Secure Two-party Threshold ECDSA from ECDSA Assumptions

Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat

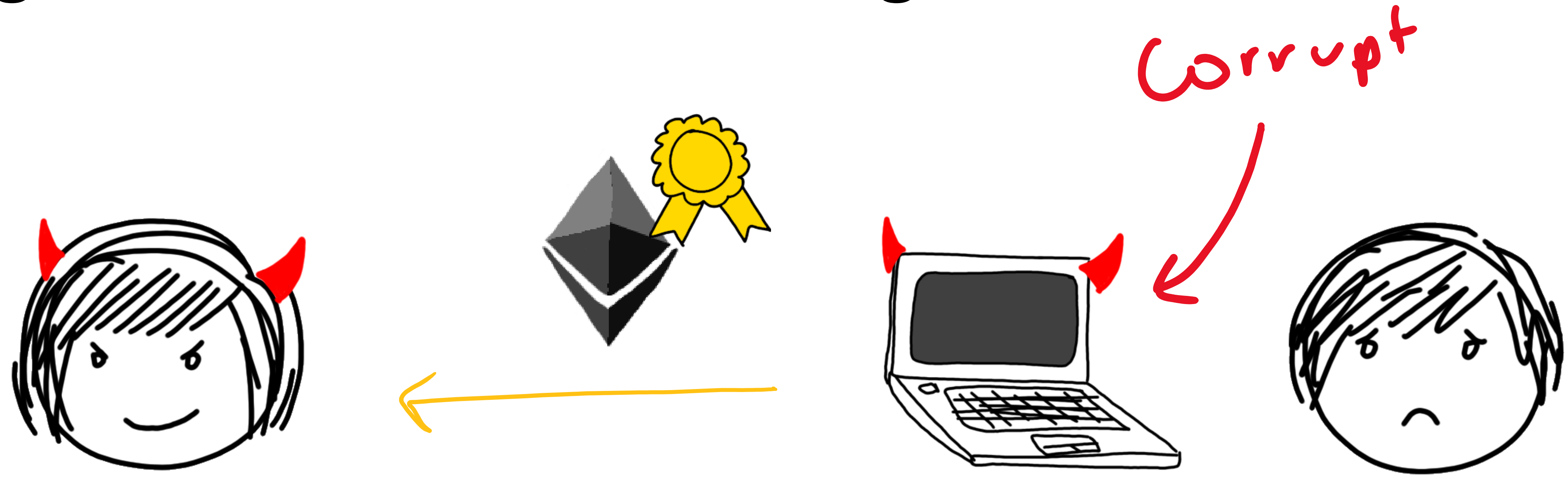
Northeastern University

Elliptic Curve Digital Signature Algorithm

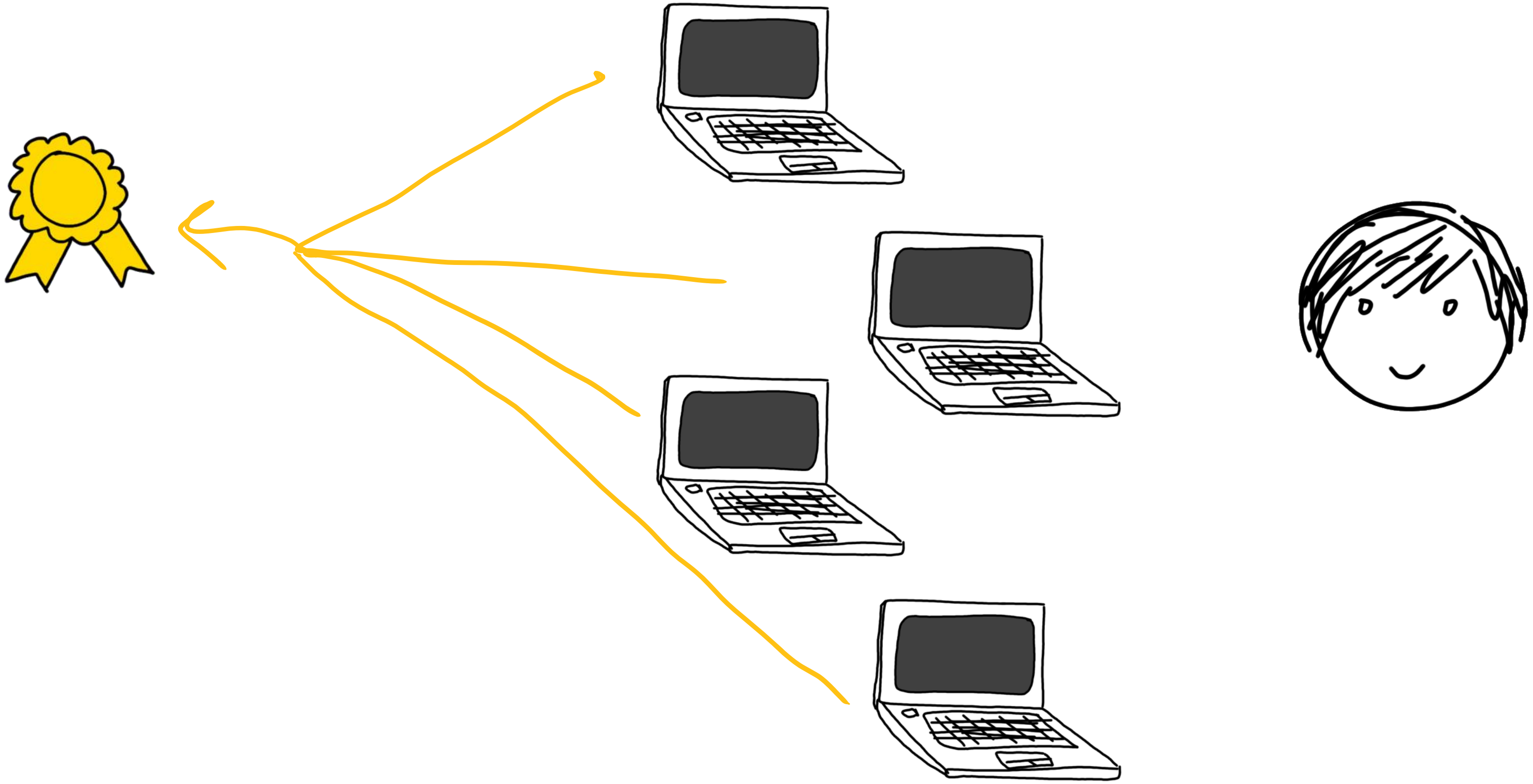
- Digital Signature Algorithm with elliptic curves
  - Smaller signature (512 bits) and key sizes (256-bit)
  - Security proof in “generic group model”
- Used pervasively in:
  - TLS
  - DNSSEC
  - Cryptocurrencies (Bitcoin, Ethereum, …)

Why Threshold Signatures?

Single Point of Failure for Signer



Distribute Signing Key Among Many Devices



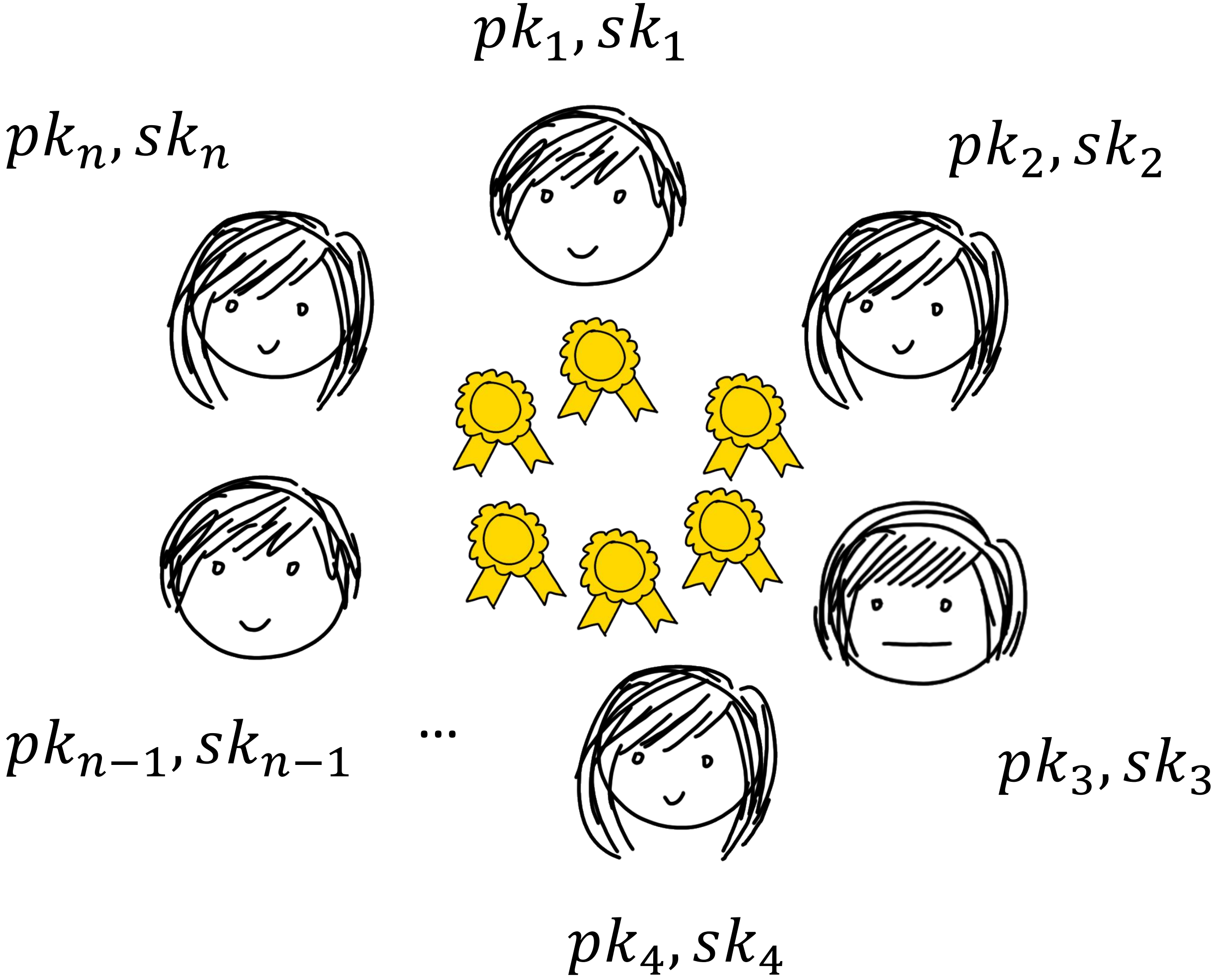
Multi-Signature

n parties

Each party has their own key pair

To sign a message, each party produces a signature under their public key

Signature: $\sigma_1, \sigma_2, \ldots, \sigma_n$



Why not Multi-Signatures?

- High bandwidth
  - Need to produce *n* signatures
  - Major bugs in implementations trying to reduce bandwidth
- Participating signers publicly known

On July 19 the ethereum community was warned that the Parity client version 1.5 and above contained a critical vulnerability in the multi-signature wallet feature. Further, a group of multi-signature "black hat exploiters" has managed to drain 150,000 ether from multi-sig wallets and ICO projects.

A Postmortem on the Parity Multi-Sig Library Self-Destruct

15 November 2017

On Monday November 6th 2017 02:33:47 PM UTC, a vulnerability in the "library" smart contract code, deployed as a shared component of all Parity multi-sig wallets deployed after July 20th 2017, was found by an anonymous user. The user decided to exploit this vulnerability and made himself the "owner" of the library contract. Subsequently, the user destructed this component. Since Parity multi-signature wallets depend on this component, this action blocked funds in 587 wallets holding a total amount of 513,774.16 Ether as well as additional tokens. Subsequent to destroying the library component, someone (purportedly this same user) posted under the username of "devops199" issue #6995 that prompted our investigation into this matter.

t-of-n Threshold Signature Scheme

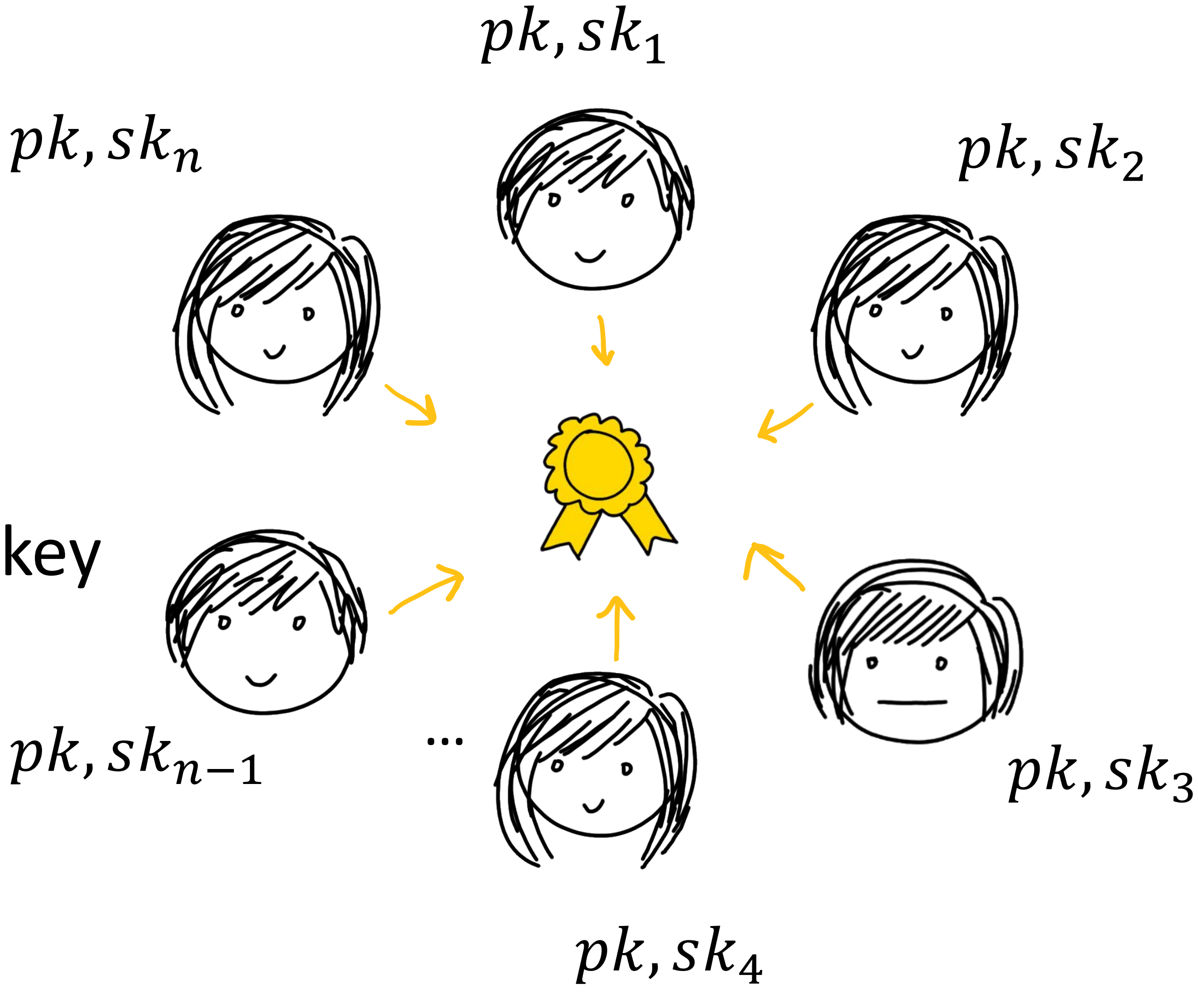
n parties

Jointly compute a *single* public key

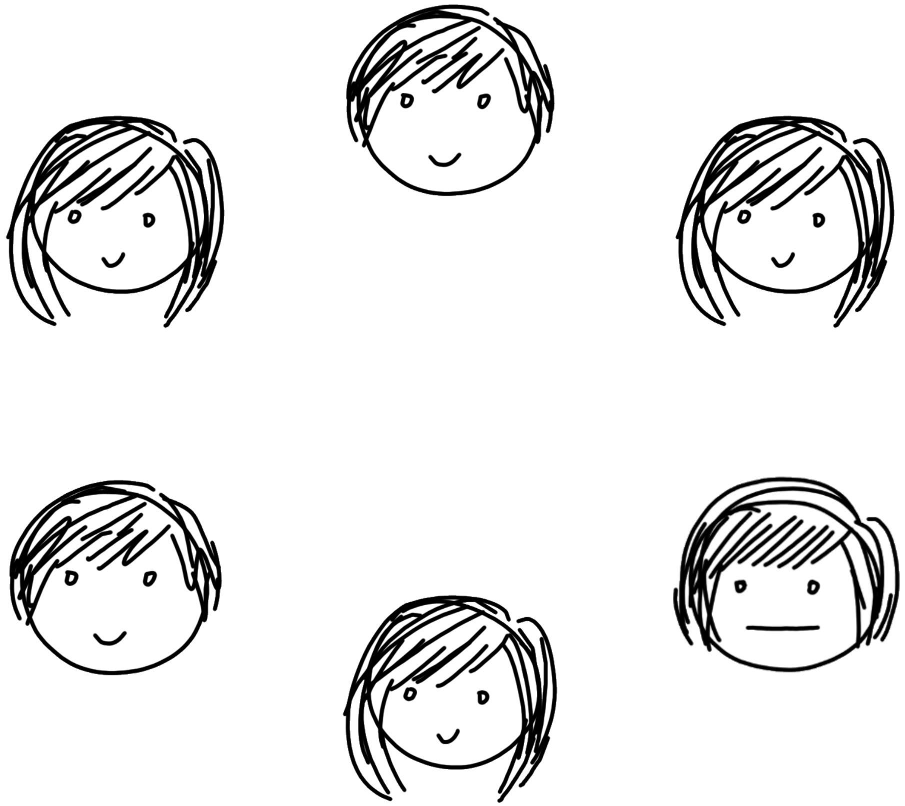
Each party has a share of the secret key

