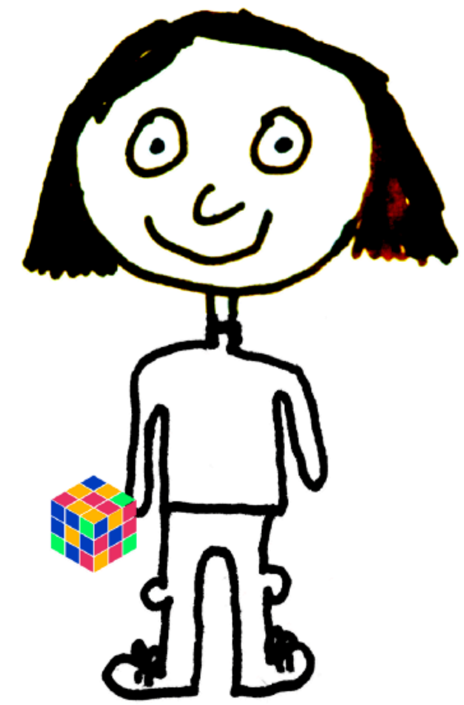
Multi-Theorem Preprocessing NIZKs from Lattices

Sam Kim and David J. Wu
Stanford University

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$

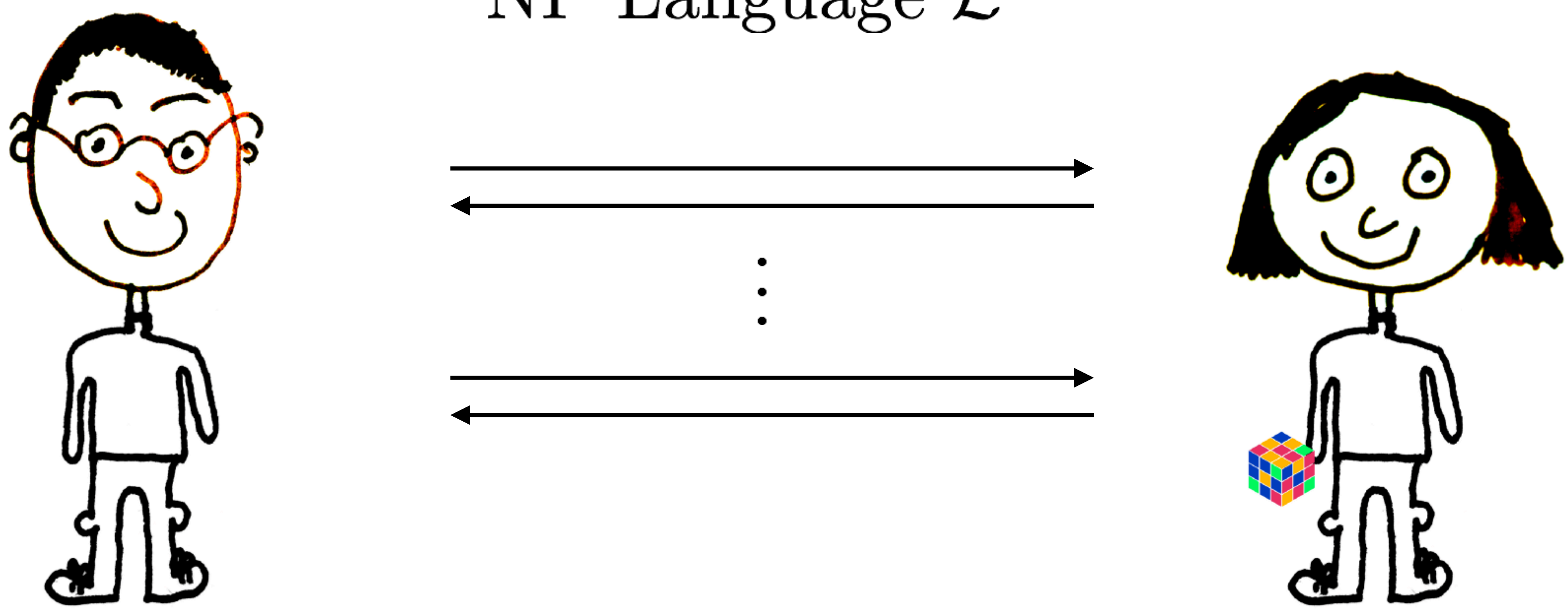
Prover(x, w)



Verifier(x)

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$

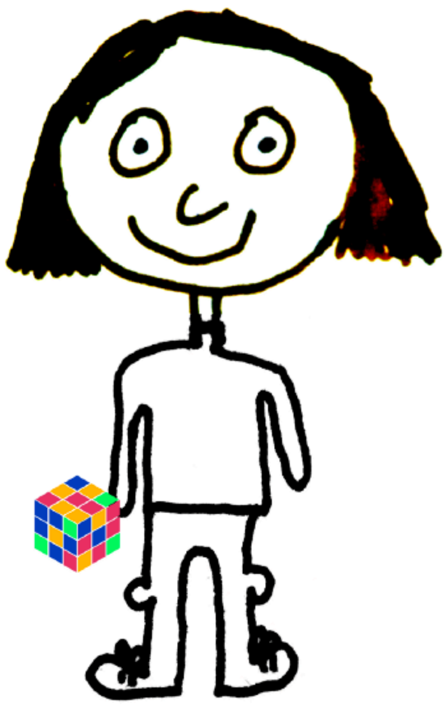


Prover(x, w) Verifier(x)

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$

Prover(x, w)

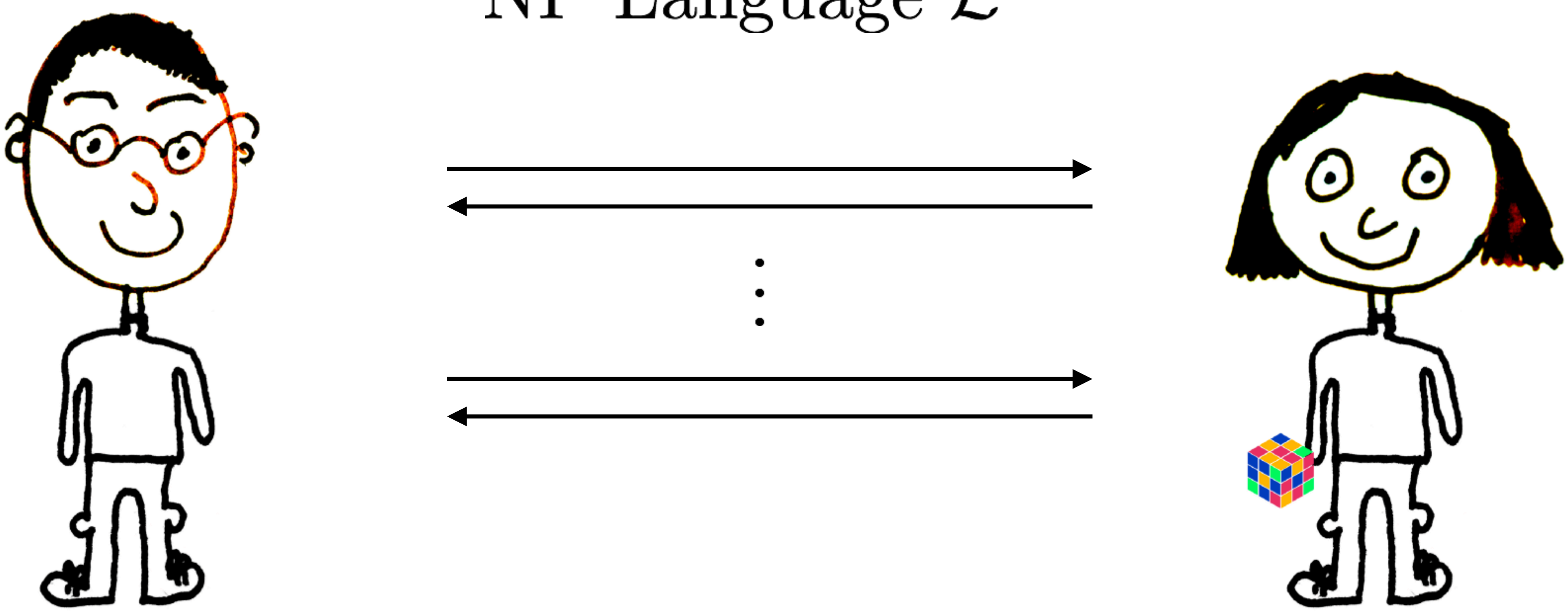


Verifier(x)

Accept / Reject

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$



Prover(x, w)

Verifier(x)

$\downarrow$

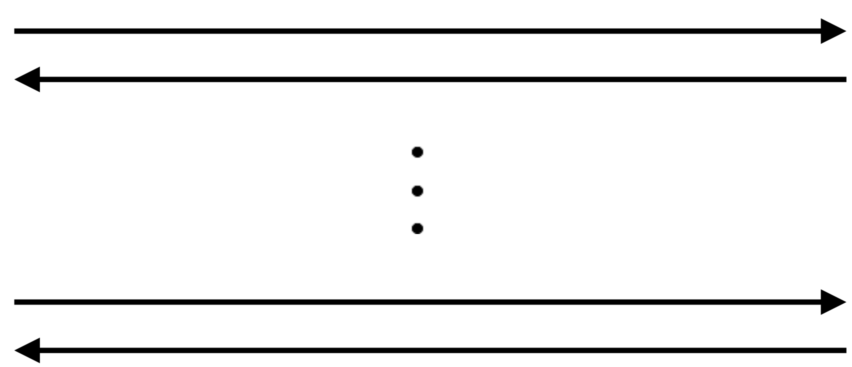
Accept

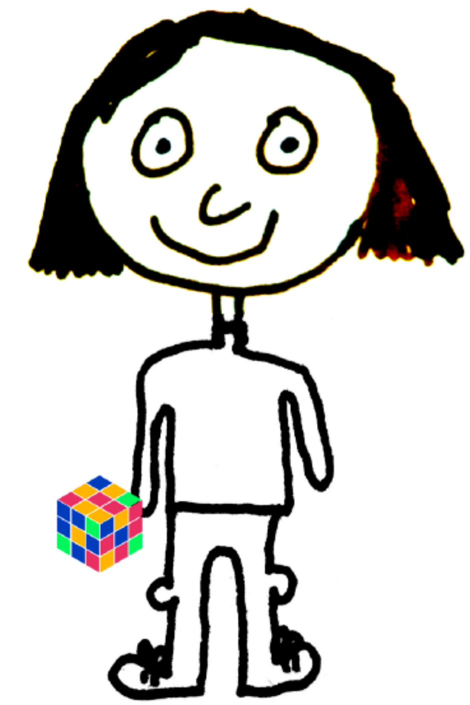
Requirements:

1. Completeness

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$





Prover$(x \notin \mathcal{L})$

Verifier(x)

