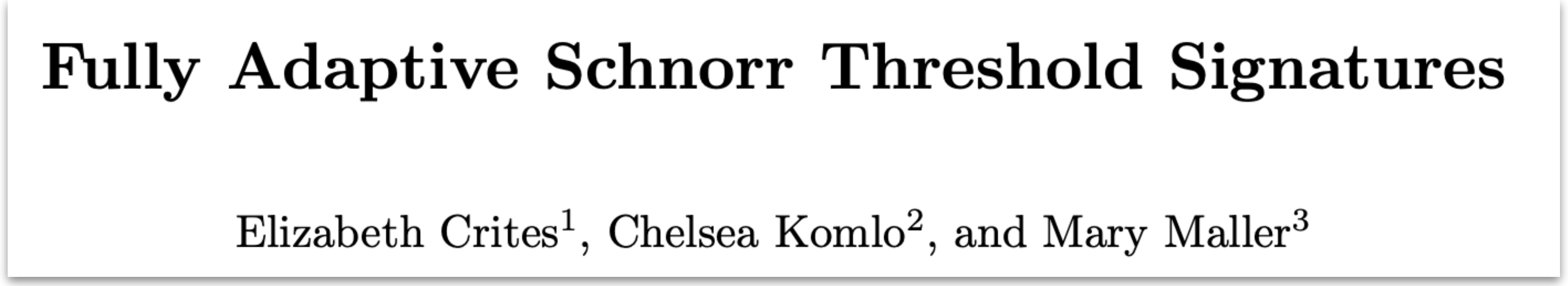
How to Prove Schnorr Assuming Schnorr: Security of Multi- and Threshold Signatures

Elizabeth Crites[1], Chelsea Komlo[2], and Mary Maller[3]



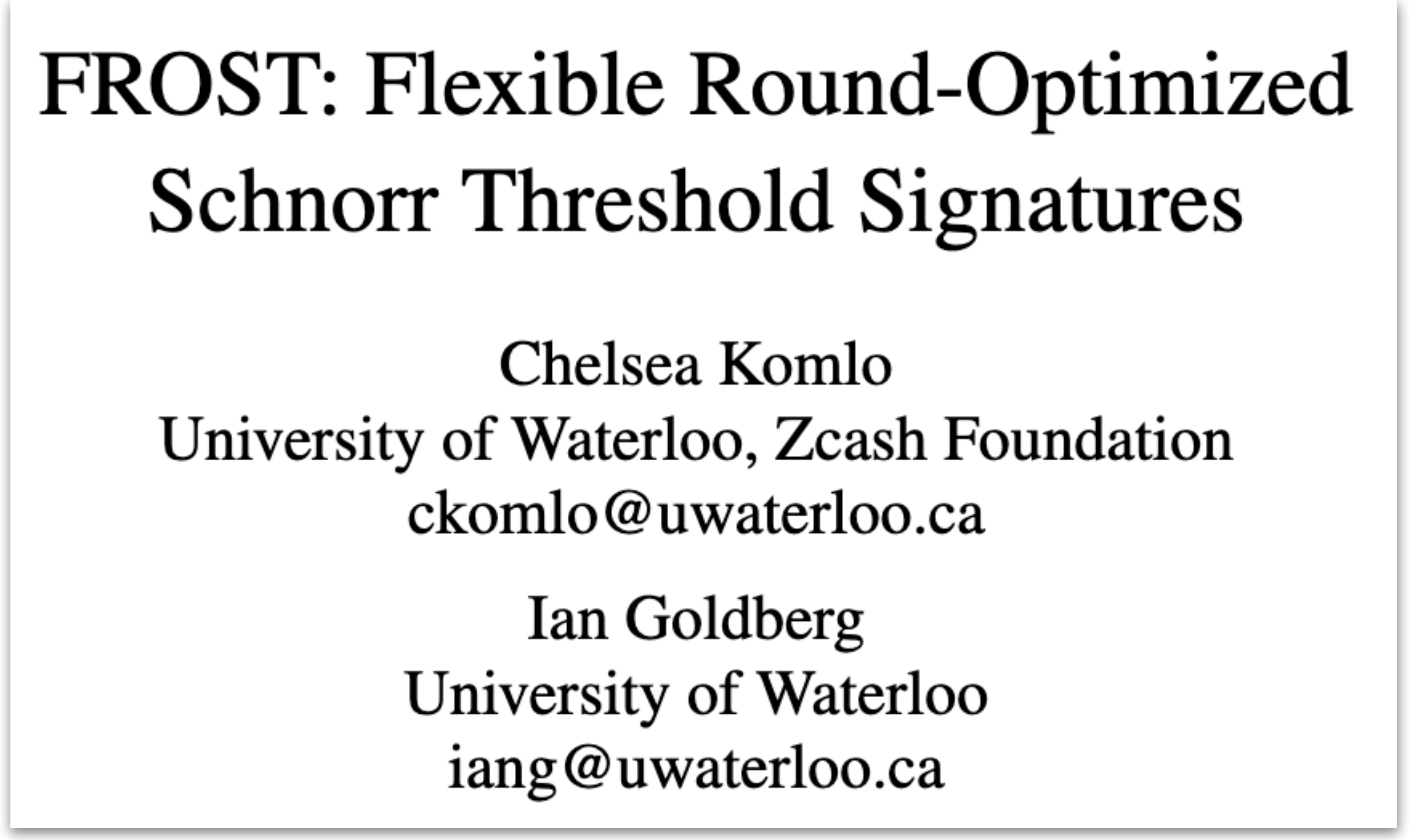
Better than Advertised Security for Non-interactive Threshold Signatures

Mihir Bellare[1], Elizabeth Crites[2], Chelsea Komlo[3], Mary Maller[4], Stefano Tessaro[5], and Chenzhi Zhu[5](✉)



Fully Adaptive Schnorr Threshold Signatures

Elizabeth Crites[1], Chelsea Komlo[2], and Mary Maller[3]



FROST: Flexible Round-Optimized Schnorr Threshold Signatures

Chelsea Komlo
University of Waterloo, Zcash Foundation
ckomlo@uwaterloo.ca

Ian Goldberg
University of Waterloo
iang@uwaterloo.ca

ROAST: Robust Asynchronous Schnorr Threshold Signatures

Tim Ruffing
Blockstream
crypto@timruffing.de

Viktoria Ronge
Friedrich-Alexander-Universität Erlangen-Nürnberg
ronge@cs.fau.de

Elliott Jin
Blockstream
eyj@blockstream.com

Jonas Schneider-Bensch
CISPA Helmholtz Center for Information Security
jonas.schneider-bensch@cispa.de

Dominique Schröder
Friedrich-Alexander-Universität Erlangen-Nürnberg
dominique.schroeder@fau.de

A Formal Treatment of Distributed Key Generation, and New Constructions

Chelsea Komlo, Ian Goldberg, Douglas Stebila

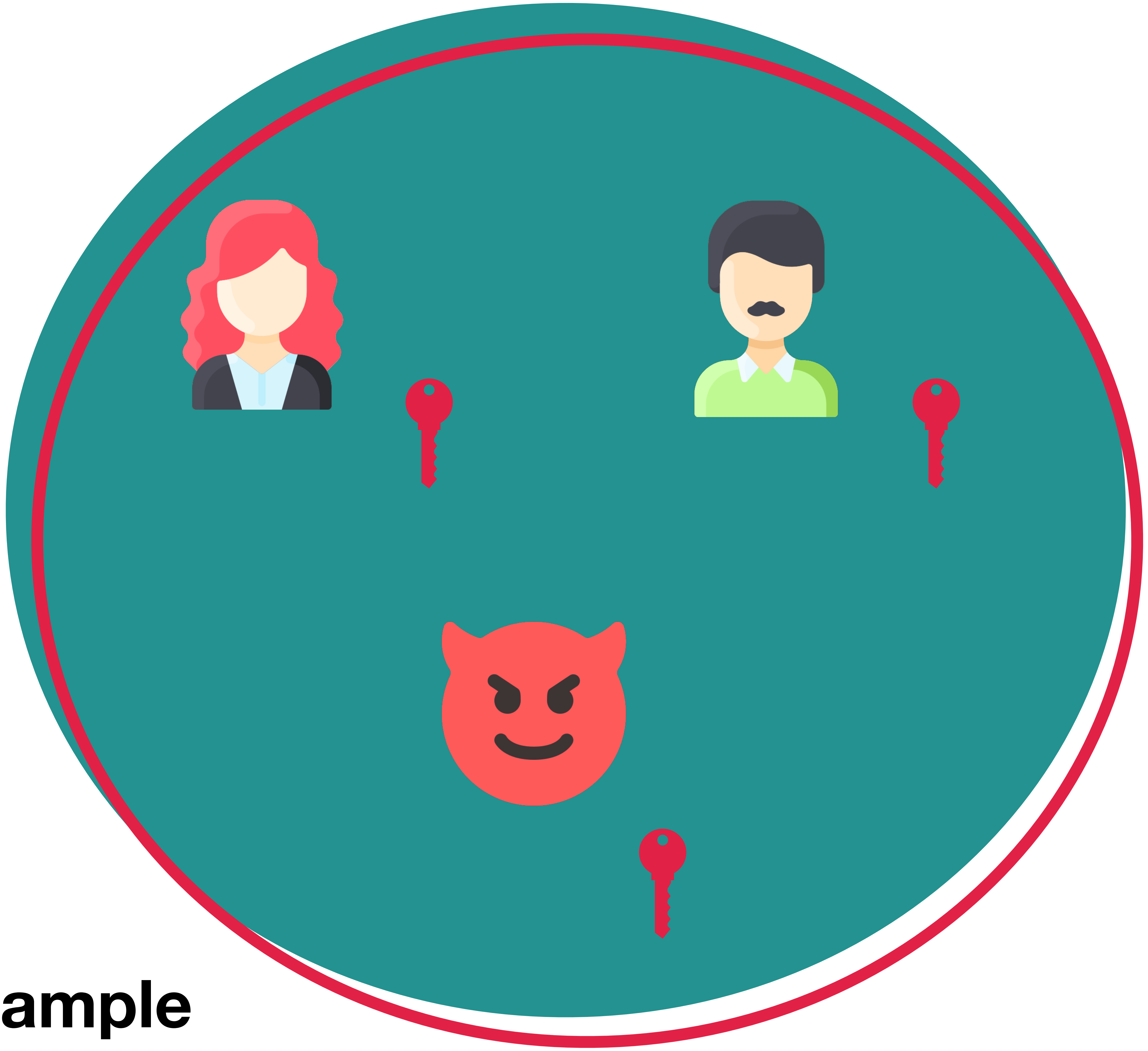
From Theory to Practice:
A Hitchhiker's Guide

Unforgeability:
Attacker cannot forge signatures.

Liveness:
System can always create signatures.

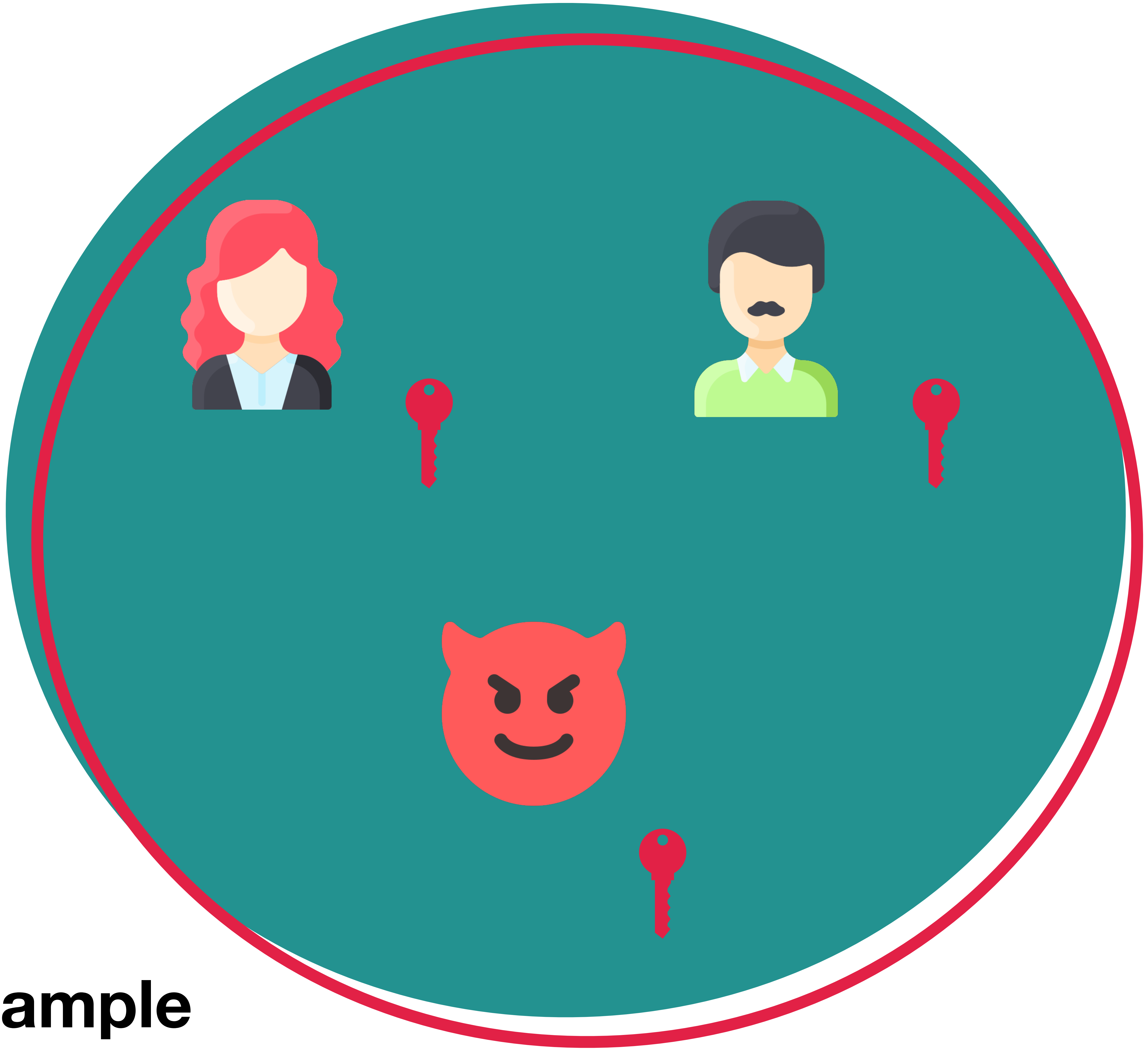
Multi-Signatures vs. Threshold Signatures

Multi-Signatures do not Guarantee Liveness



(3,3) Example

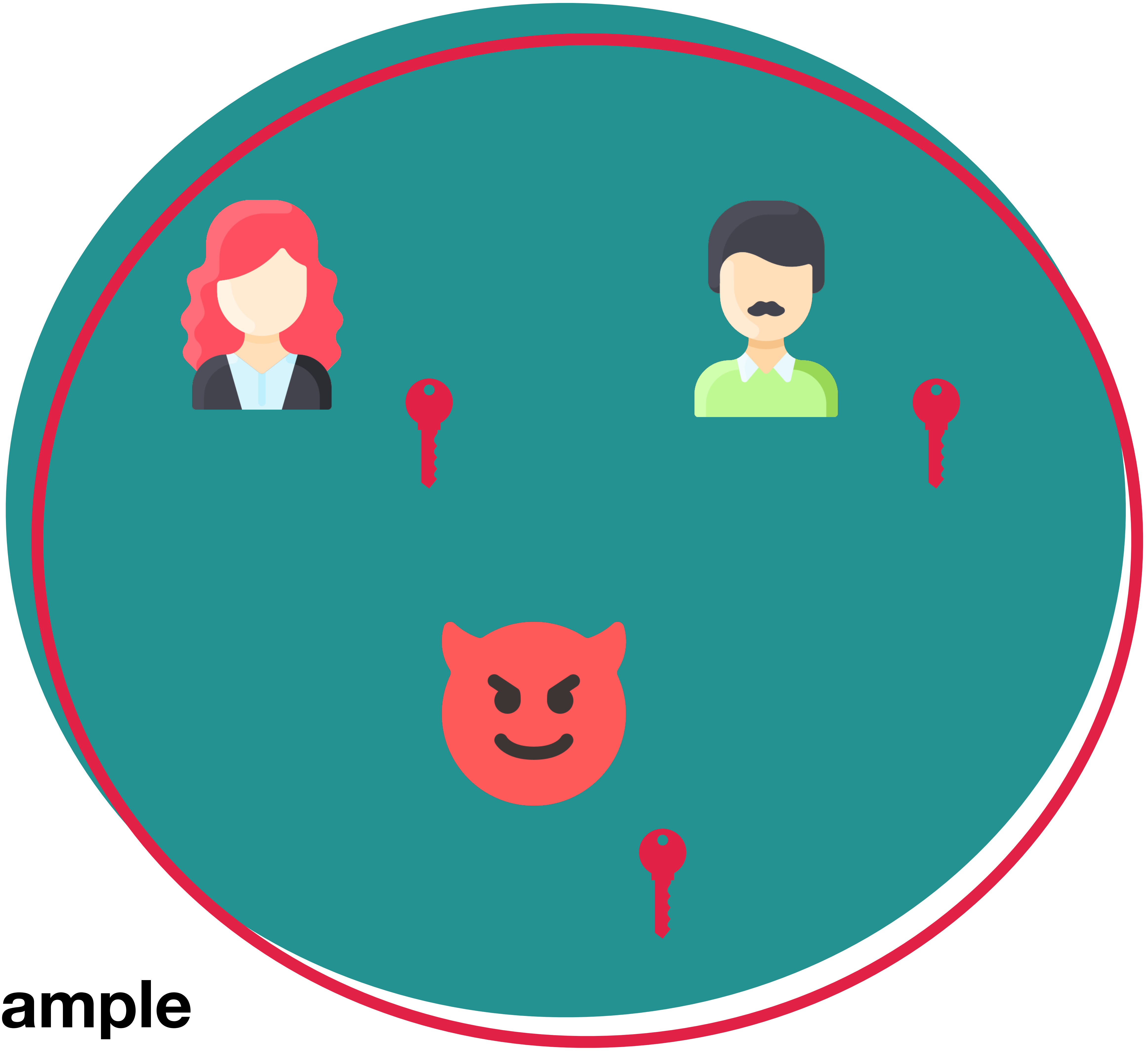
Multi-Signatures do not Guarantee Liveness



(3,3) Example

- If only one signer is unavailable, signing is not possible.

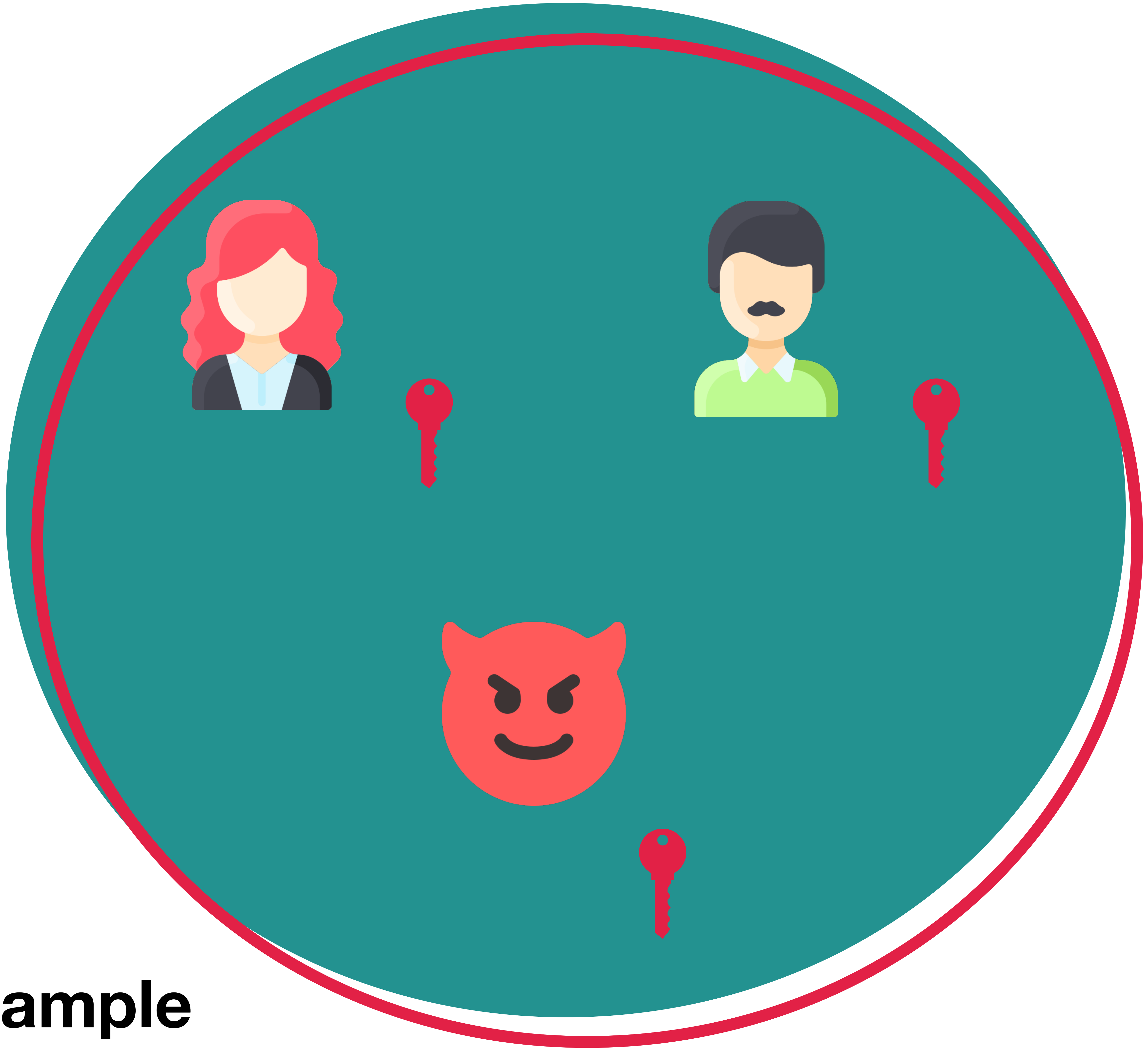
Multi-Signatures do not Guarantee Liveness



(3,3) Example

- If only one signer is unavailable, signing is not possible.
- Needs to be handled on a different layer of the system.