t parties needed to generate new signatures

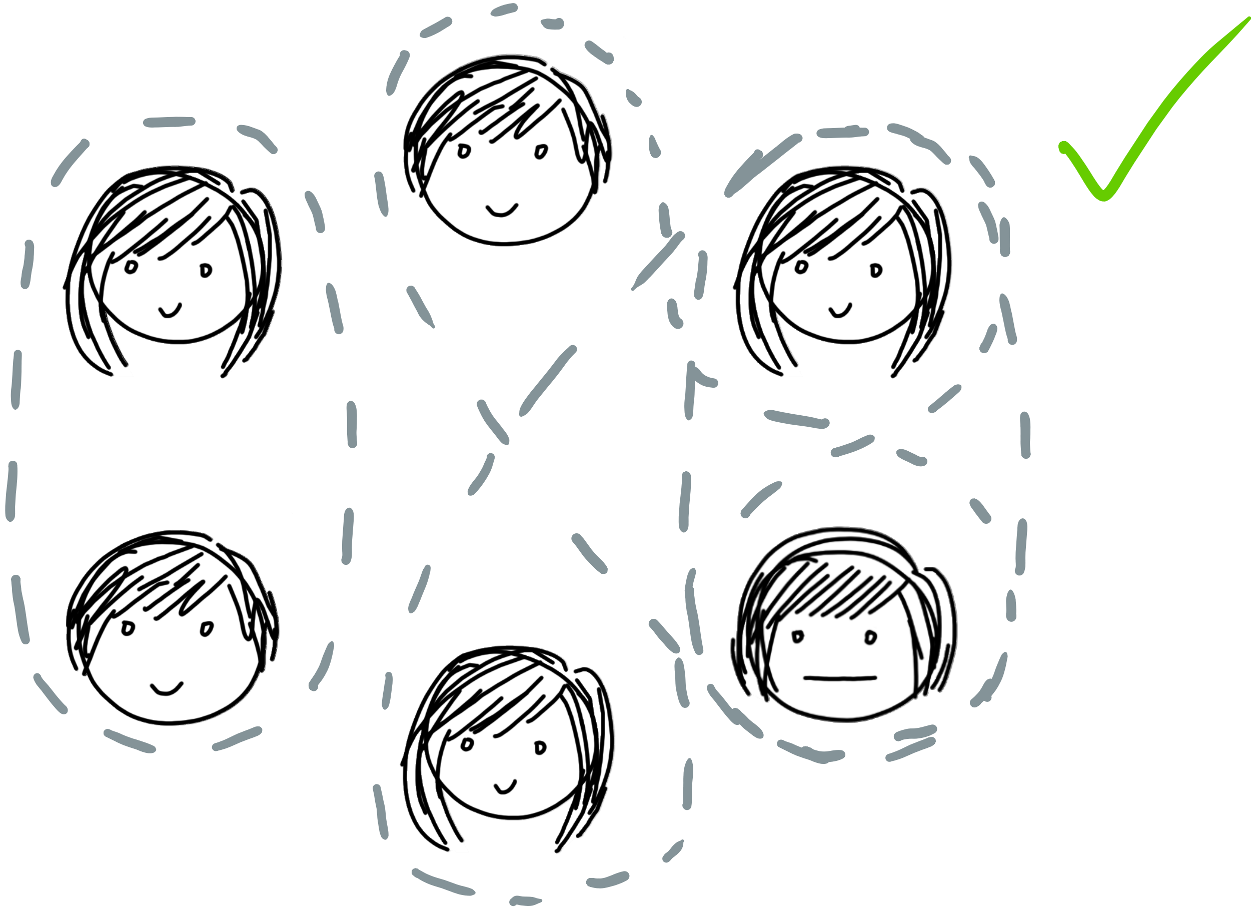
Signature: σ



2-of-*n* Threshold Signature Scheme

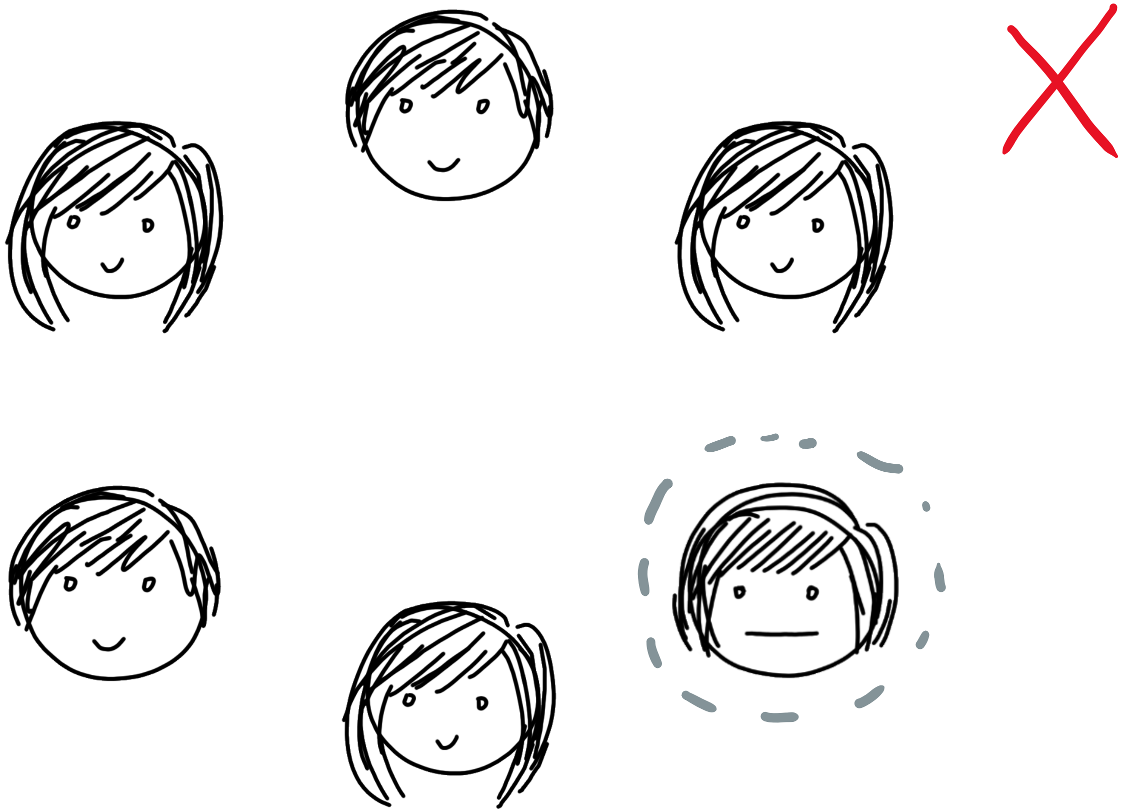


2-of-*n* Threshold Signature Scheme



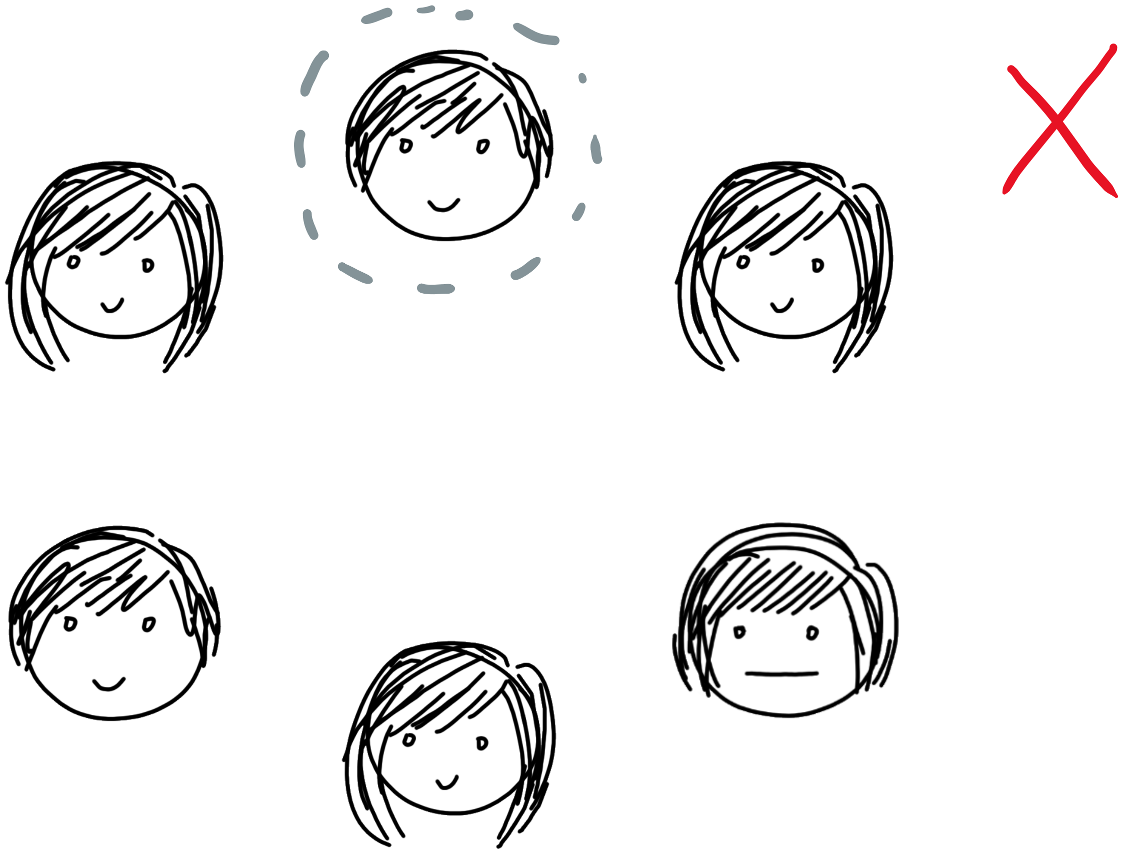
Participation of *2* parties needed to generate new signatures