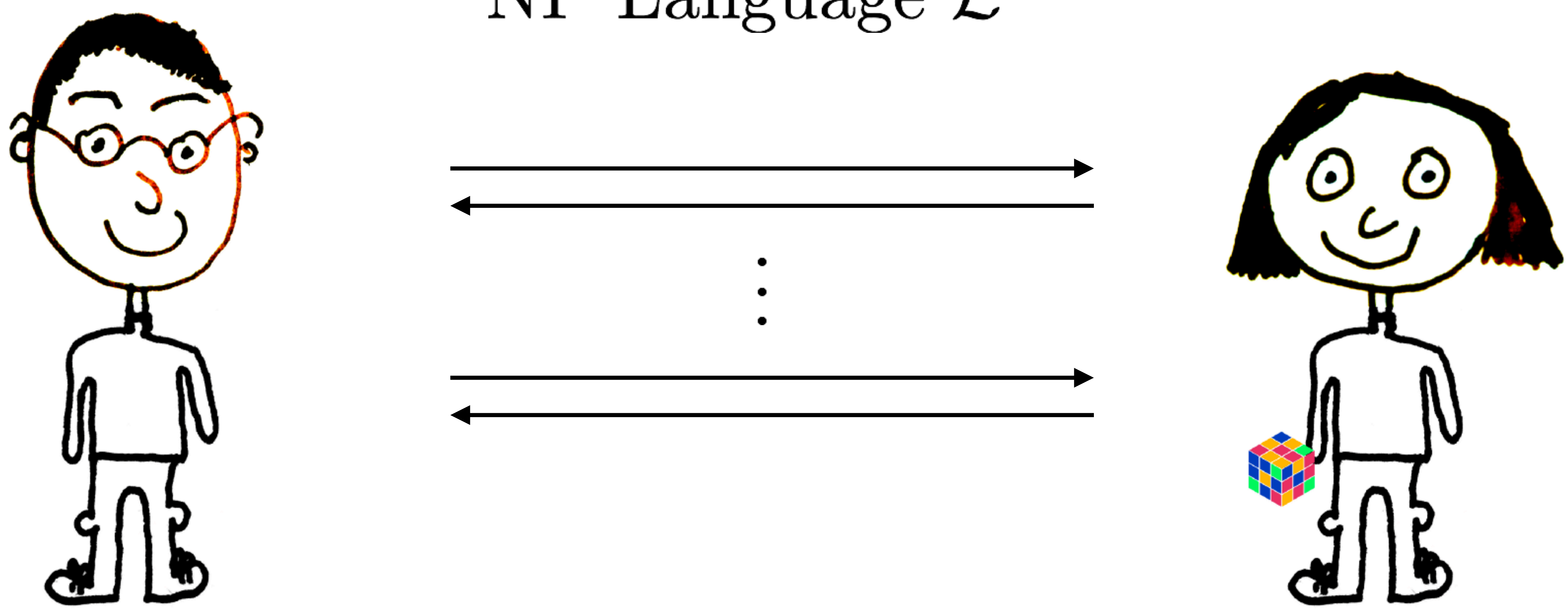
$\downarrow$

Reject

Requirements:
1. Completeness
2. Soundness

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$



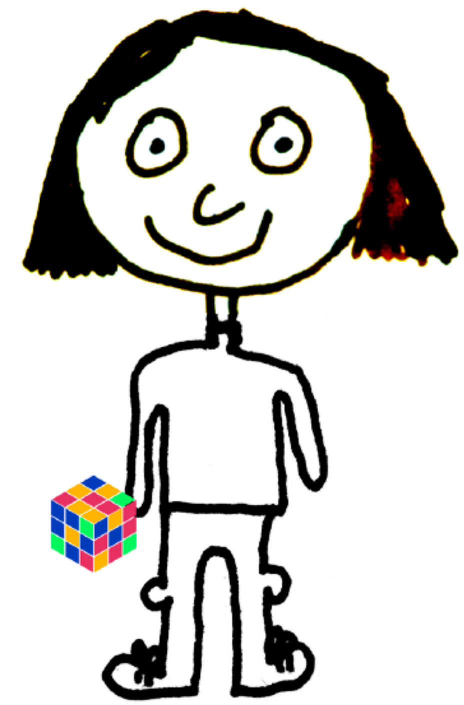
Prover(x, w) Verifier(x)

Requirements:

1. Completeness
2. Soundness
3. Zero-Knowledge

Zero-Knowledge Proofs [GMR85]

NP Language $\mathcal{L}$



$\mathsf{Sim}(x \in \mathcal{L})$

$\mathsf{Verifier}(x)$

$\downarrow$

???

Requirements:

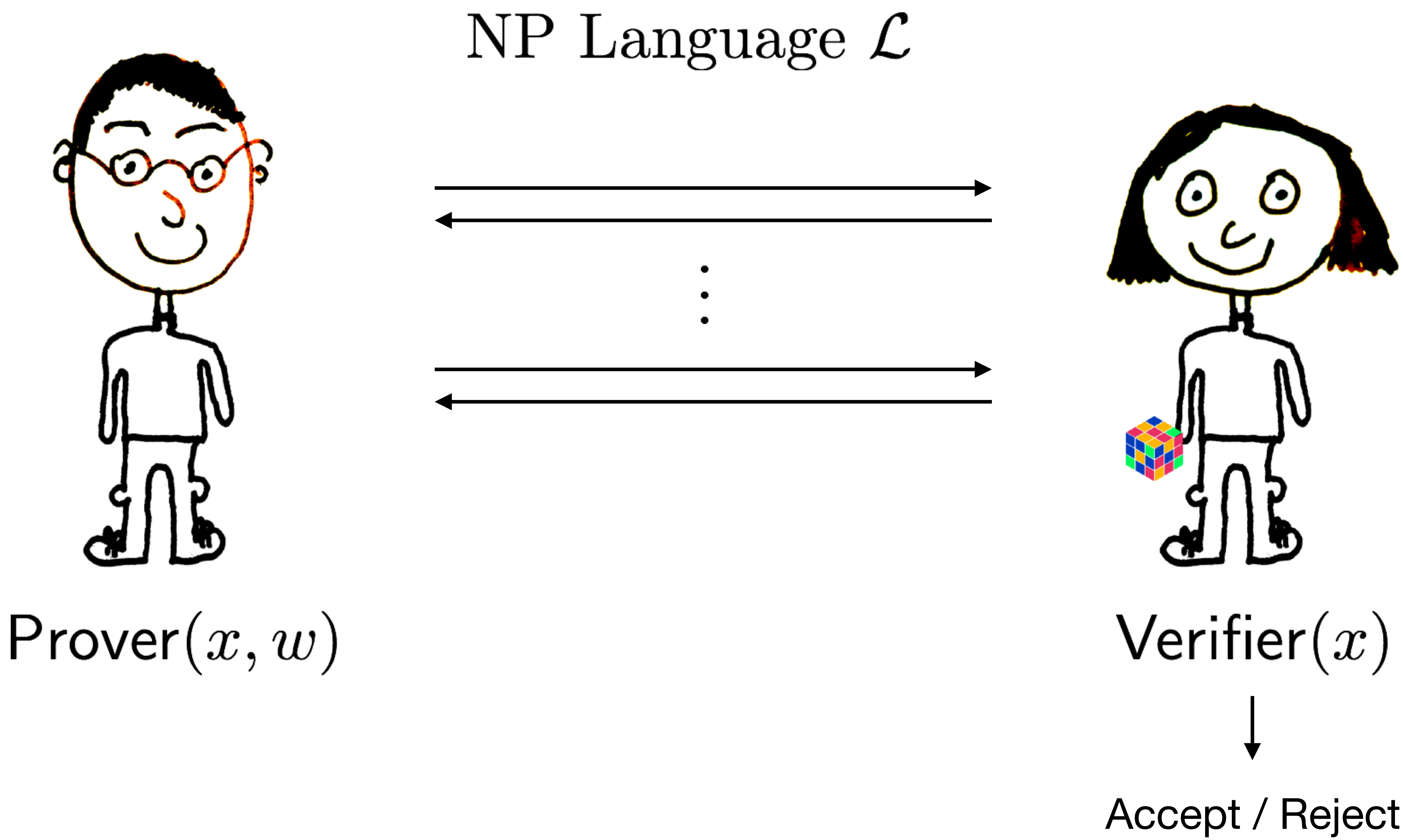
1. Completeness
2. Soundness
3. Zero-Knowledge

Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have “one-shot” ZK proofs?

Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have “one-shot” ZK proofs?

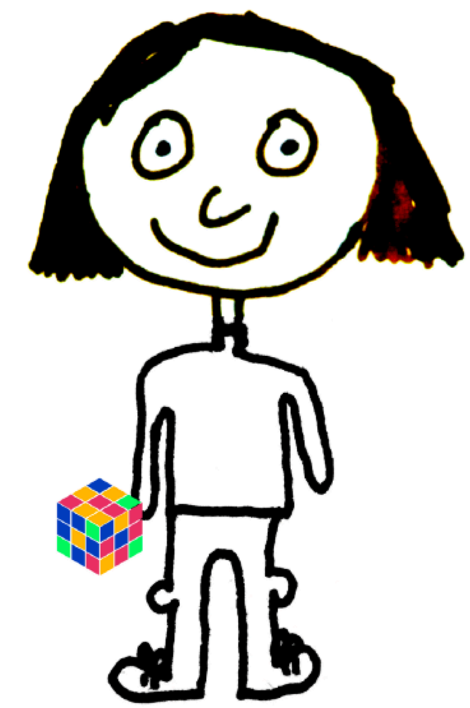


Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have "one-shot" ZK proofs?

$\text{NP Language } \mathcal{L}$

$$\xrightarrow{\quad \pi \quad}$$



Prover(x, w)

Verifier(x)

$\downarrow$

Accept / Reject

Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have “one-shot” ZK proofs?

Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have “one-shot” ZK proofs?

Can only achieve for “easy” languages in standard model [GO94]

Soundness + **Zero-Knowldge** implies Efficient Decision Algorithm

Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have "one-shot" ZK proofs?

Can only achieve for "easy" languages in standard model [GO94]

Soundness + **Zero-Knowldge** implies Efficient Decision Algorithm

Work with **weaker models**:

- Random Oracle Model
- CRS Model

Non-Interactive Zero-Knowledge (NIZK)

Natural to ask: Can we have “one-shot” ZK proofs?

Can only achieve for “easy” languages in <u>standard model</u> [GO94]

Soundness + **Zero-Knowldge** implies Efficient Decision Algorithm

Work with **weaker models**:

- Random Oracle Model
- CRS Model

This Work: We focus on the **CRS Model** (or **preprocessing model**)

Non-Interactive Zero-Knowledge (NIZK)

Constructions for all of NP ?

(w/ efficient provers, reusable CRS, publicly verifiable, …)

Non-Interactive Zero-Knowledge (NIZK)

Constructions for all of NP ?
(w/ efficient provers, reusable CRS, publicly verifiable, …)

1. Trapdoor Permutations [FLS90, …]
2. Pairings [GOS06, …]
3. Indistinguishability Obfuscation [SW14, …]

Non-Interactive Zero-Knowledge (NIZK)

Constructions for all of NP ?
(w/ efficient provers, reusable CRS, publicly verifiable, …)

1. Trapdoor Permutations [FLS90, …]
2. Pairings [GOS06, …]
3. Indistinguishability Obfuscation [SW14, …]

Still no construction from LWE.

- NIZK for specific languages [PV08, APS18, RSS18, …]

Our Results

1. Construct NIZK for NP in preprocessing model from LWE

Our Results

1. Construct NIZK for NP in preprocessing model from LWE

2. Show how to do preprocessing
   - Blind Homomorphic Signatures (BHS)

Our Results

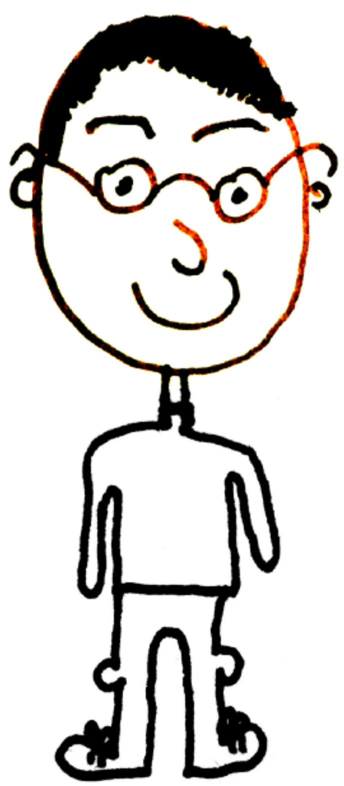
1. Construct NIZK for NP in preprocessing model from LWE

2. Show how to do preprocessing
   - Blind Homomorphic Signatures (BHS)

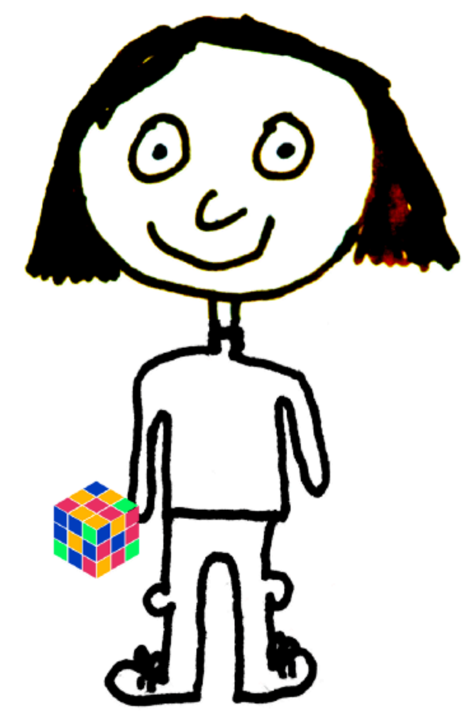
3. Applications to MPC

NIZK with Preprocessing [DMP88]