Multi-Signatures do not Guarantee Liveness



(3,3) Example

- If only one signer is unavailable, signing is not possible.
- Needs to be handled on a different layer of the system.
- Possible to use non-interactive key aggregation instead of DKG.

DKGs can be Cumbersome

DKGs can be Cumbersome

- Distributed Key Generation (DKG)

DKGs can be Cumbersome

- Distributed Key Generation (DKG)
- Major pain point: DKGs require some kind of broadcast channel

DKGs can be Cumbersome

- Distributed Key Generation (DKG)
- Major pain point: DKGs require some kind of broadcast channel
    - Protocol descriptions often just assume that all communication takes place over reliable broadcast (= consensus/BFT)

DKGs can be Cumbersome

- Distributed Key Generation (DKG)
- Major pain point: DKGs require some kind of broadcast channel
    - Protocol descriptions often just assume that all communication takes place over reliable broadcast (= consensus/BFT)
    - Implementers often fail to understand this, or simply ignore it

How to Choose (n, t)
for Threshold Signatures ?

How to Choose (n, t)
for Threshold Signatures ?

FROST supports any choice,
but that just makes the problem harder!

Honest Majority (Classic)

n = 5

t = 3

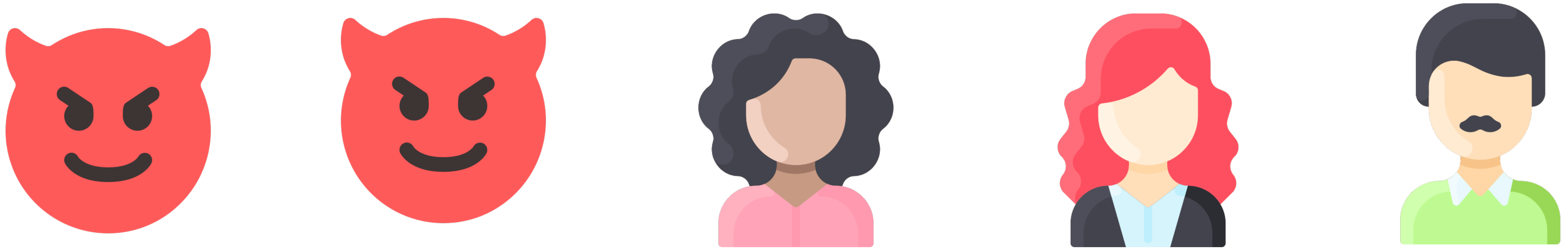
Maximum number of tolerable bad signers for

unforgeability:
$t - 1 = 2$

liveness:
$n - t = 2$

Honest Majority (Classic)

n = 5



t = 3

Maximum number of tolerable bad signers for

unforgeability:
t - 1 = 2

liveness:
n - t = 2

may be required in consensus systems anyway

Honest Minority

n = 5

t = 4

Maximum number of tolerable bad signers for

unforgeability:
t - 1 = 3

liveness:
n - t = 1

Honest Minority

n = 5

t = 4

Maximum number of tolerable bad signers for

unforgeability:
t - 1 = 3

liveness:
n - t = 1

no progress with 2 bad signers
but also no forgery

Full Threshold

n = 5

t = 5

Maximum number of tolerable bad signers for

unforgeability:
t - 1 = 4

liveness:
n - t = 0

Full Threshold

$n = 5$

$t = 5$

Maximum number of tolerable bad signers for

unforgeability:
$t - 1 = 4$

multi-signatures possible:
(non-interactive
key aggregation, no DKG)

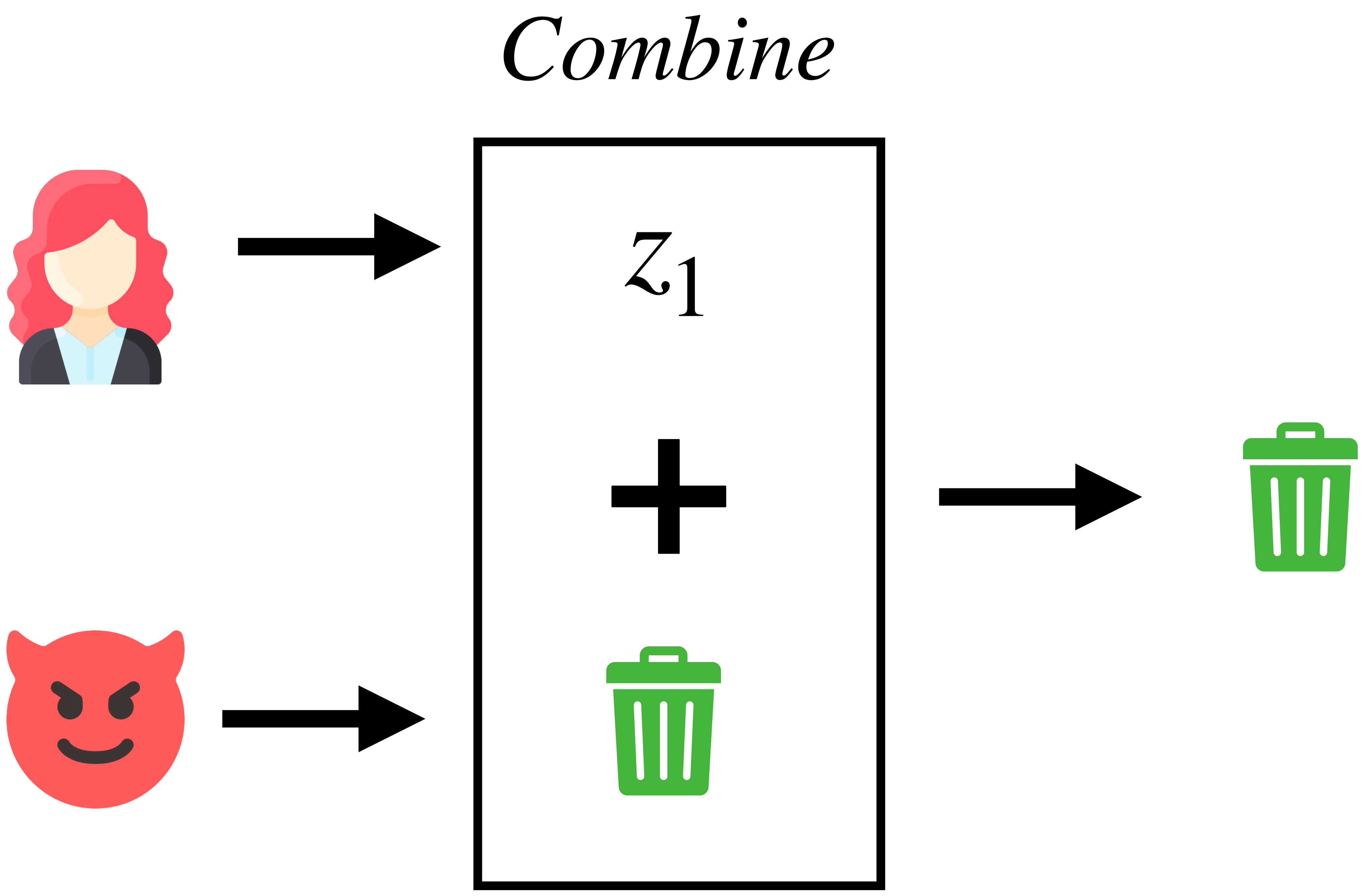
liveness:
$n - t = 0$

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

(required for liveness!)

FROST and Robustness

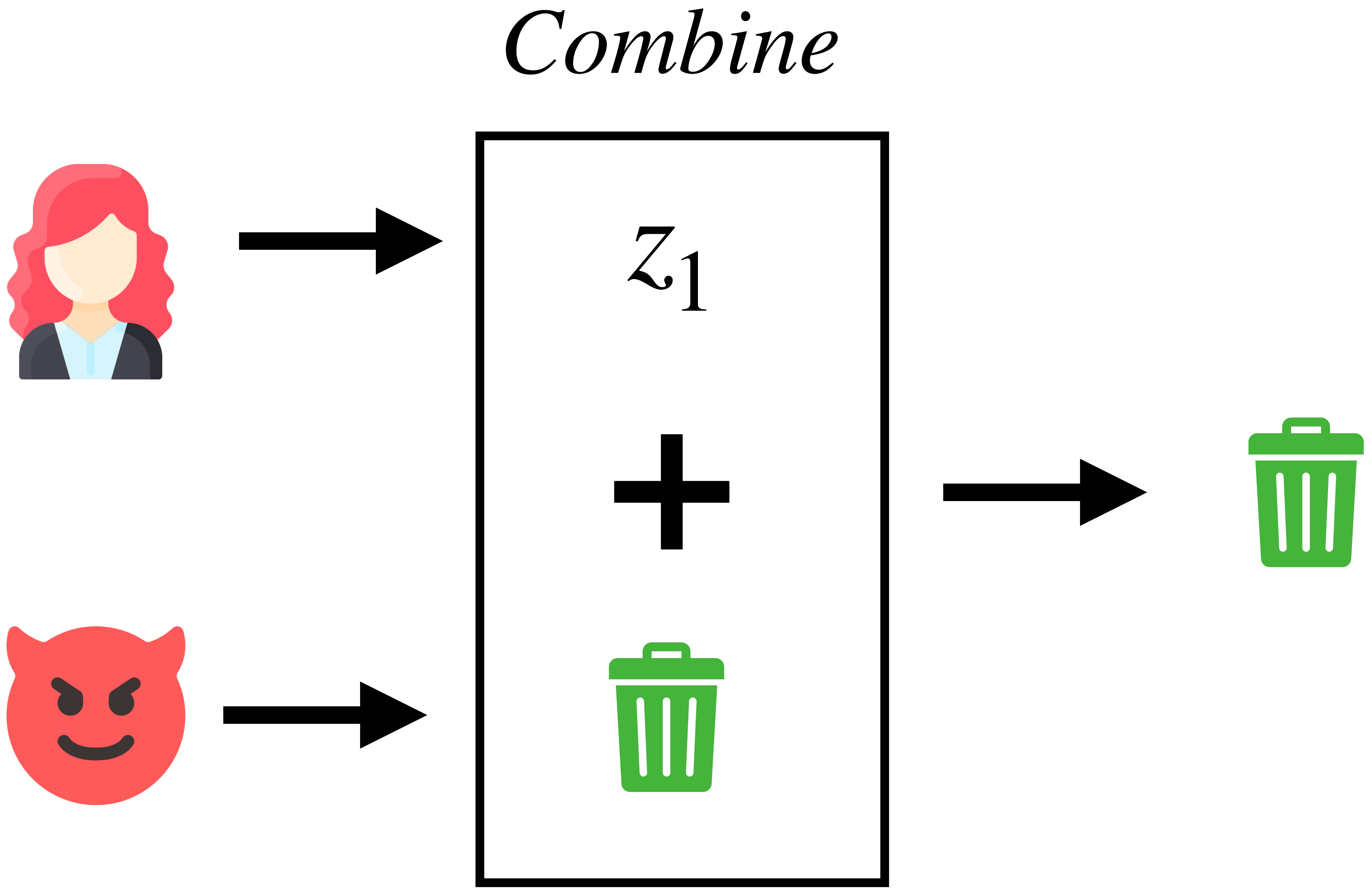
- FROST is **not** robust.



(2,3) Example

FROST and Robustness

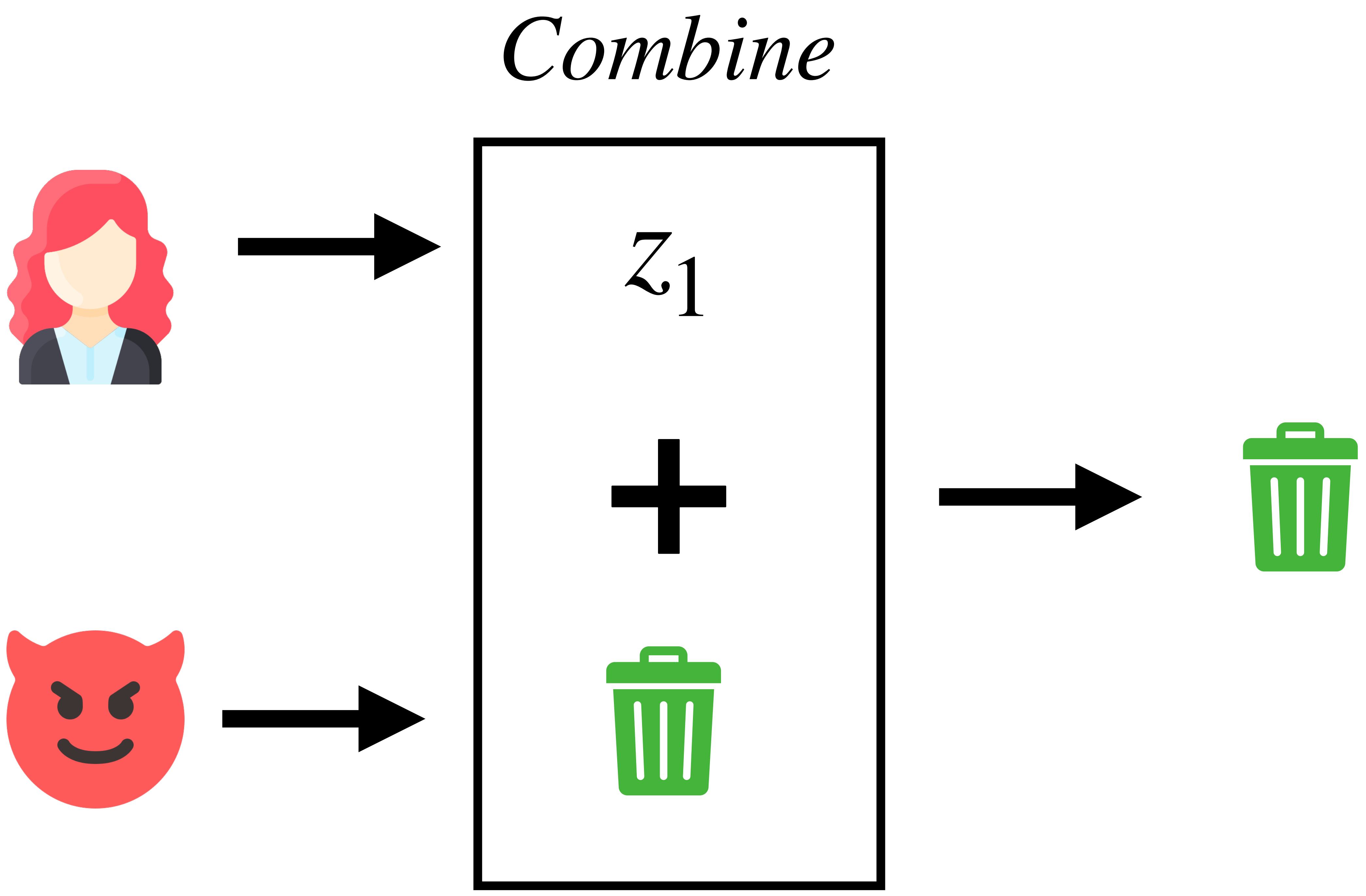
- FROST is **not** robust.
- If even one FROST signer issues garbage, the resulting signature is garbage



(2,3) Example

FROST and Robustness

- FROST is **not** robust.
- If even one FROST signer issues garbage, the resulting signature is garbage
- Then the protocol must be re-run with a different subset of signers.



(2,3) Example

ROAST: Making FROST Robust

ROAST: Making FROST Robust

ROAST: Making FROST Robust

- ROAST is a wrapper picks subsets in a clever way
- At most $n - t + 1$ FROST runs necessary
- Resulting protocol is robust and asynchronous (no timeouts)

Standardization and Deployment

Standardization

IRTF

CFRG
Internet-Draft
Intended status: Informational
Expires: 28 July 2023

D. Connolly
Zcash Foundation
C. Komlo
University of Waterloo, Zcash Foundation
I. Goldberg
University of Waterloo
C. A. Wood
Cloudflare
24 January 2023

Two-Round Threshold Schnorr Signatures with FROST
draft-irtf-cfrg-frost-12

https://datatracker.ietf.org/doc/draft-irtf-cfrg-frost/

Standardization

```
ZIP: 312
Title: FROST for Spend Authorization Signatures
Owners: Conrado Gouvea <conrado@zfnd.org>
        Chelsea Komlo <ckomlo@uwaterloo.ca>
        Deirdre Connolly <deirdre@zfnd.org>
Status: Draft
Category: Wallet
Created: 2022-08-dd
License: MIT
Discussions-To: <https://github.com/zcash/zips/issues/382>
Pull-Request: <https://github.com/zcash/zips/pull/662>
```

https://github.com/ZcashFoundation/zips/blob/zip-frost/zip-0312.rst

Standardization

```
BIP: 327
Title: MuSig2 for BIP340-compatible Multi-Signatures
Author: Jonas Nick <jonasd.nick@gmail.com>
        Tim Ruffing <crypto@timruffing.de>
        Elliott Jin <elliott.jin@gmail.com>
Status: Draft
License: BSD-3-Clause
Type: Informational
Created: 2022-03-22
```

https://github.com/bitcoin/bips/blob/master/bip-0327.mediawiki

Standardization

NIST COMPUTER SECURITY RESOURCE CENTER CSRC

NISTIR 8214C (Draft)

NIST First Call for Multi-Party Threshold Schemes

Date Published: January 25, 2023

Comments Due: April 10, 2023

Email Comments to: nistir-8214C-comments@nist.gov

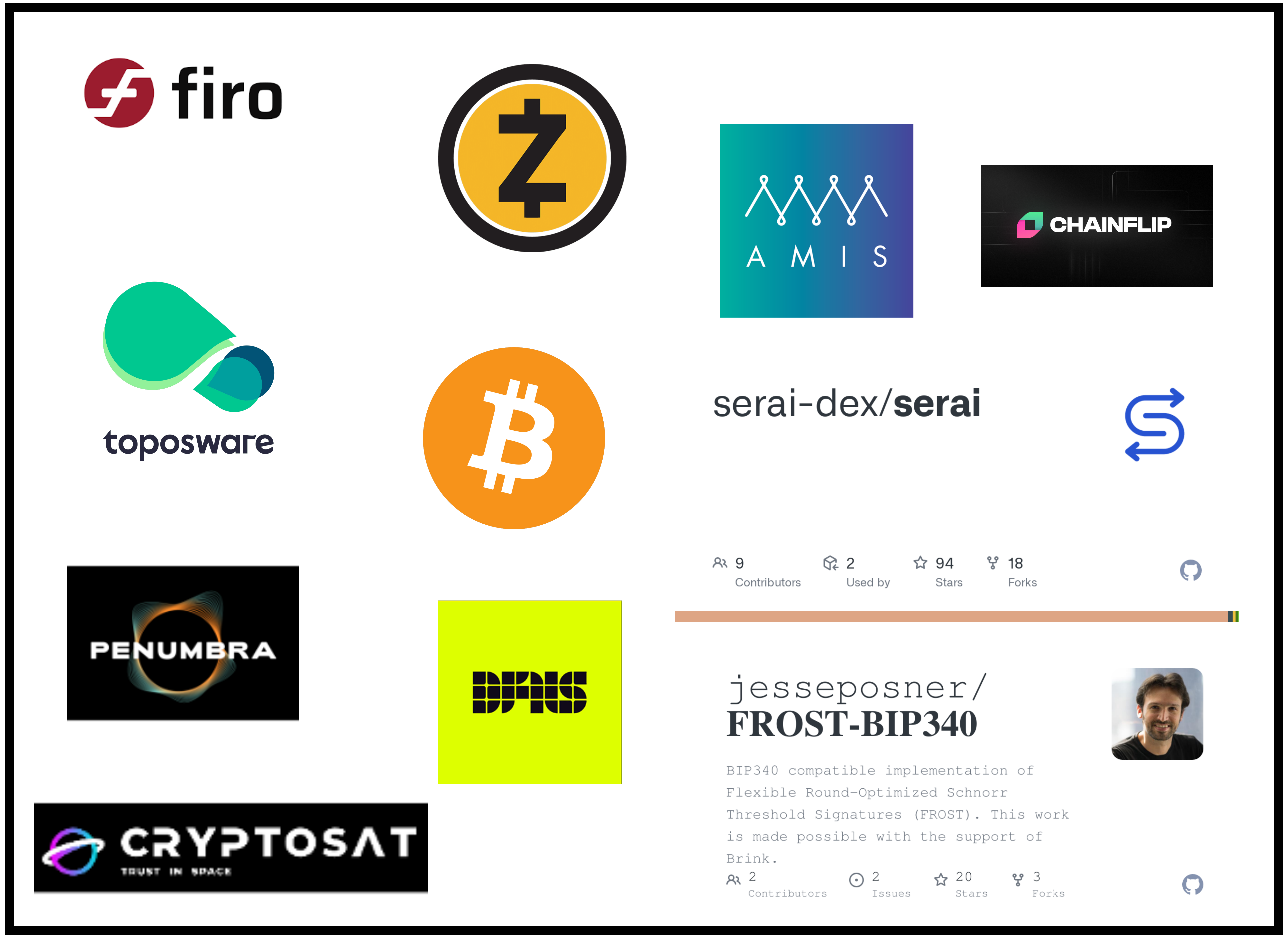
Author(s)

Luís T. A. N. Brandão (Strativia), Rene Peralta (NIST)

https://csrc.nist.gov/publications/detail/nistir/8214c/draft

FROST and MuSig2 in Practice, Today

FROST



serai-dex/**serai**

9 Contributors · 2 Used by · 94 Stars · 18 Forks

jesseposner/
FROST-BIP340

BIP340 compatible implementation of Flexible Round-Optimized Schnorr Threshold Signatures (FROST). This work is made possible with the support of Brink.