2-of-*n* Threshold Signature Scheme



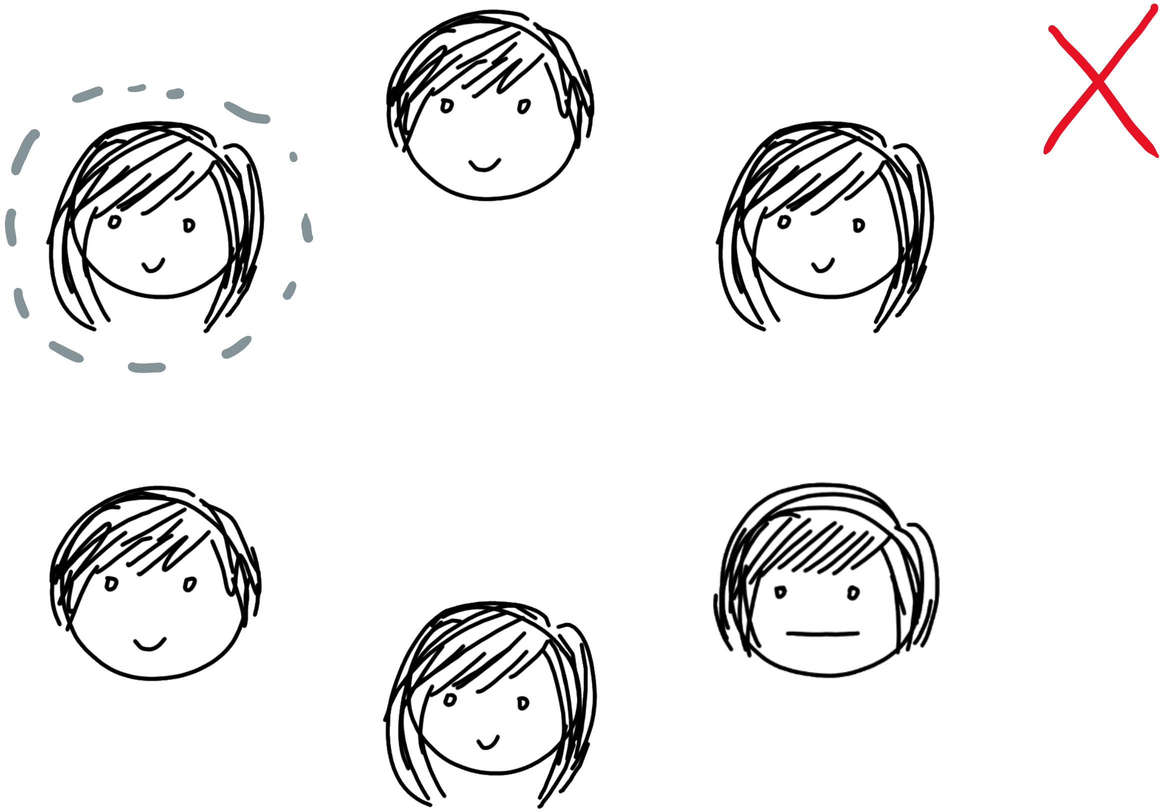
Single users cannot forge a signature

2-of-n Threshold Signature Scheme



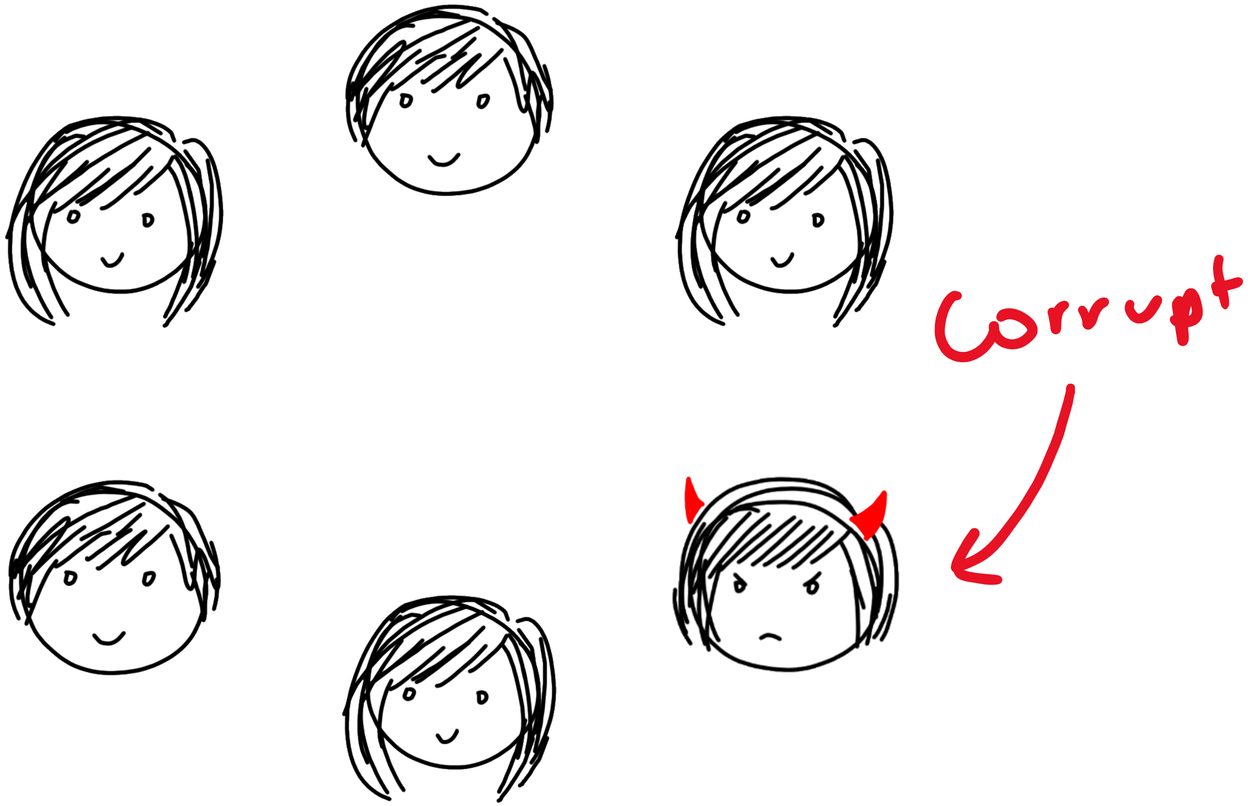
Single users cannot forge a signature

2-of-*n* Threshold Signature Scheme



Single users cannot forge a signature

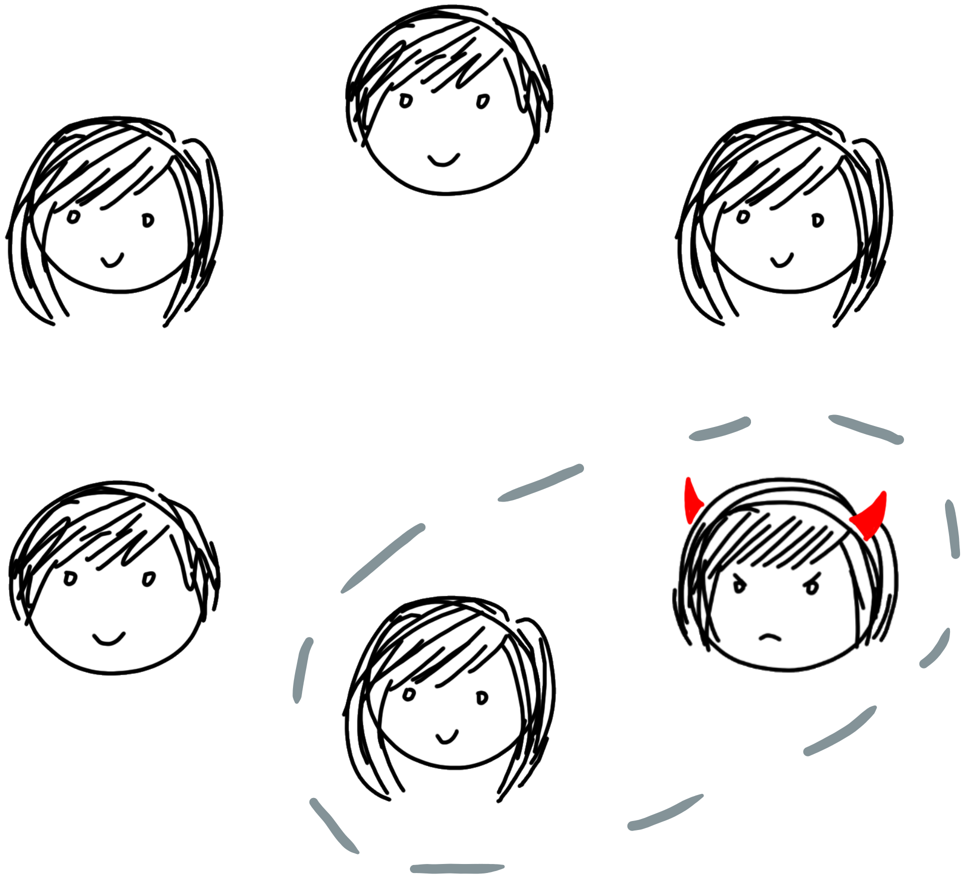
Handling Corruptions



Handling Corruptions

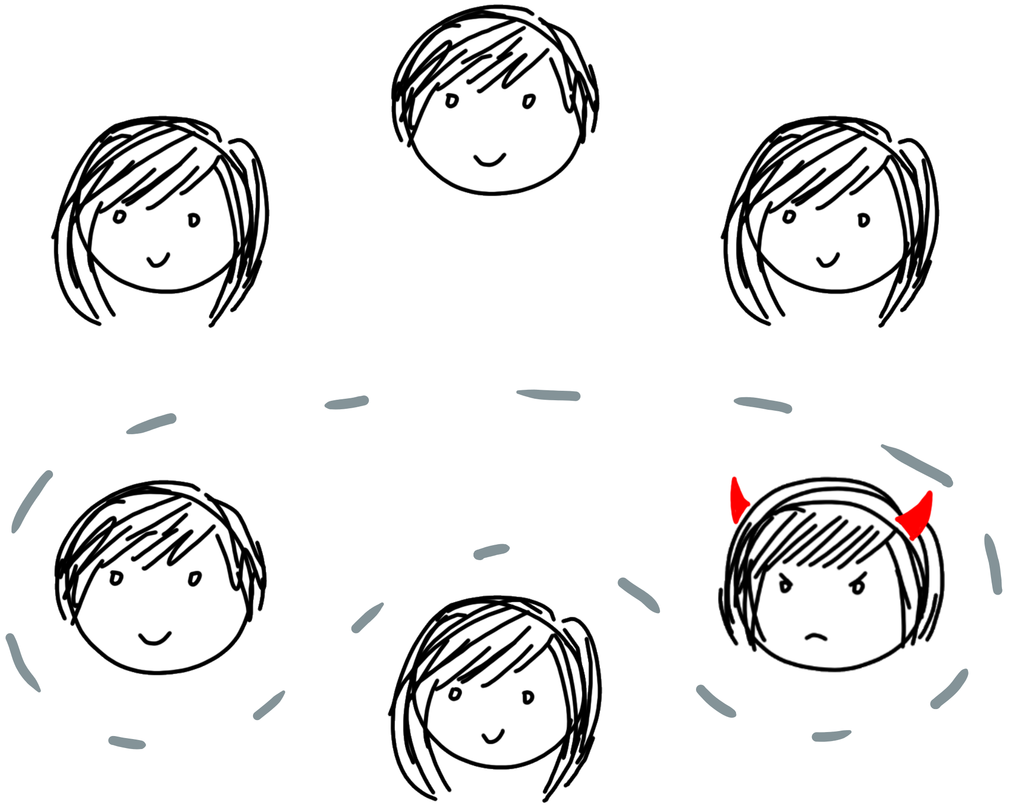
Adversary can interact with parties

Handling Corruptions



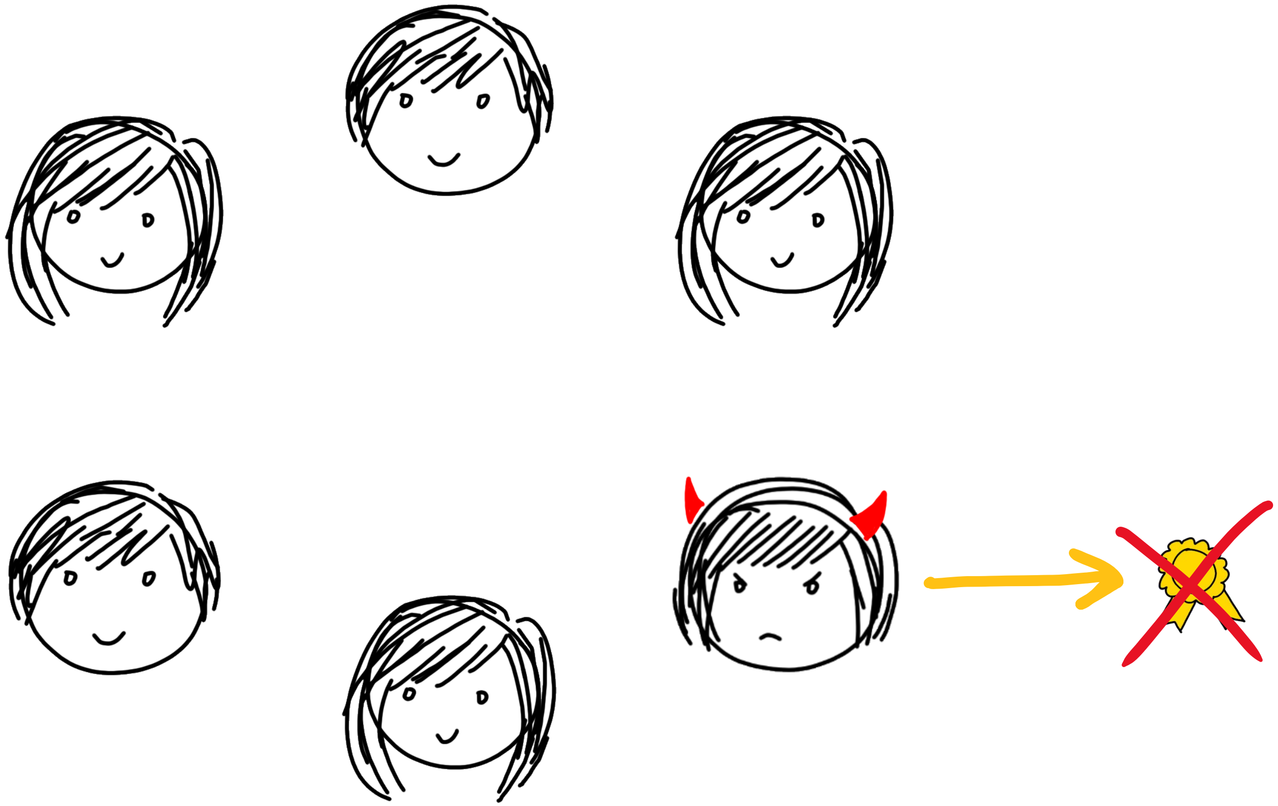
Adversary can interact with parties

Handling Corruptions



Adversary can interact with parties

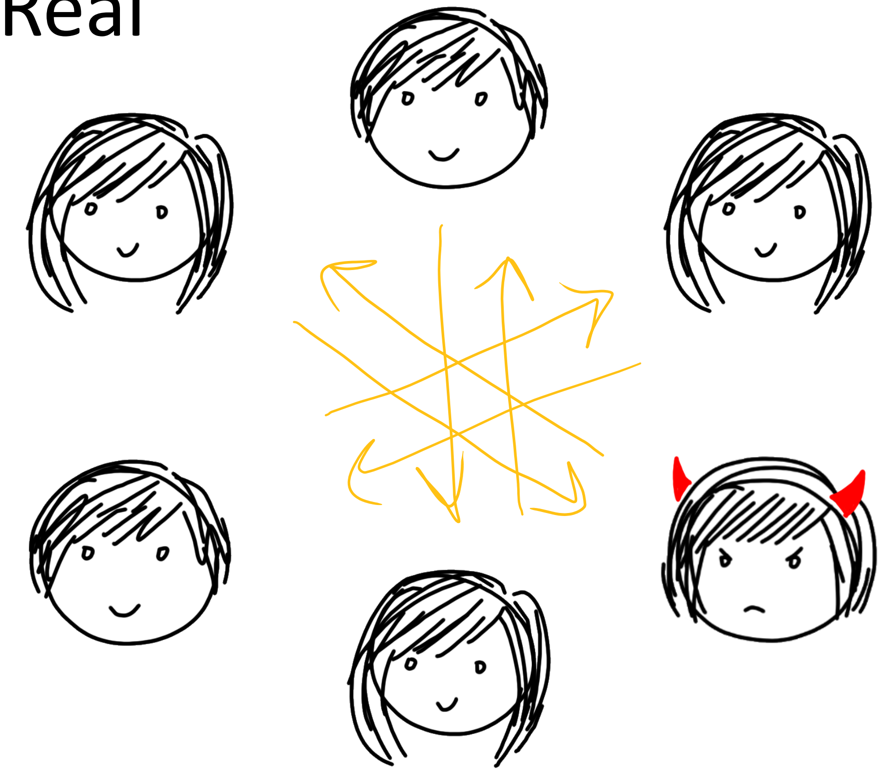
Handling Corruptions



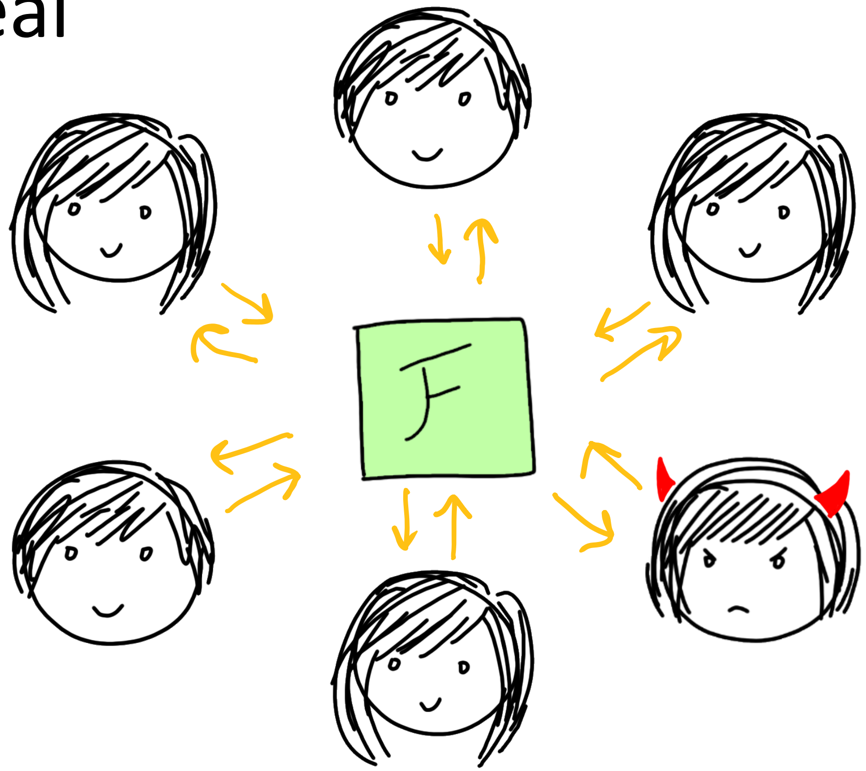
Adversary still shouldn't be able to forge a signature

Security Model

Real



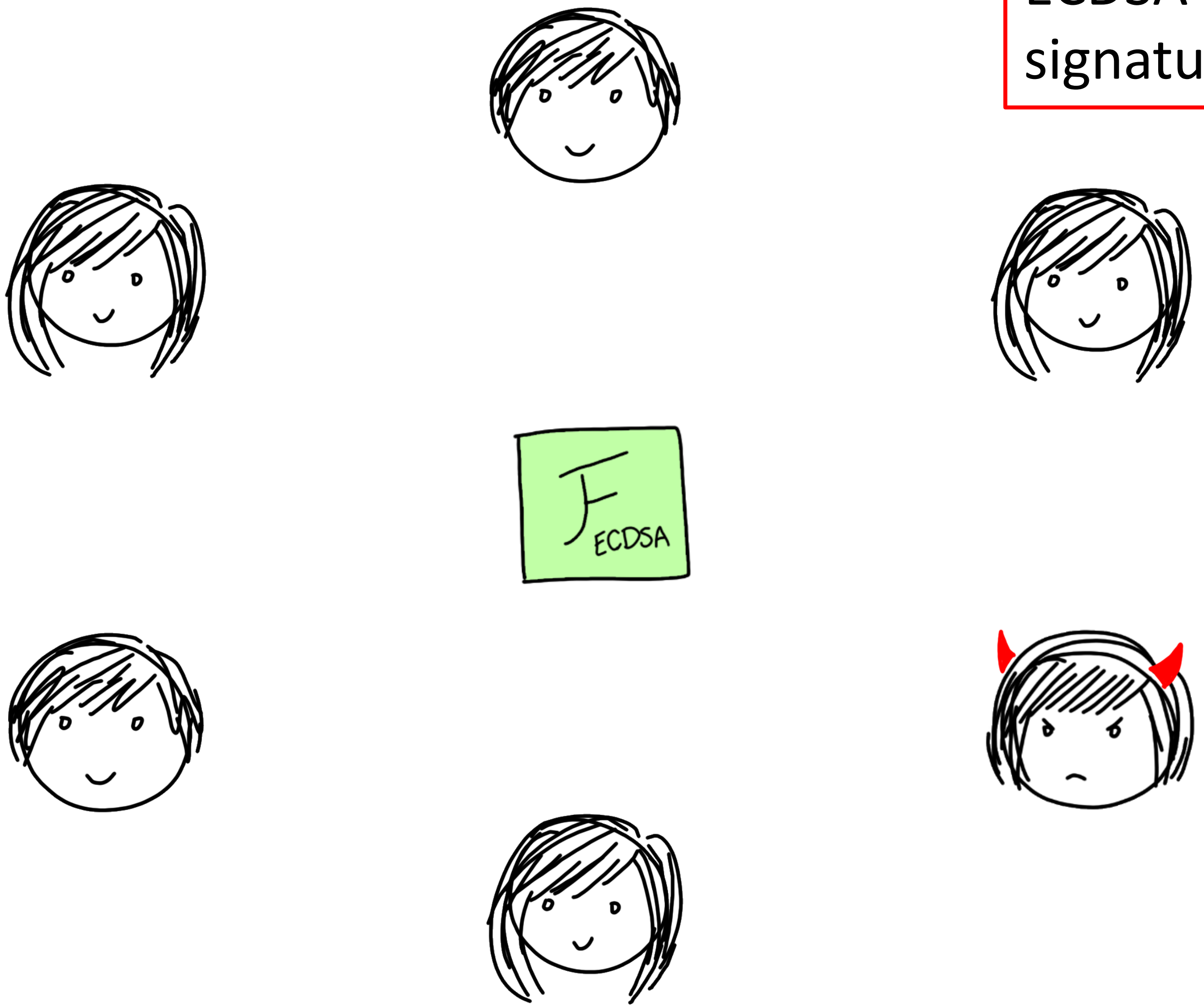
Ideal



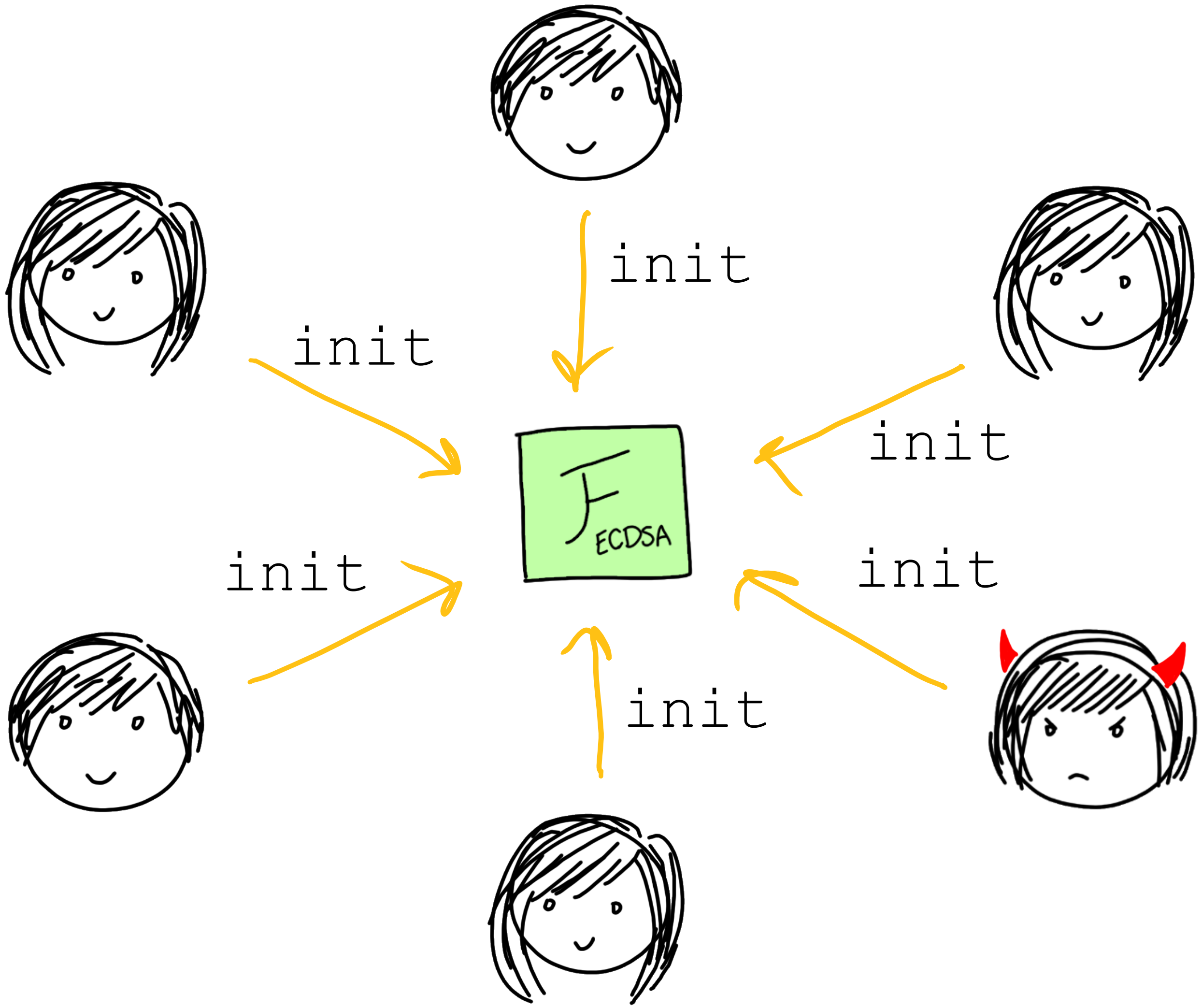
Any Adv in the real world can be mapped to one in the ideal world