NP Language $\mathcal{L}$

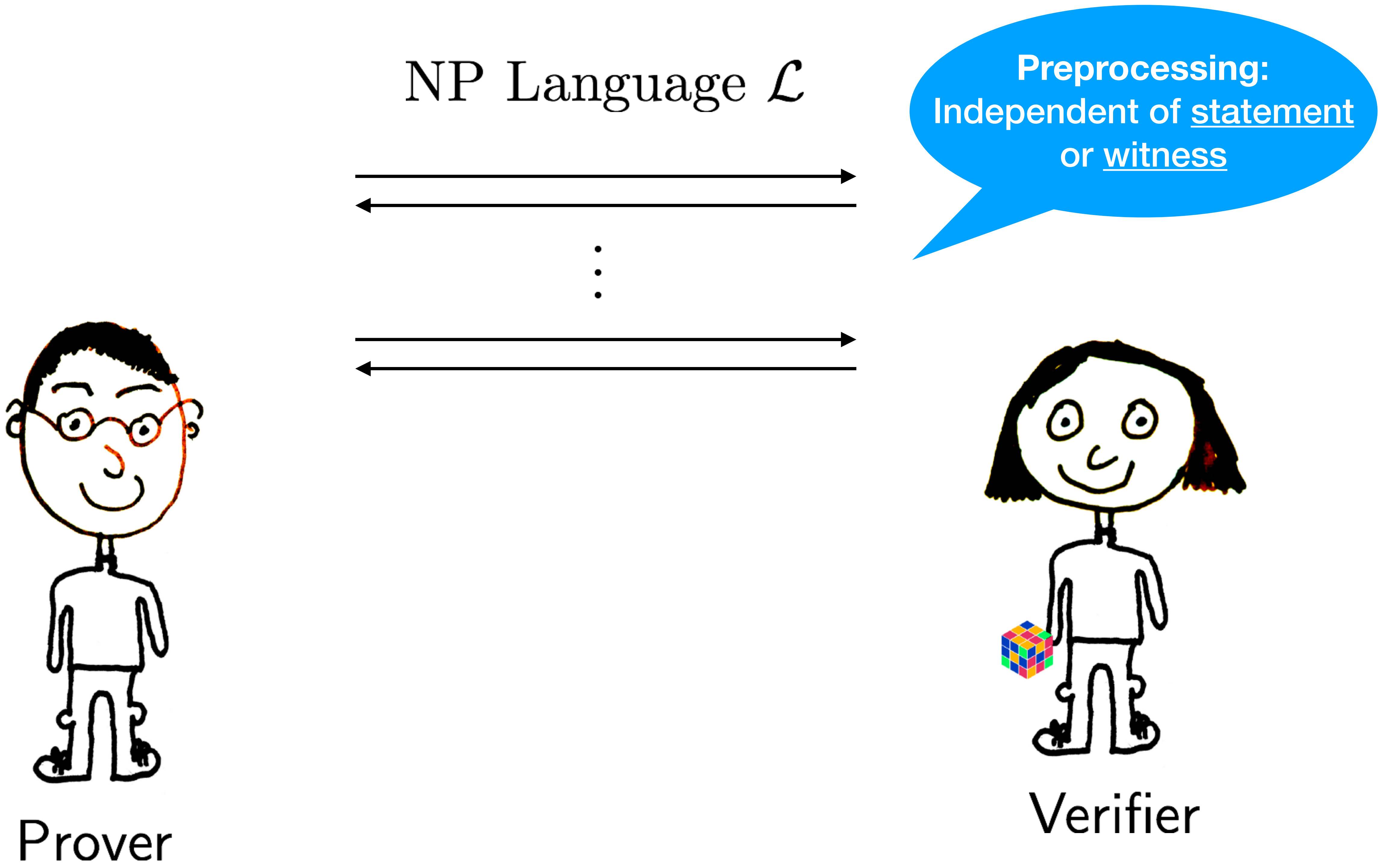


Prover



Verifier

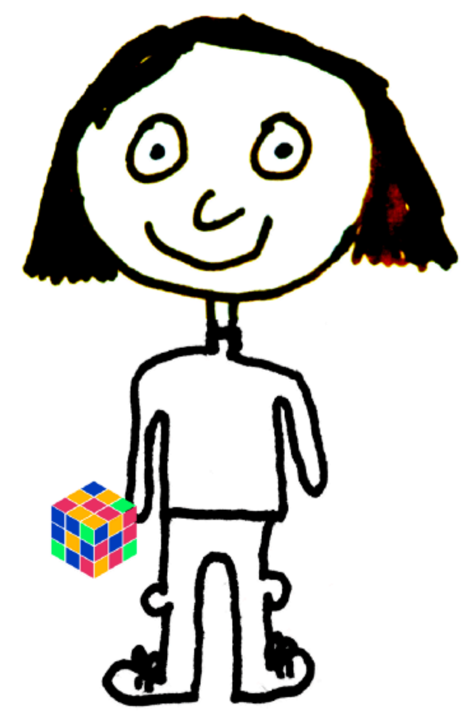
NIZK with Preprocessing [DMP88]



NIZK with Preprocessing [DMP88]

NP Language $\mathcal{L}$

Prover
k_P



Verifier
k_V

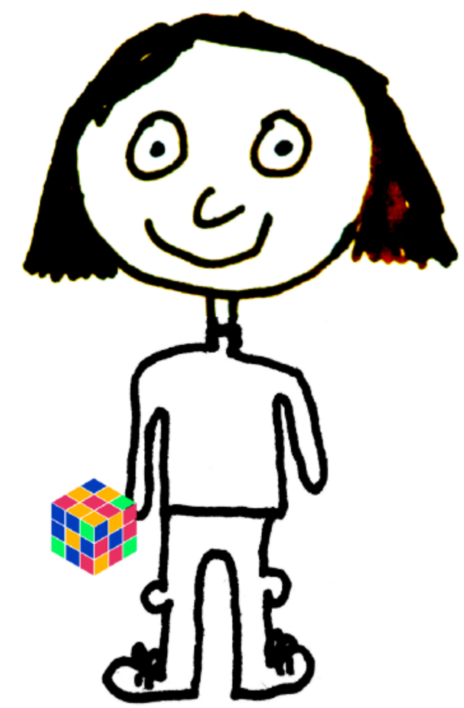
NIZK with Preprocessing [DMP88]

NP Language $\mathcal{L}$

Prover

k_P

$$\xrightarrow{\pi_x}$$

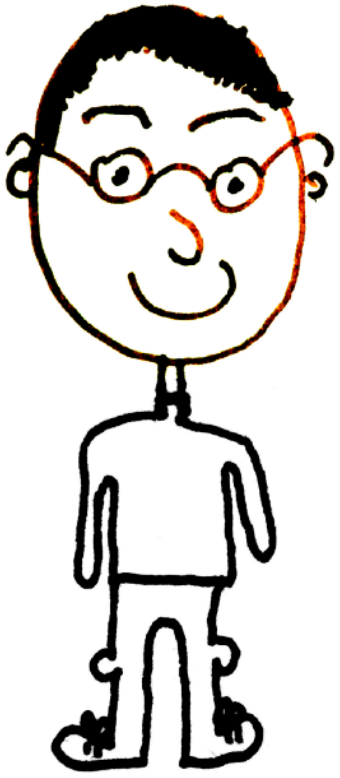


Verifier

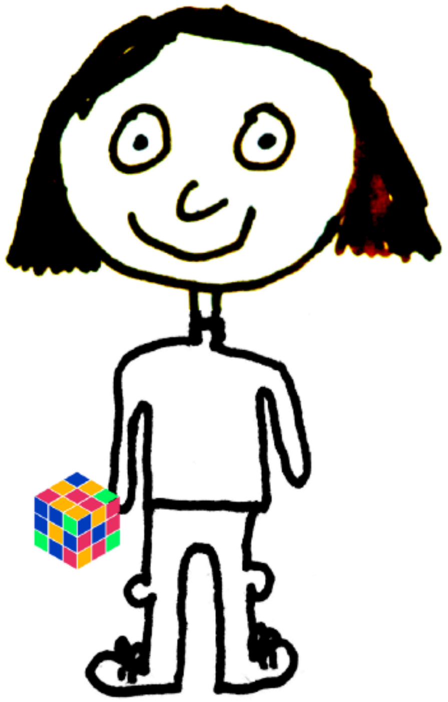
k_V

NIZK with Preprocessing [DMP88]
(Single-Theorem)

NP Language $\mathcal{L}$



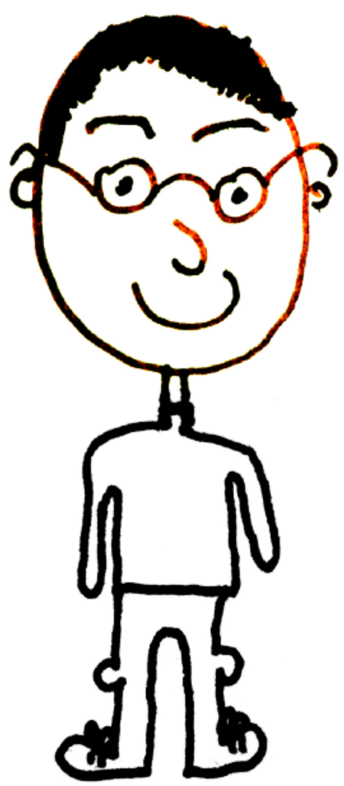
Prover
k_P



Verifier
k_V

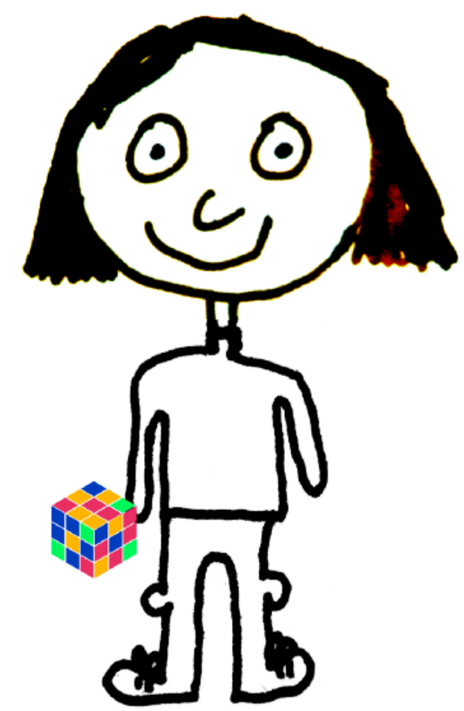
NIZK with Preprocessing [DMP88]
(Single-Theorem)

NP Language $\mathcal{L}$



Prover
k_P

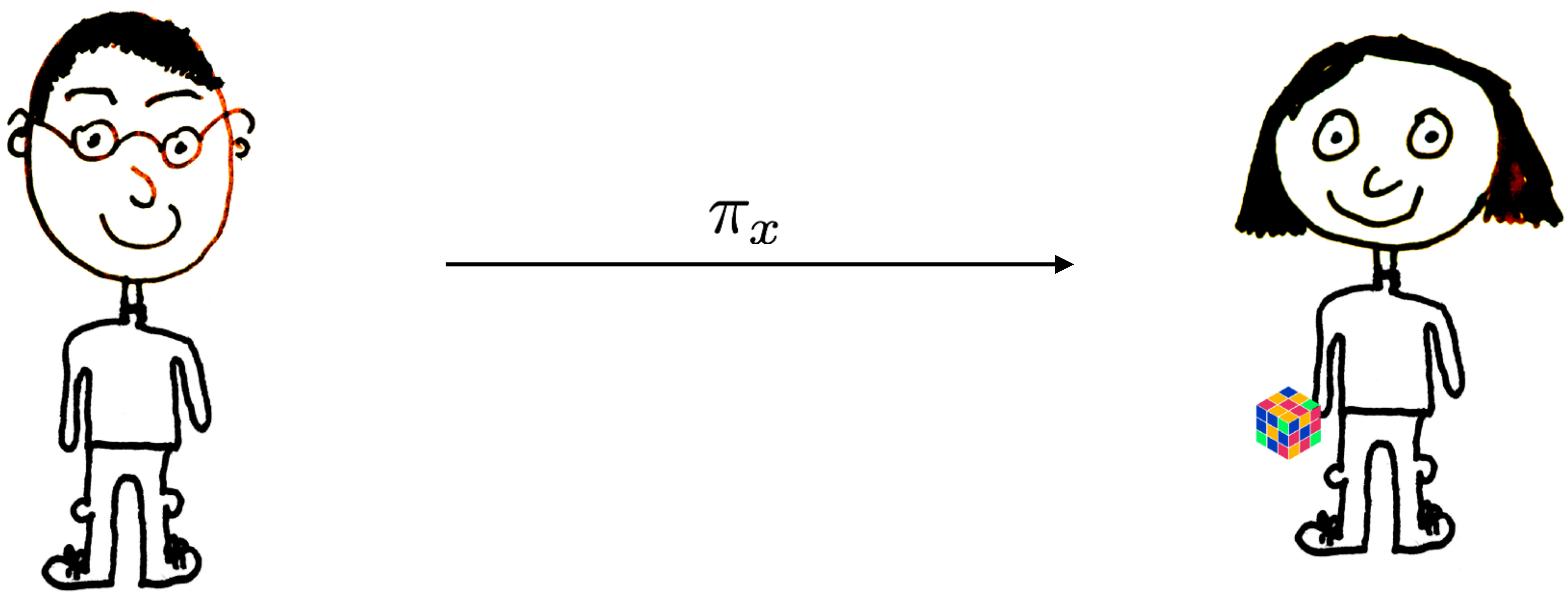
$$\xrightarrow{\quad\pi_x\quad}$$



Verifier
k_V

NIZK with Preprocessing [DMP88]
(Single-Theorem)