2 Contributors · 2 Issues · 20 Stars · 3 Forks

MuSig2

BlockstreamResearch / **secp256k1-zkp**

LLFourn / **secp256kfun**

input-output-hk / **musig2**

bitcoin/bips

#1372 **Add BIP MuSig2**

BlockstreamResearch/secp256k1-zkp

#223 **musig: Update to BIP v1.0.0-rc.4 (Check pubnonce in NonceGe...**

0 comments · 3 reviews · 3 files · +15 −7

real-or-random • March 3, 2023 · 1 commit

From Practice to Theory: What open problems exist?

Efficient Deterministic Signatures

(Single-Party) EdDSA Signature Scheme

To generate a key pair:

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

(Single-Party) EdDSA Signature Scheme

To generate a key pair:

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

To sign a message m:

$$r \leftarrow H(m, sk) \ ; \ R \leftarrow g^{r}$$

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

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

(Single-Party) EdDSA Signature Scheme

To generate a key pair:

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

To sign a message m:

$$\boxed{r \leftarrow H(m, sk)} \,;\;\; R \leftarrow g^{r}$$

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

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

(Single-Party) EdDSA Signature Scheme

To generate a key pair:

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

To sign a message m:

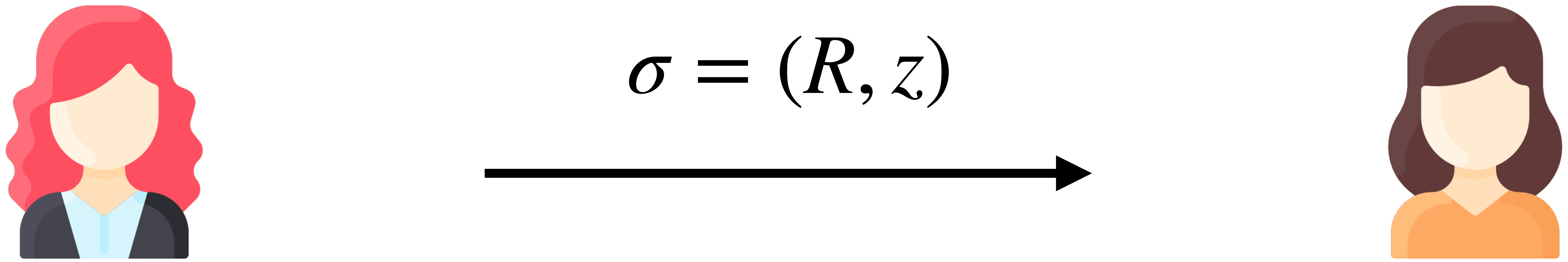
$$\boxed{r \leftarrow H(m, sk)} \,;\ R \leftarrow g^{r}$$

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

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

Helps prevent issues arising from bad randomness.

(Single-Party) EdDSA Signature Scheme



$\sigma = (R, z)$

To generate a key pair:

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

To sign a message m:

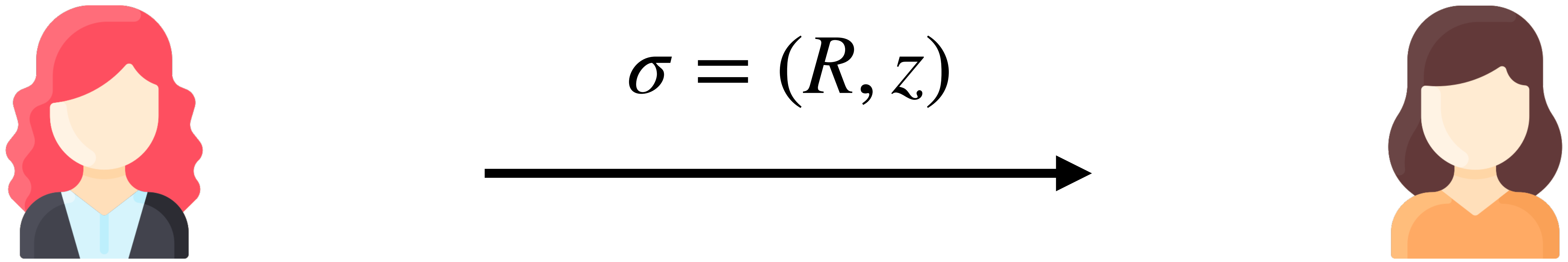
$$\boxed{r \leftarrow H(m, sk)}\ ;\ \ R \leftarrow g^{r}$$

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

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

Helps prevent issues arising from bad randomness.

(Single-Party) EdDSA Signature Scheme



$$\sigma = (R, z)$$

To generate a key pair:

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

To sign a message m:

$$\boxed{r \leftarrow H(m, sk)} \; ; \;\; R \leftarrow g^{r}$$

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

$$z \leftarrow r + csk$$

Helps prevent issues arising from bad randomness.

To verify (PK, σ, m):

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

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

output accept/reject

Naively applying EdDSA-style determinism to randomized multi-party Schnorr schemes is not secure!

Naively applying EdDSA-style determinism to randomized multi-party Schnorr schemes is not secure!

Deterministic multi-party Schnorr schemes exist, but are performance-intensive. [NRSW20, GKMN21]

