RSAConference2021
May 17 – 20 | Virtual Experience

RESILIENCE

SESSION ID: CRYP-R03C

Non-interactive Half-aggregation Of EdDSA And Variants Of Schnorr Signatures

Yashvanth Kondi

PhD Candidate
Northeastern University

Joint work with: Konstantinos Chalkias, François Garillot, Valeria Nikolaenko (Novi/Facebook)

In This Work, We:

- Study non-interactive aggregation of Schnorr/EdDSA signatures using methods that are blackbox in the hash function and the group
- Design and implement two constructions:
  - 50% compression, loose security, no computation overhead
  - 50-e% compression, tight security, high computation overhead
- Show that 50% compression is optimal for blackbox techniques

RSAConference2021

Schnorr Signatures

Recap of characteristics

Schnorr Signatures

Schnorr Signatures

- What's good:

Schnorr Signatures

- What's good:
  - Security under conservative, well-studied assumption (Discrete Logarithm problem)

Schnorr Signatures

- What's good:
  - Security under conservative, well-studied assumption (Discrete Logarithm problem)
  - Individual signatures are compact, fast to generate and verify

Schnorr Signatures

- What's good:
  - Security under conservative, well-studied assumption (Discrete Logarithm problem)
  - Individual signatures are compact, fast to generate and verify
  - Linear structure allows efficient interactive aggregation (i.e. threshold/multisignature friendly)

Schnorr Signatures

- What's good:
  - Security under conservative, well-studied assumption (Discrete Logarithm problem)
  - Individual signatures are compact, fast to generate and verify
  - Linear structure allows efficient interactive aggregation (i.e. threshold/multisignature friendly)
- However, no native non-interactive aggregation procedure (unlike BLS)

RSAConference2021

Aggregate Signatures

What are they?