NP Language $\mathcal{L}$



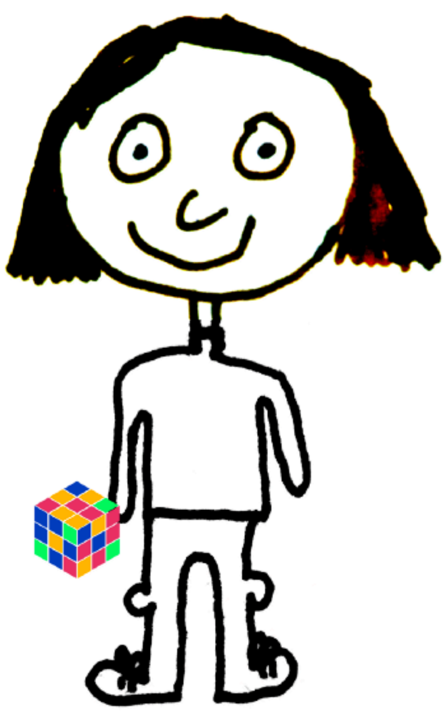
Prover $\cancel{k_P}$

Verifier $\cancel{k_V}$

NIZK with Preprocessing [DMP88]
(Single-Theorem)

NP Language $\mathcal{L}$

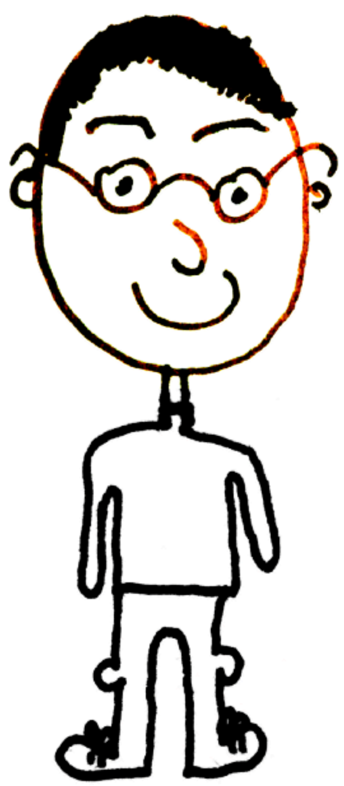
Prover



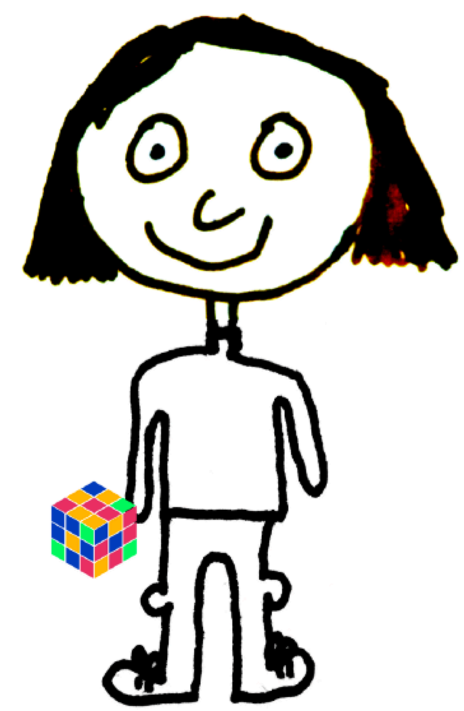
Verifier

NIZK with Preprocessing [DMP88]
(Single-Theorem)

NP Language $\mathcal{L}$



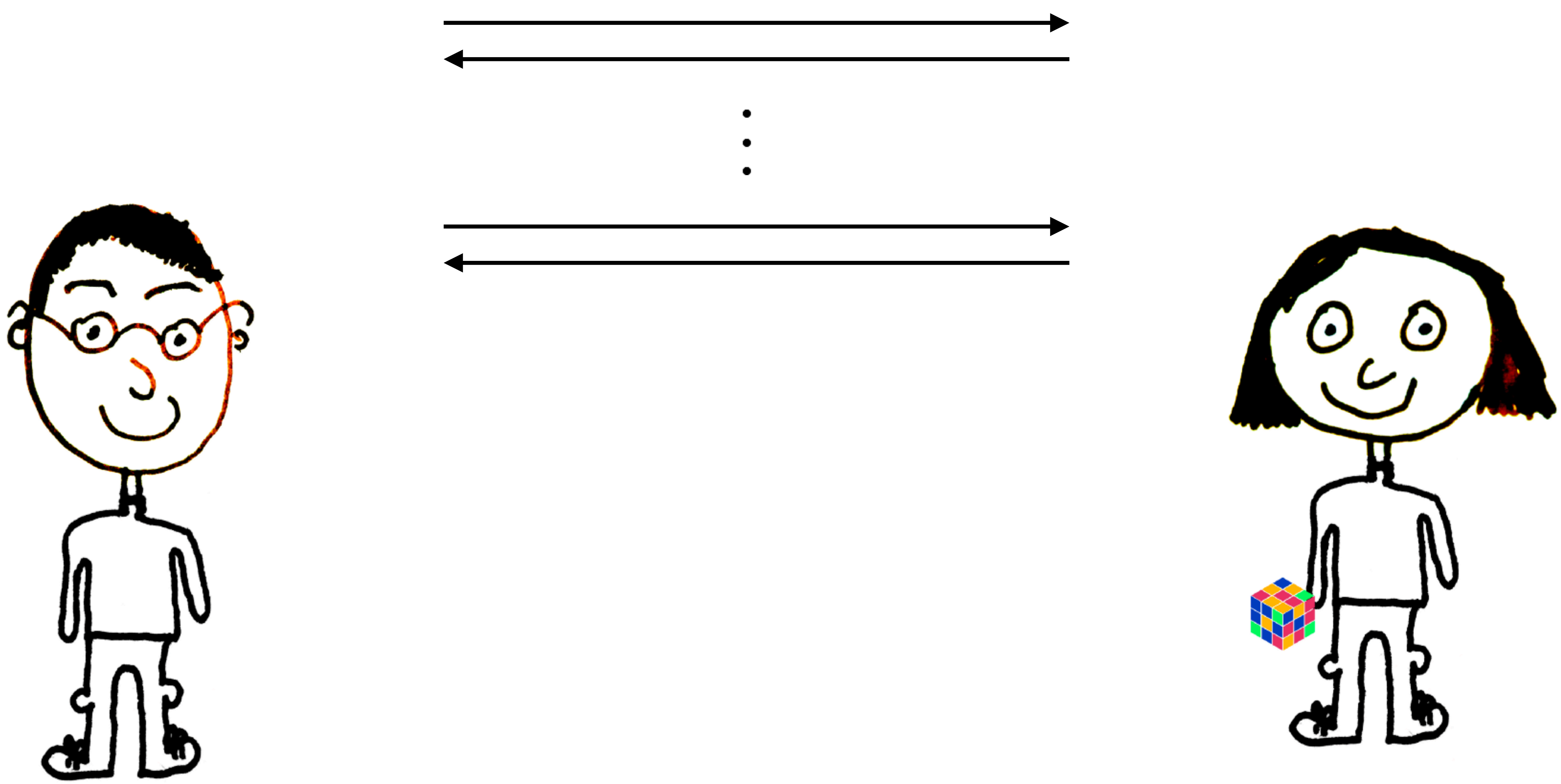
Prover



Verifier

NIZK with Preprocessing [DMP88]
(Single-Theorem)

NP Language $\mathcal{L}$



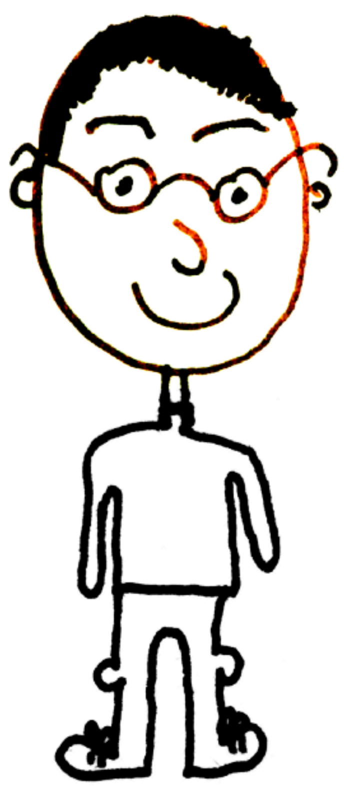
Prover

Verifier

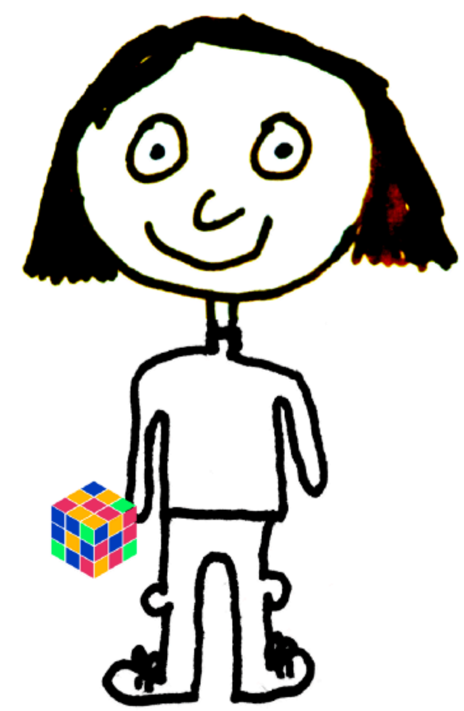
Easier to construct: follows from OWF [DMP88, …]

NIZK with Preprocessing [DMP88]
(Multi-Theorem)

NP Language $\mathcal{L}$



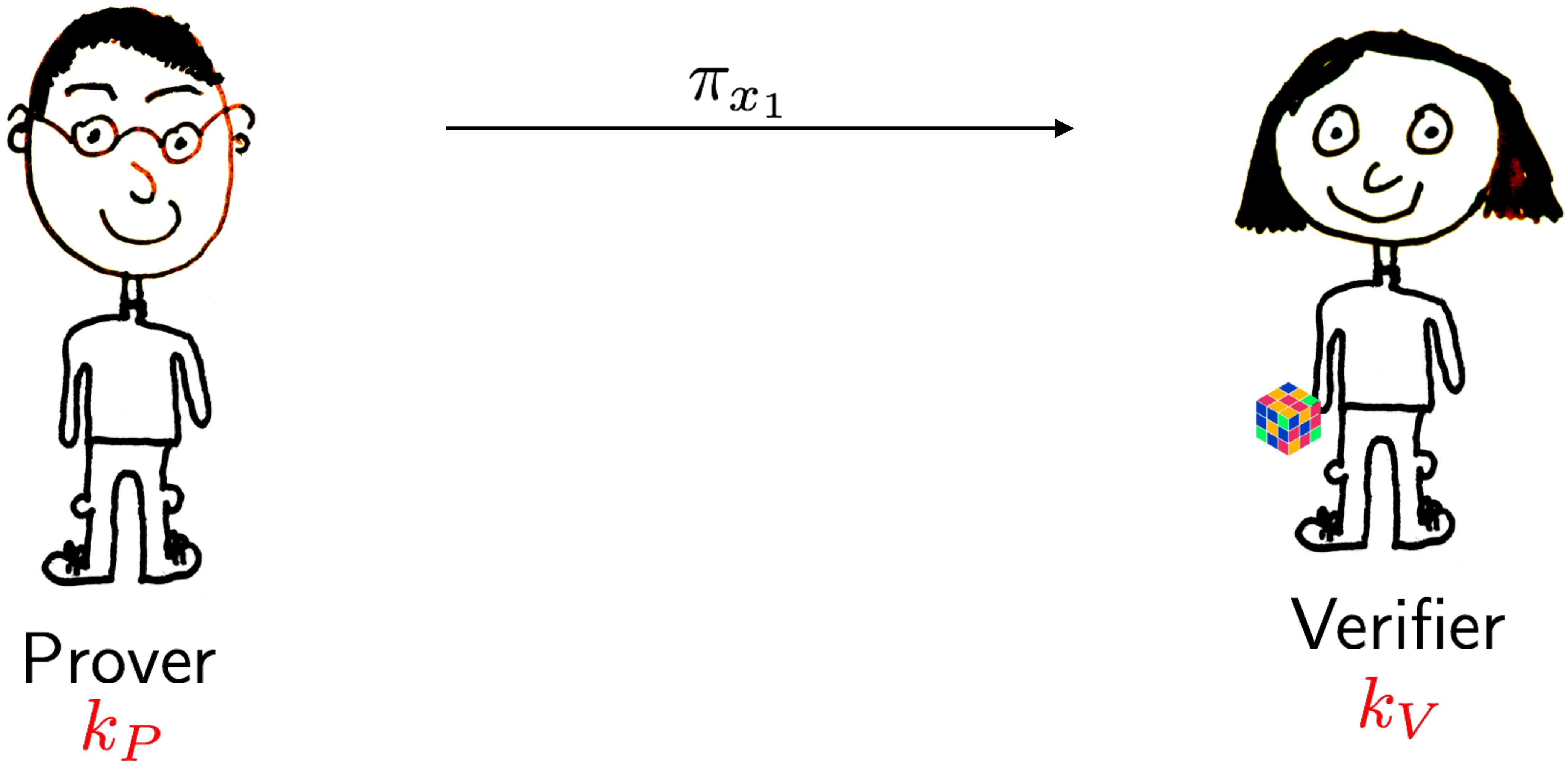
Prover
k_P



Verifier
k_V

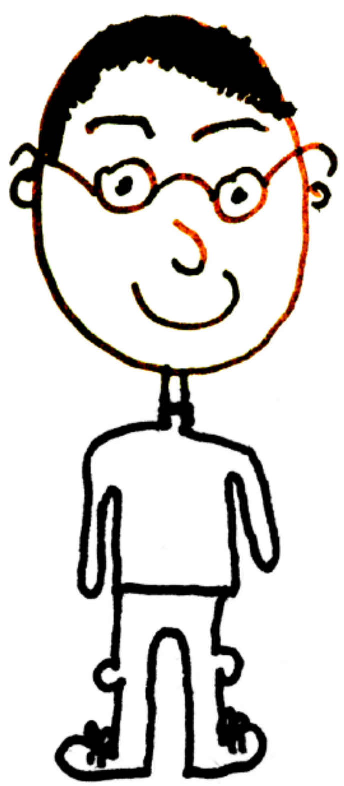
NIZK with Preprocessing [DMP88]
(Multi-Theorem)