ECDSA Functionality

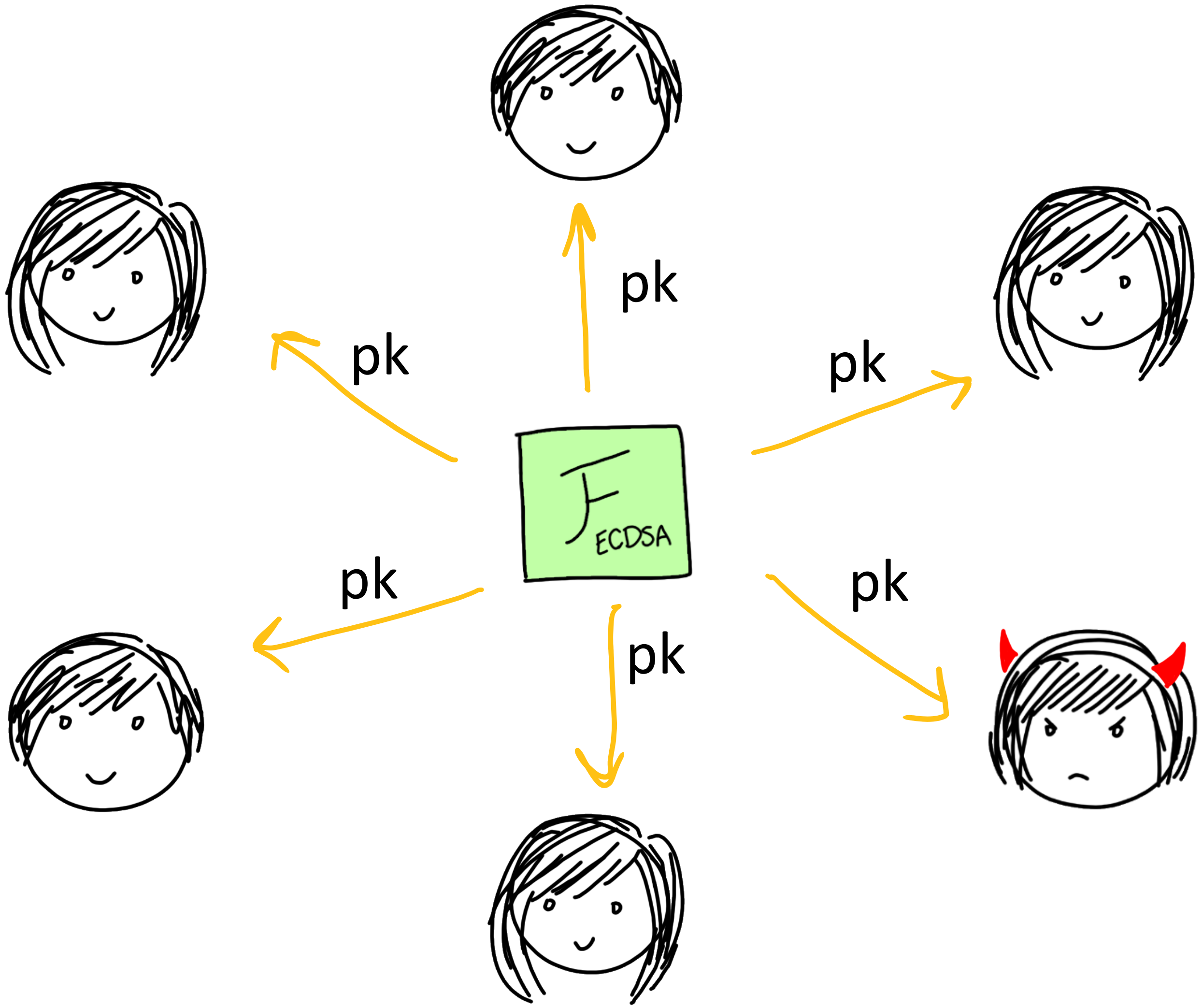
Note: Our functionality concretely implements the ECDSA algorithm and is not a signature algorithm



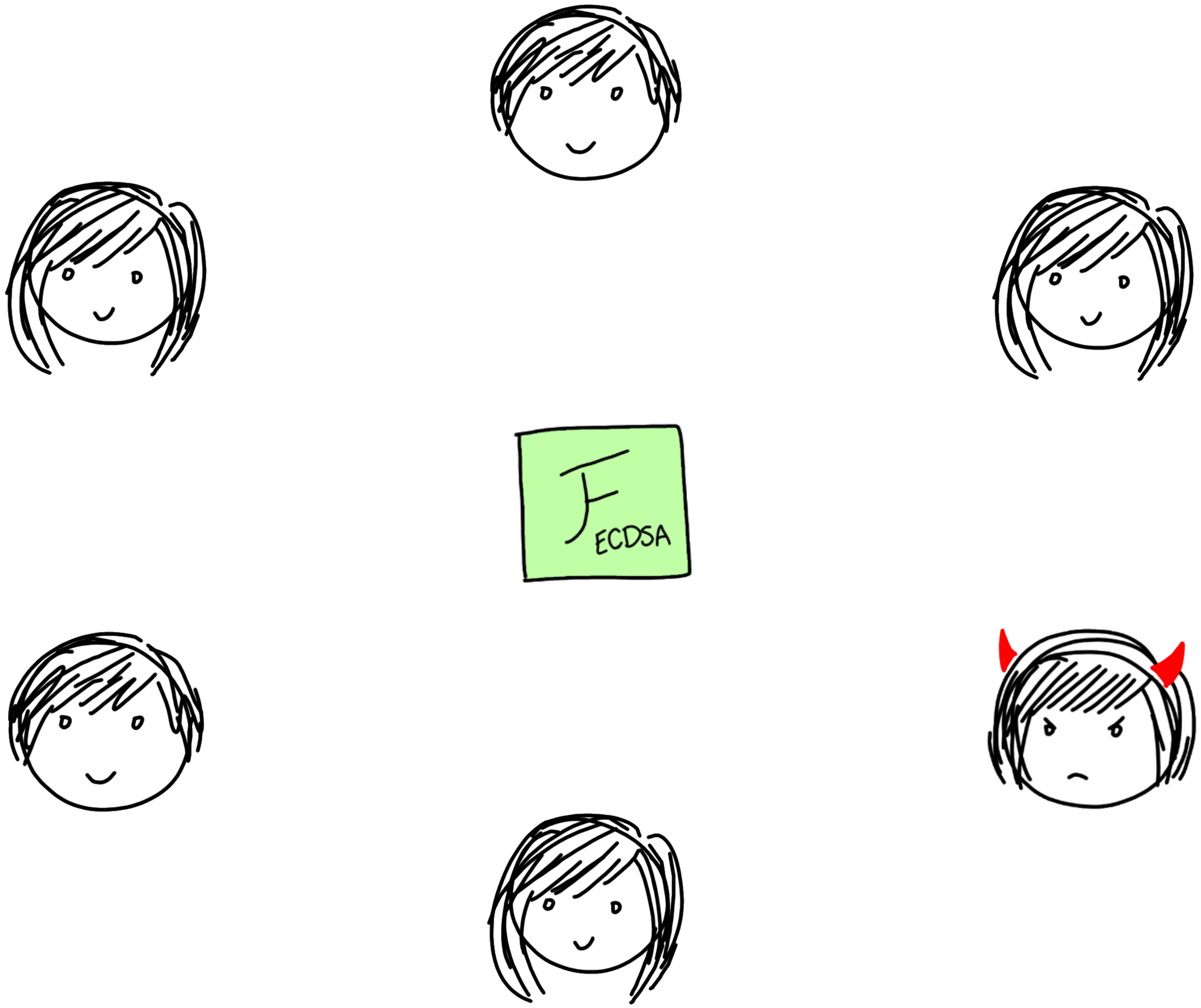
ECDSA Functionality



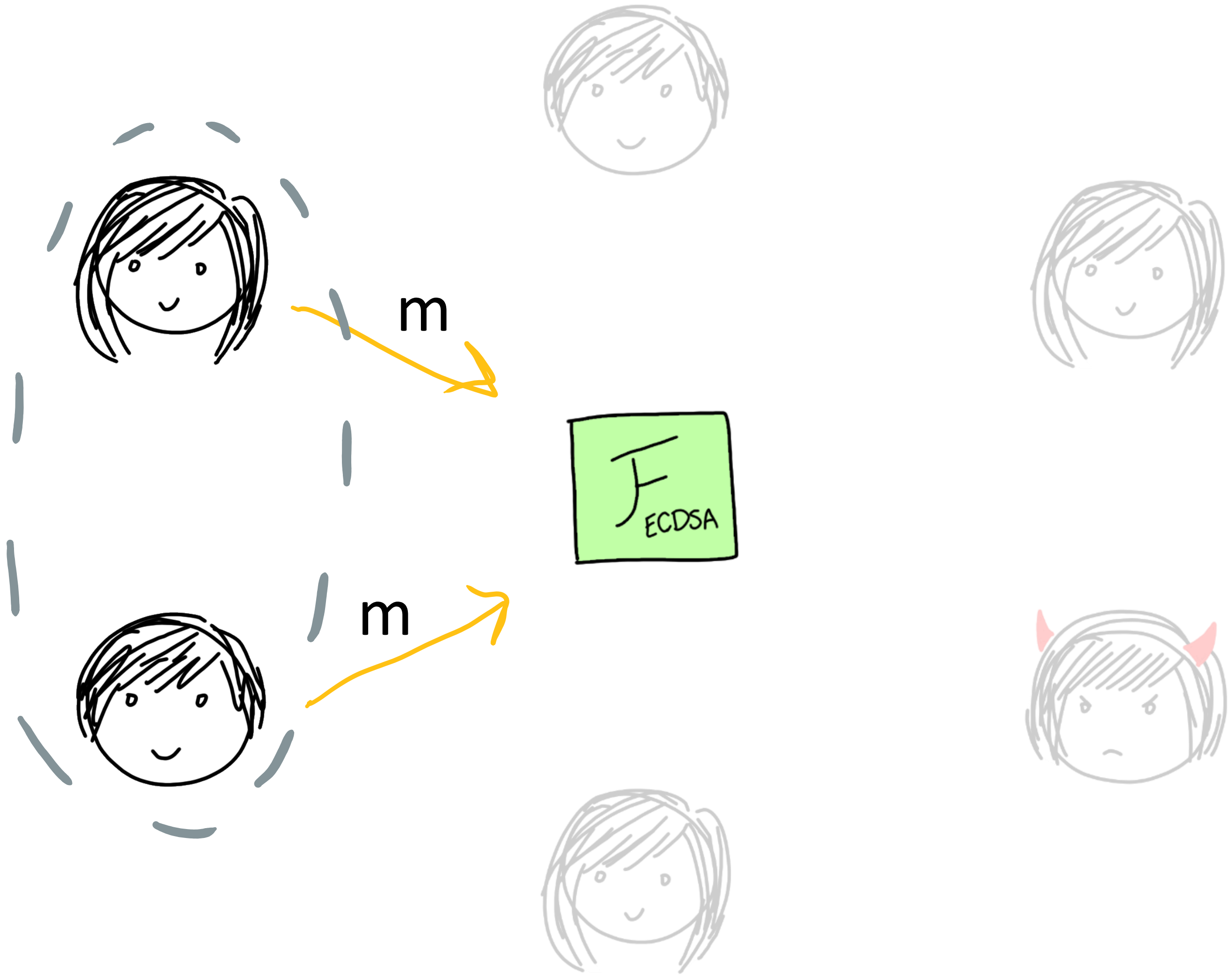
ECDSA Functionality



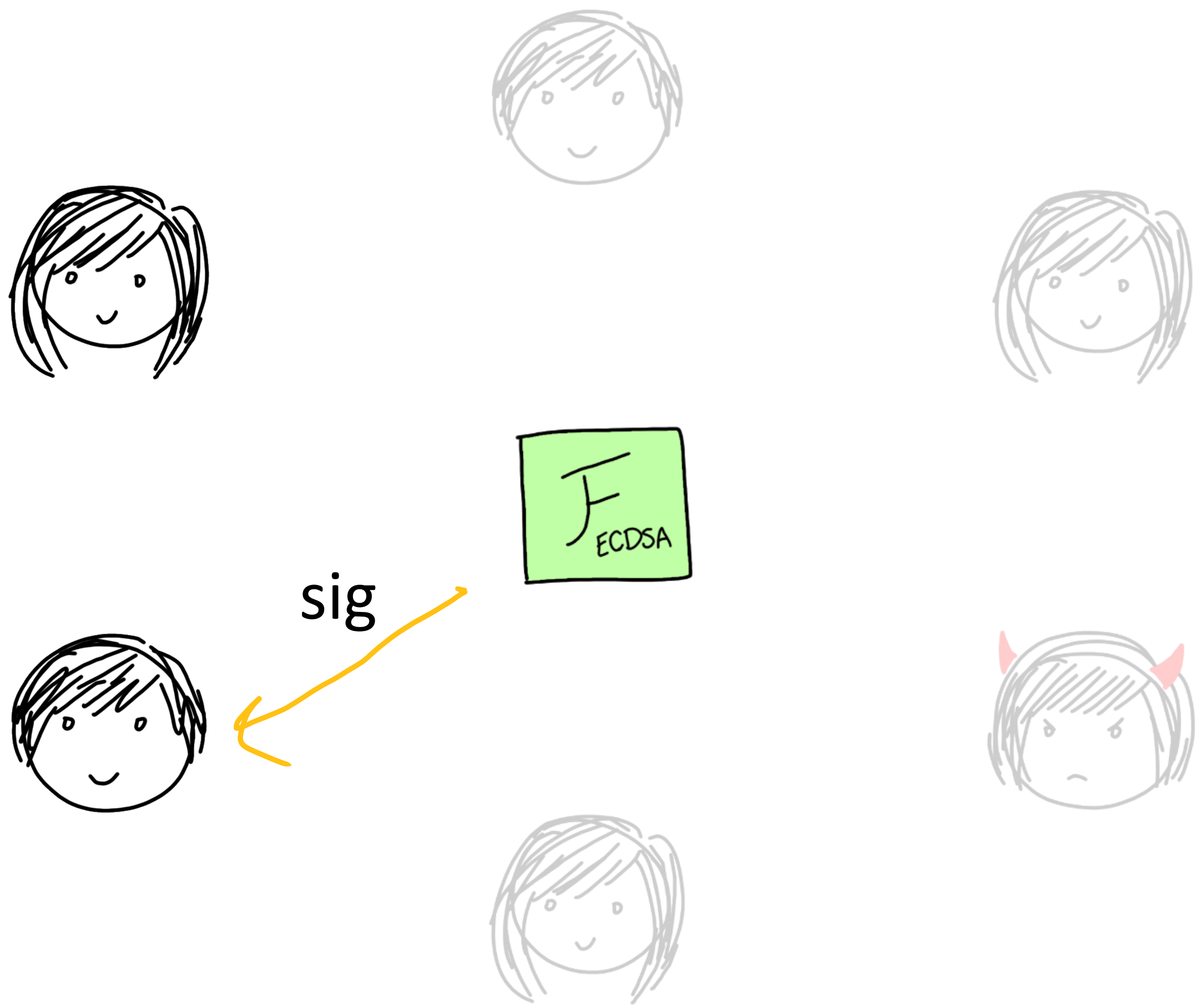
ECDSA Functionality



ECDSA Functionality



ECDSA Functionality



(Preview) Prior Works on Threshold ECDSA

- Some not proven via real/ideal
- Some have long complex, setup (several minutes), semi-honest
- All need additional assumptions