Signature Aggregation

Signature Aggregation

pk_1

m_1

Signature Aggregation

pk_1

m_1

pk_2

m_2

Signature Aggregation

pk_1

m_1

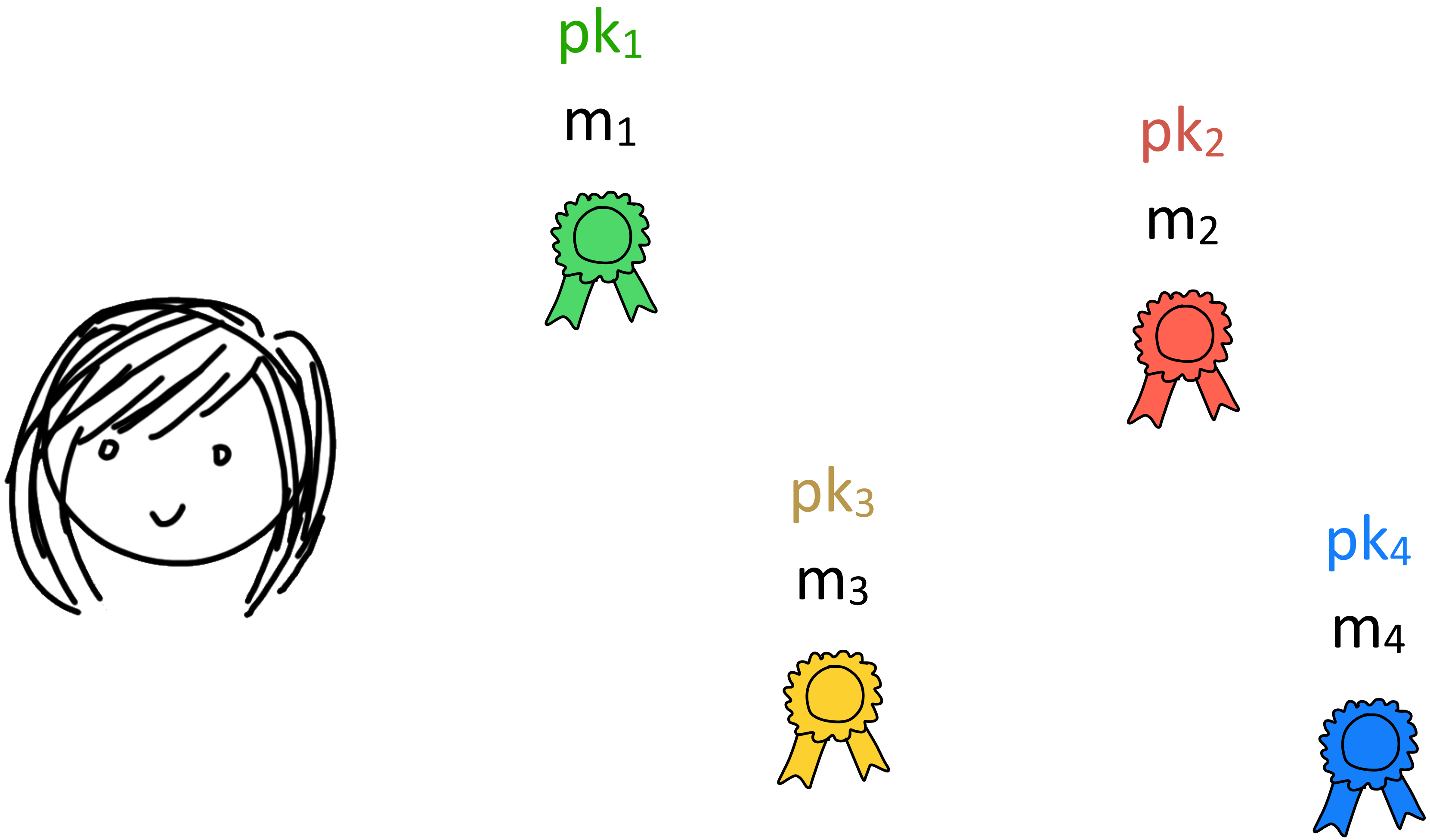
pk_2

m_2

pk_3

m_3

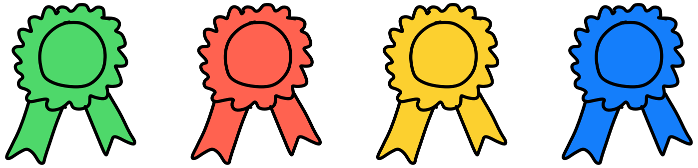
Signature Aggregation



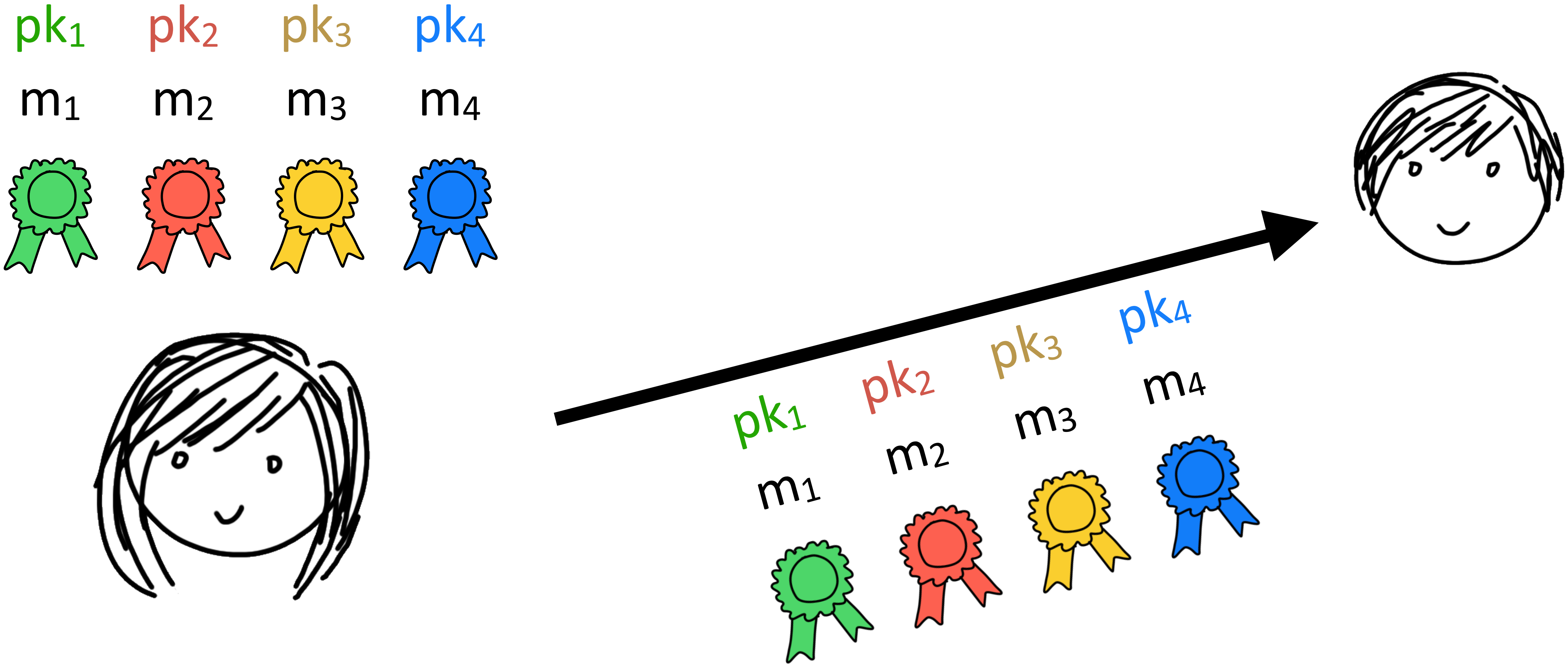
Signature Aggregation

pk_1 pk_2 pk_3 pk_4

m_1 m_2 m_3 m_4



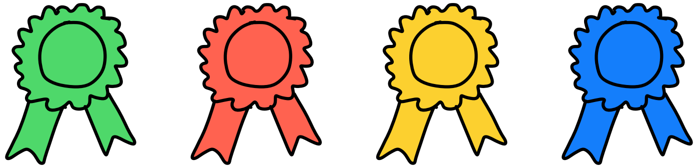
Signature Aggregation



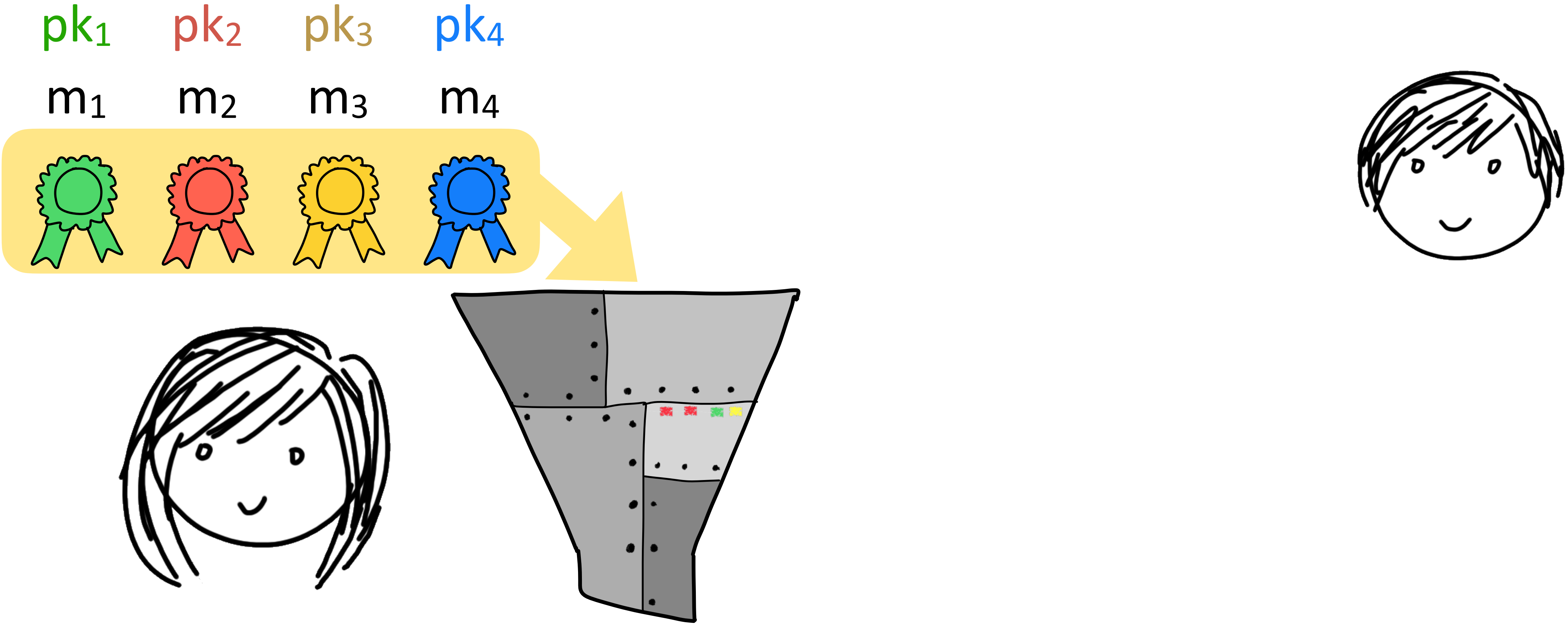
Signature Aggregation

pk_1 pk_2 pk_3 pk_4