NP Language $\mathcal{L}$

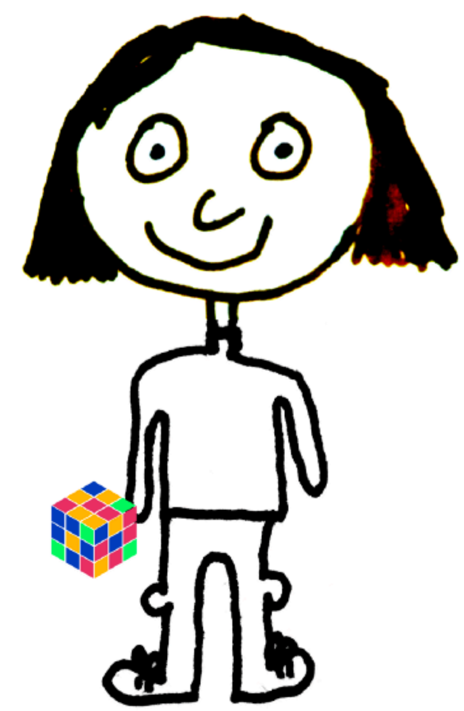


NIZK with Preprocessing [DMP88]
(Multi-Theorem)

NP Language $\mathcal{L}$



$\xrightarrow{\pi_{x_2}}$



Prover
k_P

Verifier
k_V

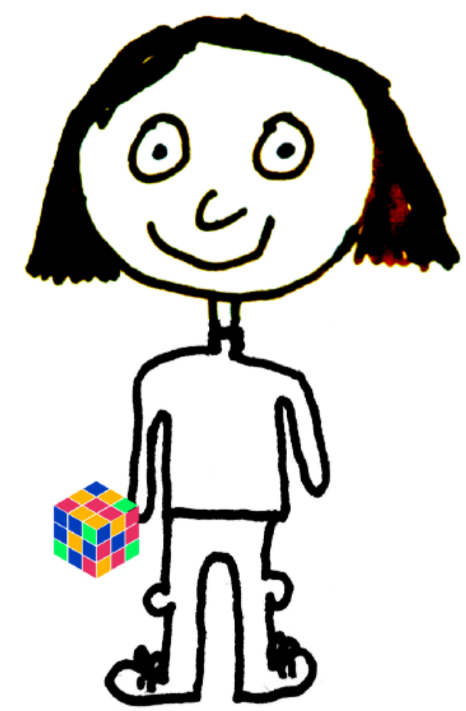
NIZK with Preprocessing [DMP88]
(Multi-Theorem)

NP Language $\mathcal{L}$

Prover

k_P

$$\xrightarrow{\pi_{x_3}}$$

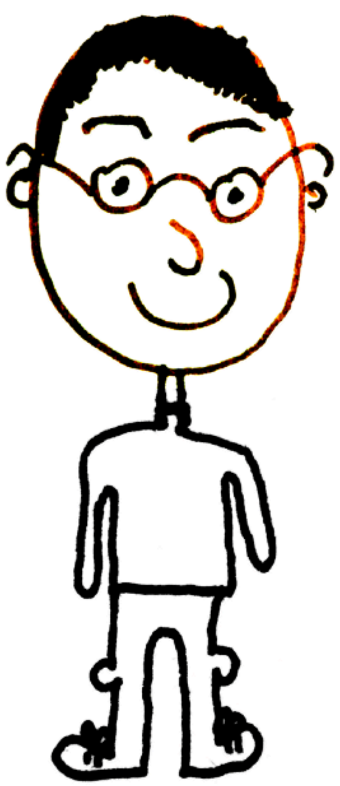


Verifier

k_V

NIZK with Preprocessing [DMP88]
(Multi-Theorem)

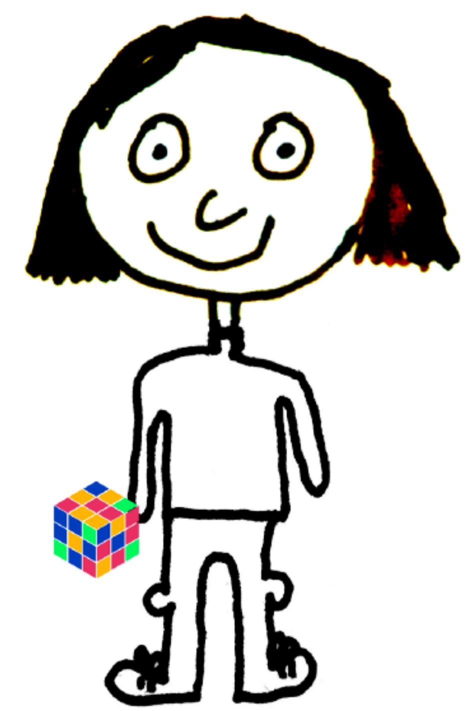
NP Language $\mathcal{L}$



Prover
k_P

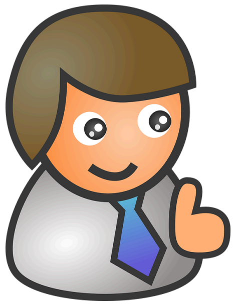
$$\xrightarrow{\pi_{x_3}}$$

$\vdots$



Verifier
k_V

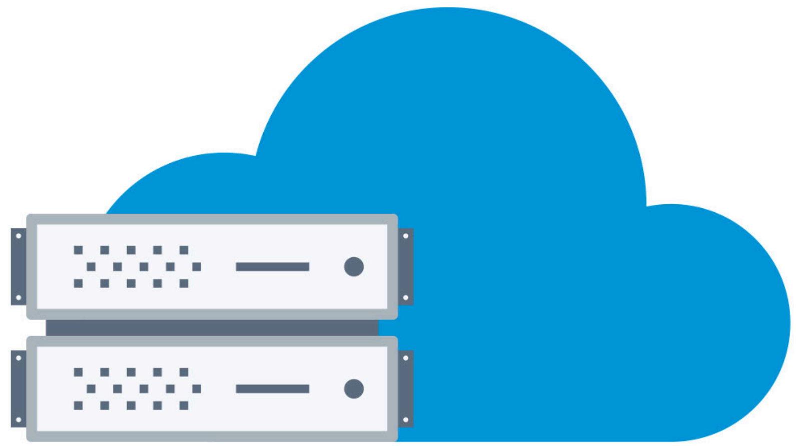
Homomorphic Signatures [BF11, GVW15]



$(\mathsf{sk}, \mathsf{pk})$

Homomorphic Signatures [BF11, GVW15]

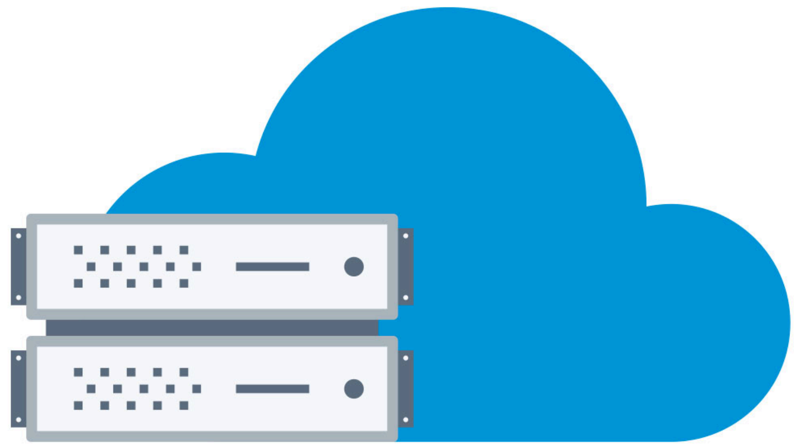
(sk, pk)



$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

Homomorphic Signatures [BF11, GVW15]

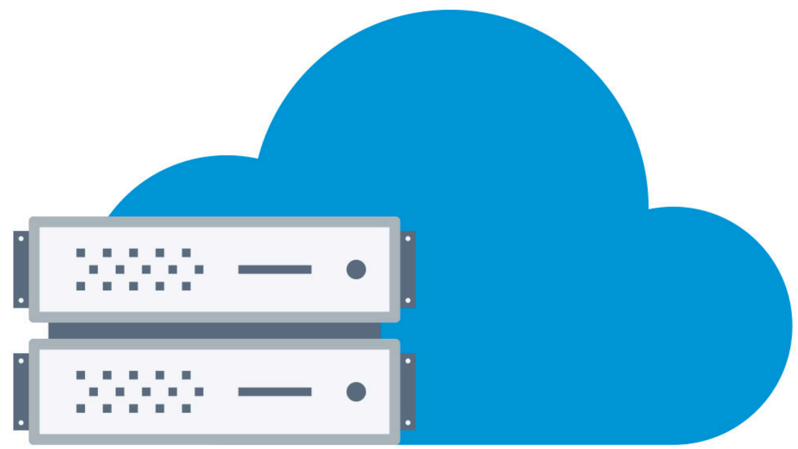
(sk, pk)



$$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$$

Homomorphic Signatures [BF11, GVW15]

$(\mathsf{sk}, \mathsf{pk})$

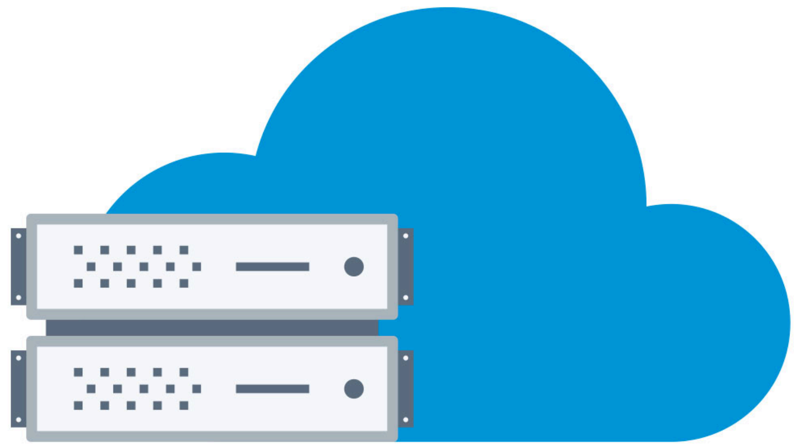


$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

$y = f(\mathbf{x}), \ \sigma^*_{\mathbf{x},f}$

Homomorphic Signatures [BF11, GVW15]

(sk, pk)



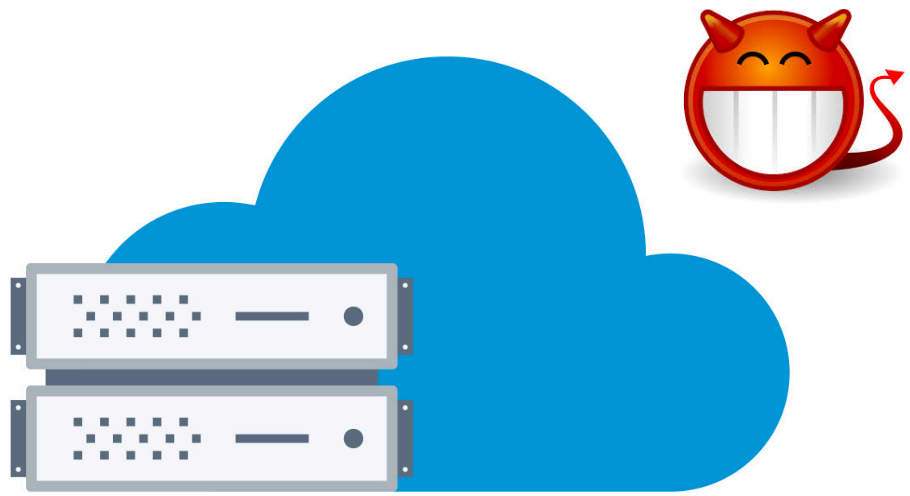
$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

$y = f(\mathbf{x}), \; \sigma^*_{\mathbf{x},f} \longrightarrow$

$\mathsf{Verify}_{\mathsf{pk}}(f, y, \sigma^*_{\mathbf{x},f})$

Homomorphic Signatures [BF11, GVW15]

(sk, pk)



$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

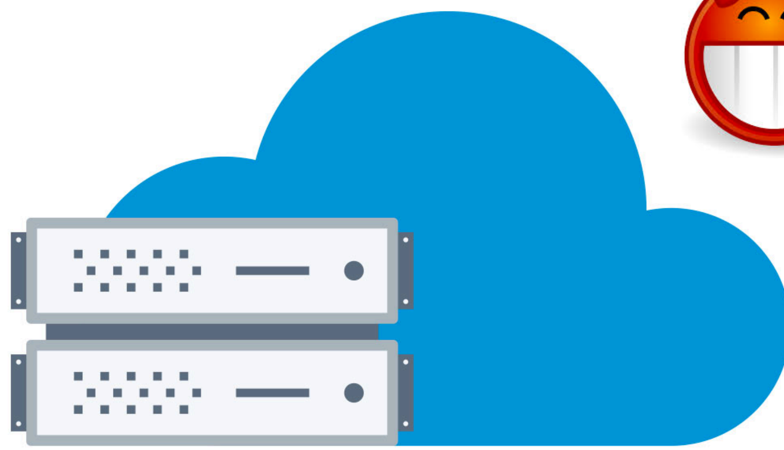
$y = f(\mathbf{x}), \ \sigma^*_{\mathbf{x},f}$

Unforgeability?