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Corrupt

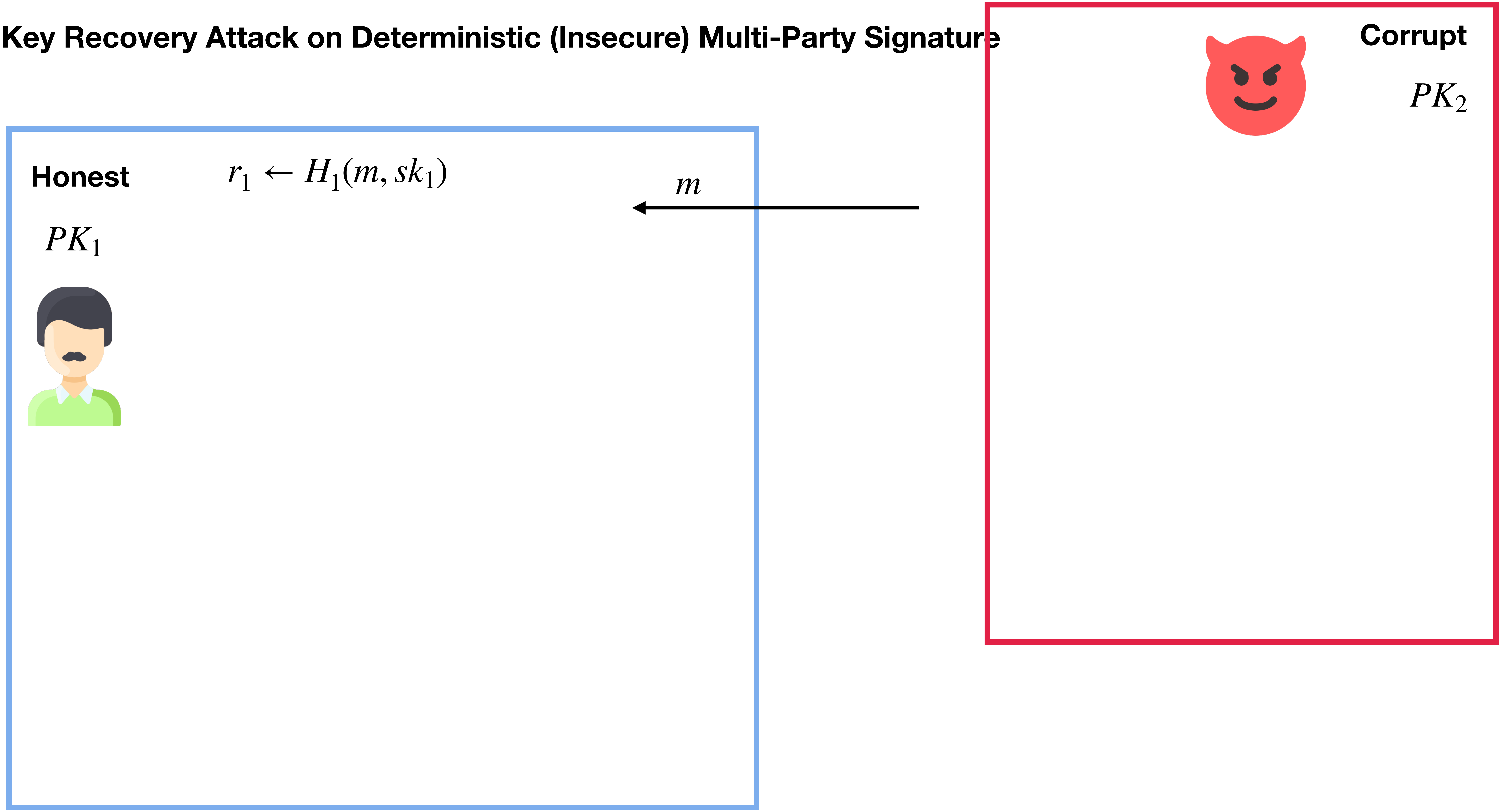
PK_2

Honest

PK_1

$\xleftarrow{m}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Corrupt

PK_2

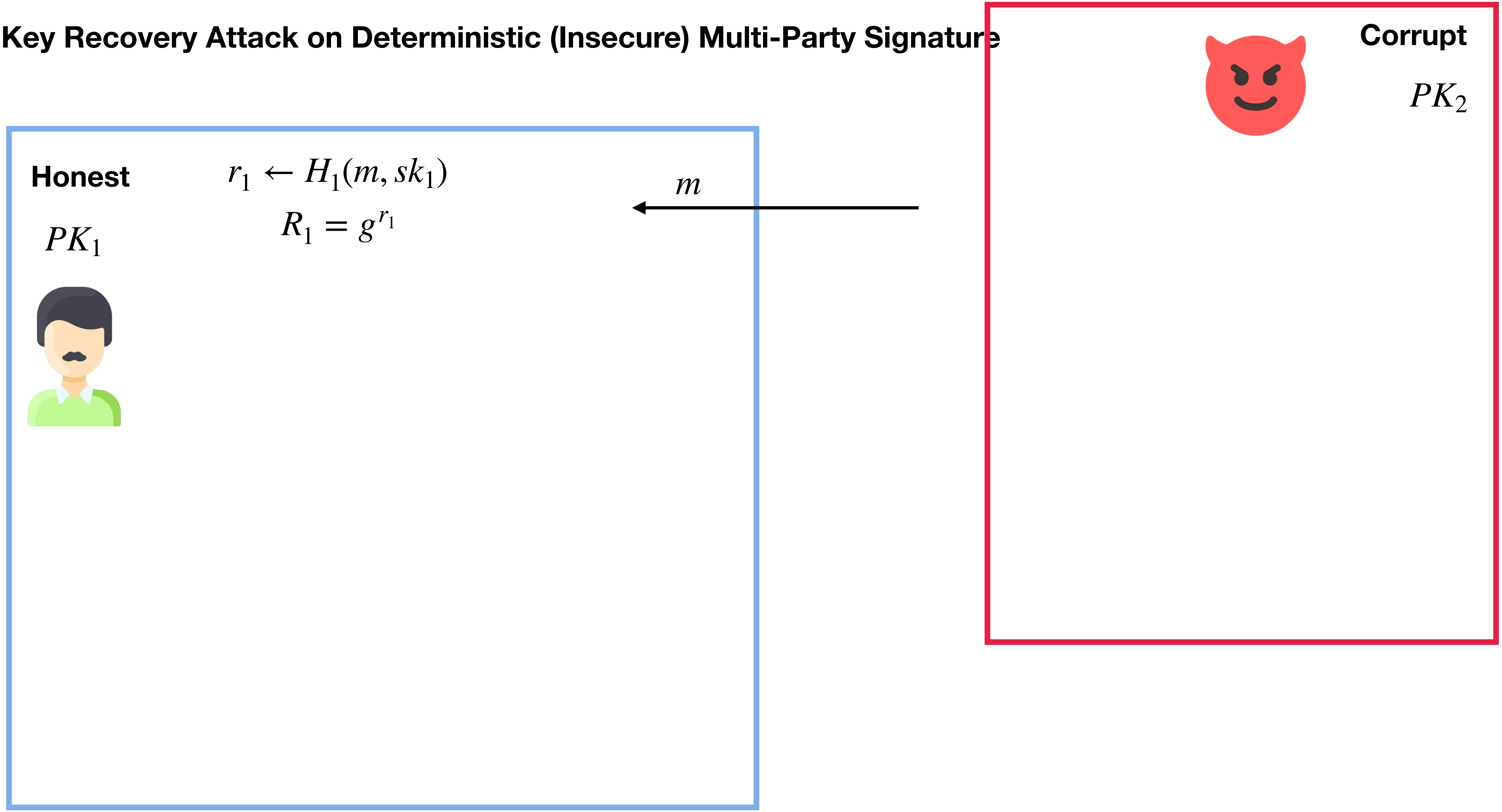
m

Honest

$r_1 \leftarrow H_1(m, sk_1)$

PK_1

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

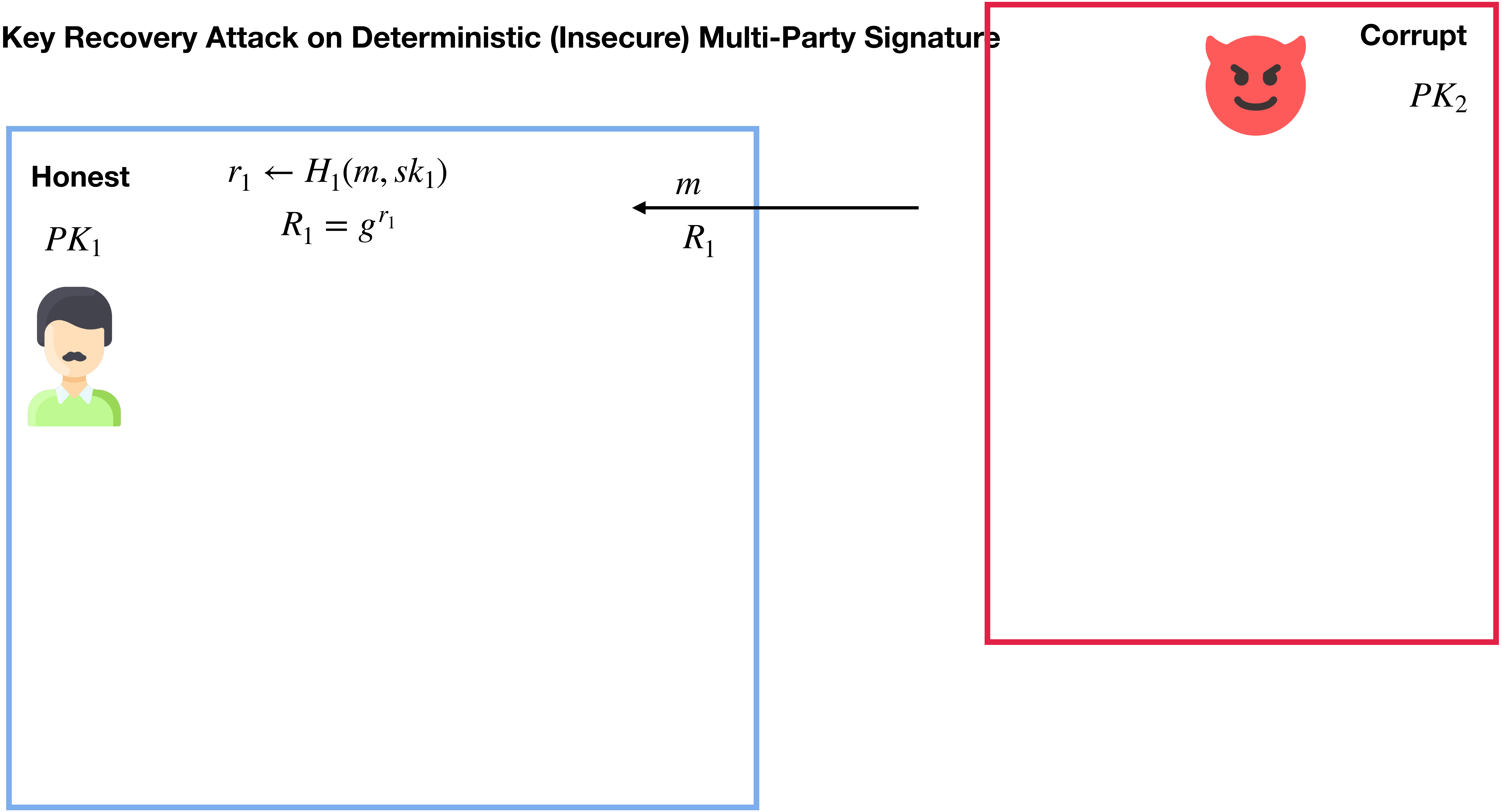
$R_1 = g^{r_1}$

m

Corrupt

PK_2

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

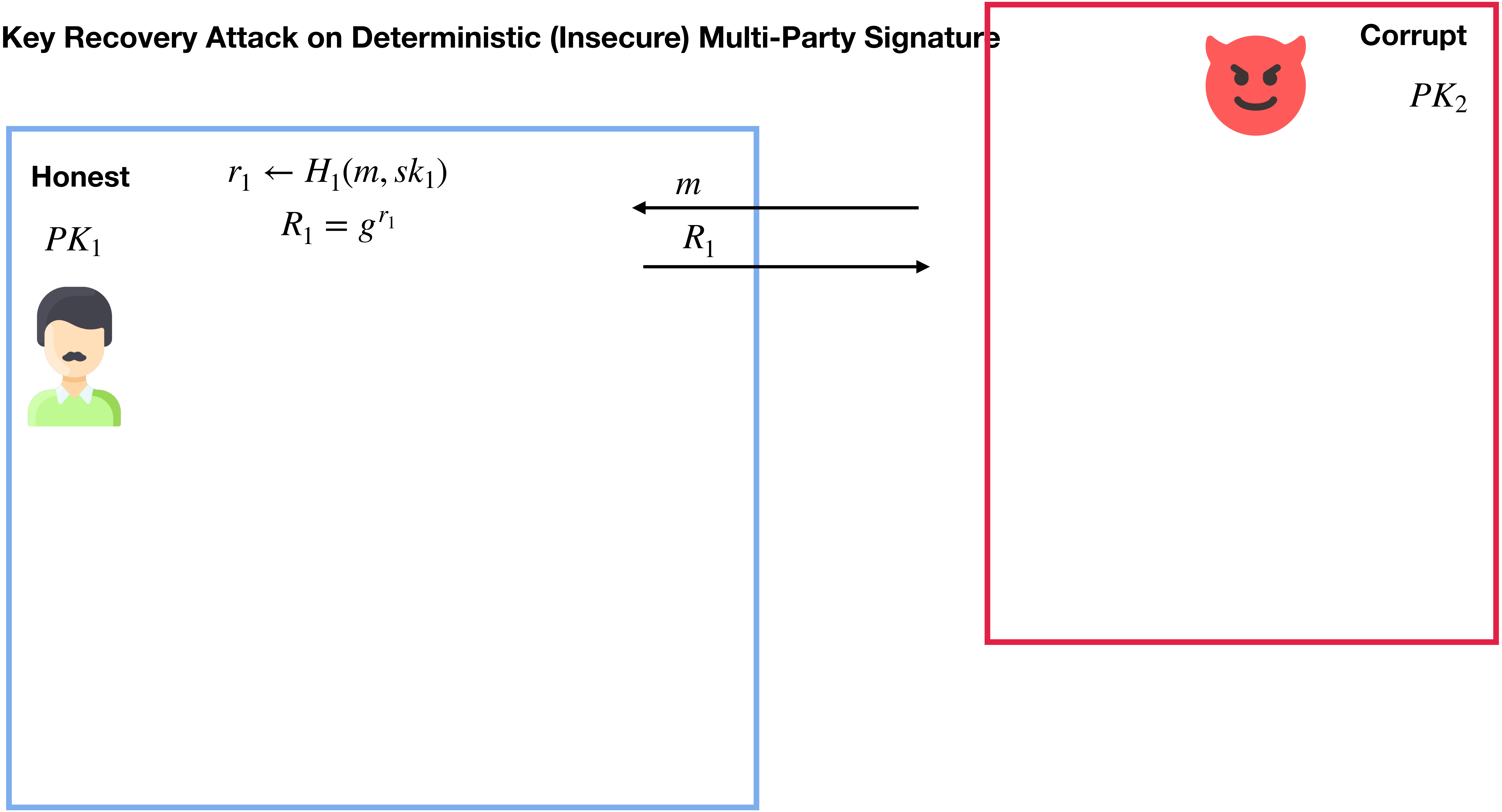
m

R_1

Corrupt

PK_2

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

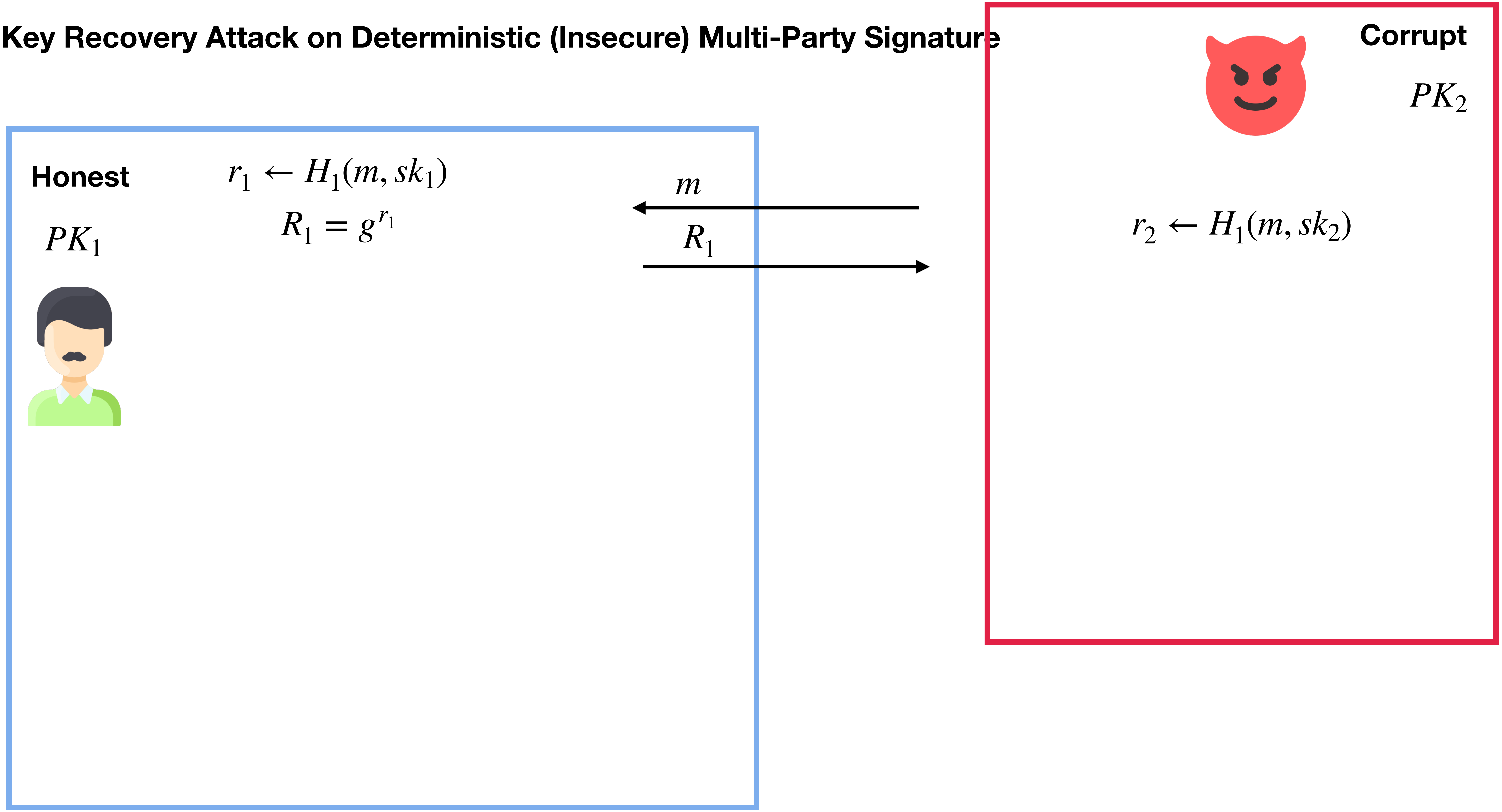
$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

Corrupt

PK_2

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$\xleftarrow{m}$

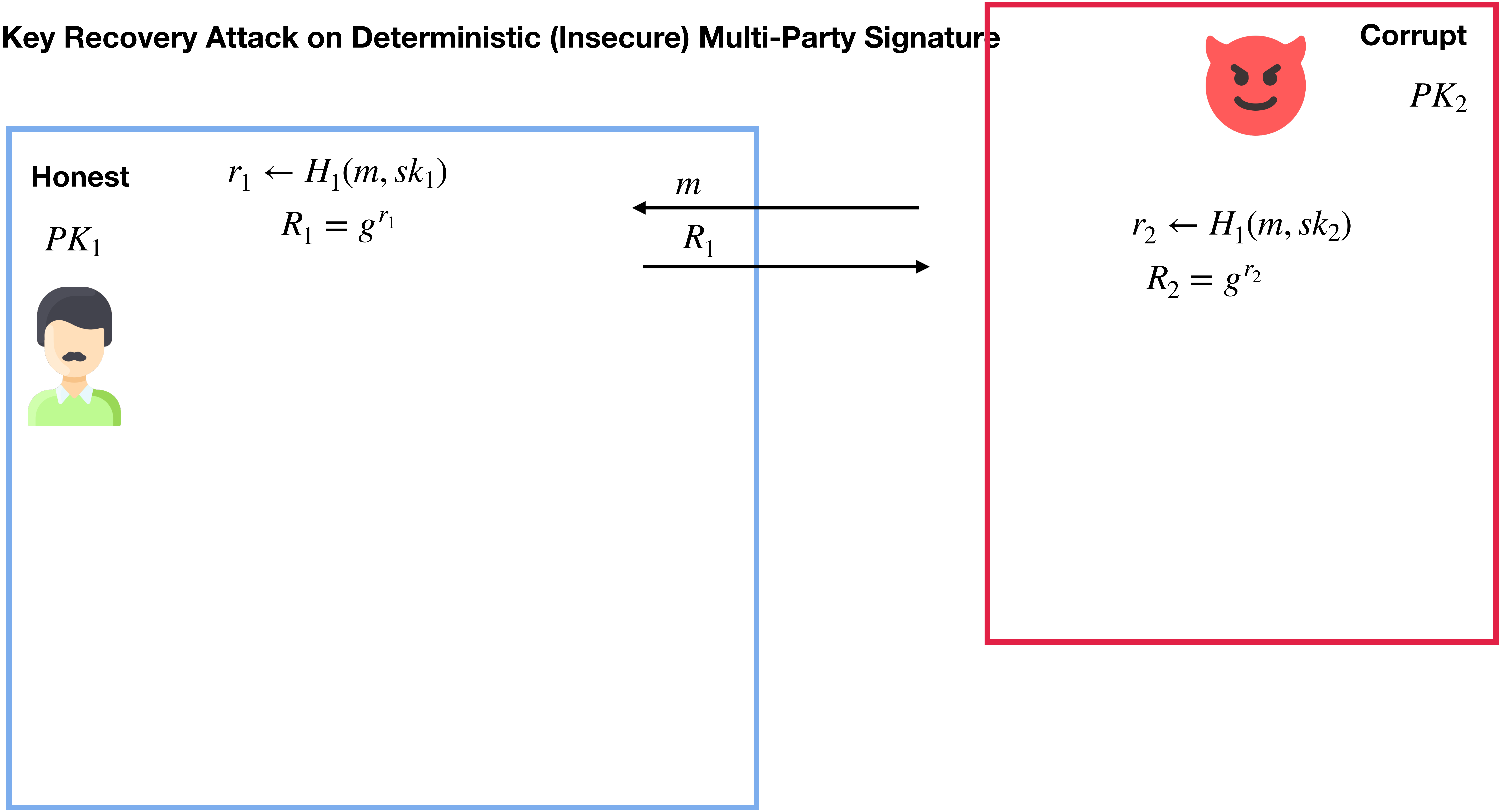
$\xrightarrow{R_1}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

m

R_1

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

m

R_1

R_2

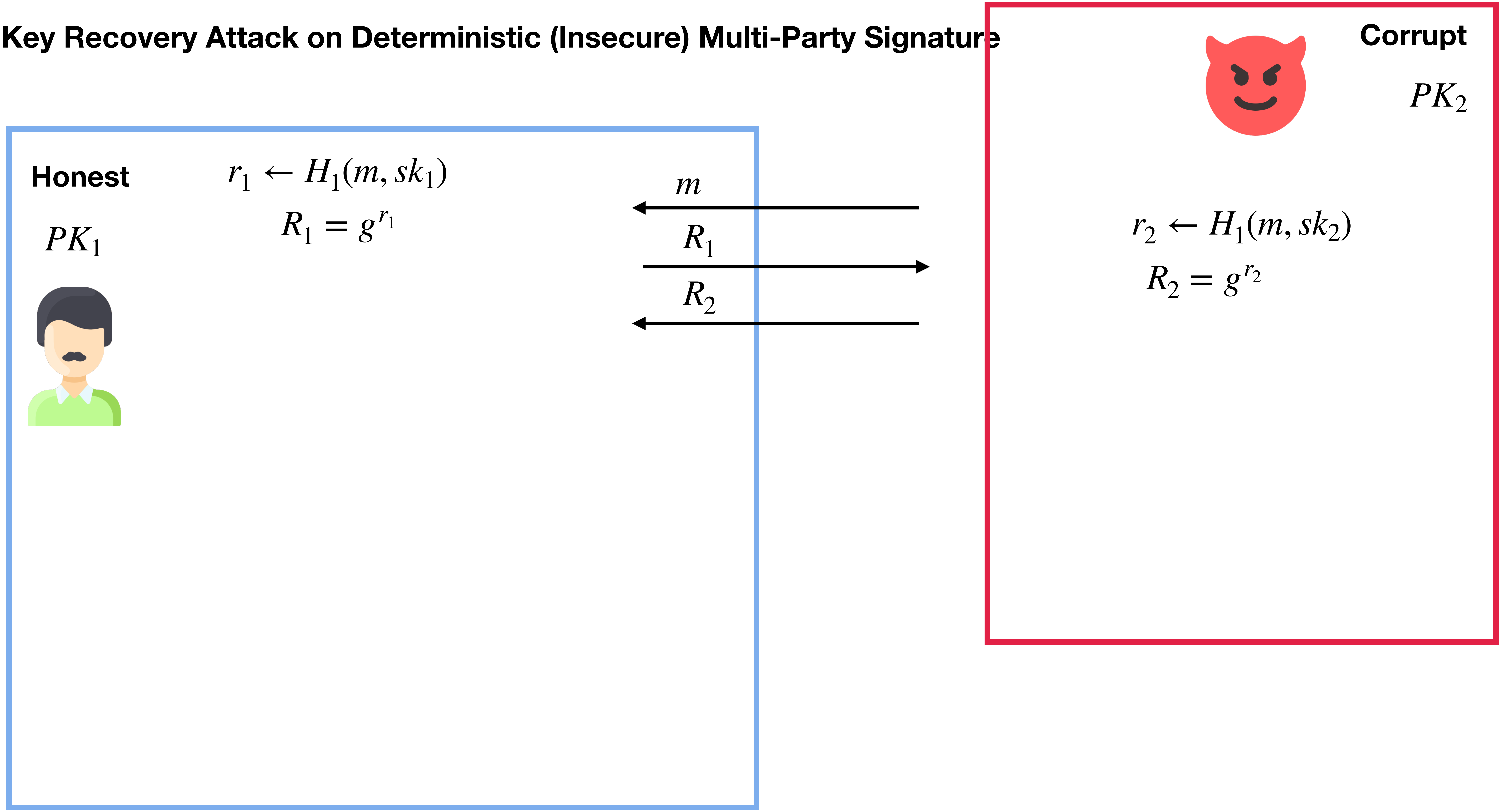
Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

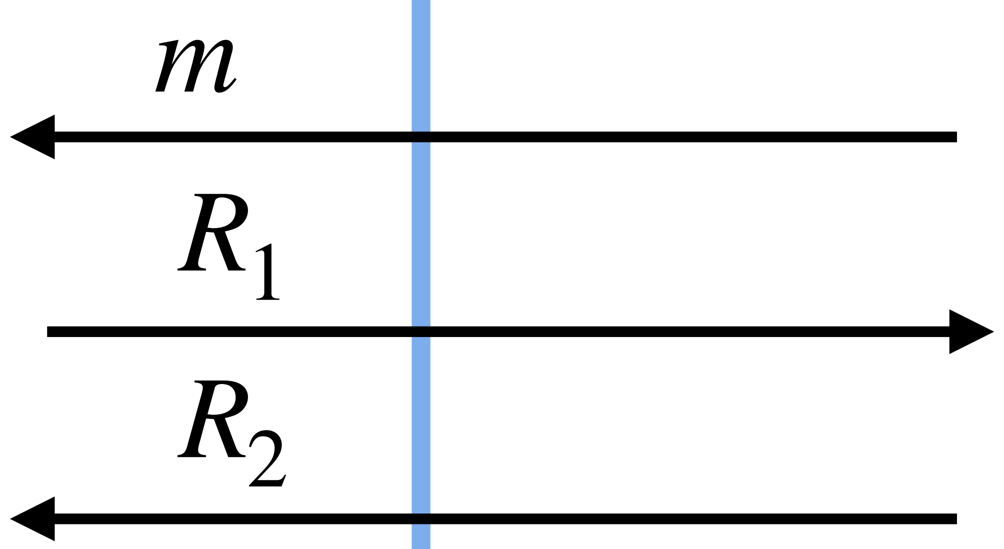
Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$



m

R_1

R_2

Corrupt

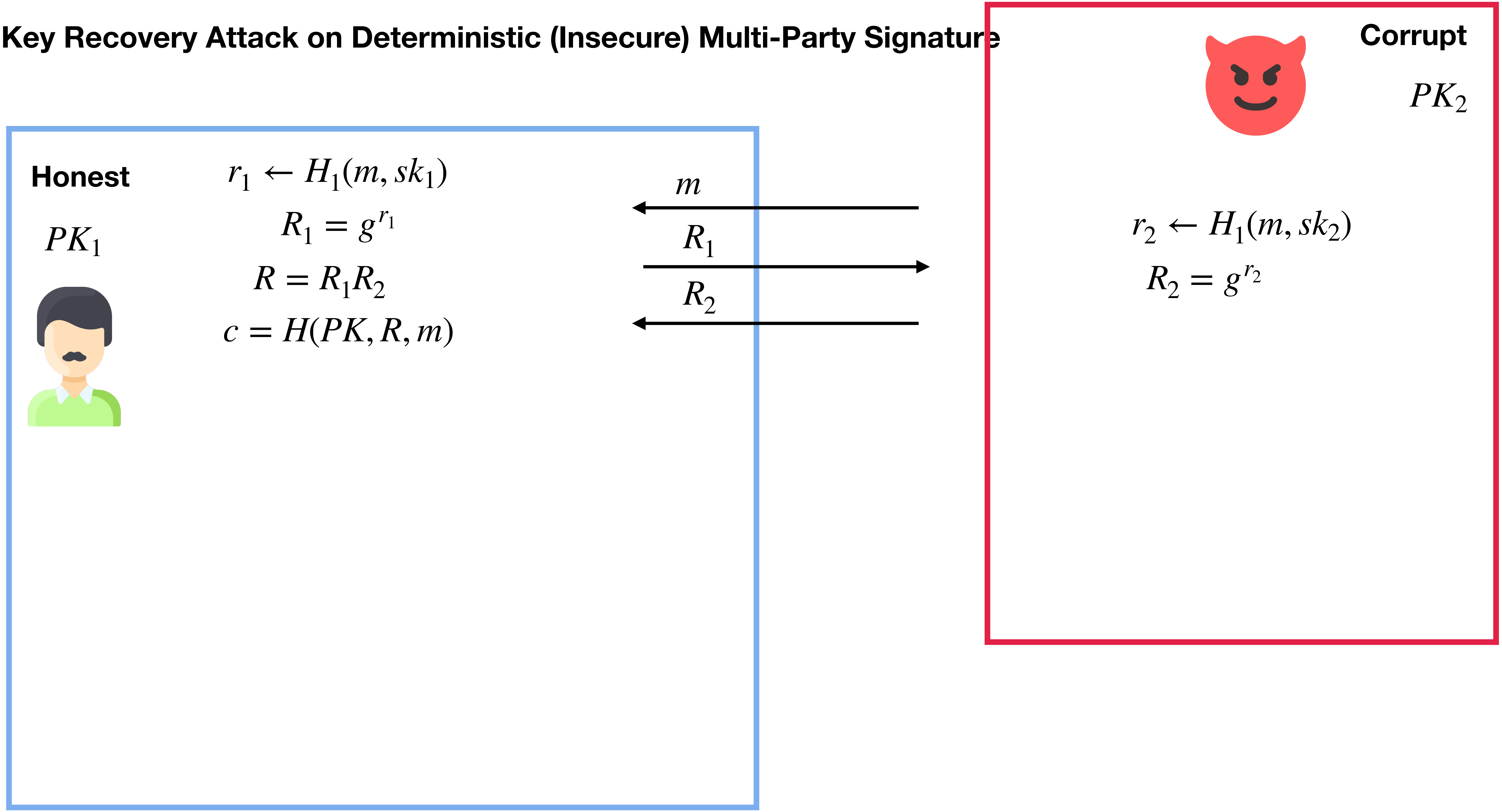
😈

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

m

R_1

R_2

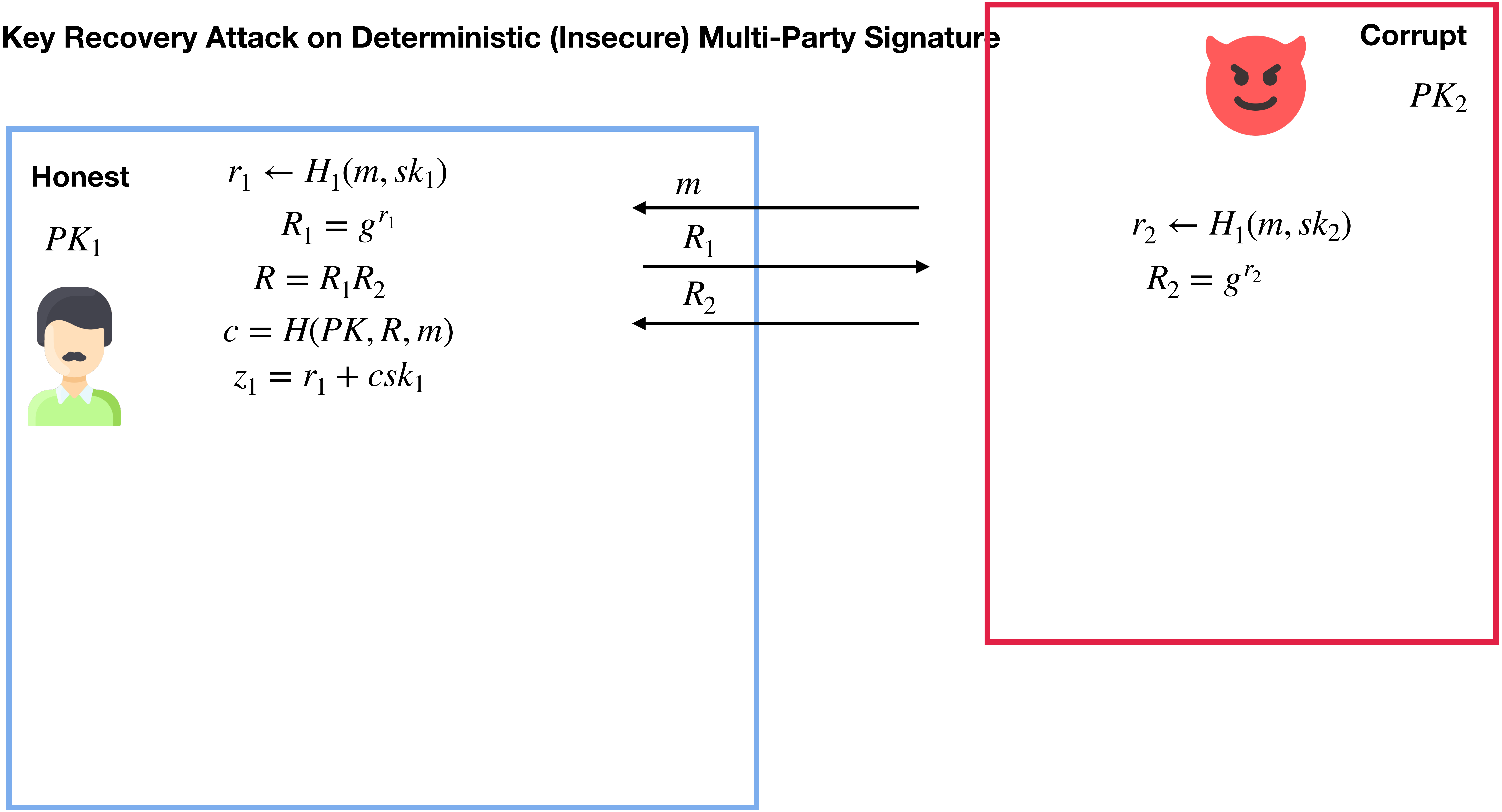
Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$$r_1 \leftarrow H_1(m, sk_1)$$