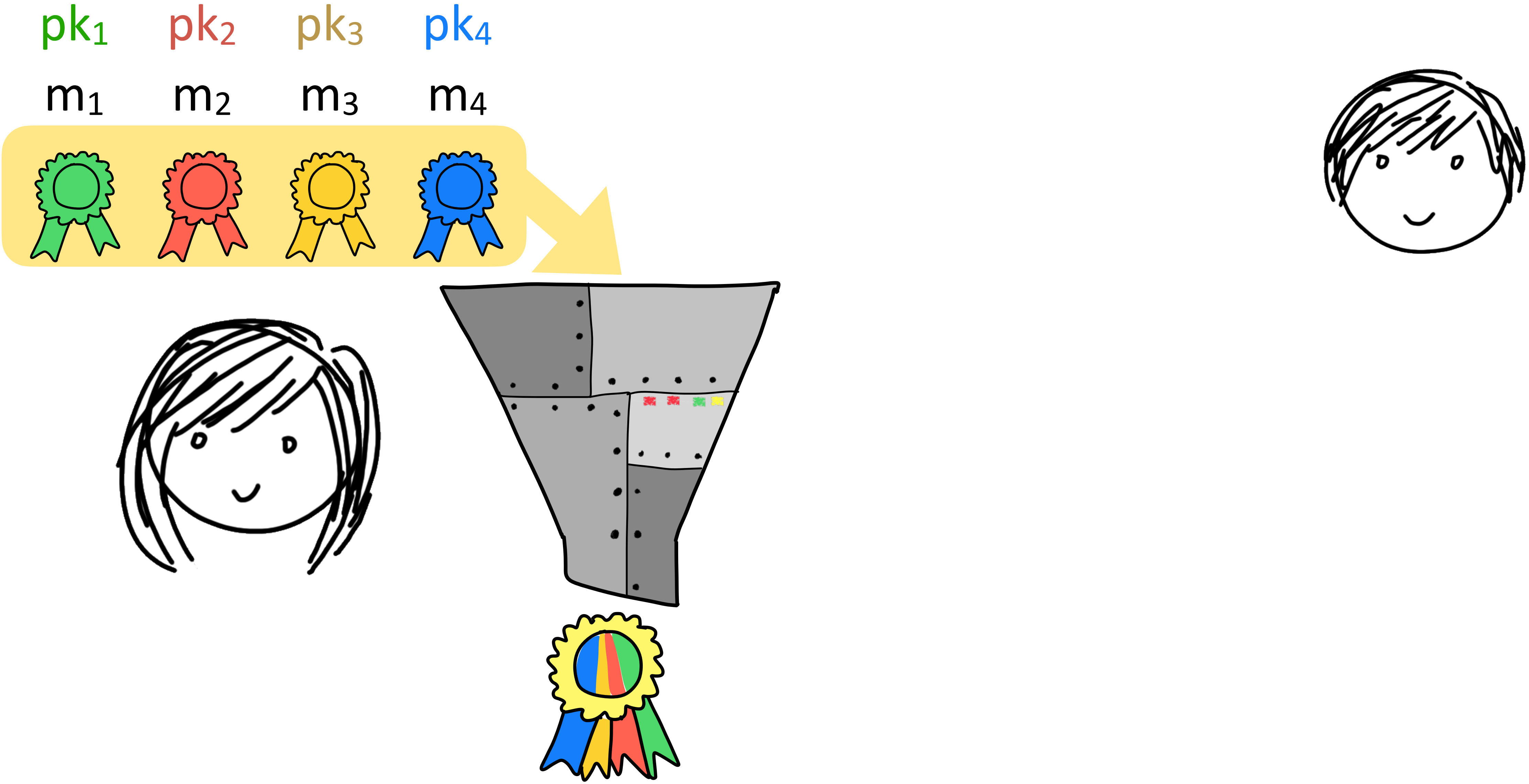
m_1 m_2 m_3 m_4



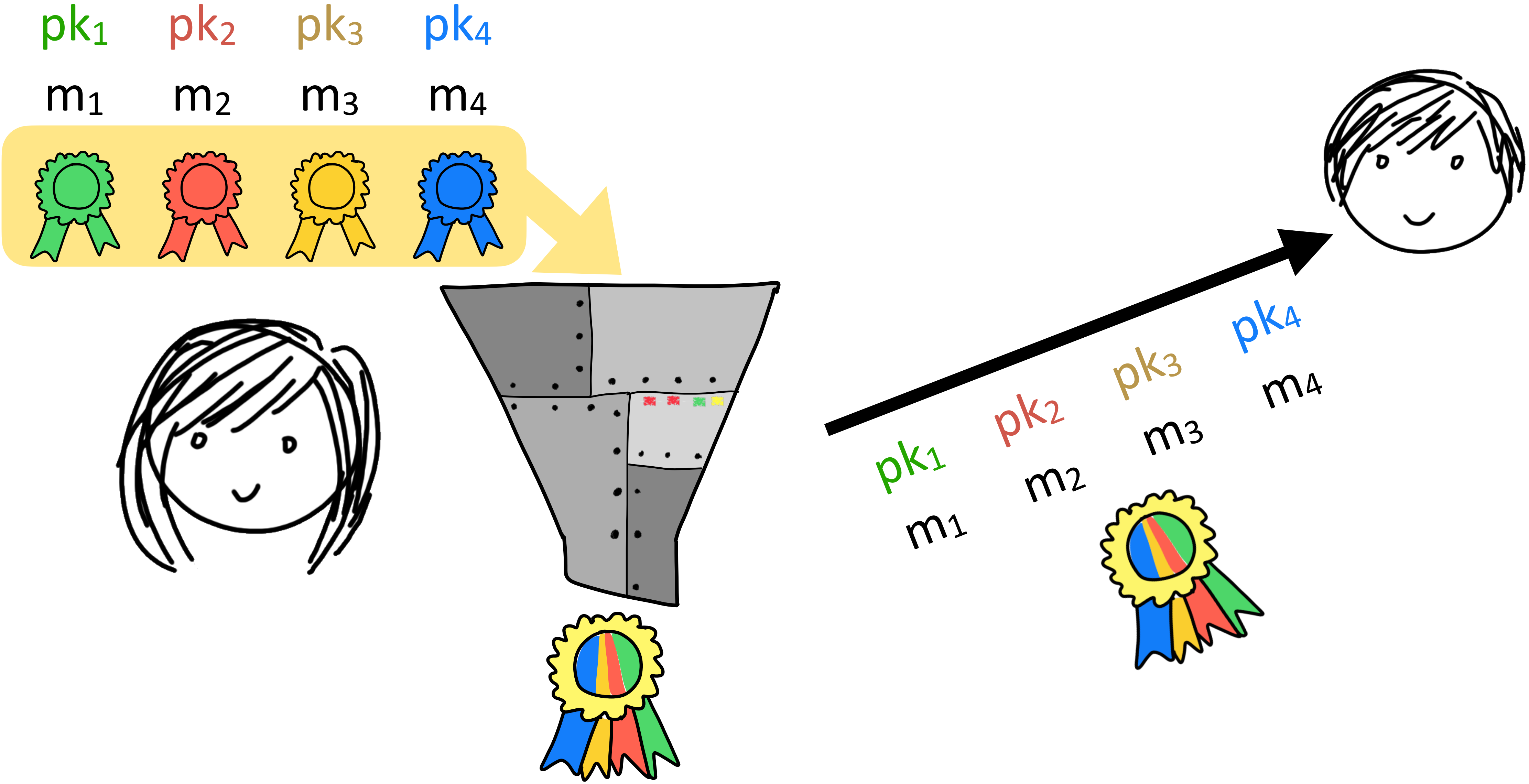
Signature Aggregation



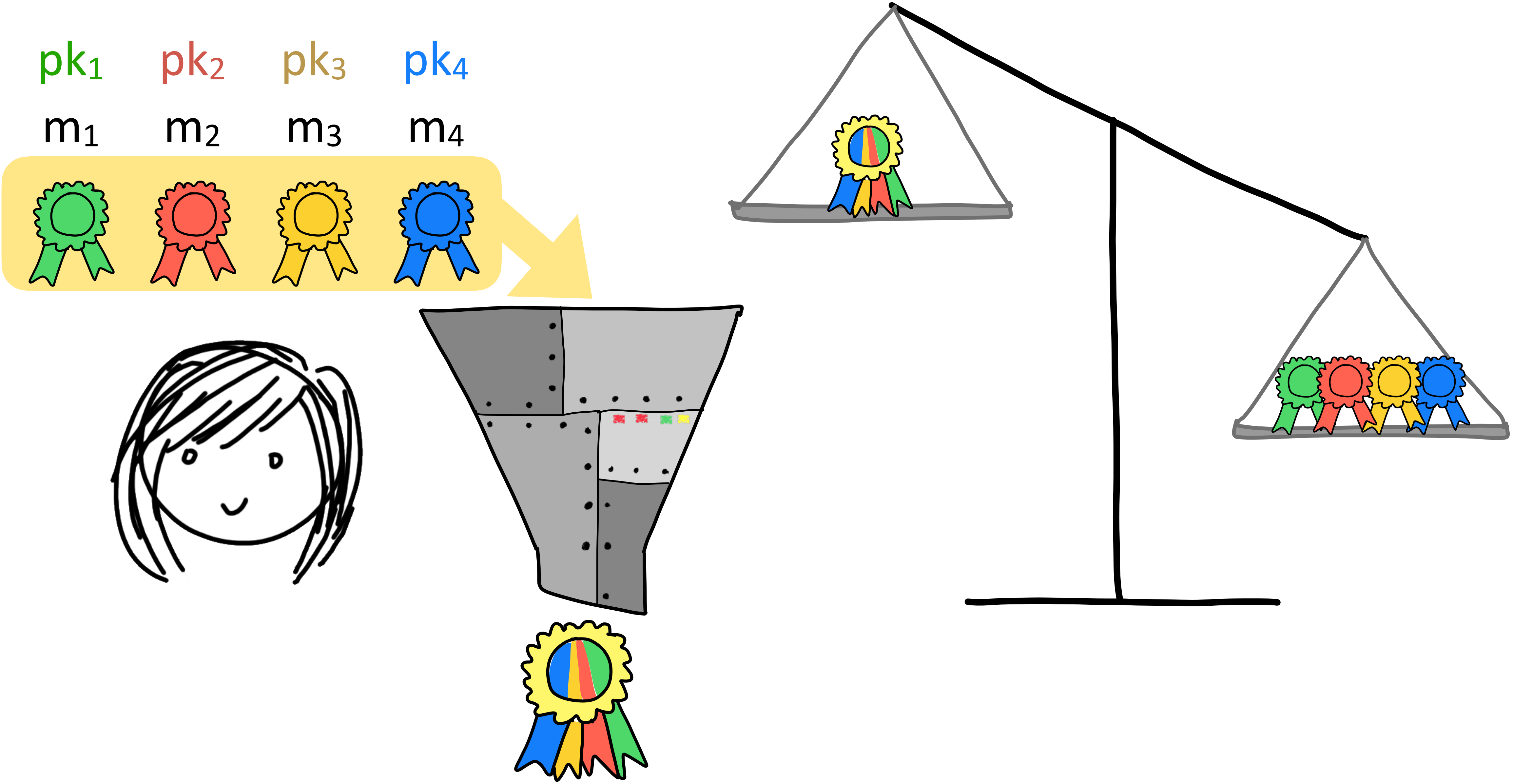
Signature Aggregation



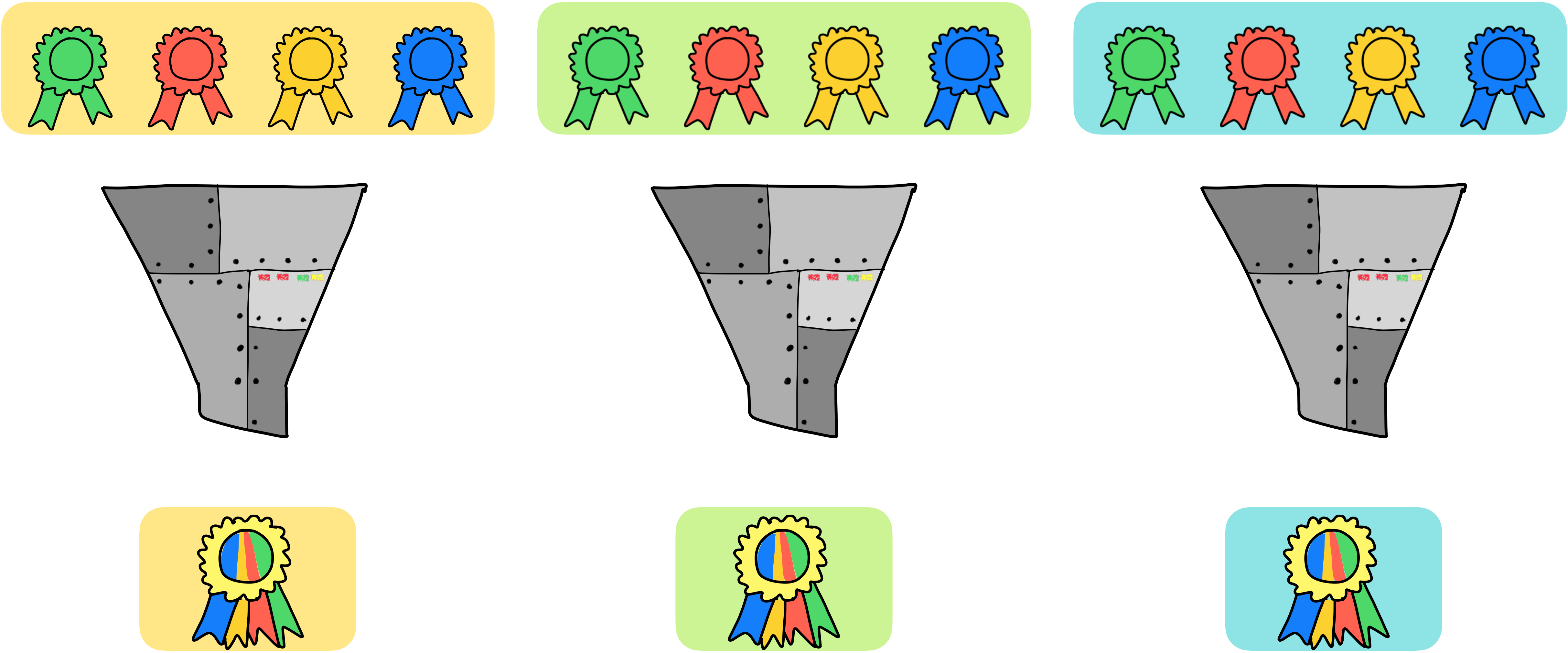
Signature Aggregation



Signature Aggregation



Application: Compressing Blockchains



Problem Scope

Problem Scope

- We formulate the problem as constructing a “proof of knowledge” (PoK) for the language of Schnorr signatures

Problem Scope

- We formulate the problem as constructing a "proof of knowledge" (PoK) for the language of Schnorr signatures
- i.e. Aggregate signature is a (non-interactive) proof that the aggregator has seen corresponding Schnorr signatures

Problem Scope

- We formulate the problem as constructing a "proof of knowledge" (PoK) for the language of Schnorr signatures
- i.e. Aggregate signature is a (non-interactive) proof that the aggregator has seen corresponding Schnorr signatures
- Drop-in replacement in any larger protocol

Problem Scope

- We formulate the problem as constructing a “proof of knowledge” (PoK) for the language of Schnorr signatures
- i.e. Aggregate signature is a (non-interactive) proof that the aggregator has seen corresponding Schnorr signatures
- Drop-in replacement in any larger protocol
- Nice composition guarantees: don’t have to re-prove security of larger protocol upon replacement by PoK

Problem Scope

Problem Scope

- We already know how to build compressing Proofs of Knowledge: eg. IOPs, Bulletproofs, etc.