$\mathsf{Verify}_{\mathsf{pk}}(f, y, \sigma^*_{\mathbf{x},f})$

Homomorphic Signatures [BF11, GVW15]

(sk, pk)



$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

Unforgeability?

$y' \neq f(\mathbf{x}), \ \sigma^*_{\mathbf{x},f}$

$\mathsf{Verify}_{\mathsf{pk}}(f, y', \sigma^*_{\mathbf{x},f})$

Homomorphic Signatures [BF11, GVW15]

(sk, pk)

$$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$$

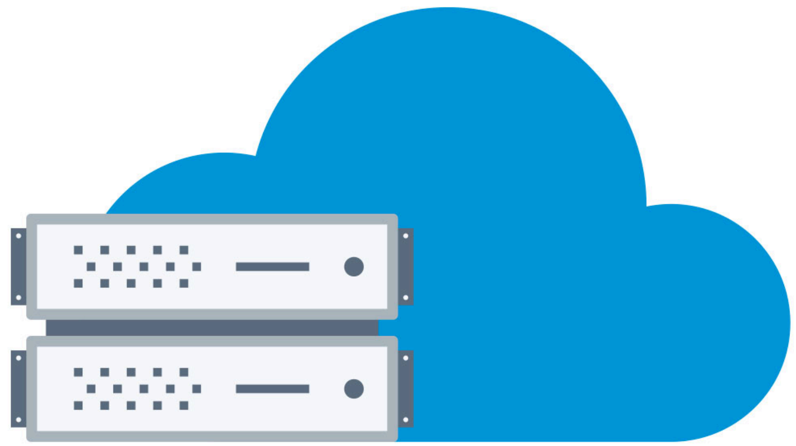
Unforgeability?

$$y' \neq f(\mathbf{x}), \ \sigma^*_{\mathbf{x},f}$$

$\mathsf{Verify}_{\mathsf{pk}}(f, y', \sigma^*_{\mathbf{x},f})$

Homomorphic Signatures [BF11, GVW15]

(sk, pk)



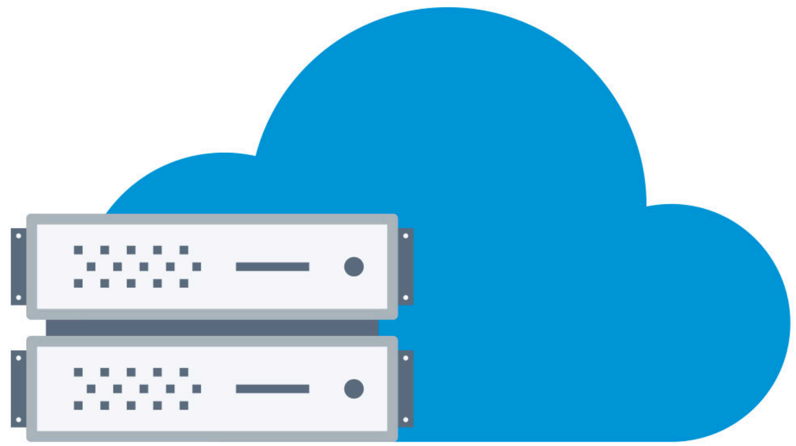
$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

$y = f(\mathbf{x}),\ \sigma^*_{\mathbf{x},f}$

Compactness: signature size $|\sigma^*_{\mathbf{x},f}|$ independent of $|\mathbf{x}|$

Homomorphic Signatures [BF11, GVW15]

(sk, pk)



$\mathbf{x} \in \{0,1\}^N, \quad \sigma_{\mathbf{x}}$

$y = f(\mathbf{x}),\ \sigma^*_{\mathbf{x},f} \longrightarrow$

$\mathsf{Verify}_{\mathsf{pk}}(f, y, \sigma^*_{\mathbf{x},f})$

Context-Hiding: $\sigma^*_{\mathbf{x},f}$ not reveal any more info about $\mathbf{x}$

Constructing Hom. Signatures [GVW15]

pp $= \mathbf{C}_1, \ldots, \mathbf{C}_N, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$

Constructing Hom. Signatures [GVW15]

$\mathsf{pp} = \mathbf{C}_1, \ldots, \mathbf{C}_N, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{sk} = \mathsf{td}_{\mathbf{A}}$

Constructing Hom. Signatures [GVW15]

$\mathsf{pp} = \mathbf{C}_1, \ldots, \mathbf{C}_N, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{sk} = \mathsf{td}_{\mathbf{A}}$

Signature for $\mathbf{x} = (x_1, \ldots, x_N)$ consists of short matrices

$\sigma_{\mathbf{x}} = \mathbf{R}_1, \ldots, \mathbf{R}_N \in \mathbb{Z}^{m \times m}$ such that

$$\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G} = \mathbf{C}_1$$

$$\vdots$$

$$\mathbf{A} \cdot \mathbf{R}_N + x_N \cdot \mathbf{G} = \mathbf{C}_N$$

Constructing Hom. Signatures [GVW15]

$\mathsf{pp} = \mathbf{C}_1, \ldots, \mathbf{C}_N, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{sk} = \mathsf{td}_{\mathbf{A}}$

Signature for $\mathbf{x} = (x_1, \ldots, x_N)$ consists of short matrices

$\sigma_{\mathbf{x}} = \mathbf{R}_1, \ldots, \mathbf{R}_N \in \mathbb{Z}^{m \times m}$ such that

$$\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G} = \mathbf{C}_1$$

$$\vdots$$

$$\mathbf{A} \cdot \mathbf{R}_N + x_N \cdot \mathbf{G} = \mathbf{C}_N$$

$$\xrightarrow{\text{GSW}} \quad \mathbf{A} \cdot \mathbf{R}_f + f(\mathbf{x}) \cdot \mathbf{G} = \mathbf{C}_f$$

Constructing Hom. Signatures [GVW15]

$\mathsf{pp} = \mathbf{C}_1, \ldots, \mathbf{C}_N, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$\mathsf{sk} = \mathsf{td}_{\mathbf{A}}$

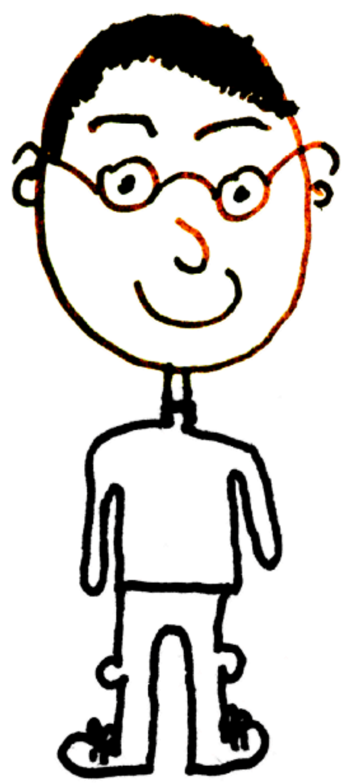
Signature for $\mathbf{x} = (x_1, \ldots, x_N)$ consists of short matrices

$\sigma_{\mathbf{x}} = \mathbf{R}_1, \ldots, \mathbf{R}_N \in \mathbb{Z}^{m \times m}$ such that

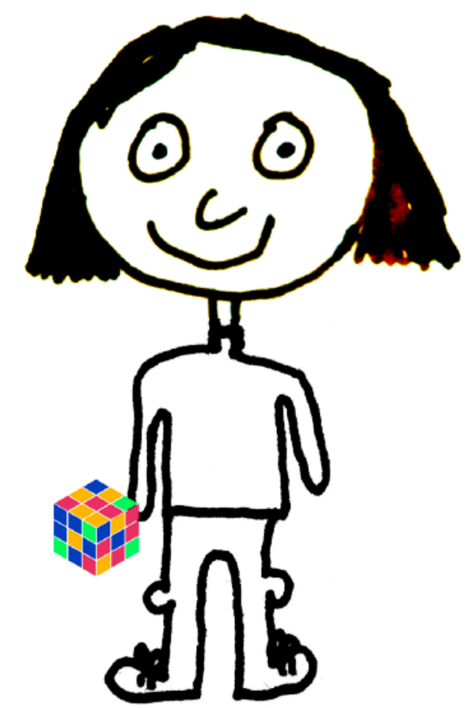
$$\begin{matrix} \mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G} = \mathbf{C}_1 \\ \vdots \\ \mathbf{A} \cdot \mathbf{R}_N + x_N \cdot \mathbf{G} = \mathbf{C}_N \end{matrix} \xrightarrow{\text{GSW}} \mathbf{A} \cdot \mathbf{R}_f + f(\mathbf{x}) \cdot \mathbf{G} = \mathbf{C}_f$$

Need extra step for context-hiding

Homomorphic Signatures to NIZK?

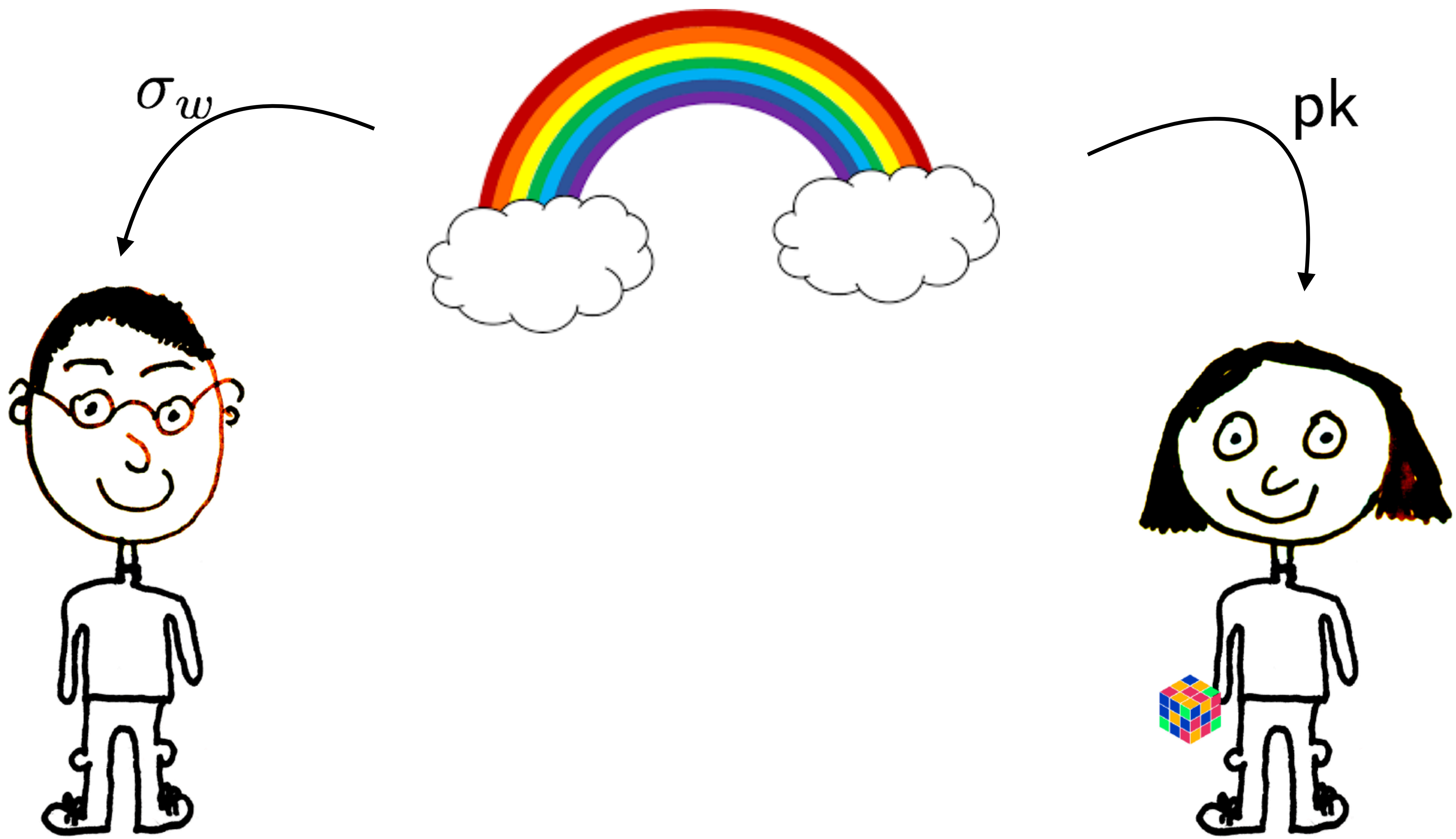


Prover(x, w)



Verifier(x)

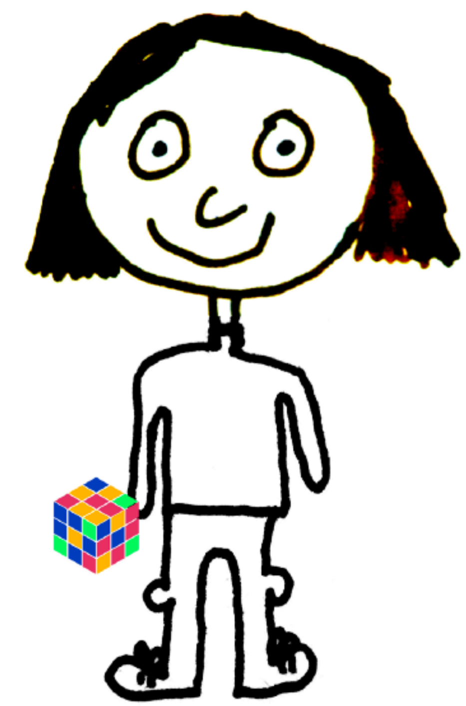
Homomorphic Signatures to NIZK?