This Work

- Maliciously secure threshold ECDSA
  - 2-round with relaxed definition
  - Maliciously secure multiplication with external checks
- No additional assumptions
  - Threshold ECDSA scheme from only ECDSA
- Improved efficiency
  - ~3 ms to sign
- Open source implementation in Rust

This Talk

- 2-of-2 Threshold ECDSA
  - Extended to 2-of-n in paper
- Optimizations

Threshold Schnorr

$\mathrm{SchnorrSign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)

Threshold Schnorr

SchnorrSign(sk, m):

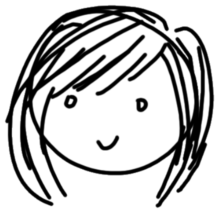
Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)



sk_a

k_a

$\mathrm{sk} = \mathrm{sk}_a + \mathrm{sk}_b$

$k = k_a + k_b$

$R = k_a \cdot G + k_b \cdot G$

sk_b

k_b

$\sigma = k - \mathrm{sk} \cdot e$

$k_a + k_b - (\mathrm{sk}_a + \mathrm{sk}_b) \cdot e$

Threshold Schnorr

SchnorrSign(sk, m):

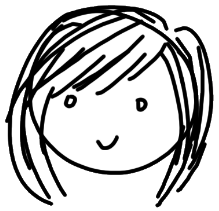
Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)



sk_a

k_a

$\mathrm{sk} = \mathrm{sk}_a + \mathrm{sk}_b$

$k = k_a + k_b$

sk_b

k_b

$R = k_a \cdot G + k_b \cdot G$

$\sigma = k - \mathrm{sk} \cdot e$

$k_a + k_b - (\mathrm{sk}_a + \mathrm{sk}_b) \cdot e$

Threshold Schnorr

SchnorrSign(sk, m):

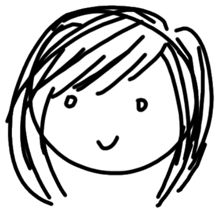
Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\rightarrow \sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)



$\mathrm{sk} = \mathrm{sk}_a + \mathrm{sk}_b$

$k = k_a + k_b$

sk_a

k_a

$R = k_a \cdot G + k_b \cdot G$

sk_b

k_b

$\sigma = k - \mathrm{sk} \cdot e$

$\downarrow$

$k_a + k_b - (\mathrm{sk}_a + \mathrm{sk}_b) \cdot e$

Threshold Schnorr

SchnorrSign(sk, m):

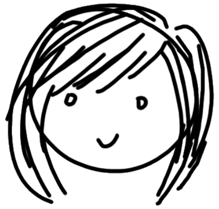
Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

→ Output (σ, e)



$\mathrm{sk} = \mathrm{sk}_a + \mathrm{sk}_b$

$k = k_a + k_b$

sk_a

k_a

$R = k_a \cdot G + k_b \cdot G$

sk_b

k_b

$\sigma = k - \mathrm{sk} \cdot e$

$k_a + k_b - (\mathrm{sk}_a + \mathrm{sk}_b) \cdot e$

$\sigma_a = k_a - \mathrm{sk}_a \cdot e$

$\sigma_b = k_b - \mathrm{sk}_b \cdot e$

$\xrightarrow{\sigma_a}$

$\sigma = \sigma_a + \sigma_b$

What makes ECDSA difficult?

$\mathrm{SchnorrSign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)

$\mathrm{ECDSASign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q^*$

$R = k \cdot G$

$e = H(m)$

$\sigma = \frac{e}{k} + \frac{\mathrm{sk}}{k} \cdot r_x$

Output (σ, r_x)

What makes ECDSA difficult?

$\mathrm{SchnorrSign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)

$\mathrm{ECDSASign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q^*$

$R = k \cdot G$

$e = H(m)$

$\sigma = \frac{e}{k} + \frac{\mathrm{sk}}{k} \cdot r_x$

Output (σ, r_x)

What makes ECDSA difficult?

$\mathrm{SchnorrSign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q$

$R = k \cdot G$

$e = H(R \parallel m)$

$\sigma = k - \mathrm{sk} \cdot e$

Output (σ, e)

$\mathrm{ECDSASign}(\mathrm{sk}, m)$:

Sample instance key $k \leftarrow \mathbb{Z}_q^*$

$R = k \cdot G$

$e = H(m)$

$\sigma = \frac{e}{k} + \frac{\mathrm{sk}}{k} \cdot r_x$

Output (σ, r_x)

Need shares of k and k^{-1}

Prior Approaches

Gennaro-Goldfeder-Narayanan16

Lindell17

Boneh-Gennaro-Goldfeder17

1. Multiplicative shares of the secret and instance keys

$$k = k_a \cdot k_b \qquad \mathrm{sk} = \mathrm{sk}_a \cdot \mathrm{sk}_b$$

$$\frac{1}{k} \cdot (H(m) + \mathrm{sk} \cdot r_x)$$

Prior Approaches

Gennaro-Goldfeder-Narayanan16

Lindell17

Boneh-Gennaro-Goldfeder17

1. Multiplicative shares of the secret and instance keys
2. Use additively homomorphic Paillier encryption

$$k = k_a \cdot k_b \qquad \mathrm{sk} = \mathrm{sk}_a \cdot \mathrm{sk}_b$$

$$\frac{1}{k} \cdot (H(m) + \mathrm{sk} \cdot r_x)$$

$$\frac{1}{k_a} \cdot \underbrace{\left(\frac{1}{k_b} \cdot H(m) + \frac{\mathrm{sk}}{k_b} \cdot r_x\right)}_{\text{Paillier encryption}}$$

Prior Approaches

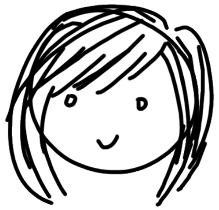
GGN16, BGG17

- t-of-n, 4 rounds (reduced from 6 rounds)
- Expensive setup; not implemented or not reported
- Additional assumptions:
  - Decisional Composite Residuosity
  - Strong RSA

Lindell17

- Only 2-of-2, 4 rounds
- Additional assumptions:
  - Decisional Composite Residuosity
  - Paillier-EC (new, construction-specific)

Our Approach to Threshold Signing



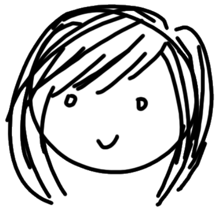
sk_a

$$\mathrm{pk} = \mathrm{sk} \cdot G$$

$$\mathrm{sk} = \mathrm{sk}_a \cdot \mathrm{sk}_b$$

sk_b

Our Approach to Threshold Signing



sk_a

k_a

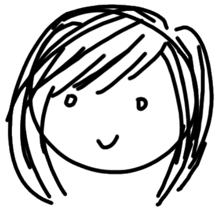
$$R = k \cdot G$$

$$k = k_a \cdot k_b$$

sk_b

k_b

Our Approach to Threshold Signing



sk_a

k_a

$$\frac{1}{k} \cdot H(m) + \frac{\mathrm{sk}}{k} \cdot r_x$$

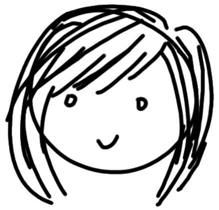
$$\Downarrow$$

$$\frac{1}{k_a} \cdot \frac{1}{k_b} \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$

sk_b

k_b

Our Approach to Threshold Signing



sk_a

k_a

$$\frac{1}{k} \cdot H(m) + \frac{\mathrm{sk}}{k} \cdot r_x$$

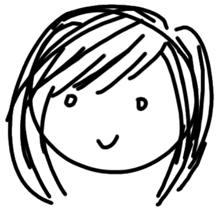
$$\Downarrow$$

$$\frac{1}{k_a} \cdot \frac{1}{k_b} \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$

sk_b

k_b

Our Approach to Threshold Signing



sk_a

k_a

$$\frac{1}{k} \cdot H(m) + \frac{\mathrm{sk}}{k} \cdot r_x$$

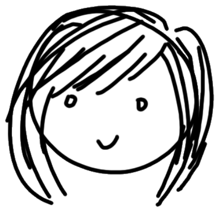
$$\downarrow$$

$$\frac{1}{k_a} \cdot \frac{1}{k_b} \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$

sk_b

k_b

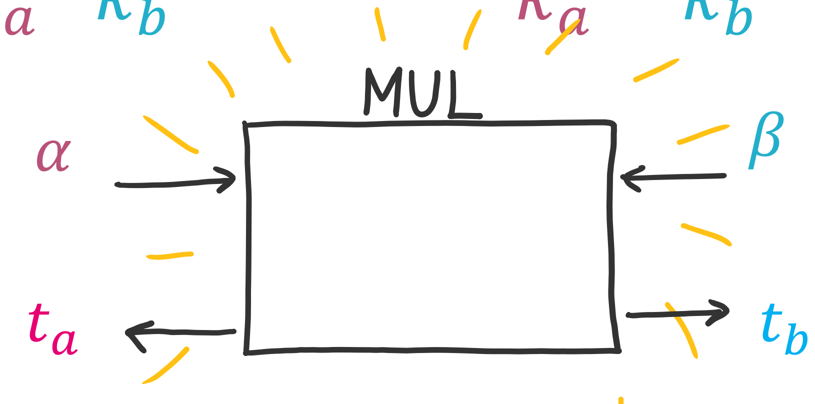
Our Approach to Threshold Signing



sk_a

k_a

$$\frac{1}{k_a} \cdot \frac{1}{k_b} \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$

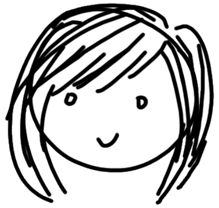


$$t_a + t_b = \alpha \cdot \beta$$

sk_b

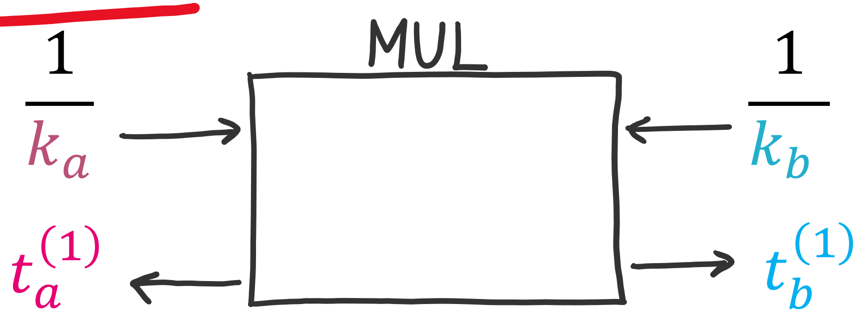
k_b

Our Approach to Threshold Signing



sk_a
k_a

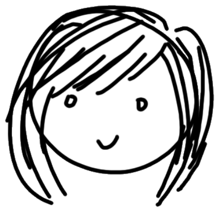
$$\underline{\frac{1}{k_a} \cdot \frac{1}{k_b}} \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$



sk_b
k_b

$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k_a} \cdot \frac{1}{k_b}$$

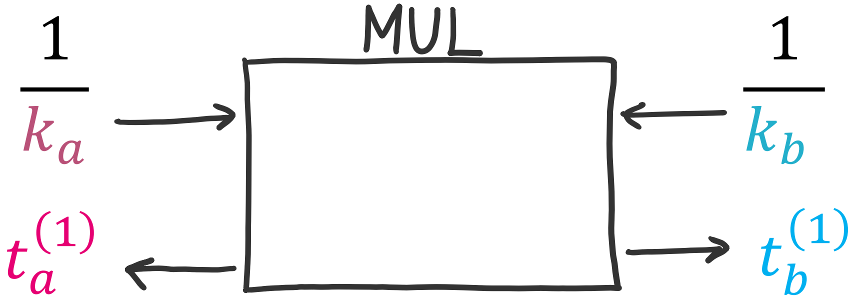
Our Approach to Threshold Signing



sk_a
k_a

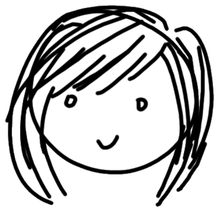
$$\left(\underline{t_a^{(1)} + t_b^{(1)}}\right) \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$

sk_b
k_b



$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k_a} \cdot \frac{1}{k_b}$$

Our Approach to Threshold Signing



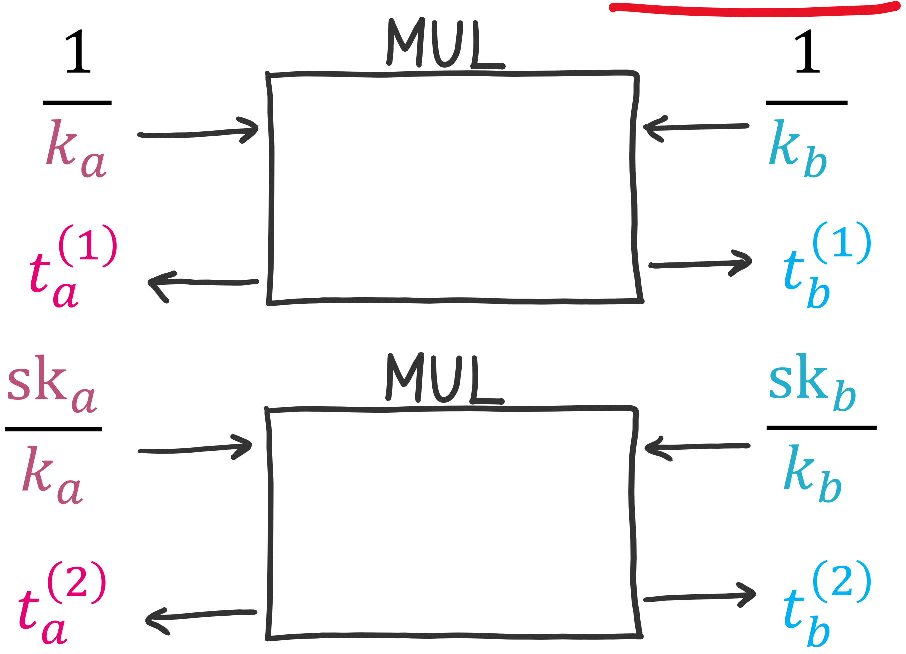
sk_a

k_a

$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot H(m) + \underline{\frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b}} \cdot r_x$$

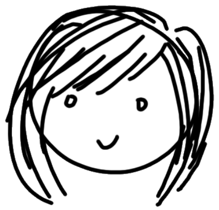
sk_b

k_b



$$t_a^{(2)} + t_b^{(2)} = \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b}$$

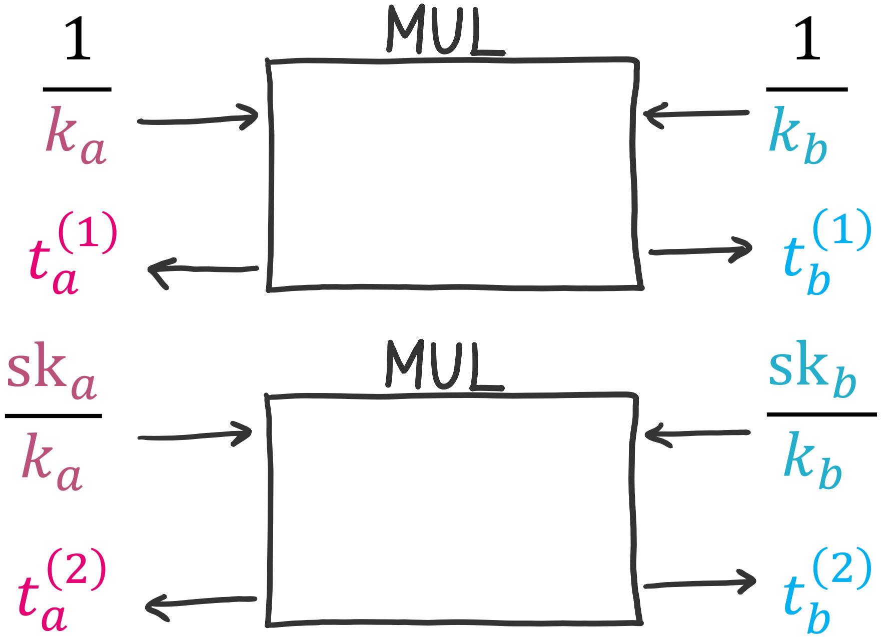
Our Approach to Threshold Signing



$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot H(m) + \underline{\left(t_a^{(2)} + t_b^{(2)}\right)} \cdot r_x$$

sk_a
k_a

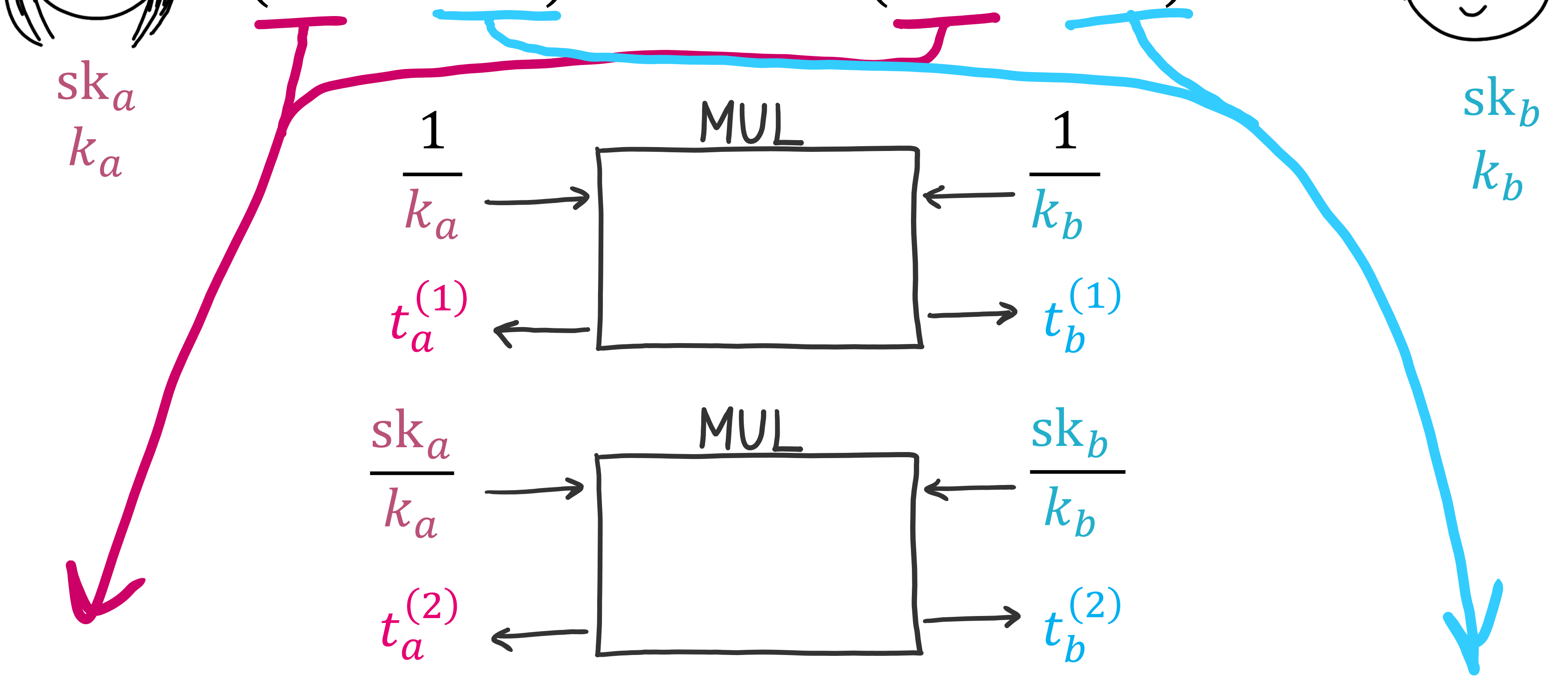
sk_b
k_b



$$t_a^{(2)} + t_b^{(2)} = \frac{\mathrm{sk}_a}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b}$$