Problem Scope

- We already know how to build compressing Proofs of Knowledge: eg. IOPs, Bulletproofs, etc.
- Establishes feasibility, but too slow for most applications

Problem Scope

- We already know how to build compressing Proofs of Knowledge: eg. IOPs, Bulletproofs, etc.
- Establishes feasibility, but too slow for most applications
- Bottleneck for such techniques: standard hash functions (eg. SHA2 for EdDSA) and elliptic curve group operations have huge circuit representations

Problem Scope

- We already know how to build compressing Proofs of Knowledge: eg. IOPs, Bulletproofs, etc.
- Establishes feasibility, but too slow for most applications
- Bottleneck for such techniques: standard hash functions (eg. SHA2 for EdDSA) and elliptic curve group operations have huge circuit representations
- Constraint: must be **blackbox** in hash function and curve group (i.e. use them like oracles)

RSAConference2021

Our Techniques

Sigma protocols and non-interactive proofs

Quick recap: Sigma Protocol for relation *R*

$P(X, w)$

X

$V(X)$

Quick recap: Sigma Protocol for relation *R*

$P(X, w)$

X

$\xrightarrow{\quad a \quad}$

$V(X)$

Quick recap: Sigma Protocol for relation *R*

$P(X, w)$

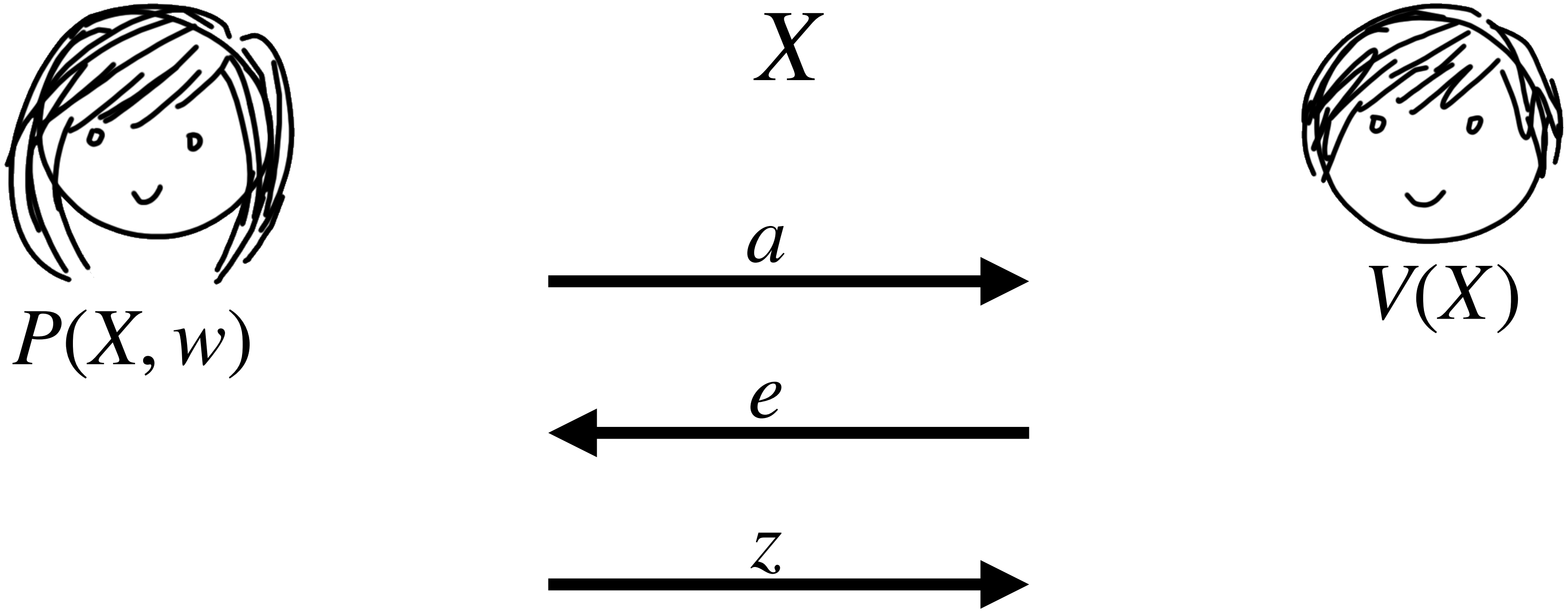
X

a →

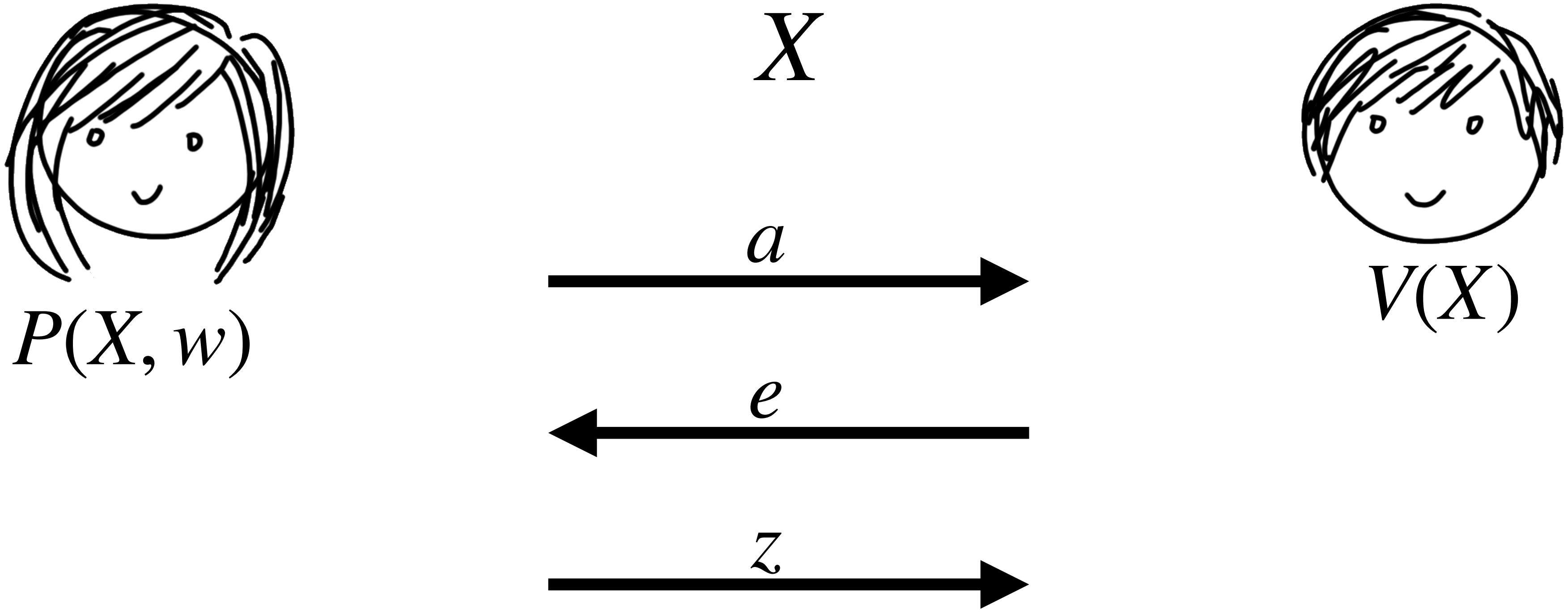
← e

$V(X)$

Quick recap: Sigma Protocol for relation *R*



Quick recap: Sigma Protocol for relation *R*



n-special soundness:

$$\text{Ext}(X, a, (e_1, z_1), \cdots, (e_n, z_n)) \text{ outputs } w \text{ s.t. } R(X, w) = 1$$