$$R_1 = g^{r_1}$$

$$R = R_1 R_2$$

$$c = H(PK, R, m)$$

$$z_1 = r_1 + c sk_1$$

$$\xleftarrow{m}$$

$$\xrightarrow{R_1}$$

$$\xleftarrow{R_2}$$

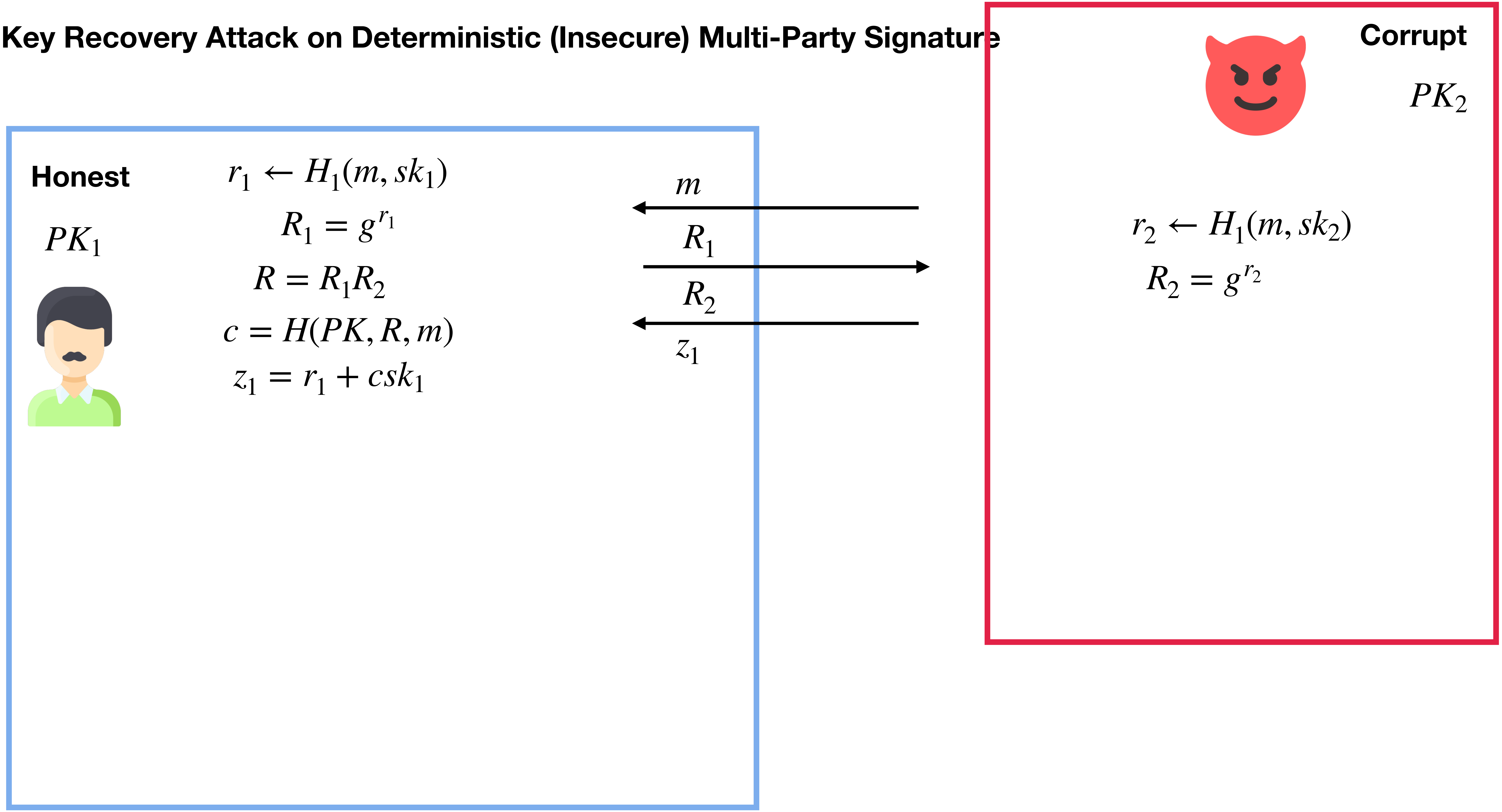
Corrupt

PK_2

$$r_2 \leftarrow H_1(m, sk_2)$$

$$R_2 = g^{r_2}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

m

R_1

R_2

z_1

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

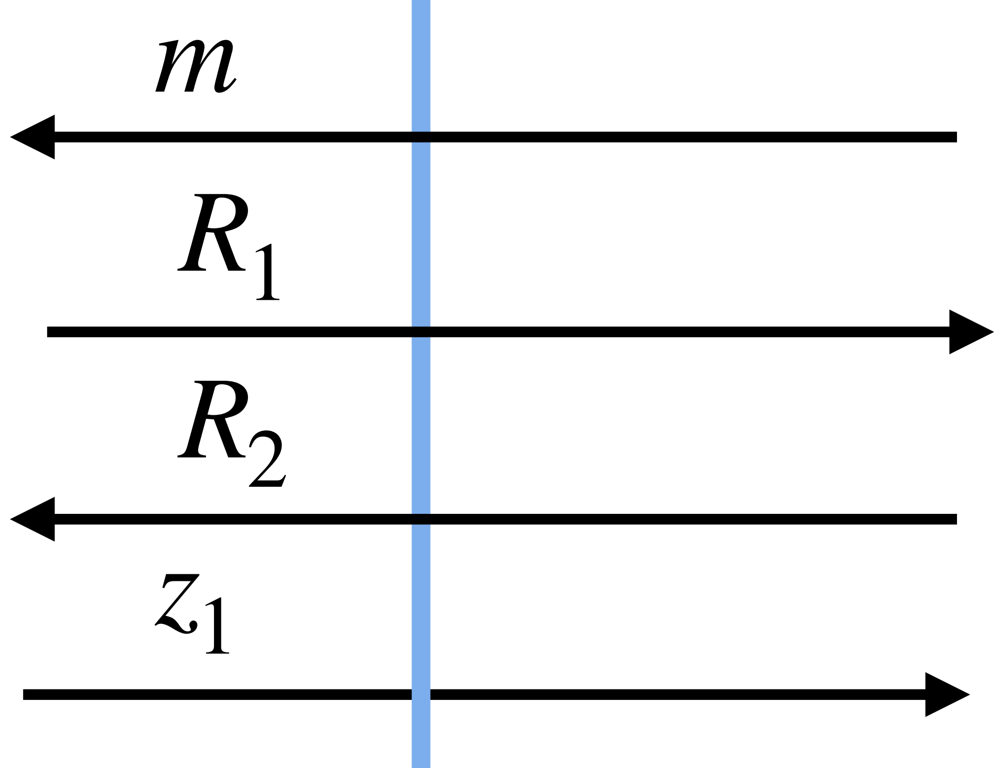
$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$



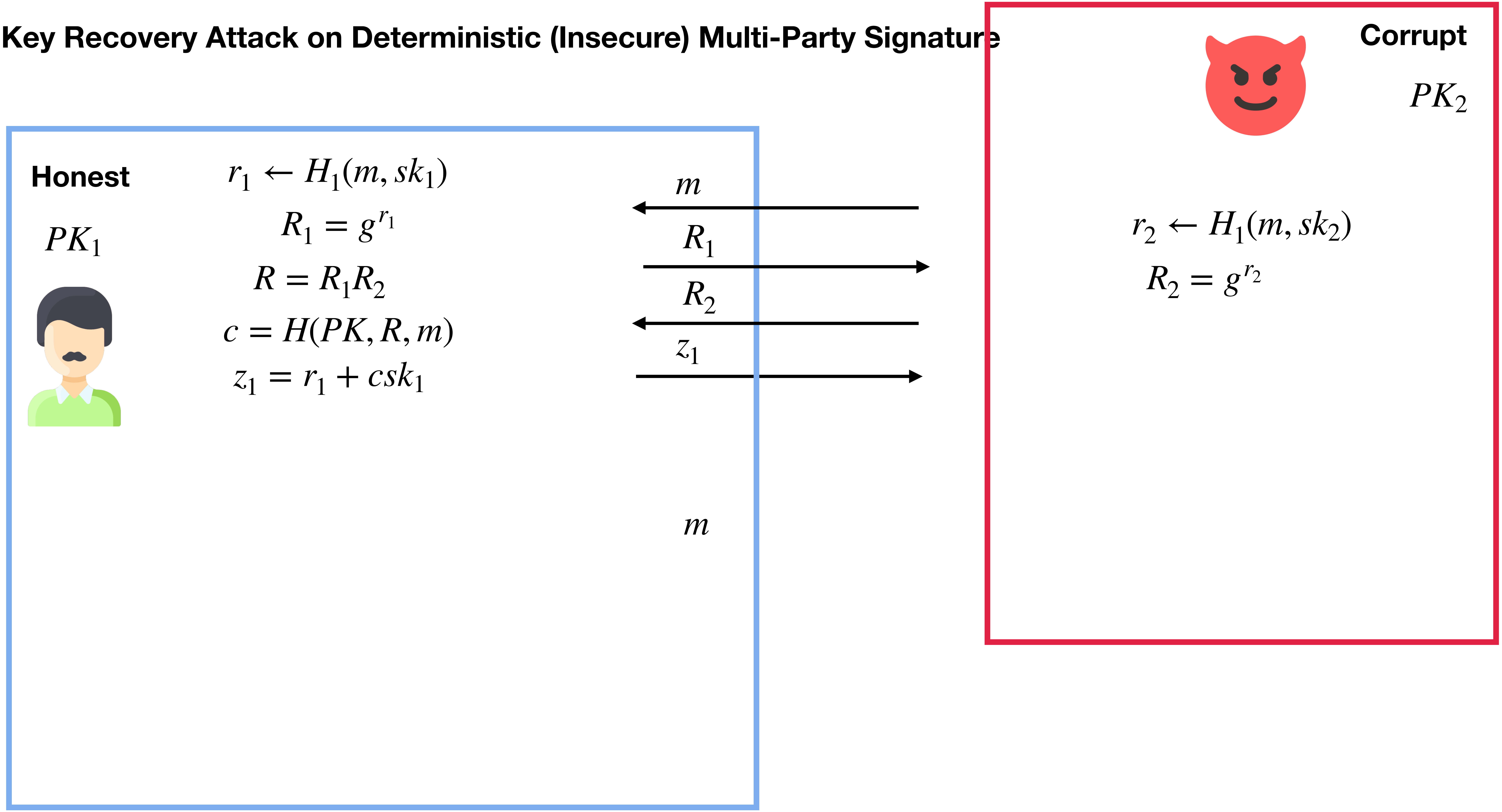
Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

$\xleftarrow{\quad R_2 \quad}$

$\xrightarrow{\quad z_1 \quad}$

m

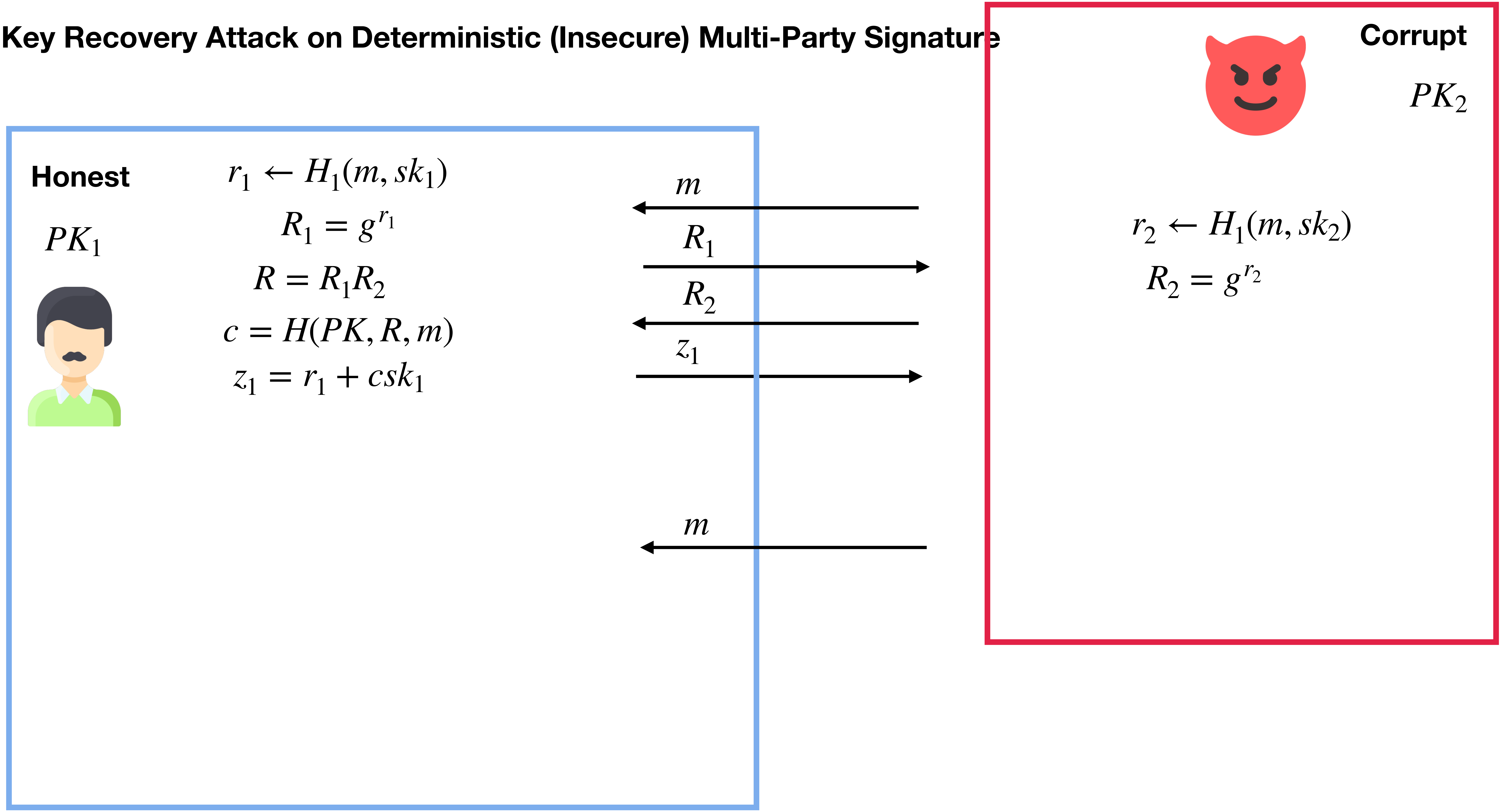
Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$

m

R_1

R_2

z_1

m

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

$r_1 \leftarrow H_1(m, sk_1)$

$\xleftarrow{m}$

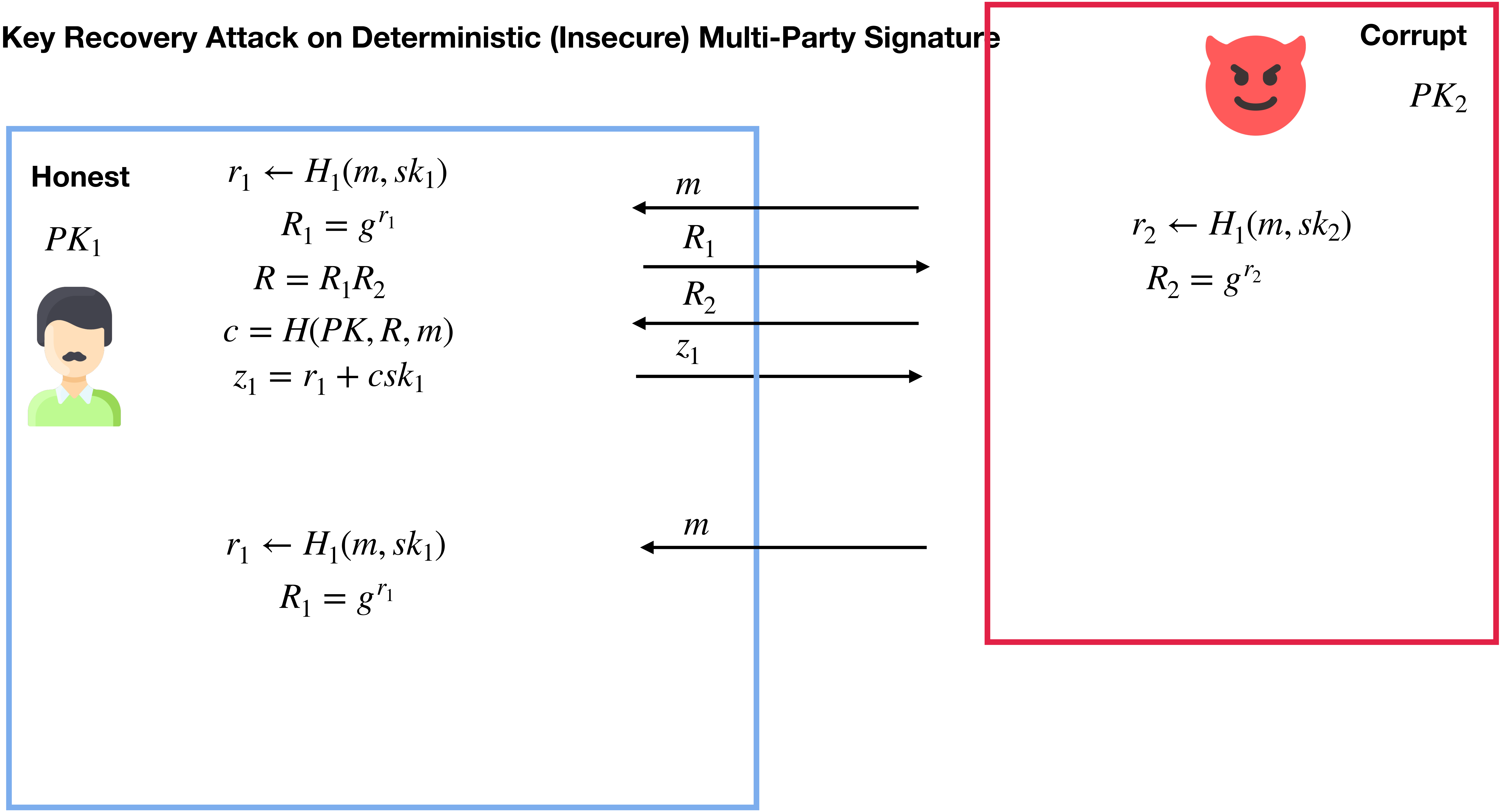
Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$

m

R_1

R_2

z_1

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

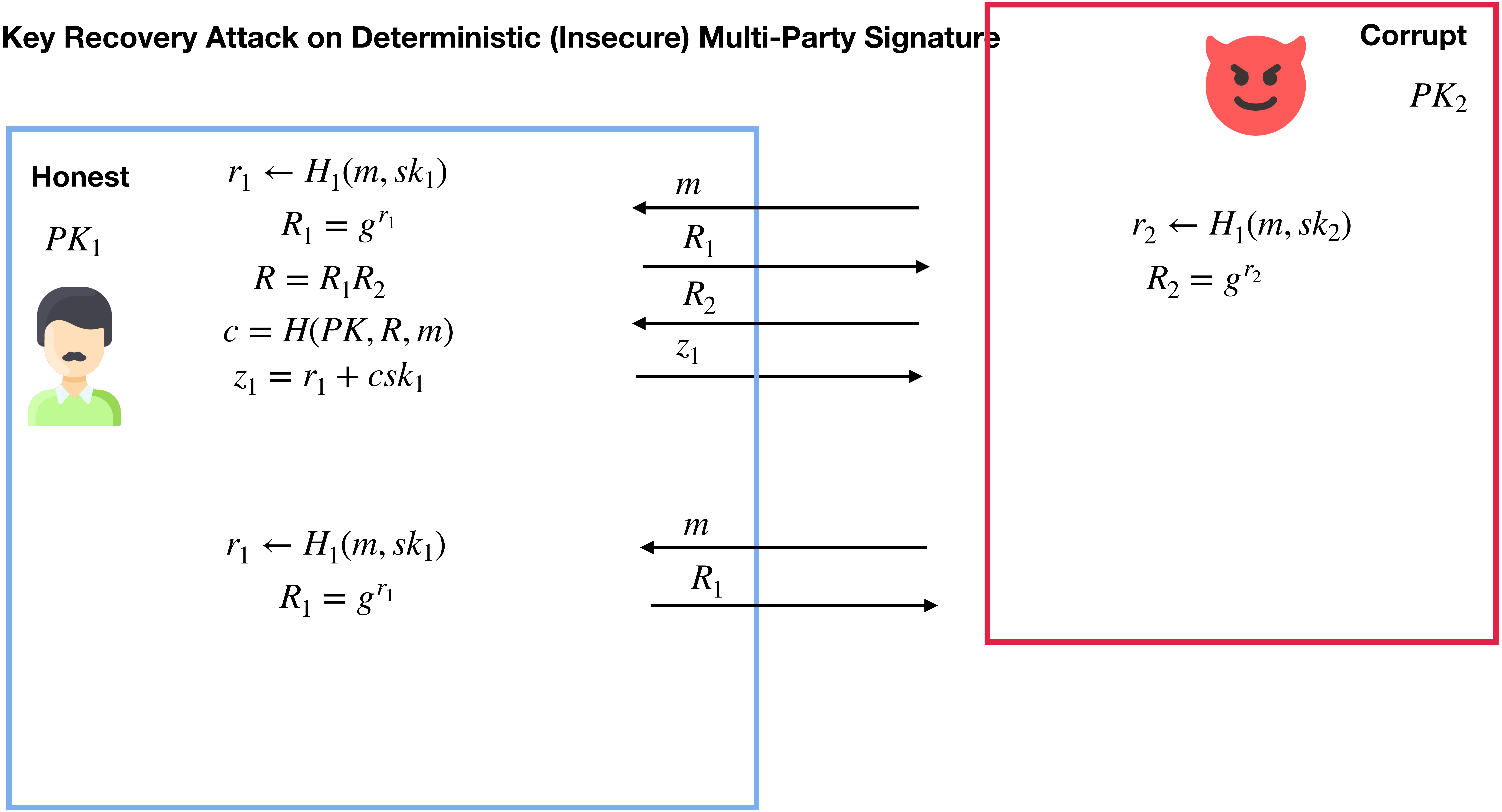
$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