Our Approach to Threshold Signing

$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot H(m) + \left(t_a^{(2)} + t_b^{(2)}\right) \cdot r_x$$

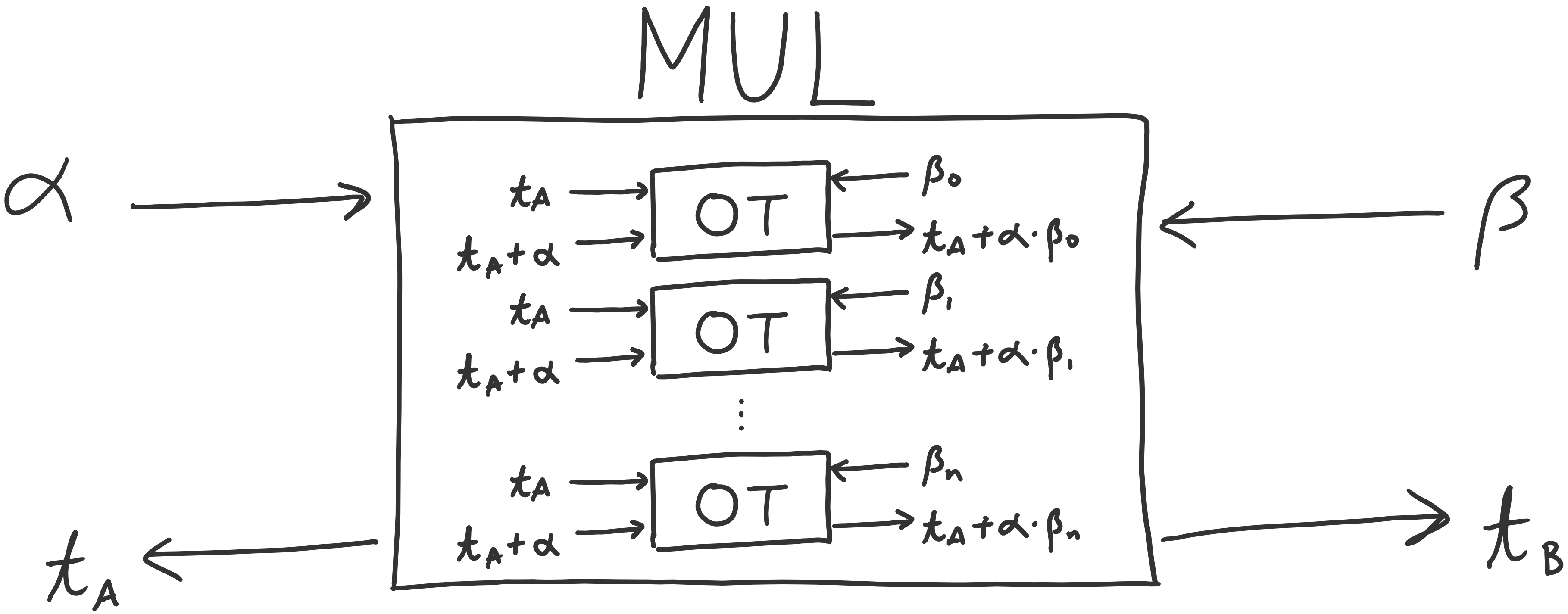


$t_a^{(1)} \cdot H(m) + t_a^{(2)} \cdot r_x$

$t_b^{(1)} \cdot H(m) + t_b^{(2)} \cdot r_x$

(Semi-honest)

[Gilboa99] Multiplication by Oblivious Transfer

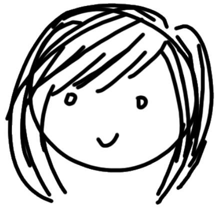


# of OTs proportional to security parameter

Efficient with OT extension (symmetric key operations)

Skeleton Protocol

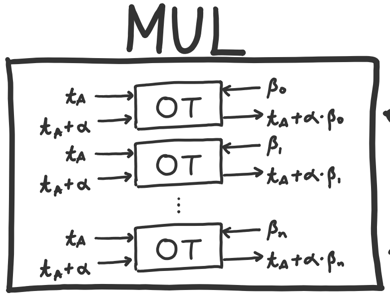
$$pk = \mathrm{sk}_a \cdot \mathrm{sk}_b \cdot G$$
$$R = k_a \cdot k_b \cdot G$$



sk_a
k_a

sk_b
k_b

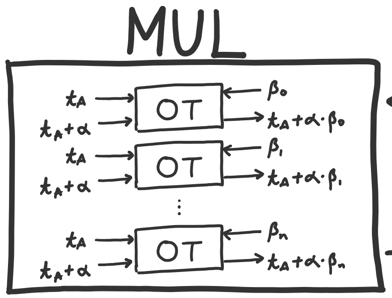
$\frac{1}{k_a}$ $\rightarrow$



$\leftarrow$ $\frac{1}{k_b}$

$t_a^{(1)}$ $\leftarrow$ $\rightarrow$ $t_b^{(1)}$

$\frac{\mathrm{sk}_a}{k_a}$ $\rightarrow$



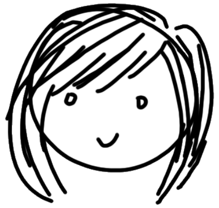
$\leftarrow$ $\frac{\mathrm{sk}_b}{k_b}$

$t_a^{(2)}$ $\leftarrow$ $\rightarrow$ $t_b^{(2)}$

$$\sigma_a = t_a^{(1)} \cdot H(m) + t_a^{(2)} \cdot r_x \xrightarrow{\sigma_a} \quad t_b^{(1)} \cdot H(m) + t_b^{(2)} \cdot r_x$$

Hardening for Malicious Security

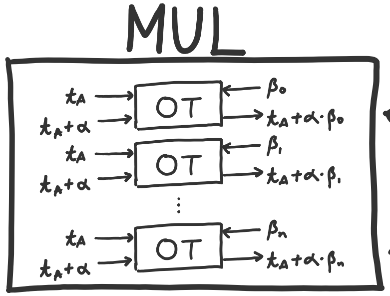
$$pk = \mathrm{sk}_a \cdot \mathrm{sk}_b \cdot G$$
$$R = k_a \cdot k_b \cdot G$$



sk_a
k_a

sk_b
k_b

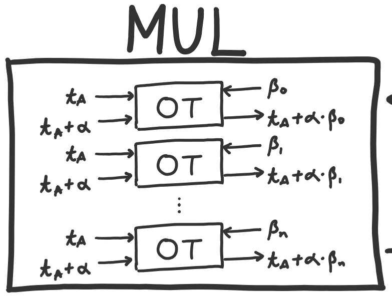
$\frac{1}{k_a}$ →



← $\frac{1}{k_b}$

$t_a^{(1)}$ ← → $t_b^{(1)}$

$\frac{\mathrm{sk}_a}{k_a}$ →



← $\frac{\mathrm{sk}_b}{k_b}$

$t_a^{(2)}$ ← → $t_b^{(2)}$

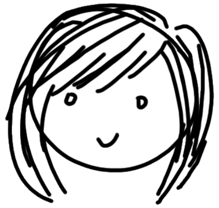
$$\sigma_a = t_a^{(1)} \cdot H(m) + t_a^{(2)} \cdot r_x$$

$$\xrightarrow{\sigma_a}$$

$$t_b^{(1)} \cdot H(m) + t_b^{(2)} \cdot r_x$$

Hardening for Malicious Security

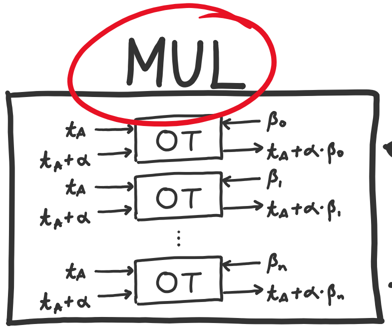
$pk = \mathrm{sk}_a \cdot \mathrm{sk}_b \cdot G$
$R = k_a \cdot k_b \cdot G$



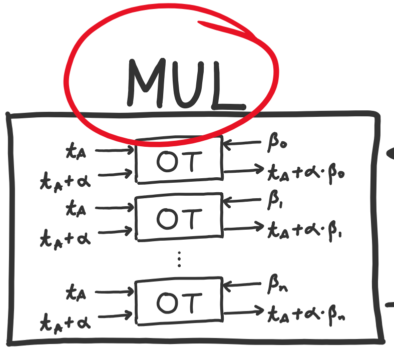
sk_a
k_a

sk_b
k_b

1. Maliciously secure multiplication

$\frac{1}{k_a}$ →  ← $\frac{1}{k_b}$

$t_a^{(1)}$ ← → $t_b^{(1)}$

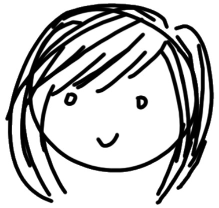
$\frac{\mathrm{sk}_a}{k_a}$ →  ← $\frac{\mathrm{sk}_b}{k_b}$

$t_a^{(2)}$ ← → $t_b^{(2)}$

$\sigma_a = t_a^{(1)} \cdot H(m) + t_a^{(2)} \cdot r_x$ $\xrightarrow{\sigma_a}$ $t_b^{(1)} \cdot H(m) + t_b^{(2)} \cdot r_x$

Hardening for Malicious Security

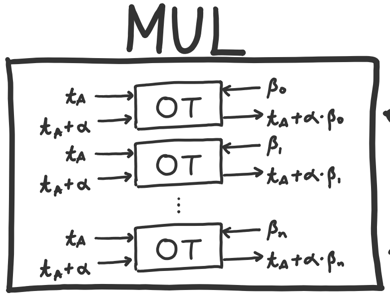
$$pk = \mathrm{sk}_a \cdot \mathrm{sk}_b \cdot G$$
$$R = k_a \cdot k_b \cdot G$$



sk_a
k_a

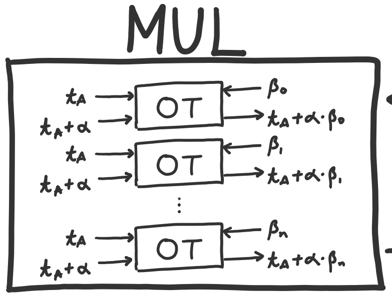
sk_b
k_b

$\frac{1}{k_a}$ → MUL ← $\frac{1}{k_b}$



$t_a^{(1)}$ ← → $t_b^{(1)}$

$\frac{\mathrm{sk}_a}{k_a}$ → MUL ← $\frac{\mathrm{sk}_b}{k_b}$



$t_a^{(2)}$ ← → $t_b^{(2)}$

1. Maliciously secure multiplication
2. Enforce input consistency

$$\sigma_a = t_a^{(1)} \cdot H(m) + t_a^{(2)} \cdot r_x \xrightarrow{\sigma_a} t_b^{(1)} \cdot H(m) + t_b^{(2)} \cdot r_x$$

Hardening for Malicious Security

Malicious Multiplication

1. Checks per OT
   - ½ probability getting caught per OT
2. High entropy encoding scheme
   - Bob encodes his input into multiplication

Input Consistency

Verify output is an ECDSA signature

A new consistency check

Assumptions Needed

Malicious Multiplication

1. Checks per OT
2. High entropy encoding scheme

statistical in ROM

Input Consistency

Verify output is an ECDSA signature

ECDSA is a signature scheme

CDH implied by generic group model, which is what ECDSA is proven in

A new consistency check

Computational Diffie-Hellman

Security Against Malicious Bob

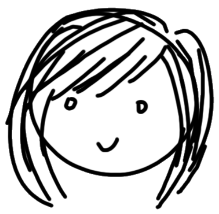
$t_a^{(1)}$
$t_a^{(2)}$

$$\left.\begin{aligned} t_a^{(1)} + t_b^{(1)} &= \frac{1}{k} \\ t_a^{(2)} + t_b^{(2)} &= \frac{\mathrm{sk}}{k} \end{aligned}\right\}$$

Goal: Enforce Consistency

$t_b^{(1)}$
$t_b^{(2)}$

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

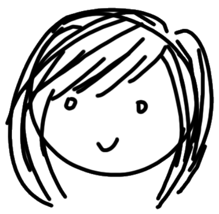
$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k}$$

$$t_a^{(2)} + t_b^{(2)} = \frac{\mathrm{sk}}{k}$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

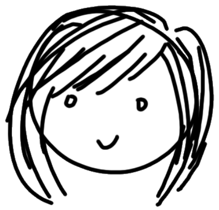
$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k}$$

$$\left(t_a^{(2)} + t_b^{(2)}\right) \cdot G = \left(\frac{\mathrm{sk}}{k}\right) \cdot G$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k}$$

$$\left(t_a^{(2)} + t_b^{(2)}\right) \cdot G = \left(\frac{\mathrm{sk}}{k}\right) \cdot G$$

$$= \frac{1}{k} \cdot \mathrm{pk}$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob

$t_a^{(1)}$

$t_a^{(2)}$

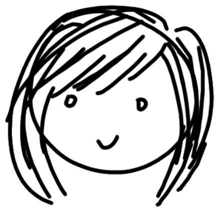
$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k}$$

$$\left(t_a^{(2)} + t_b^{(2)}\right) \cdot G = \frac{1}{k} \cdot \mathrm{pk}$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

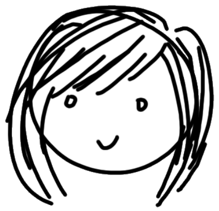
$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k}$$

$$\left(t_a^{(2)} + t_b^{(2)}\right) \cdot G = \frac{1}{k} \cdot \mathrm{pk}$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

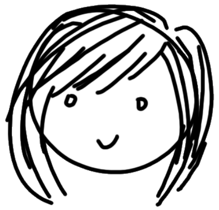
$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot \mathrm{pk} = \frac{1}{k} \cdot \mathrm{pk}$$

$$\left(t_a^{(2)} + t_b^{(2)}\right) \cdot G = \frac{1}{k} \cdot \mathrm{pk}$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot pk = \frac{1}{k} \cdot pk$$

$$\left(t_a^{(2)} + t_b^{(2)}\right) \cdot G = \frac{1}{k} \cdot pk$$

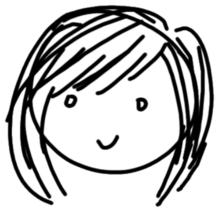
$$\Downarrow$$

$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot pk = \left(t_a^{(2)} + t_b^{(2)}\right) \cdot G$$

$t_b^{(1)}$

$t_b^{(2)}$

Security Against Malicious Bob



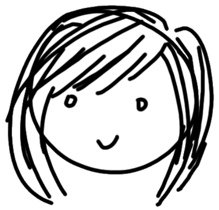
$t_a^{(1)}$
$t_a^{(2)}$

$$\left(t_a^{(1)} + t_b^{(1)}\right) \cdot pk = \left(t_a^{(2)} + t_b^{(2)}\right) \cdot G$$

$t_b^{(1)}$
$t_b^{(2)}$

$$t_a^{(1)} \cdot pk - t_a^{(2)} \cdot G = t_b^{(2)} \cdot G - t_b^{(1)} \cdot pk$$

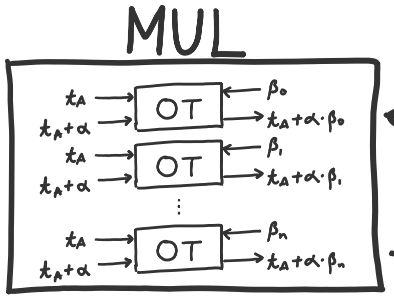
Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

$\dfrac{1}{k_a}$



$\dfrac{1}{k_b} + \delta$

$t_a^{(1)}$ $\quad$ $t_b^{(1)}$

$t_b^{(1)}$

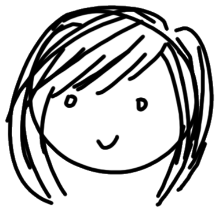
$t_b^{(2)}$

$$t_a^{(1)} + t_b^{(1)} = \frac{1}{k} + \frac{\delta}{k_a}$$

$$t_a^{(1)} \cdot pk - t_a^{(2)} \cdot G = t_b^{(2)} \cdot G - t_b^{(1)} \cdot pk + \frac{\delta}{k_a} \cdot pk$$

Computing this is as hard as CDH!