Sigma Protocol for PoK of Schnorr Signature

$$\mathsf{pk} = x \cdot G \qquad R = r \cdot G$$

$$e = H(\mathsf{pk}, R, m) \qquad s = xe + r$$

Sigma Protocol for PoK of Schnorr Signature

$\mathsf{pk} = x \cdot G$ $\qquad$ $R = r \cdot G$

$e = H(\mathsf{pk}, R, m)$ $\qquad$ $s = xe + r$

$\mathsf{Verify}(\mathsf{pk}, R, s):$

Compute $S = e \cdot \mathsf{pk} + R$

Output $S \stackrel{?}{=} s \cdot G$

Sigma Protocol for PoK of Schnorr Signature

$\mathsf{pk} = x \cdot G$ $\qquad$ $R = r \cdot G$

$e = H(\mathsf{pk}, R, m)$ $\qquad$ $s = xe + r$

$\mathsf{Verify}(\mathsf{pk}, R, s)$:

Compute $S = e \cdot \mathsf{pk} + R$

Output $S \stackrel{?}{=} s \cdot G$

Sigma Protocol for PoK of Schnorr Signature

$$\mathsf{pk} = x \cdot G \qquad R = r \cdot G$$

$$e = H(\mathsf{pk}, R, m) \qquad s = xe + r$$

$\mathsf{Verify}(\mathsf{pk}, R, s)$:

Compute $S = e \cdot \mathsf{pk} + R$

Output $S \stackrel{?}{=} s \cdot G$

PoK of Schnorr signature of *m* under pk, *R* is equivalent to PoK of discrete logarithm of *S*

Compressing PoKs for *n* discrete logarithm instances

[Gennaro, Leigh, Sundaram, Yerazunis ’04]

$x_1, x_2, \cdots, x_n$

$X_1, X_2, \cdots X_n$

Compressing PoKs for *n* discrete logarithm instances

[Gennaro, Leigh, Sundaram, Yerazunis ’04]

$x_1, x_2, \cdots, x_n$

$X_1, X_2, \cdots X_n$ →

← $e \in \mathbb{Z}_q$

Compressing PoKs for *n* discrete logarithm instances

[Gennaro, Leigh, Sundaram, Yerazunis ’04]

$x_1, x_2, \cdots, x_n$

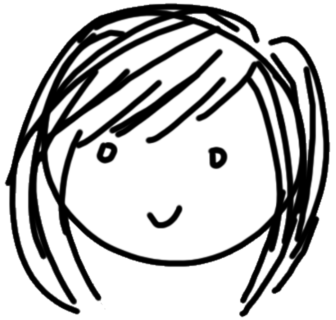
$X_1, X_2, \cdots X_n$

$e \in \mathbb{Z}_q$

$$z = \sum_{i \in [n]} x_i \cdot e^{i-1}$$

Compressing PoKs for *n* discrete logarithm instances

[Gennaro, Leigh, Sundaram, Yerazunis '04]



$x_1, x_2, \cdots, x_n$

$X_1, X_2, \cdots X_n$ →

← $e \in \mathbb{Z}_q$

z →

$$z = \sum_{i \in [n]} x_i \cdot e^{i-1}$$

Compressing PoKs for *n* discrete logarithm instances

[Gennaro, Leigh, Sundaram, Yerazunis ’04]

$x_1, x_2, \cdots, x_n$

$X_1, X_2, \cdots X_n$ →

← $e \in \mathbb{Z}_q$

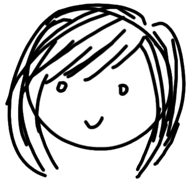
$z = \sum_{i \in [n]} x_i \cdot e^{i-1}$

z →

$z \cdot G \stackrel{?}{=} \sum_{i \in [n]} e^{i-1} \cdot X_i$

Compressing PoKs for *n* discrete logarithm instances

[Gennaro, Leigh, Sundaram, Yerazunis '04]



$x_1, x_2, \cdots, x_n$

$z = \sum_{i \in [n]} x_i \cdot e^{i-1}$

$X_1, X_2, \cdots X_n$ →

$e \in \mathbb{Z}_q$ ←

z →

$z \cdot G \stackrel{?}{=} \sum_{i \in [n]} e^{i-1} \cdot X_i$

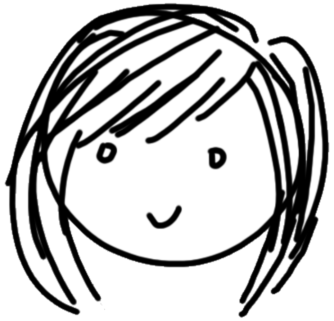
n special soundness:

Values $(e_1, z_1), \cdots, (e_n, z_n)$

Characterise *n* linearly independent combinations of x_is

Solve for each x_i

Compressed Sigma Protocol for PoK of n Schnorr Sigs

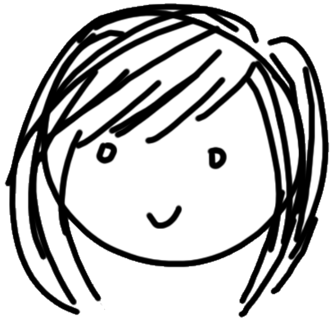


$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$

$(R_1, s_1), \cdots, (R_n, s_n)$

Compressed Sigma Protocol for PoK of n Schnorr Sigs

$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$



$R_1, \cdots, R_n$ →

$(R_1, s_1), \cdots, (R_n, s_n)$

Compressed Sigma Protocol for PoK of n Schnorr Sigs

$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$

$(R_1, s_1), \cdots, (R_n, s_n)$

$\xrightarrow{R_1, \cdots, R_n}$

$\forall i \in [n],$

$S_i = \mathsf{pk}_i^{H(\mathsf{pk}, R_i, m_i)} + R_i$

Compressed Sigma Protocol for PoK of n Schnorr Sigs

$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$

$(R_1, s_1), \cdots, (R_n, s_n)$

$R_1, \cdots, R_n$ →

$\forall i \in [n],$

$S_i = \mathsf{pk}_i^{H(\mathsf{pk}, R_i, m_i)} + R_i$

← $e \in \mathbb{Z}_q$

Compressed Sigma Protocol for PoK of n Schnorr Sigs

$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$

$(R_1, s_1), \cdots, (R_n, s_n)$

$R_1, \cdots, R_n$ →

$\forall i \in [n],$

$S_i = \mathsf{pk}_i^{H(\mathsf{pk}, R_i, m_i)} + R_i$

← $e \in \mathbb{Z}_q$

$$z = \sum_{i \in [n]} s_i \cdot e^{i-1}$$

Compressed Sigma Protocol for PoK of n Schnorr Sigs

$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$

$(R_1, s_1), \cdots, (R_n, s_n)$

$R_1, \cdots, R_n$ →

$\forall i \in [n],$

$S_i = \mathsf{pk}_i^{H(\mathsf{pk}, R_i, m_i)} + R_i$

← $e \in \mathbb{Z}_q$

$$z = \sum_{i \in [n]} s_i \cdot e^{i-1}$$

z →

Compressed Sigma Protocol for PoK of n Schnorr Sigs

$$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$$

$$(R_1, s_1), \cdots, (R_n, s_n)$$

$R_1, \cdots, R_n$ →

$\forall i \in [n],$

$$S_i = \mathsf{pk}_i^{H(\mathsf{pk}, R_i, m_i)} + R_i$$

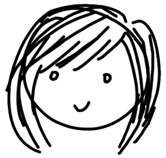
← $e \in \mathbb{Z}_q$

$$z = \sum_{i \in [n]} s_i \cdot e^{i-1}$$

z →

$$z \cdot G \stackrel{?}{=} \sum_{i \in [n]} e^{i-1} \cdot S_i$$

Compressed Sigma Protocol for PoK of n Schnorr Sigs



$(\mathsf{pk}_1, m_1), \cdots, (\mathsf{pk}_n, m_n)$

$(R_1, s_1), \cdots, (R_n, s_n)$

$\xrightarrow{R_1, \cdots, R_n}$

$\forall i \in [n],$

$S_i = \mathsf{pk}_i^{H(\mathsf{pk}, R_i, m_i)} + R_i$

$\xleftarrow{e \in \mathbb{Z}_q}$

$z = \sum_{i \in [n]} s_i \cdot e^{i-1}$ $\xrightarrow{z}$ $z \cdot G \stackrel{?}{=} \sum_{i \in [n]} e^{i-1} \cdot S_i$

What have we accomplished?

Naive transmission:

$$(R_1, s_1), \cdots, (R_n, s_n)$$

Compressed Sigma protocol:

$$z, (R_1, \cdots, R_n)$$

i.e. ~50% compression!

From Sigma Protocol to Non-interactive PoK

From Sigma Protocol to Non-interactive PoK

- Sigma protocols are interactive, but our target is a non-interactive proof

From Sigma Protocol to Non-interactive PoK

- Sigma protocols are interactive, but our target is a non-interactive proof
- Standard compilers,

From Sigma Protocol to Non-interactive PoK

- Sigma protocols are interactive, but our target is a non-interactive proof
- Standard compilers,
  - Fiat-Shamir: optimal efficiency, loose security proof

From Sigma Protocol to Non-interactive PoK

- Sigma protocols are interactive, but our target is a non-interactive proof
- Standard compilers,
  - Fiat-Shamir: optimal efficiency, loose security proof
  - Fischlin: reduced efficiency (compression approaches 50%), tight security proof

Benchmarks

Benchmarks

- We measured the performance of both constructions using the Ed25519-dalek library

Benchmarks

- We measured the performance of both constructions using the Ed25519-dalek library
- Takeaway:

Benchmarks

- We measured the performance of both constructions using the Ed25519-dalek library
- Takeaway:
  - Fiat-Shamir: aggregates 1024 sigs in <1ms, and verifying the aggregate signature costs the same as batch verifying the same number of signatures

Benchmarks

- We measured the performance of both constructions using the Ed25519-dalek library
- Takeaway:
  - Fiat-Shamir: aggregates 1024 sigs in <1ms, and verifying the aggregate signature costs the same as batch verifying the same number of signatures
  - Fischlin: 10s of seconds to aggregate 100s of sigs with >40% compression, order of magnitude slower verification

Can we do better?

Can we do better?

- We show that 50% compression is optimal for any aggregation scheme that makes oracle use of the hash function in Schnorr

Can we do better?

- We show that 50% compression is optimal for any aggregation scheme that makes oracle use of the hash function in Schnorr
- **Implication**: compressing Schnorr sigs beyond 50% must depend on the code of the hash function. All known techniques are expensive, eg. Ed25519 will need SNARKs for *n* SHA2 pre-images

See the paper for…

- Discussions on how to use these constructions
- Optimisations for concrete efficiency
- Detailed proofs
- Detailed benchmarks
- Discussion on related work

Apply these constructions

- Identify protocols that involve transmitting or storing multiple Schnorr (eg. Ed25519) signatures in a batch
- Question if the exact bit representation is important for some reason (eg. having to un-batch the signatures later, or compare with a digest for an integrity check). Can the physical signatures be replaced by a proof-of-knowledge oracle?
- Consider cutting bandwidth/storage cost in half by aggregating the signatures

Thanks!

ia.cr/2021/350

github.com/novifinancial/ed25519-dalek-fiat/tree/half-aggregation

Thanks to Eysa Lee for