m

R_1

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Corrupt

PK_2

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

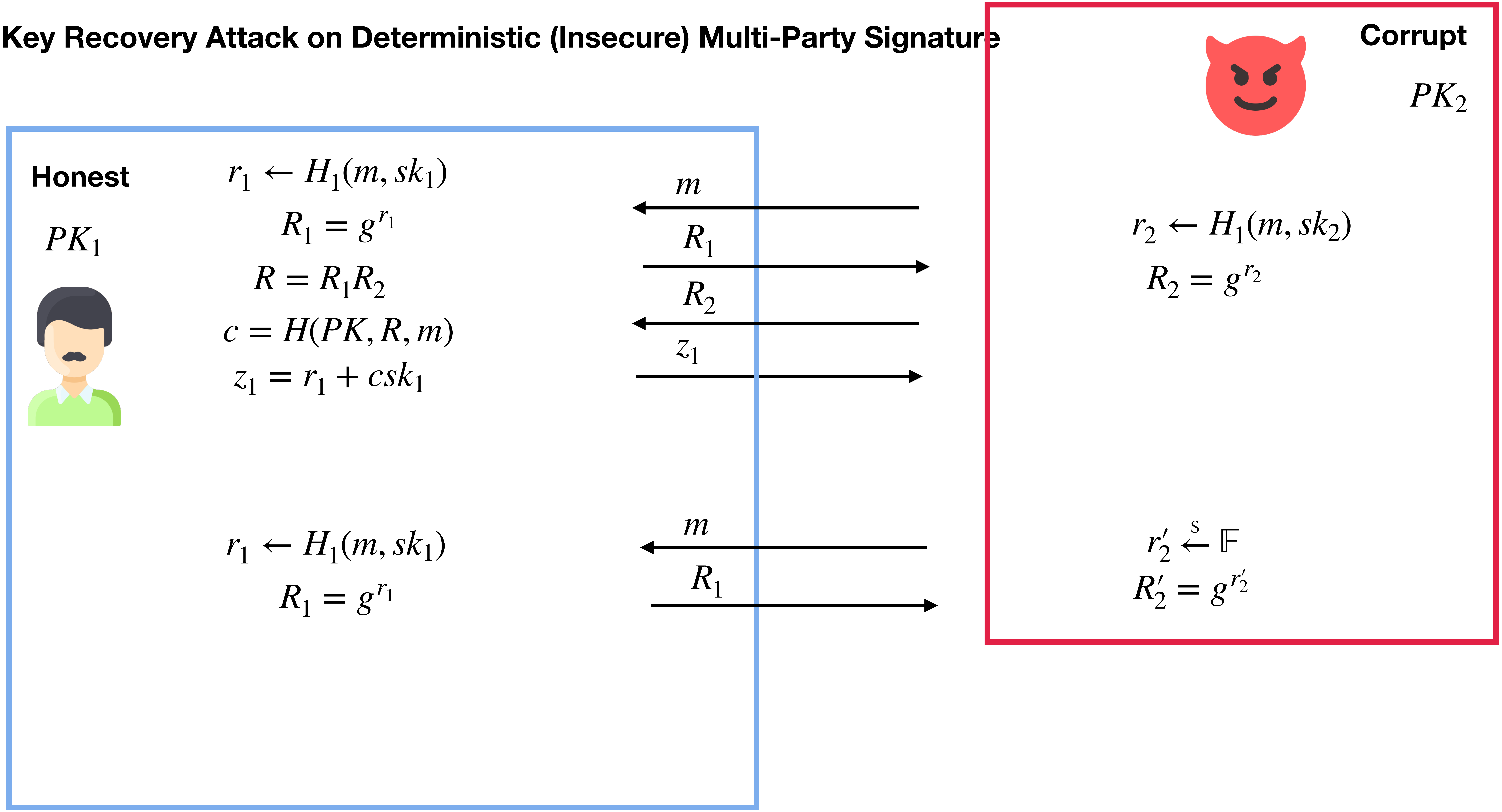
$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

$\xleftarrow{\quad R_2 \quad}$

$\xrightarrow{\quad z_1 \quad}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

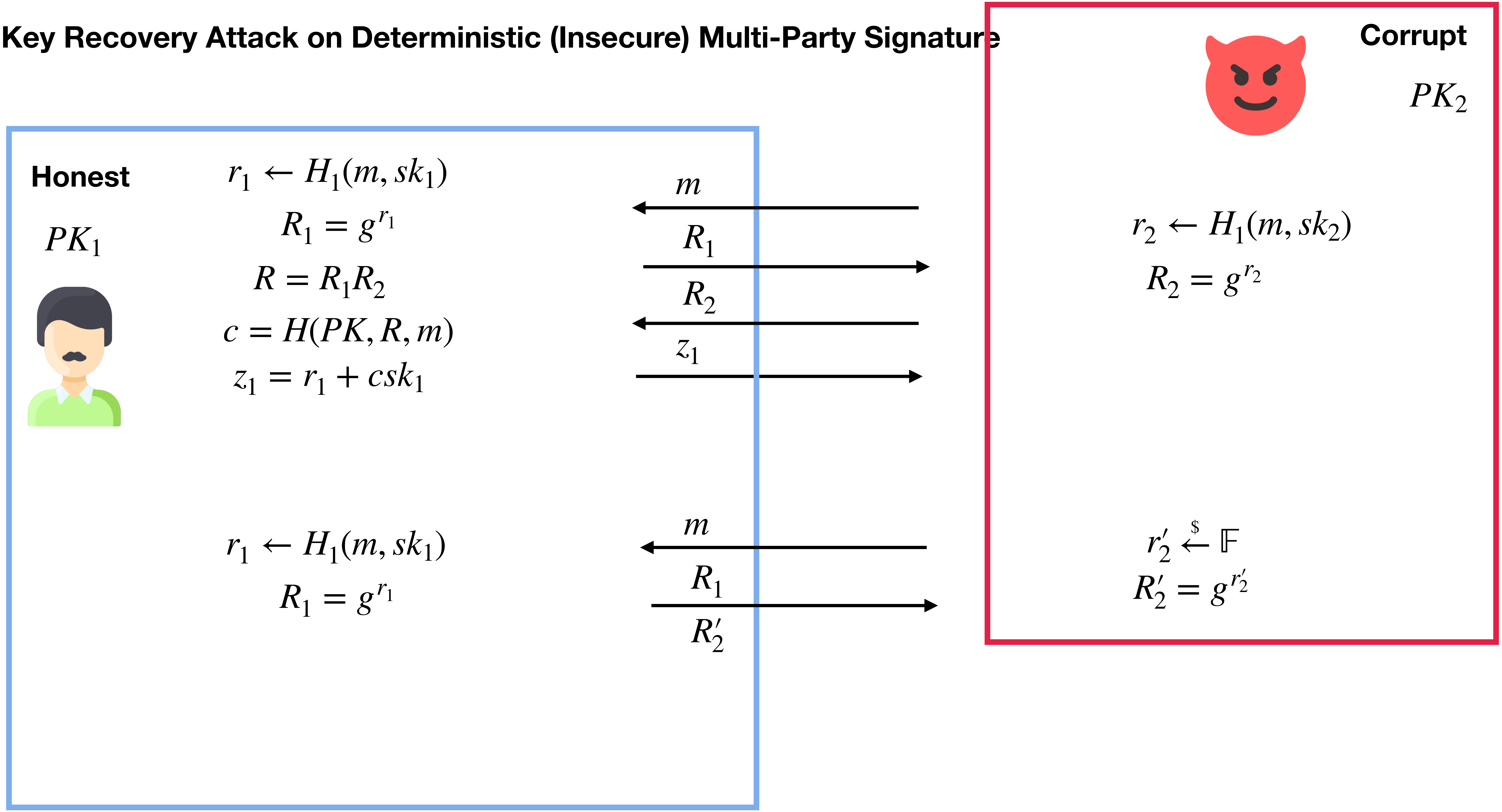
$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

$r'_2 \xleftarrow{\$} \mathbb{F}$

$R'_2 = g^{r'_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

$\xleftarrow{\quad R_2 \quad}$

$\xrightarrow{\quad z_1 \quad}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

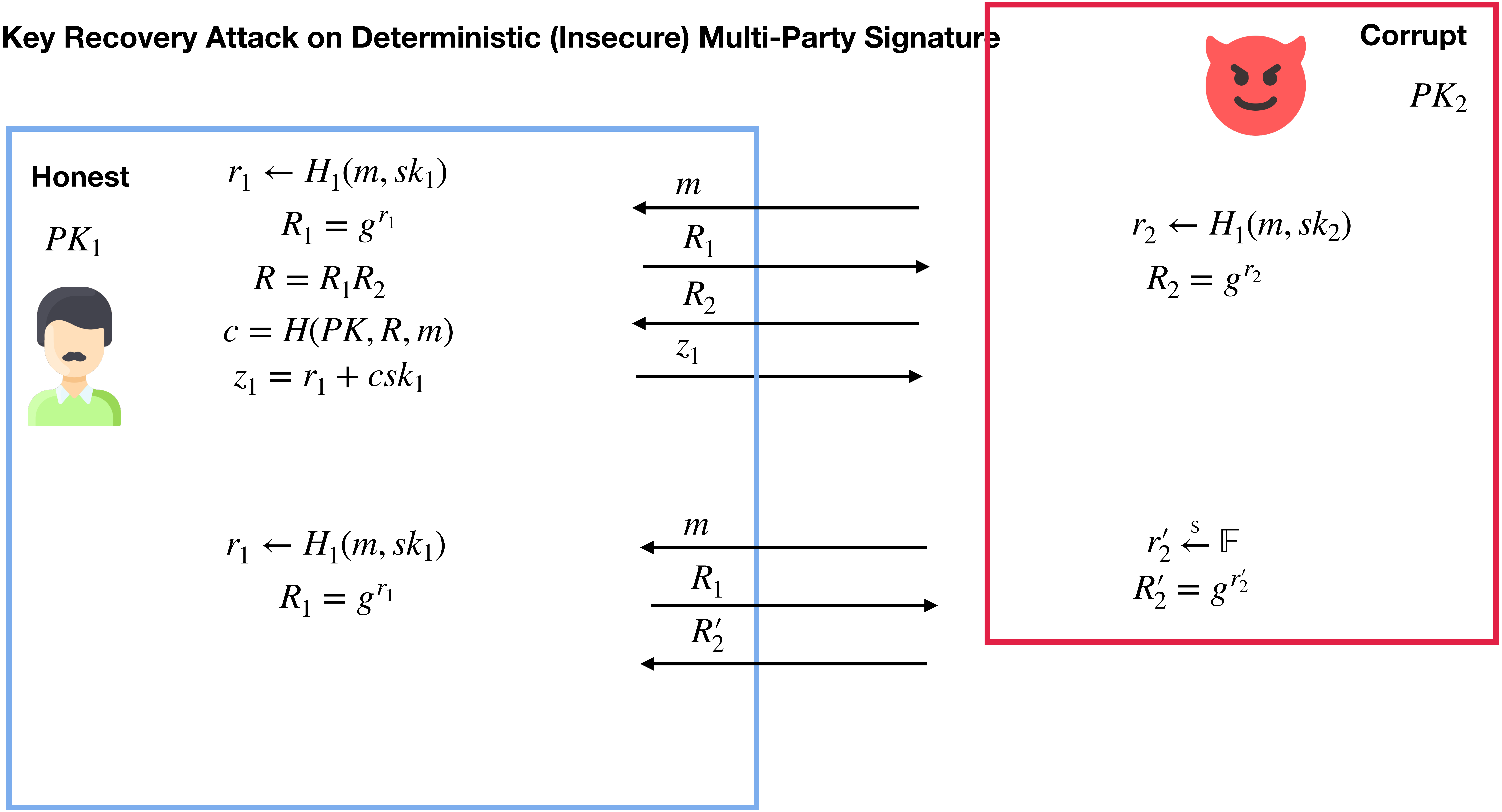
$\xleftarrow{\quad m \quad}$

$\xrightarrow[R_2']{\quad R_1 \quad}$

$r_2' \overset{\$}{\leftarrow} \mathbb{F}$

$R_2' = g^{r_2'}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$\xleftarrow{m}$

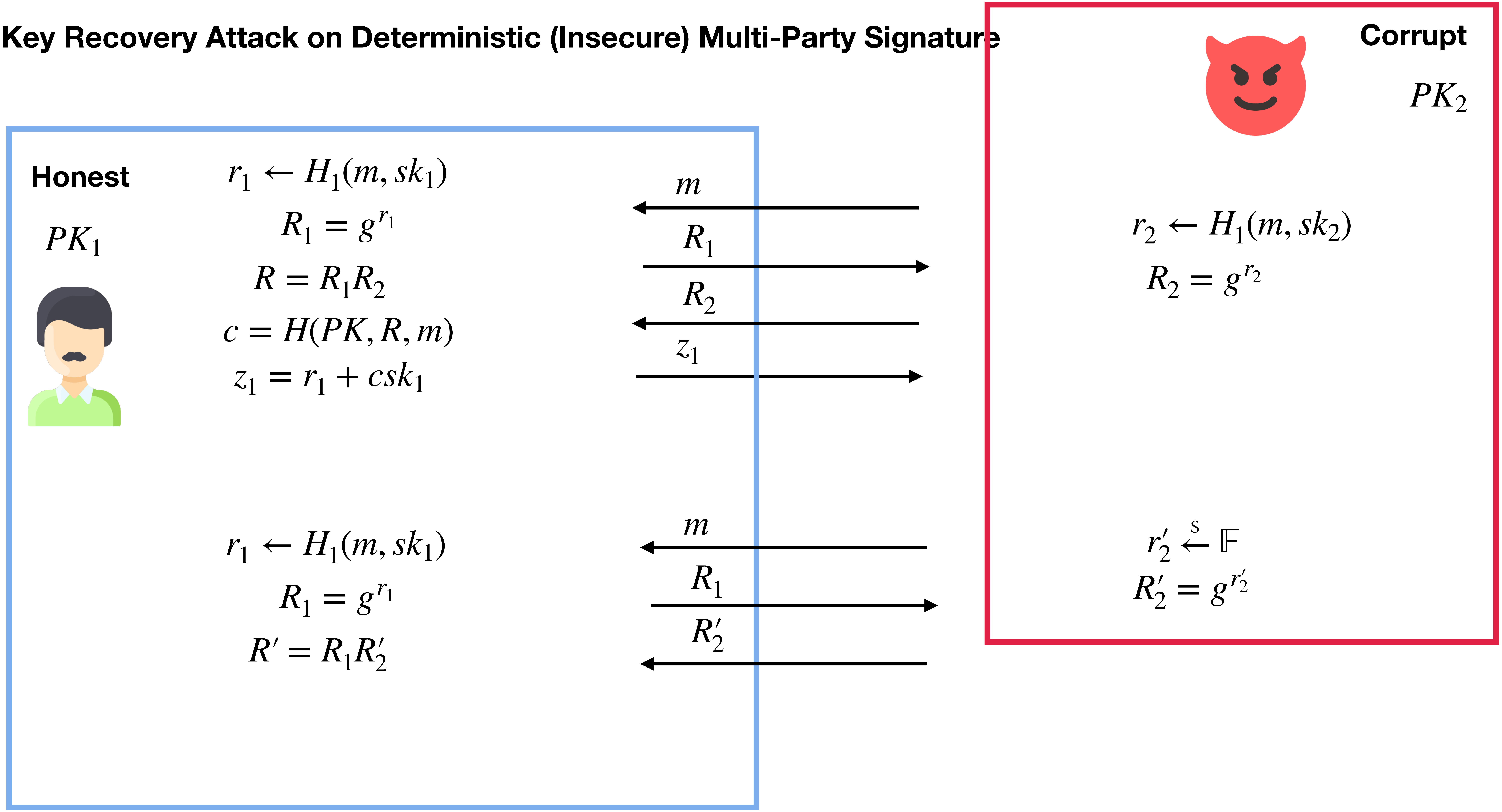
$\xrightarrow{R_1}$

$\xleftarrow{R_2'}$

$r_2' \xleftarrow{\$} \mathbb{F}$

$R_2' = g^{r_2'}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

Corrupt

PK_2

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + csk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R' = R_1 R'_2$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R'_2}$

$r'_2 \overset{\$}{\leftarrow} \mathbb{F}$

$R'_2 = g^{r'_2}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Corrupt

PK_2

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R' = R_1 R_2'$

$c' = H(PK, R', m)$

$\xleftarrow{m}$

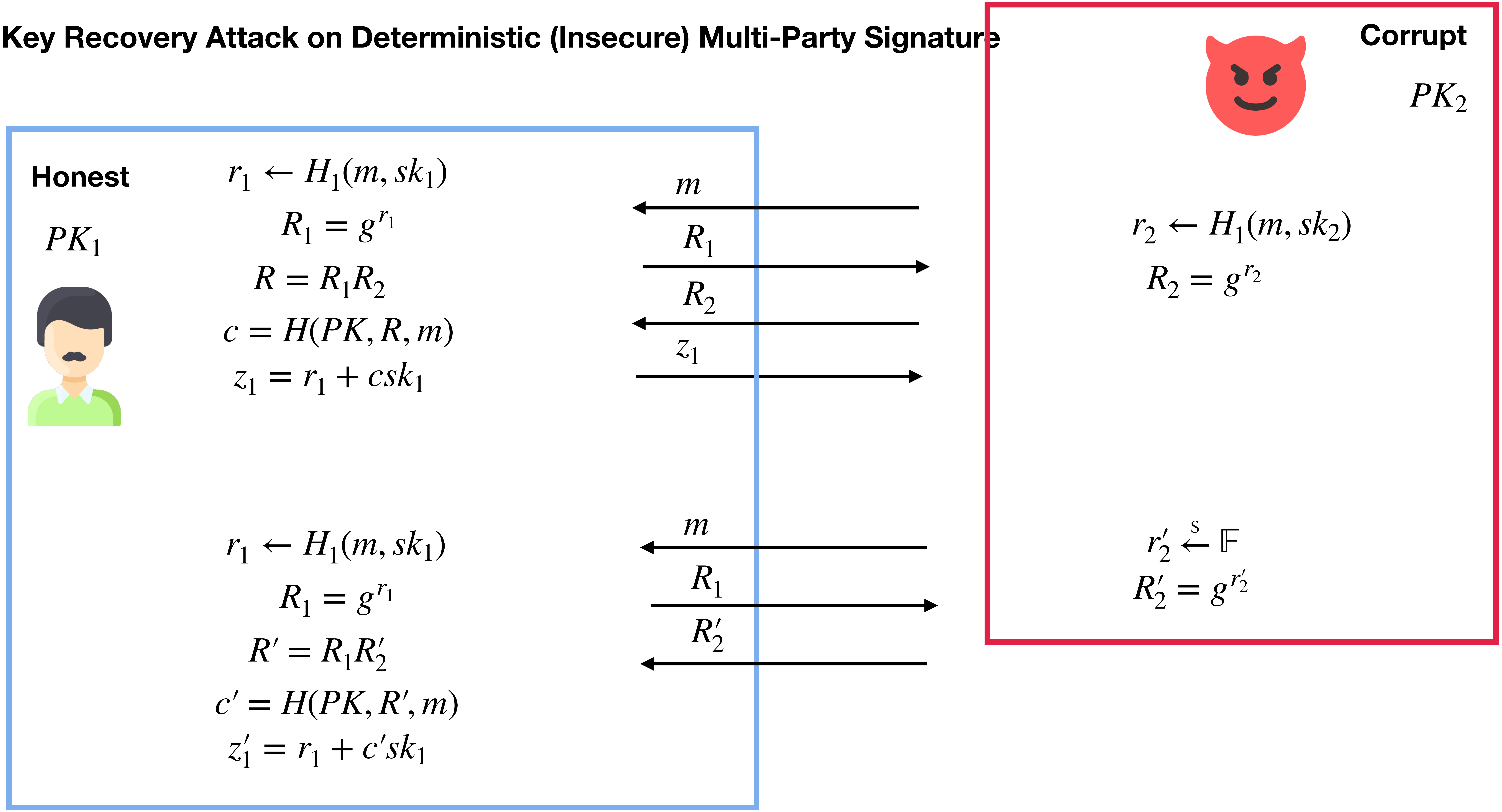
$\xrightarrow{R_1}$

$\xleftarrow{R_2'}$

$r_2' \xleftarrow{\$} \mathbb{F}$

$R_2' = g^{r_2'}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R = R_1 R_2$$
$$c = H(PK, R, m)$$
$$z_1 = r_1 + c sk_1$$

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R' = R_1 R_2'$$
$$c' = H(PK, R', m)$$
$$z_1' = r_1 + c' sk_1$$