Security Against Malicious Bob



$t_a^{(1)}$

$t_a^{(2)}$

$$\Gamma: \quad t_a^{(1)} \cdot pk - t_a^{(2)} \cdot G = t_b^{(2)} \cdot G - t_b^{(1)} \cdot pk$$

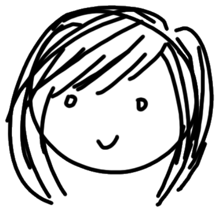
$t_b^{(1)}$

$t_b^{(2)}$

$$\xleftarrow{\quad\Gamma\quad}$$

$$\xrightarrow{\quad\sigma_a\quad}$$

Consistency Check Optimization



$t_a^{(1)}$

$t_a^{(2)}$

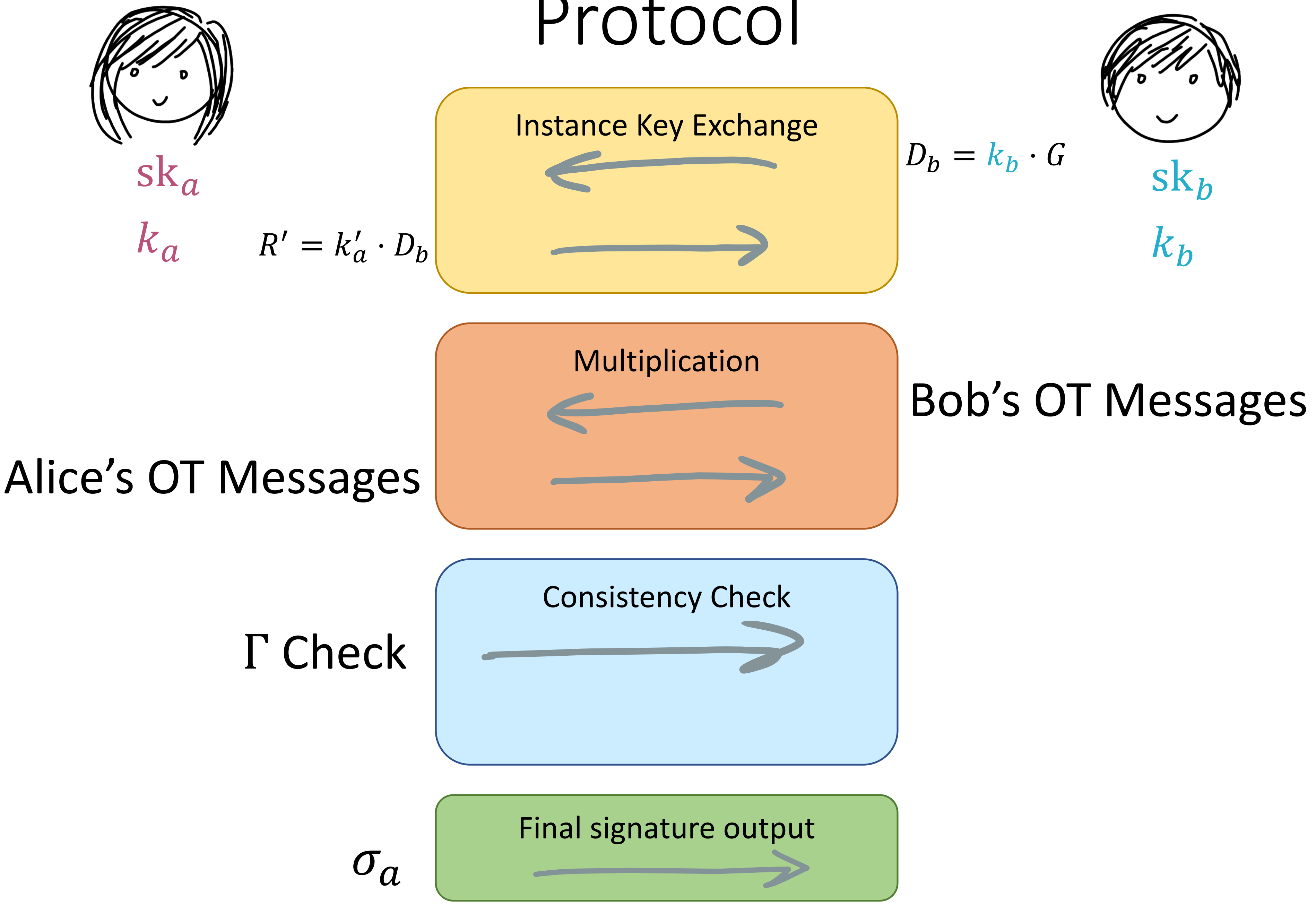
$$\Gamma: \quad t_a^{(1)} \cdot pk - t_a^{(2)} \cdot G = t_b^{(2)} \cdot G - t_b^{(1)} \cdot pk$$

$t_b^{(1)}$

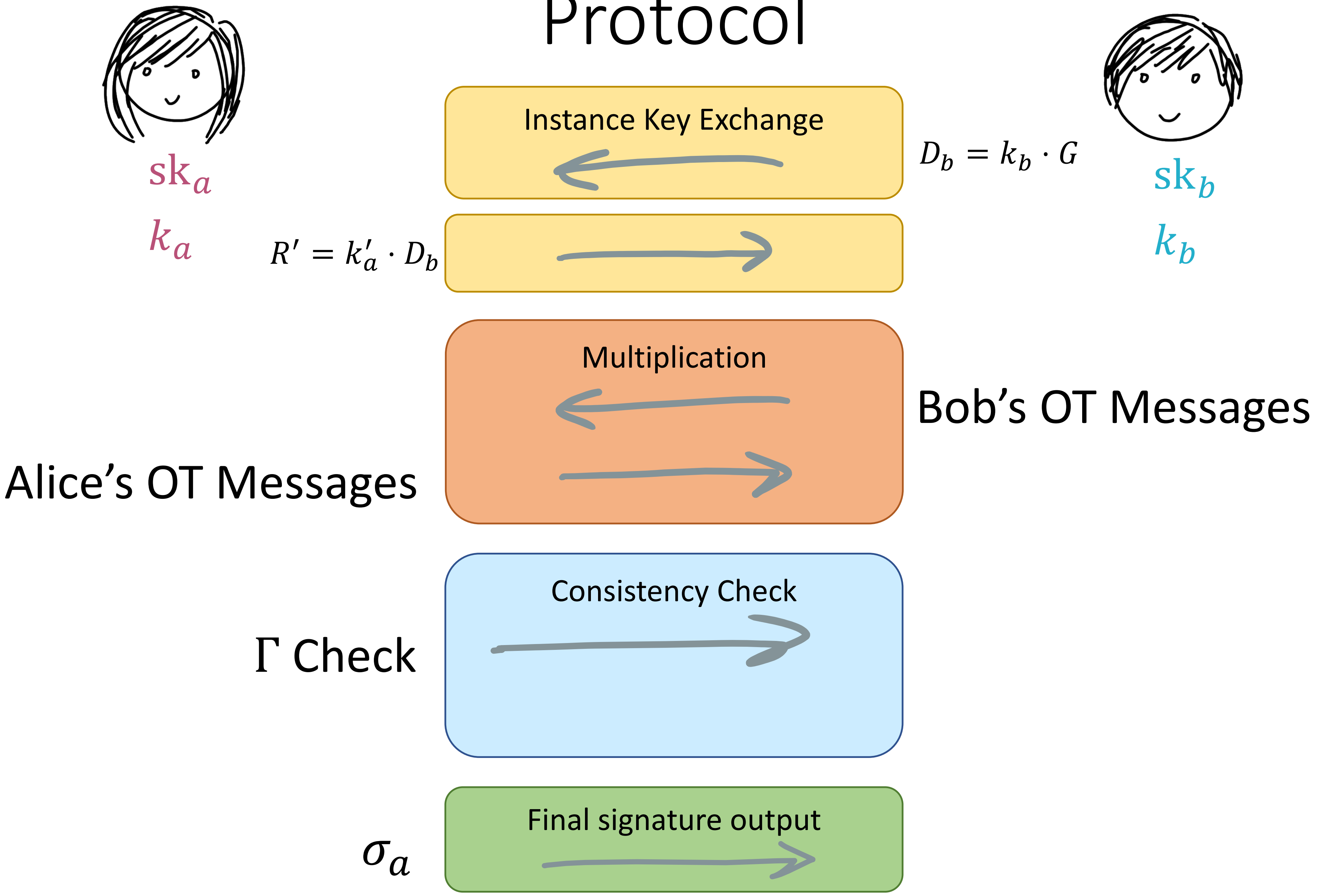
$t_b^{(2)}$

$$\xrightarrow{Enc_{\Gamma}(\sigma_a)}$$

Protocol

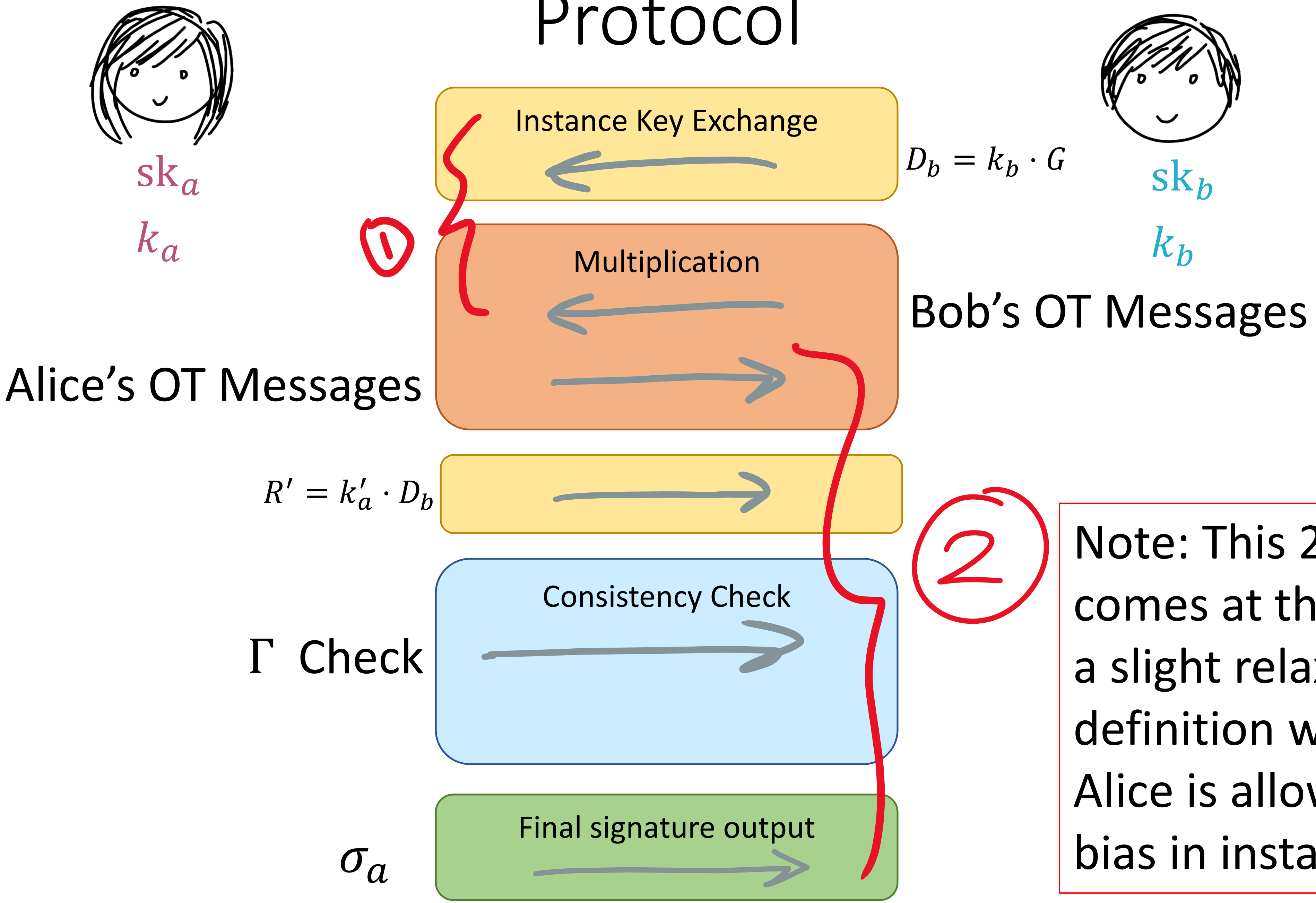

sk_a
k_a
Instance Key Exchange
$D_b = k_b \cdot G$
$R' = k'_a \cdot D_b$
sk_b
k_b
Multiplication
Bob's OT Messages
Alice's OT Messages
Consistency Check
Γ Check
Final signature output
σ_a

Protocol



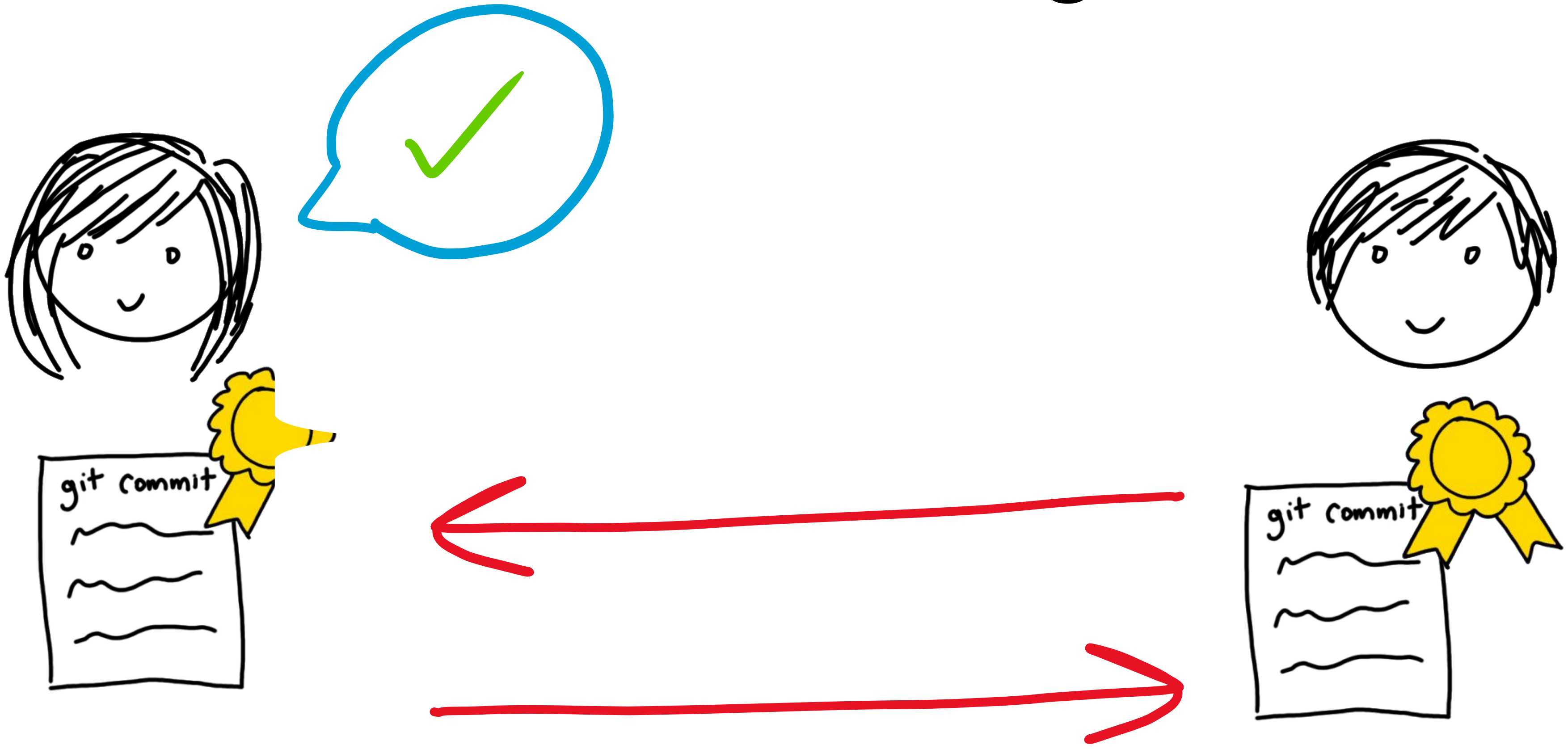
sk_a
k_a
Instance Key Exchange
$D_b = k_b \cdot G$
sk_b
k_b
$R' = k'_a \cdot D_b$
Multiplication
Bob's OT Messages
Alice's OT Messages
Consistency Check
Γ Check
Final signature output
σ_a