$\mathsf{Prover}(x, w)$ $\mathsf{Verifier}(x)$

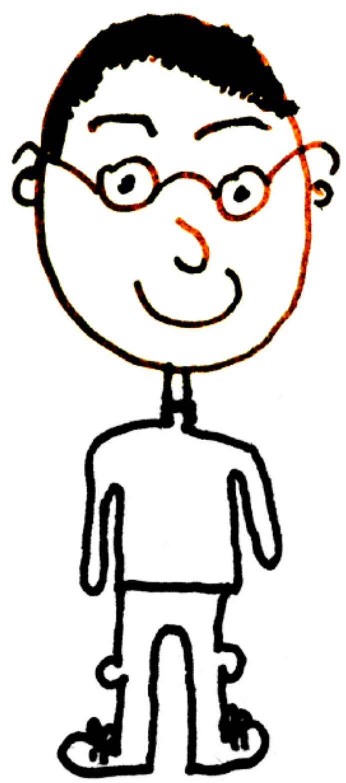
Homomorphic Signatures to NIZK?

$\mathsf{Prover}(x, w)$
σ_w

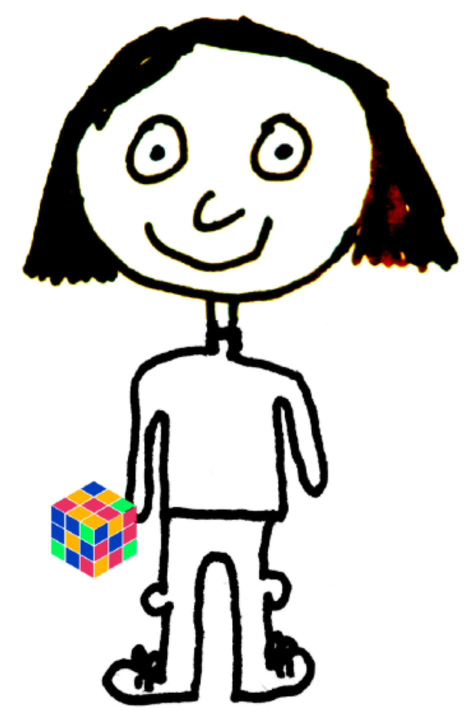


$\mathsf{Verifier}(x)$
pk

Homomorphic Signatures to NIZK?



$$\xrightarrow{\quad 1 = f_x(w),\ \sigma^*_{w,f_x} \quad}$$

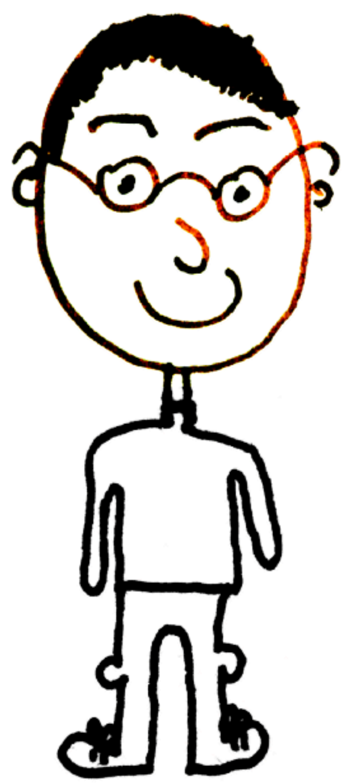


$\mathsf{Prover}(x, w)$
σ_w

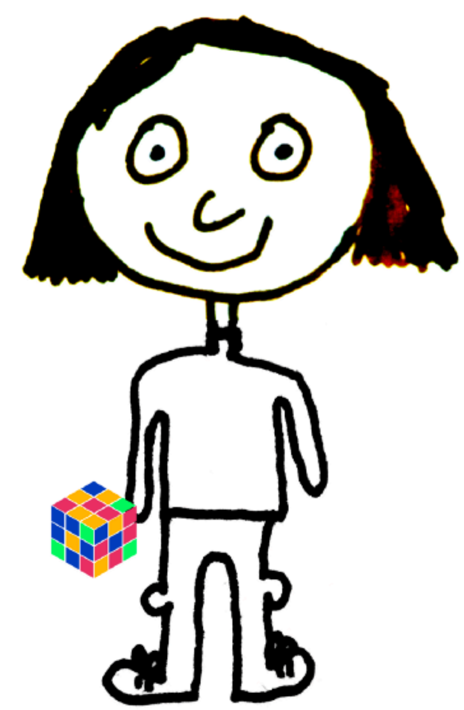
$\mathsf{Verifier}(x)$
pk

Homomorphic Signatures to NIZK?

$$f_x(w) = \mathcal{R}(x, w)$$



$$\xrightarrow{1 = f_x(w),\ \sigma^*_{w,f_x}}$$



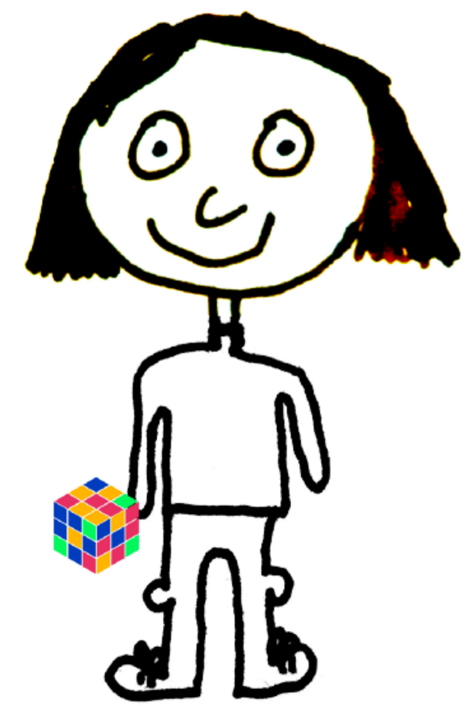
Prover(x, w)
σ_w

Verifier(x)
pk

Homomorphic Signatures to NIZK?

$$f_x(w) = \mathcal{R}(x, w)$$

$$1 = f_x(w),\ \sigma^*_{w,f_x} \longrightarrow$$



$\mathsf{Prover}(x, w)$

σ_w

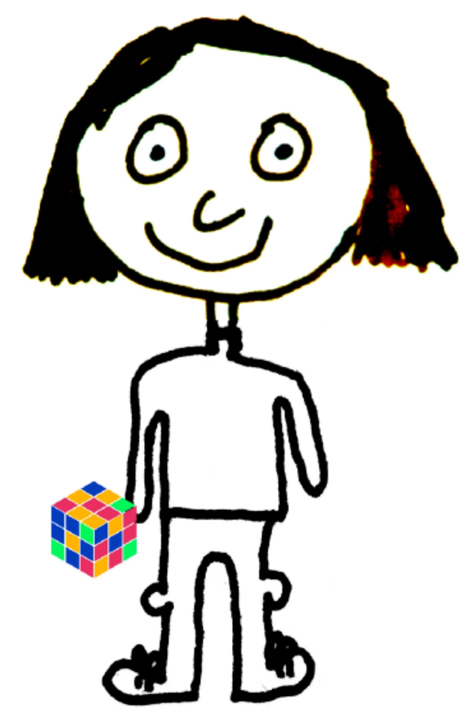
$\mathsf{Verifier}(x)$

$\mathsf{Verify}_{\mathsf{pk}}(f_x, 1, \sigma^*_{w,f_x})$

Homomorphic Signatures to NIZK?

$$f_x(w) = \mathcal{R}(x, w)$$

$$\xrightarrow{1 = f_x(w),\ \sigma^*_{w,f_x}}$$



$\mathsf{Prover}(x, w)$

σ_w

$\mathsf{Verifier}(x)$

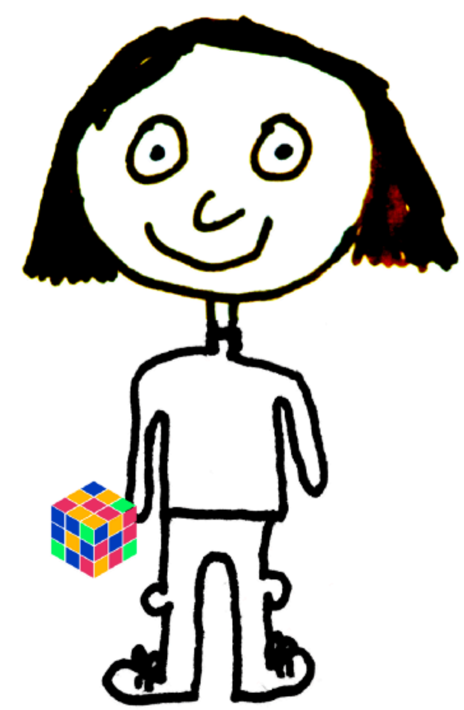
$\mathsf{Verify}_{\mathsf{pk}}(f_x, 1, \sigma^*_{w,f_x})$

1. **HS Correctness** implies **NIZK Completeness**

Homomorphic Signatures to NIZK?

$$f_x(w) = \mathcal{R}(x, w)$$

$$\xrightarrow{\quad 1 \neq f_x(w),\ \ \sigma^*_{w,f_x} \quad}$$



$\mathsf{Prover}(x, w)$

σ_w

$\mathsf{Verifier}(x)$

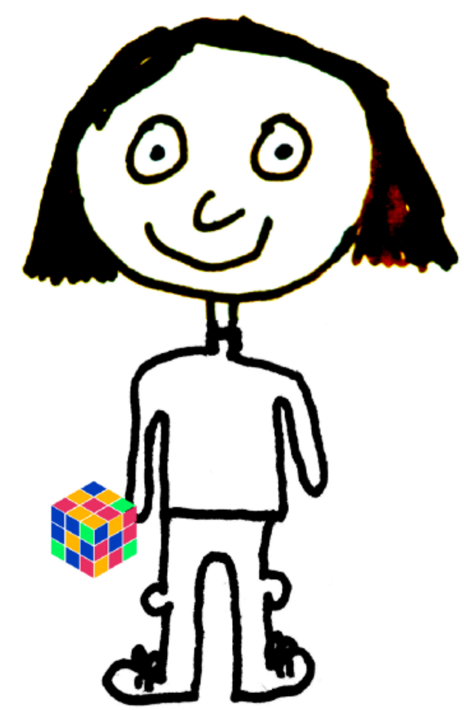
$\mathsf{Verify}_{\mathsf{pk}}(f_x, 1, \sigma^*_{w,f_x})$

2. **HS Unforgeability** implies **NIZK Soundness**

Homomorphic Signatures to NIZK?

$$f_x(w) = \mathcal{R}(x, w)$$

$$\xrightarrow{\quad 1 = f_x(w),\ \sigma^*_{w,f_x} \quad}$$



Prover(x, w)

σ_w

Verifier(x)