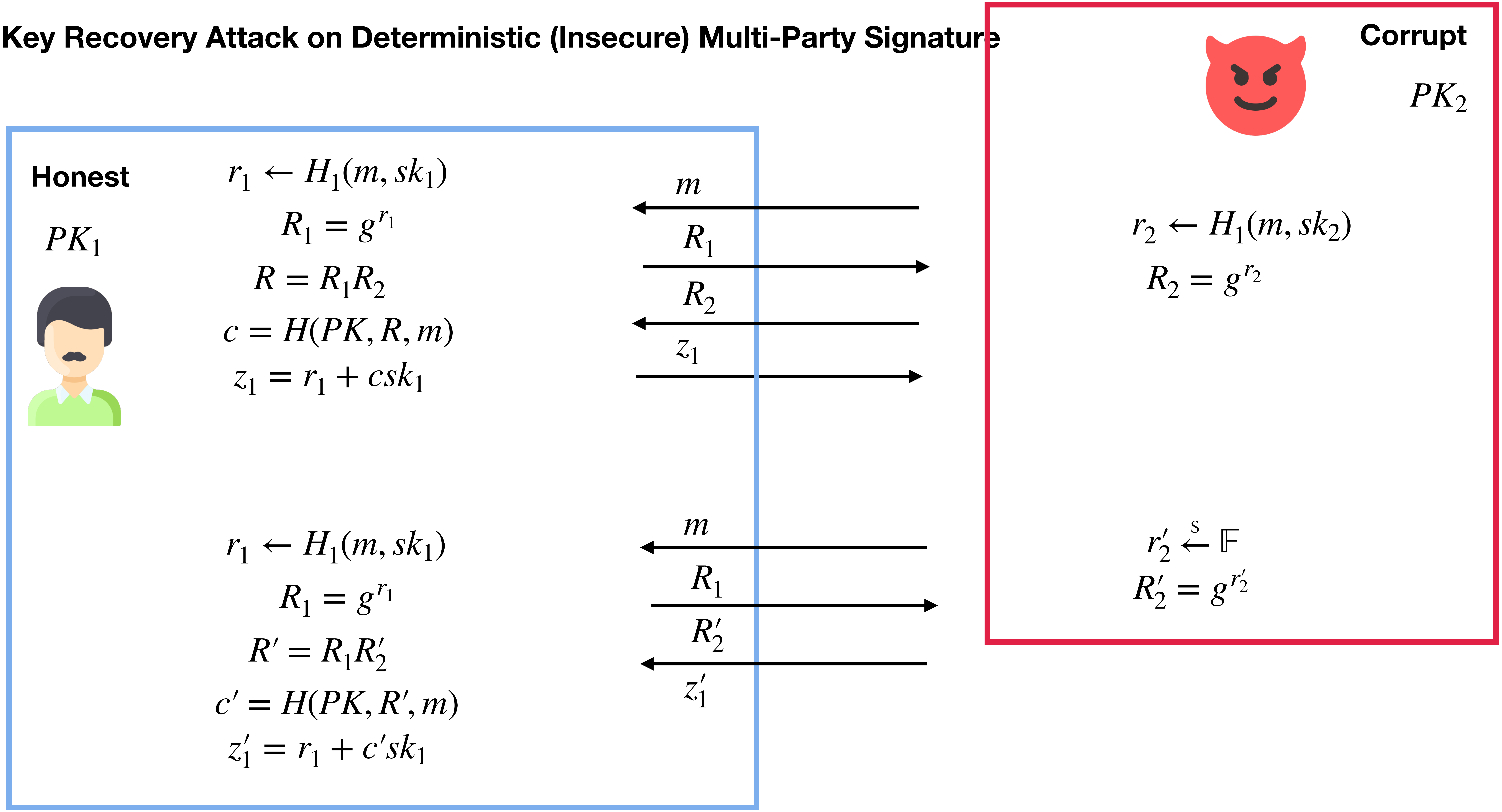
Corrupt

PK_2

$$r_2 \leftarrow H_1(m, sk_2)$$
$$R_2 = g^{r_2}$$

$$r_2' \xleftarrow{\$} \mathbb{F}$$
$$R_2' = g^{r_2'}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R = R_1 R_2$$
$$c = H(PK, R, m)$$
$$z_1 = r_1 + c sk_1$$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

Corrupt

PK_2

$$r_2 \leftarrow H_1(m, sk_2)$$
$$R_2 = g^{r_2}$$

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R' = R_1 R_2'$$
$$c' = H(PK, R', m)$$
$$z_1' = r_1 + c' sk_1$$

$\xleftarrow{m}$

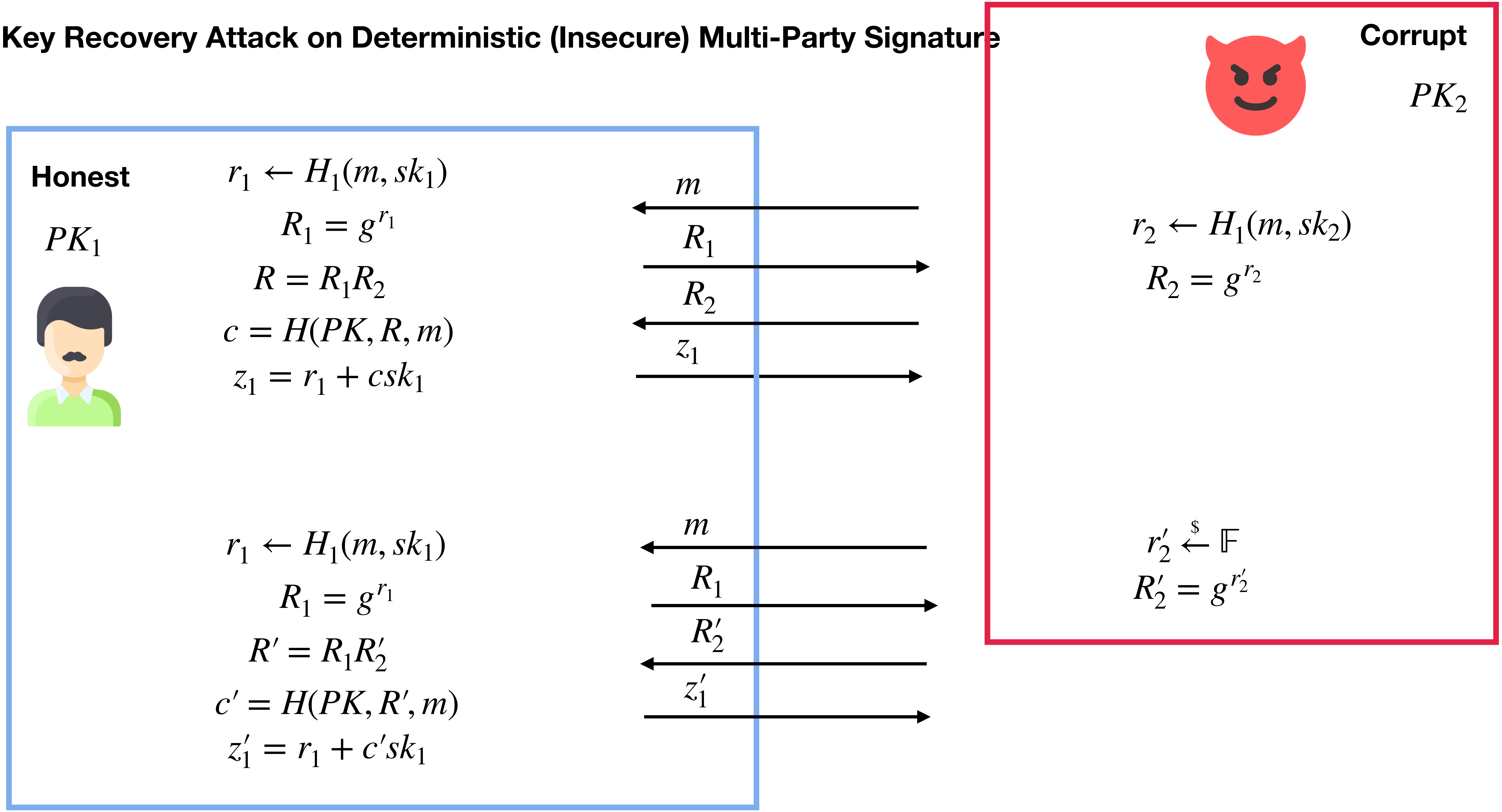
$\xrightarrow{R_1}$

$\xleftarrow{R_2'}$

z_1'

$$r_2' \overset{\$}{\leftarrow} \mathbb{F}$$
$$R_2' = g^{r_2'}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R = R_1 R_2$$
$$c = H(PK, R, m)$$
$$z_1 = r_1 + c sk_1$$

m ←

R_1 →

R_2 ←

z_1 →

Corrupt

PK_2

$$r_2 \leftarrow H_1(m, sk_2)$$
$$R_2 = g^{r_2}$$

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R' = R_1 R_2'$$
$$c' = H(PK, R', m)$$
$$z_1' = r_1 + c' sk_1$$

m ←

R_1 →

R_2' ←

z_1' →

$$r_2' \xleftarrow{\$} \mathbb{F}$$
$$R_2' = g^{r_2'}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R = R_1 R_2$$
$$c = H(PK, R, m)$$
$$z_1 = r_1 + c sk_1$$

$$\xleftarrow{\quad m \quad}$$

Honest party cannot detect that the corrupt party has deviated from the protocol!

Corrupt

PK_2

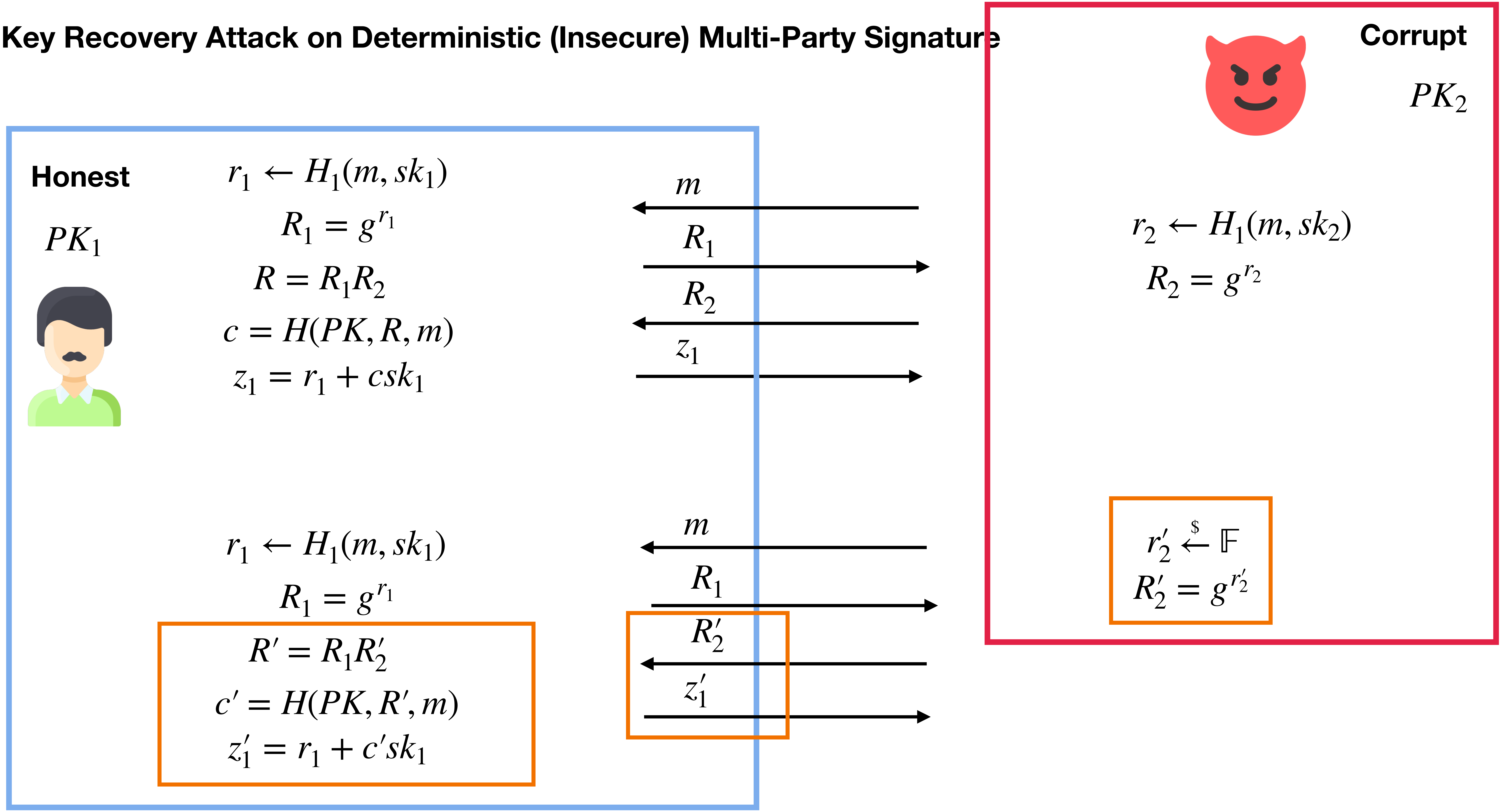
$$r_2 \leftarrow H_1(m, sk_2)$$
$$R_2 = g^{r_2}$$

$$r_1 \leftarrow H_1(m, sk_1)$$
$$R_1 = g^{r_1}$$
$$R' = R_1 R'_2$$
$$c' = H(PK, R', m)$$
$$z'_1 = r_1 + c' sk_1$$

$$\xleftarrow{\quad m \quad}$$
$$\xrightarrow{\quad R_1 \quad}$$
$$\xleftarrow{\quad R'_2 \quad}$$
$$\xrightarrow{\quad z'_1 \quad}$$

$$r'_2 \xleftarrow{\$} \mathbb{F}$$
$$R'_2 = g^{r'_2}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

$\xleftarrow{\quad R_2 \quad}$

$\xrightarrow{\quad z_1 \quad}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R' = R_1 R_2'$

$c' = H(PK, R', m)$

$z_1' = r_1 + c' sk_1$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad R_1 \quad}$

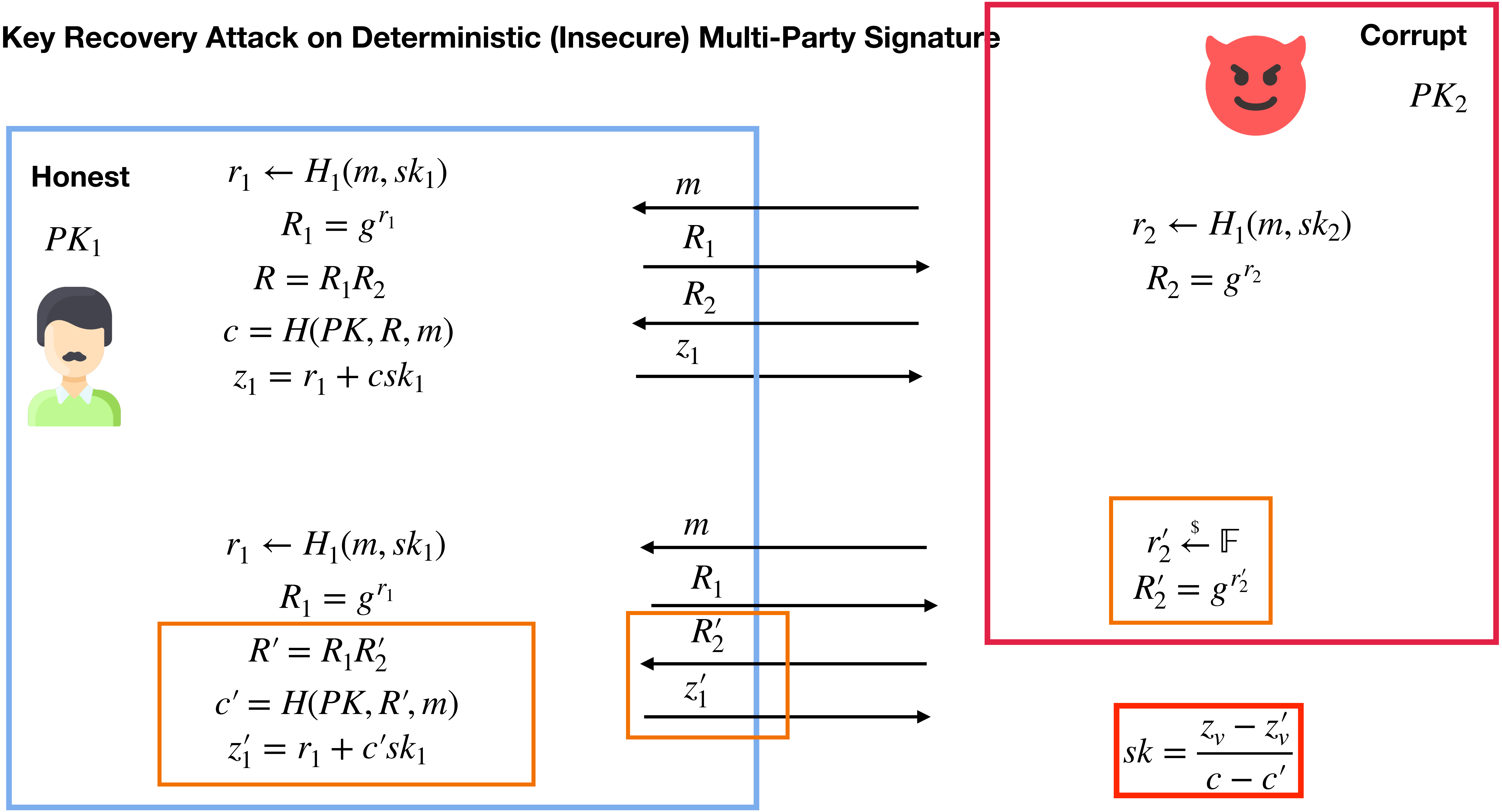
$\xleftarrow{\quad R_2' \quad}$

$\xrightarrow{\quad z_1' \quad}$

$r_2' \xleftarrow{\$} \mathbb{F}$

$R_2' = g^{r_2'}$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature



Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R' = R_1 R_2'$

$c' = H(PK, R', m)$

$z_1' = r_1 + c' sk_1$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2'}$

$\xrightarrow{z_1'}$

$r_2' \xleftarrow{\$} \mathbb{F}$

$R_2' = g^{r_2'}$

$$sk = \frac{z_v - z_v'}{c - c'}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$$\boxed{\begin{aligned} R' &= R_1 R_2' \\ c' &= H(PK, R', m) \\ z_1' &= r_1 + c' sk_1 \end{aligned}}$$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$$\boxed{\begin{aligned} r_2' &\xleftarrow{\$} \mathbb{F} \\ R_2' &= g^{r_2'} \end{aligned}}$$

$$\boxed{sk = \frac{z_v - z_v'}{c - c'}}$$

Key Recovery Attack on Deterministic (Insecure) Multi-Party Signature

Honest

PK_1

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R = R_1 R_2$

$c = H(PK, R, m)$

$z_1 = r_1 + c sk_1$

$r_1 \leftarrow H_1(m, sk_1)$

$R_1 = g^{r_1}$

$R' = R_1 R'_2$

$c' = H(PK, R', m)$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R_2}$

$\xrightarrow{z_1}$

$\xleftarrow{m}$

$\xrightarrow{R_1}$

$\xleftarrow{R'_2}$

$\xrightarrow{z'_1}$

Corrupt

PK_2

$r_2 \leftarrow H_1(m, sk_2)$

$R_2 = g^{r_2}$

$r'_2 \xleftarrow{\$} \mathbb{F}$

$R'_2 = g^{r'_2}$

$sk = \frac{z_v - z'_v}{c - c'}$

Question: Can we build a real-world (efficient) deterministic threshold signature?

Clarify the Tradeoff Between Efficiency and Security Assumptions for Signing

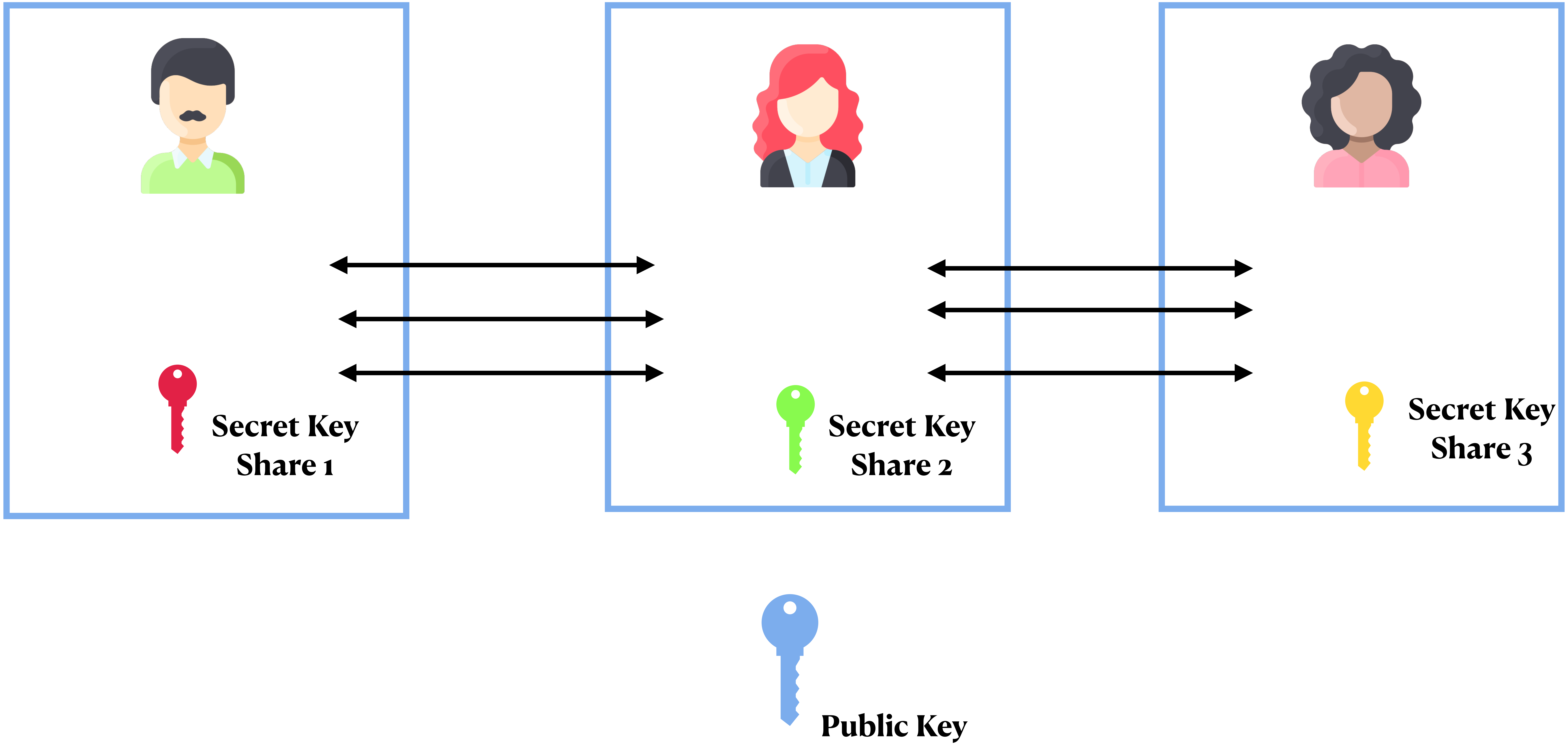
	Scheme	Assumptions	Signing Rounds
Multi-sigs	MuSig [MPSW18, BDN18] SimpleMuSig [BDN18, CKM21]	DL+ROM	3
	MuSig2 [NRS21] DWMS [AB21] SpeedyMuSig [CKM21]	OMDL+ROM	2
Threshold	Lindell22	Schnorr	3
	Sparkle [CKM23]	DL+ROM	
	FROST [KG20, BCKMTZ22] FROST2 [CKM21]	OMDL+ROM	2

	Scheme	Assumptions	Signing Rounds
Multi-sigs	MuSig [MPSW18, BDN18] SimpleMuSig [BDN18, CKM21]	DL+ROM	3
	MuSig2 [NRS21] DWMS [AB21] SpeedyMuSig [CKM21]	OMDL+ROM	2
Threshold	Lindell22	Schnorr	3
	Sparkle [CKM23]	DL+ROM	
	FROST [KG20, BCKMTZ22] FROST2 [CKM21]	OMDL+ROM	2

Question: Can we prove that two-round, efficient multi-party Schnorr requires stronger assumptions?