Protocol



Note: This 2 round comes at the cost of a slight relaxation to definition where Alice is allowed negl bias in instance key

On the Benefit of Two Messages



Why not generic MPC

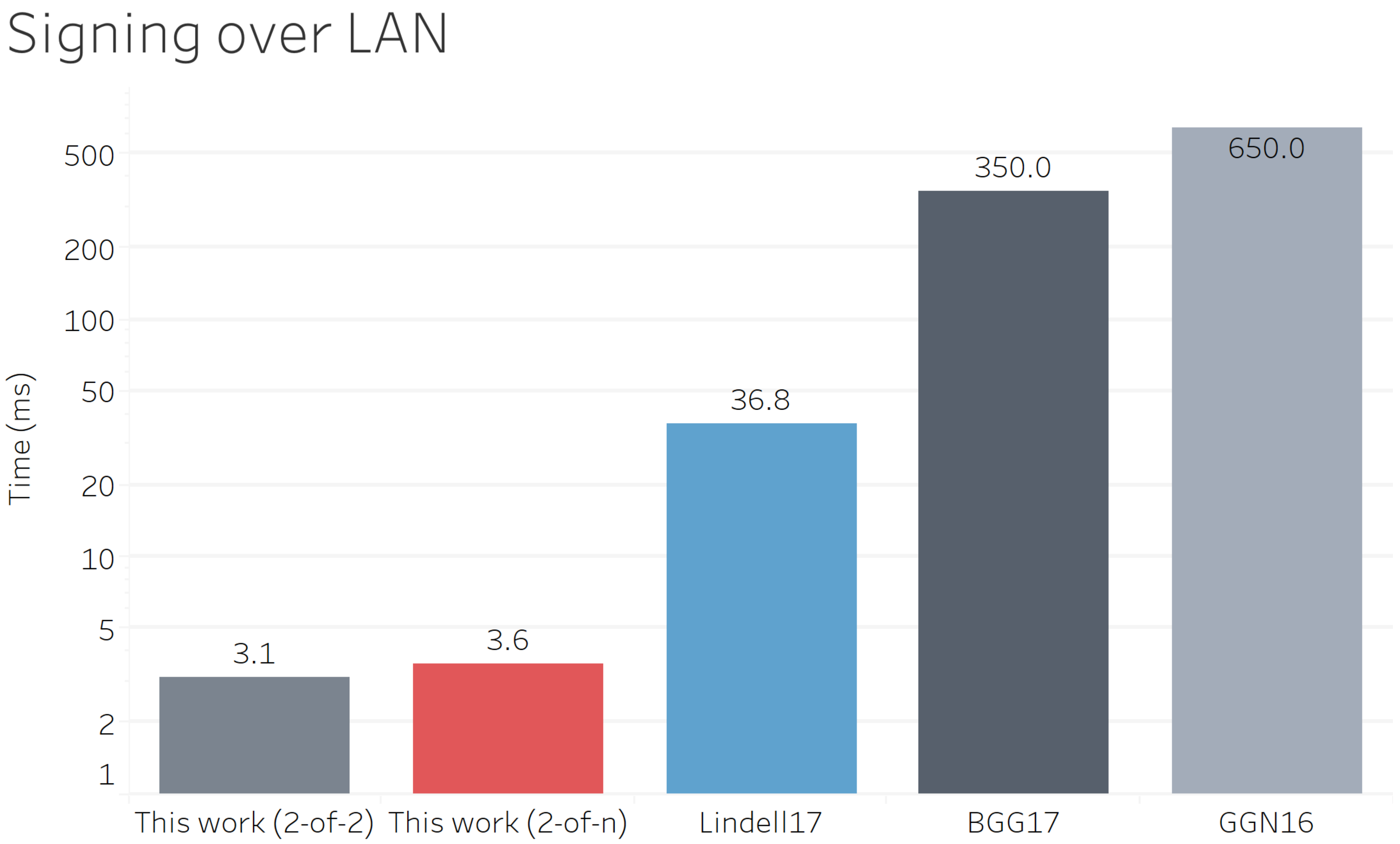
- Highly efficient multiplication in 2 rounds
  - Don’t amortize over large number of gates
- Exploit verifiability
  - Take advantage of public values with respect to signature scheme to verify inputs
  - Don’t need expensive techniques to ensure input consistency

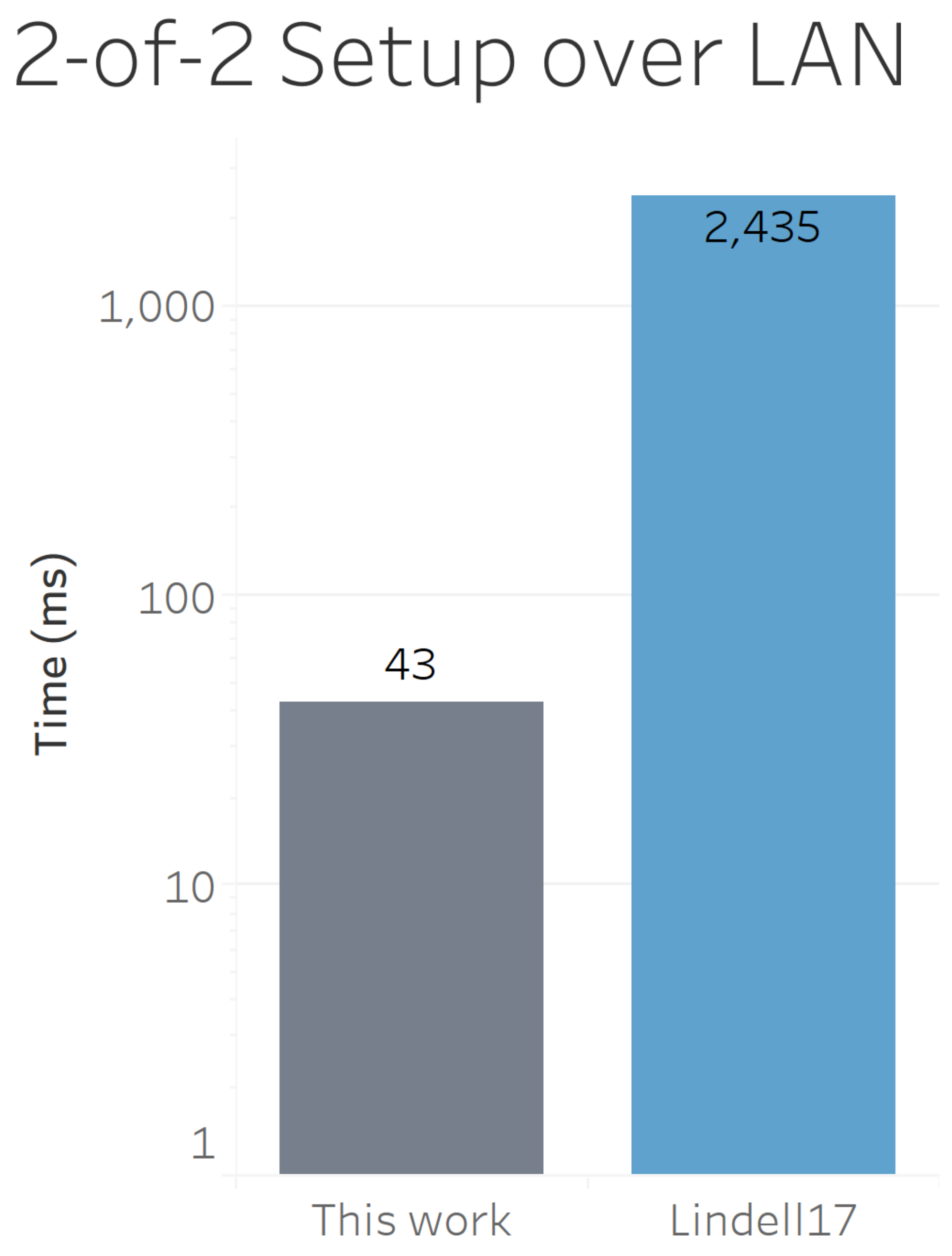
Implementation

- Open source implementation in Rust
  - SHA-256, same as ECDSA
  - 10,000 samples for setup, 100,000 samples for signing
  - Setup is 5 rounds and all n parties participate

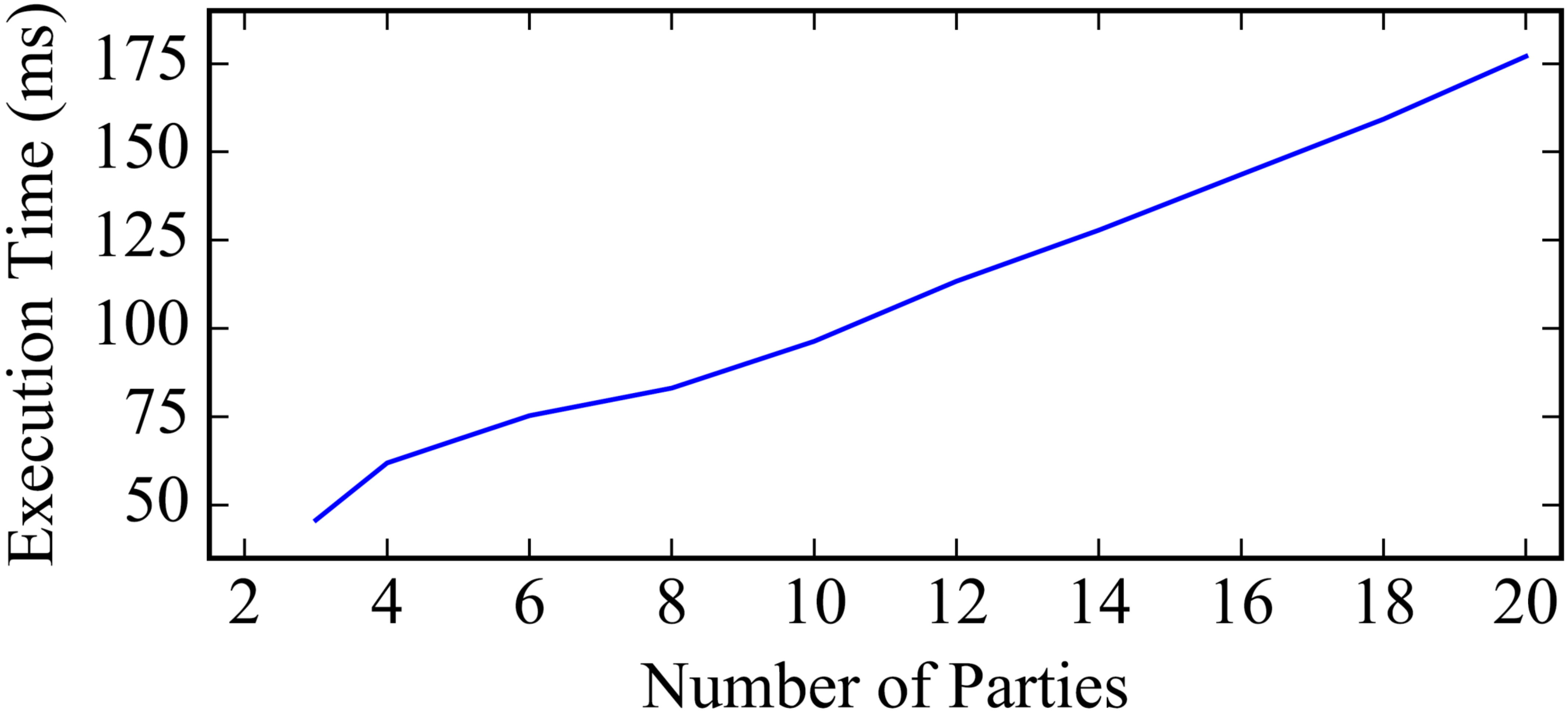
Signing Communication Costs

	$\kappa = 256$	$\kappa = 384$	$\kappa = 521$
Gennaro *et al.*	~1808 KiB	~4054 KiB	~7454 KiB
Boneh *et al.*	~1680 KiB	~3768 KiB	~6924 KiB
This Work (2-of-n)	232.8 KiB	481.3 KiB	844.7 KiB
This Work (2-of-2)	169.8 KiB	350.7 KiB	615.3 KiB
Lindell	769 B	897 B	1043 B


Signing over LAN
Time (ms)
500
200
100
50
20
10
5
2
1
3.1
3.6
36.8
350.0
650.0
This work (2-of-2)
This work (2-of-n)
Lindell17
BGG17
GGN16


2-of-2 Setup over LAN
Time (ms)
1,000
100
10
1
43
2,435
This work
Lindell17

2-of-*n* Setup over LAN



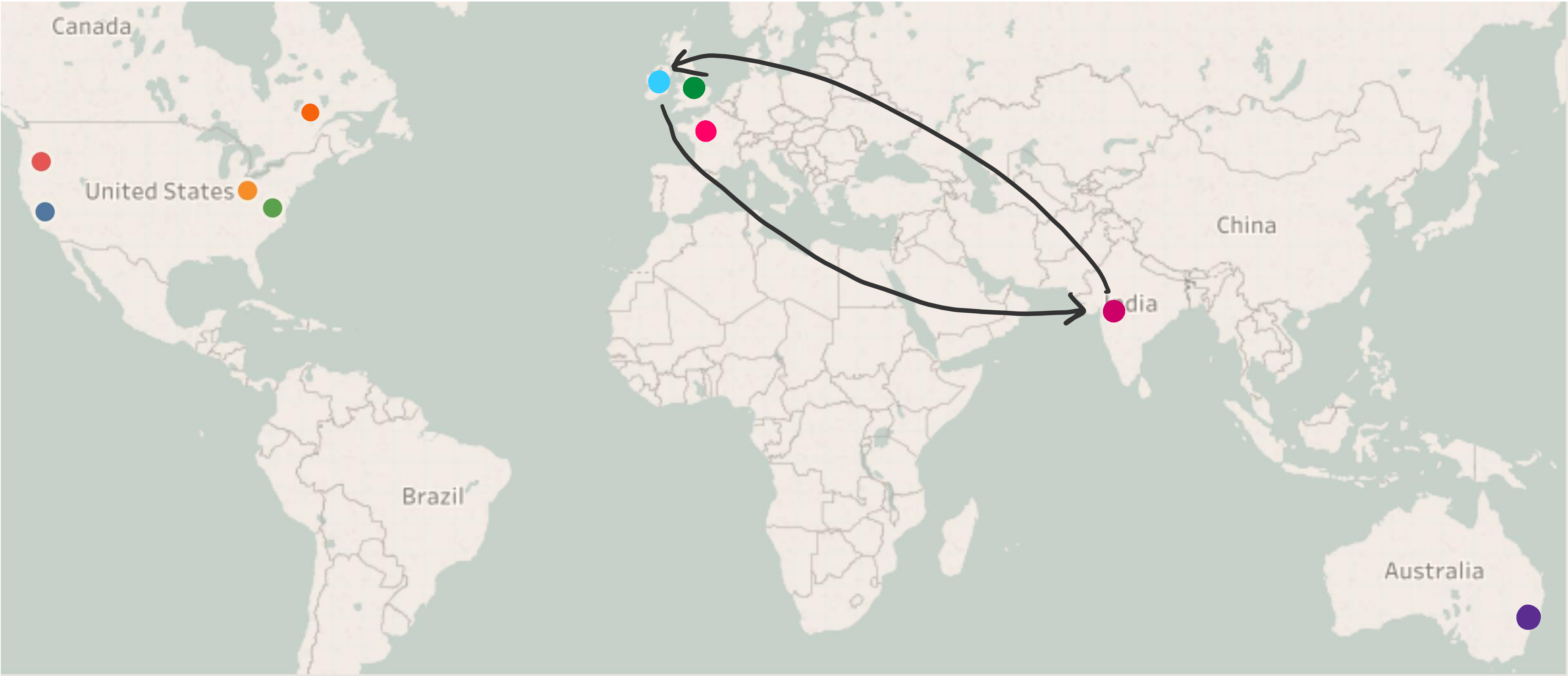
Benchmarks over WAN: 2-of-2 and 2-of-n

Round-trip latency between Virginia and Paris: 78.2 ms

Benchmarks over WAN: 2-of-4 Setup

Round-trip latency between US data centers: 11.2 ms to 79.9 ms

Benchmarks over WAN: 2-of-10 Setup



Round-trip latency between Ireland and Mumbai: 282 ms

Times in ms over WAN

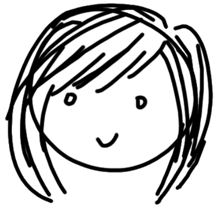
Setup			Signing	
2-of-2	2-of-4 (US)	2-of-10 (World)	2-of-2	2-of-n
354.36	376.86	1228.46	81.34	81.83

Conclusion

- ECDSA threshold with no more assumptions than ECDSA
- Improved efficiency
- Open-source implementation in Rust
  - https://gitlab.com/neucrypt/mpecdsa
- Can be extended to k-out-of-n


Thank You!

Appendix: 2-of-n Signing



$$\mathrm{sk} = \mathrm{sk}_a + \mathrm{sk}_b$$

$$\frac{1}{k_a} \cdot \frac{1}{k_b} \cdot H(m) + \frac{1}{k_a} \cdot \frac{1}{k_b} \cdot (\mathrm{sk}_a + \mathrm{sk}_b) \cdot r_x$$

$$\frac{1}{k_a} \cdot \frac{1}{k_b} \cdot H(m) + \frac{\mathrm{sk}_a}{k_a} \cdot \frac{1}{k_b} \cdot r_x + \frac{1}{k_a} \cdot \frac{\mathrm{sk}_b}{k_b} \cdot r_x$$