Investigate the Tradeoff Between Efficiency and Security Assumptions for Key Generation

Distributed Key Generation



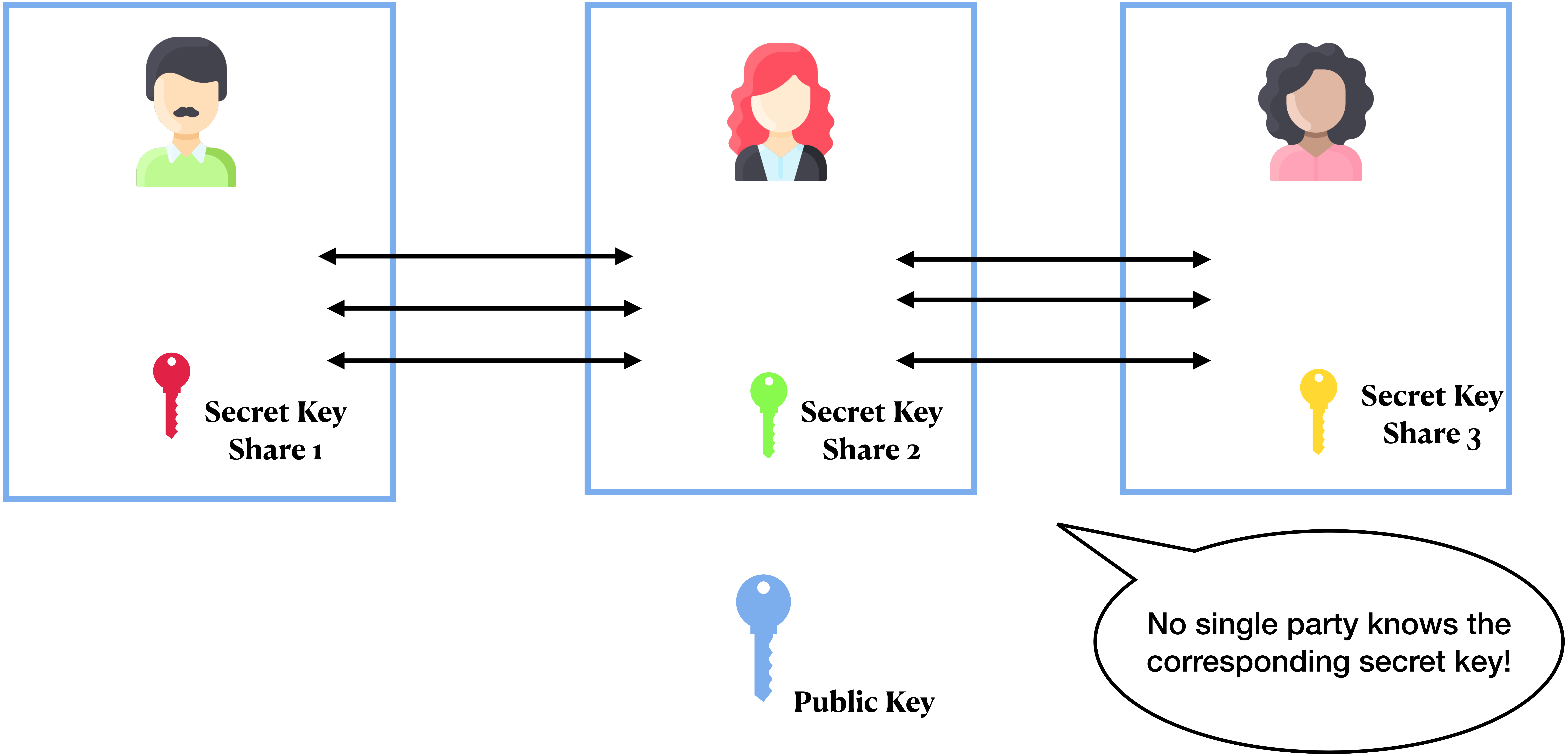
Secret Key Share 1

Secret Key Share 2

Secret Key Share 3

Public Key

Distributed Key Generation



Secret Key Share 1

Secret Key Share 2

Secret Key Share 3

Public Key

No single party knows the corresponding secret key!

Proving the security of DKGs

Proving the security of DKGs

- Two options:

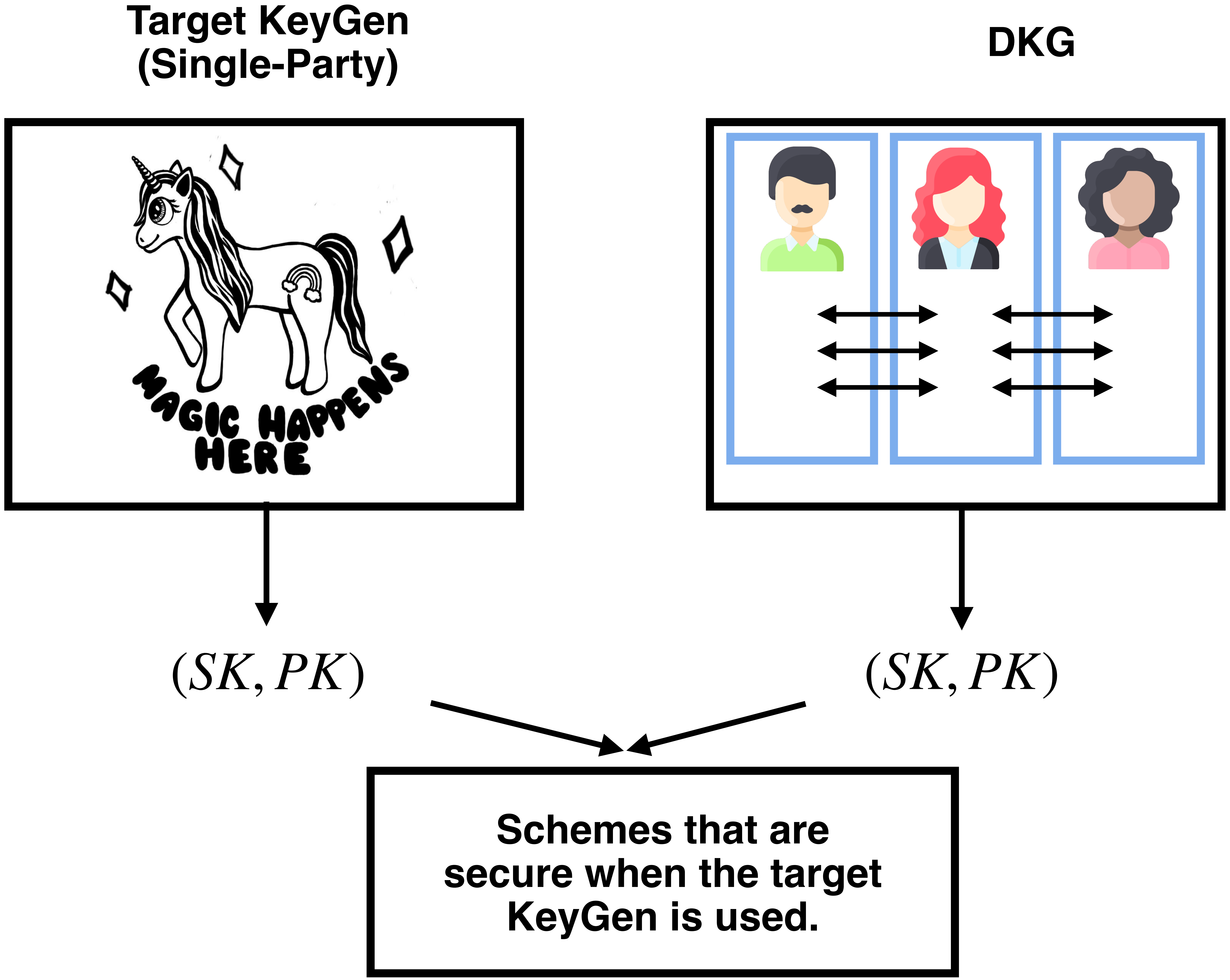
Proving the security of DKGs

- Two options:
  - Prove security of the DKG in the context of a proof for unforgeability for a threshold signature scheme

Proving the security of DKGs

- Two options:
  - Prove security of the DKG in the context of a proof for unforgeability for a threshold signature scheme
  - Prove the security of the DKG independently, i.e, via a proof of composability

Composability of Distributed Key Generation



Target KeyGen (Single-Party)

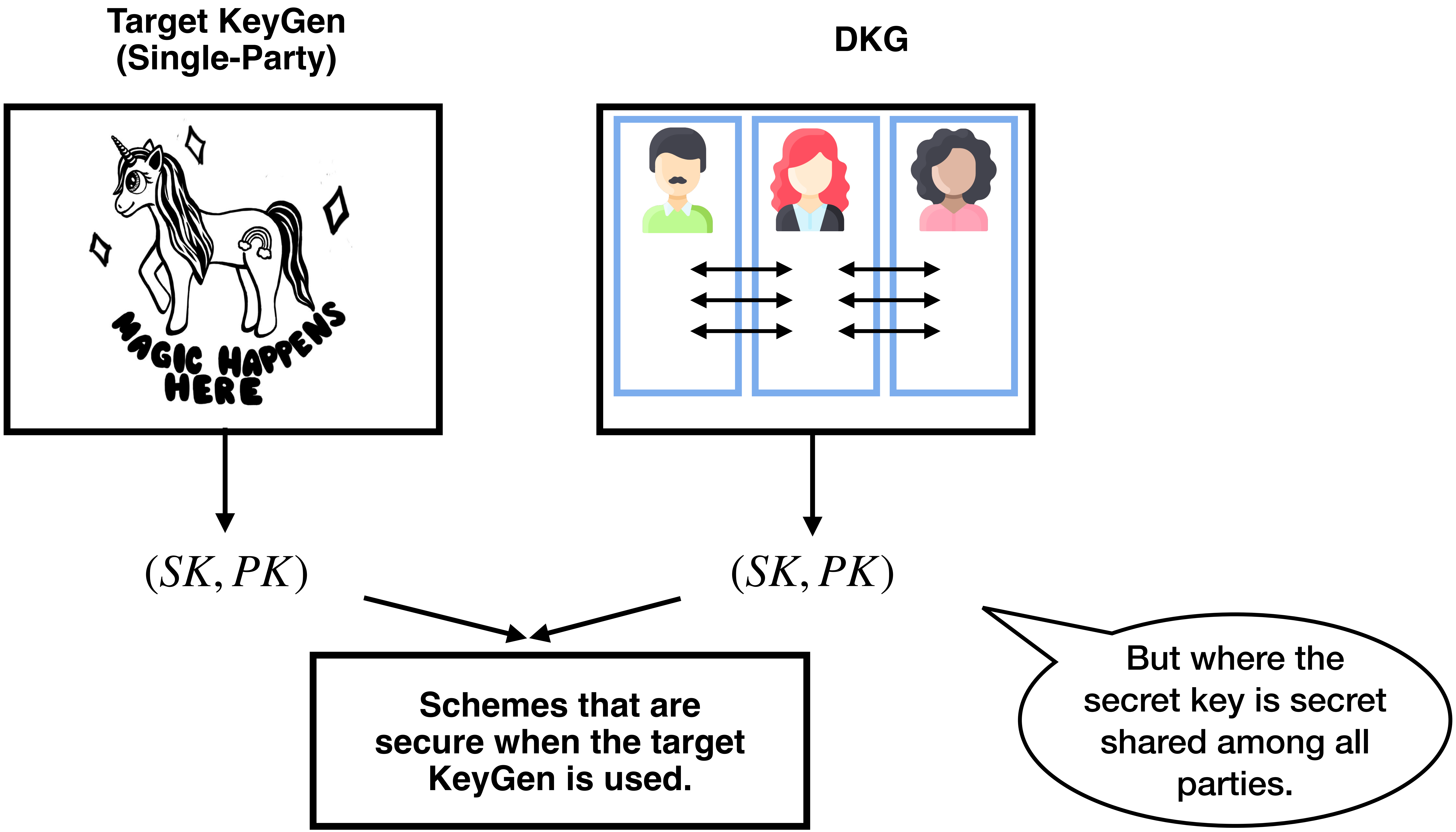
DKG

(SK, PK)

(SK, PK)

Schemes that are secure when the target KeyGen is used.

Composability of Distributed Key Generation



Target KeyGen (Single-Party)

DKG

(SK, PK)

(SK, PK)

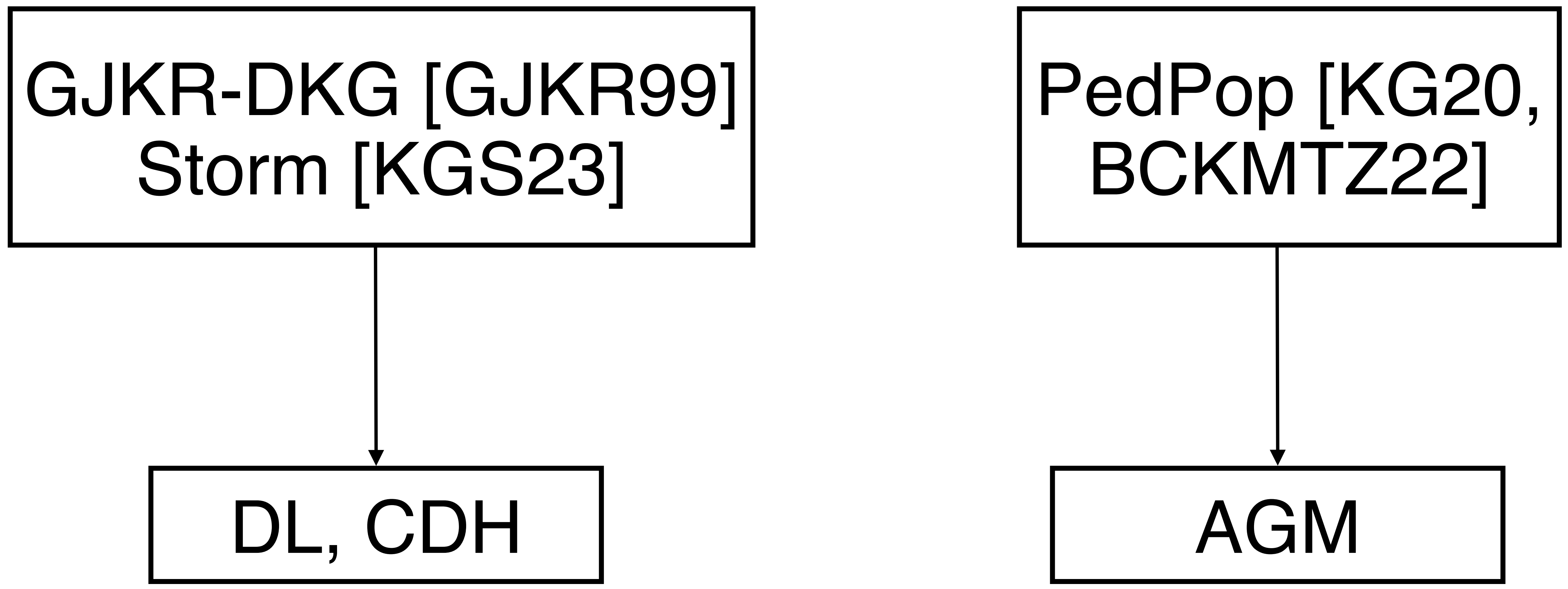
Schemes that are secure when the target KeyGen is used.

But where the secret key is secret shared among all parties.

Distributed Key Generation

3-Round (optimistically),
Composable

2-Round (total)
Proven for FROST



GJKR-DKG [GJKR99]
Storm [KGS23]

DL, CDH

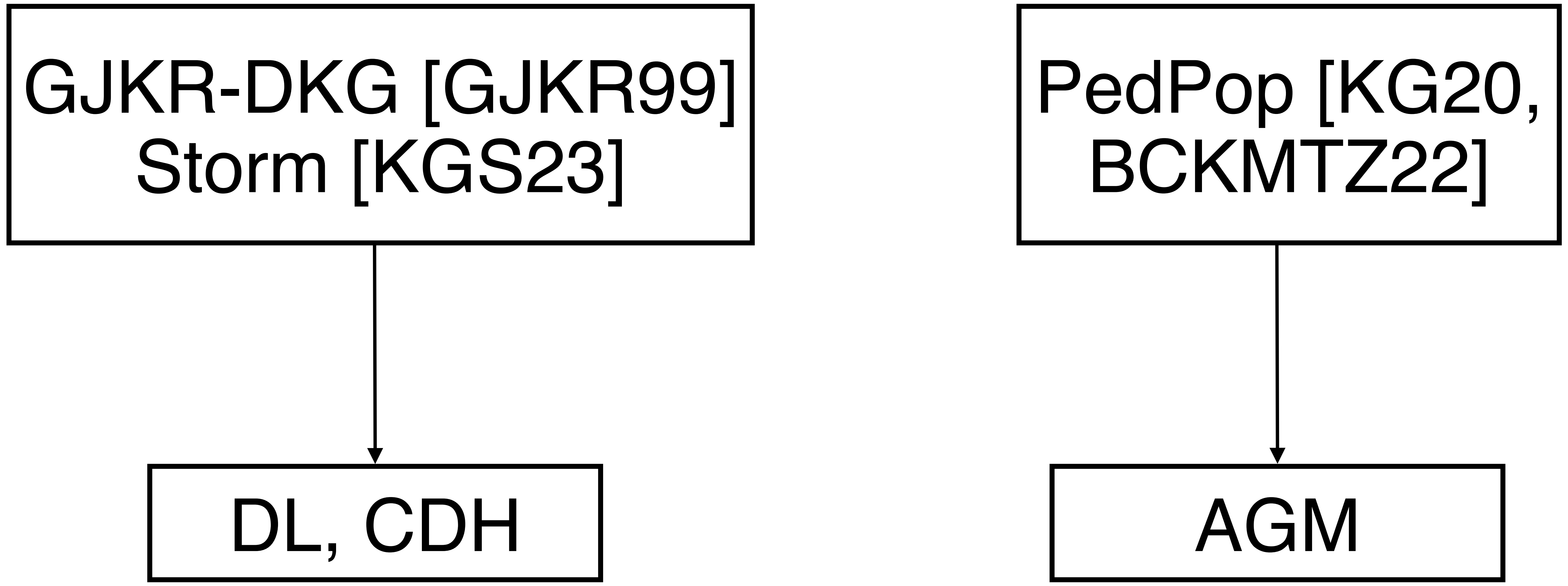
standard
assumption

PedPop [KG20,
BCKMTZ22]

AGM

stronger
assumption

Distributed Key Generation



3-Round (optimistically),
Composable

GJKR-DKG [GJKR99]
Storm [KGS23]

DL, CDH

standard
assumption

2-Round (total)
Proven for FROST

PedPop [KG20,
BCKMTZ22]

AGM

stronger
assumption

Question: Do two-round, efficient and composable DKGs exist?

Takeaways

Takeaways

- Schnorr multi-party signatures are being used in practice today!

Takeaways

- Schnorr multi-party signatures are being used in practice today!
- Many questions and challenges remain, to improving their usability and security.

Takeaways

- Schnorr multi-party signatures are being used in practice today!
- Many questions and challenges remain, to improving their usability and security.
- We would love to collaborate with anyone interested in tackling these problems or using these schemes, so please come talk to us!

Takeaways

- Schnorr multi-party signatures are being used in practice today!
- Many questions and challenges remain, to improving their usability and security.
- We would love to collaborate with anyone interested in tackling these problems or using these schemes, so please come talk to us!
- (t,n) of us will also be involved in the NIST call for threshold schemes, so please let us know if you would like to join forces.

Takeaways

- Schnorr multi-party signatures are being used in practice today!
- Many questions and challenges remain, to improving their usability and security.
- We would love to collaborate with anyone interested in tackling these problems or using these schemes, so please come talk to us!
- (t,n) of us will also be involved in the NIST call for threshold schemes, so please let us know if you would like to join forces.

Thank you!