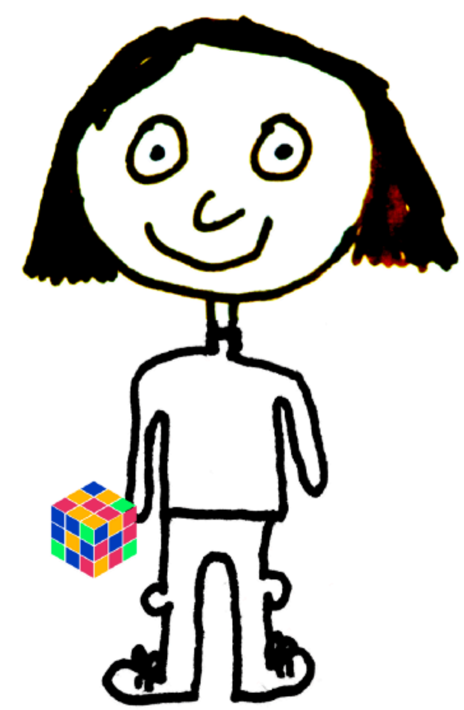
Verify$_{\mathsf{pk}}(f_x, 1, \sigma^*_{w,f_x})$

3. **HS Context-Hiding** implies **NIZK Zero-Knowledge**

Homomorphic Signatures to NIZK?

$$f_x(w) = \mathcal{R}(x, w)$$

$$1 = f_x(w),\ \sigma^*_{w,f_x}$$



$\mathsf{Prover}(x, w)$

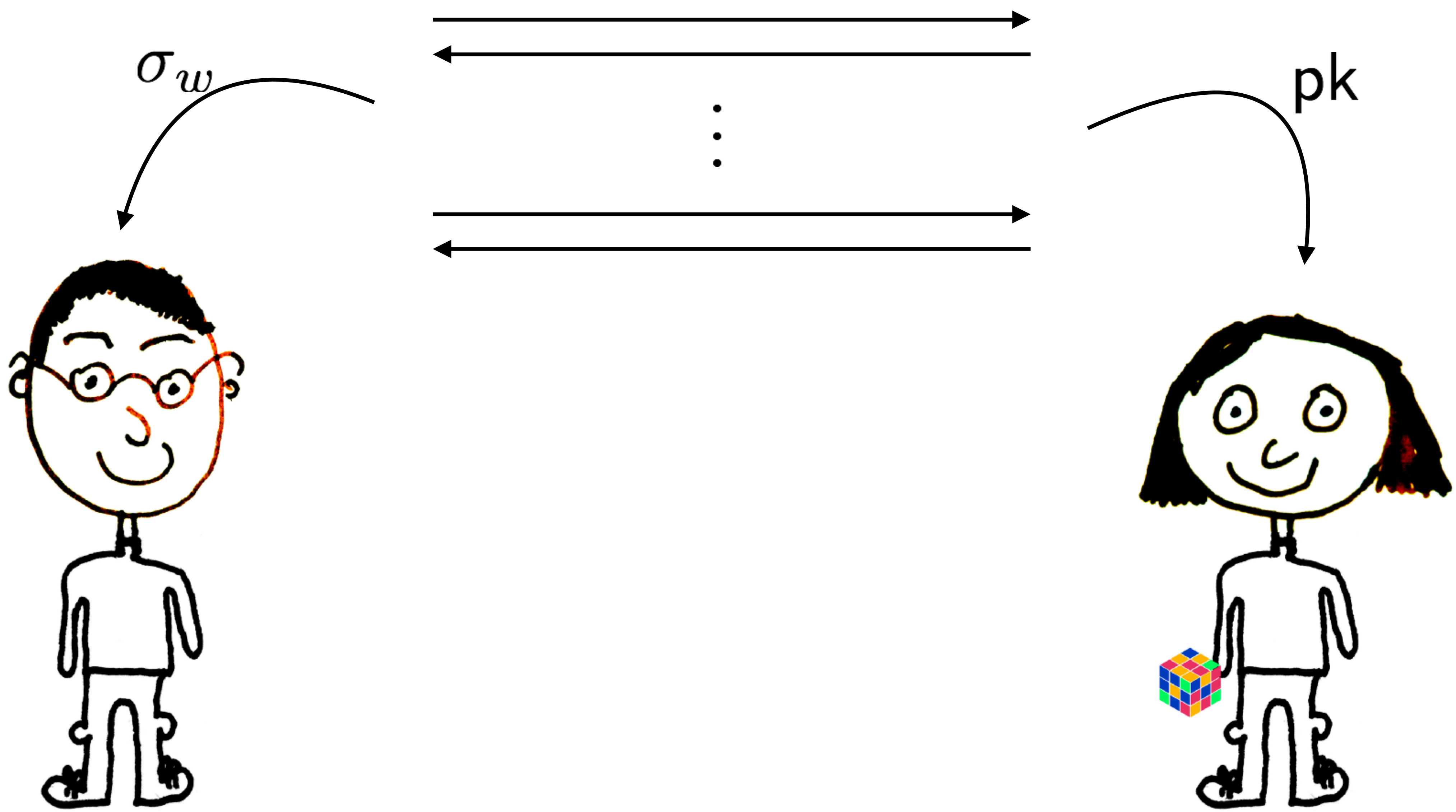
σ_w

$\mathsf{Verifier}(x)$

$\mathsf{Verify}_{\mathsf{pk}}(f_x, 1, \sigma^*_{w,f_x})$

How can the prover get its witness signed?

Homomorphic Signatures to NIZK?

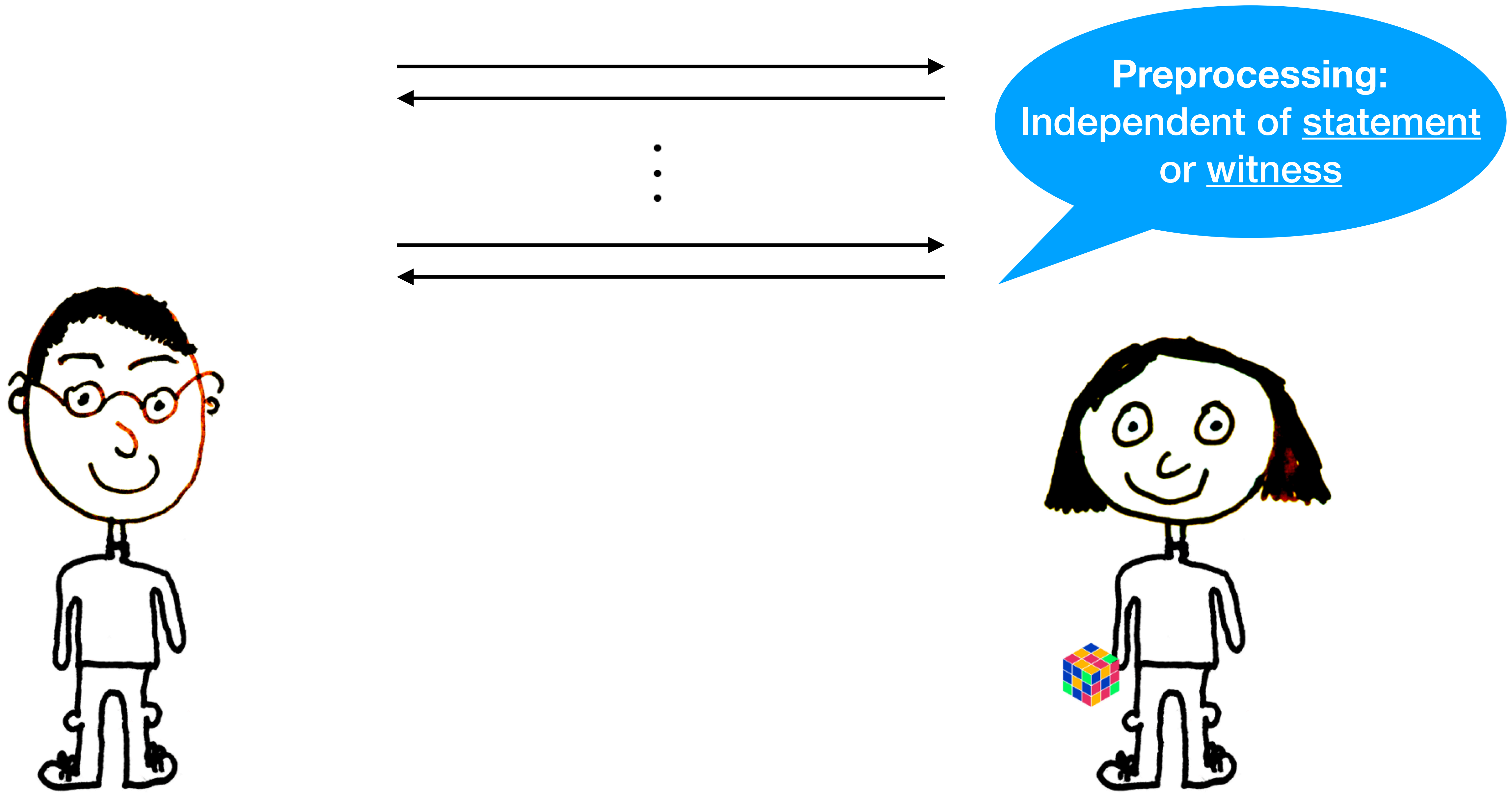


Prover(x, w) Verifier(x)

σ_w pk

How can the prover get its witness signed?

Homomorphic Signatures to NIZK?

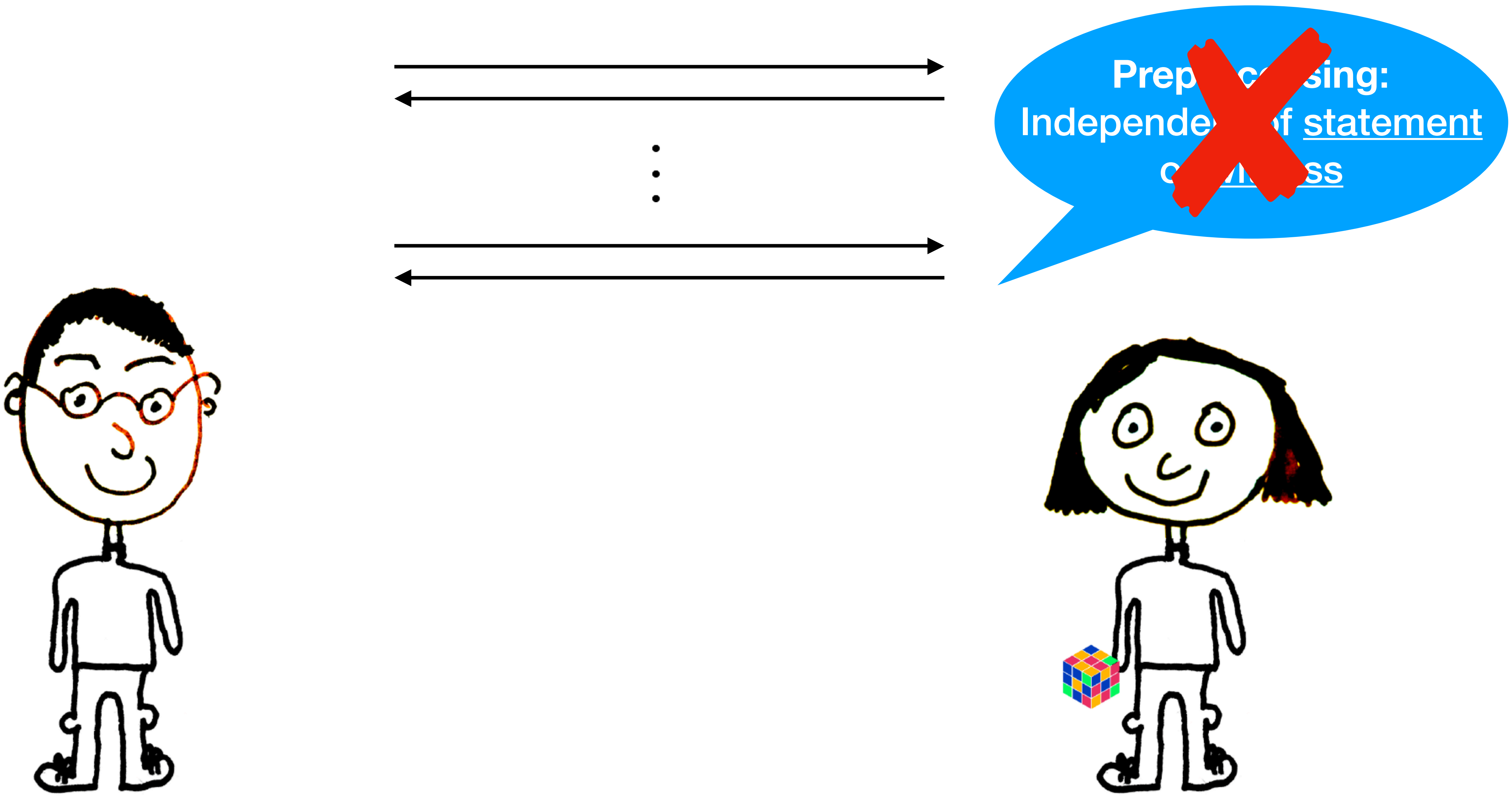


Prover(x, w)
σ_w

Verifier(x)
pk

How can the prover get its witness signed?

Homomorphic Signatures to NIZK?



Prover(x, w)
σ_w

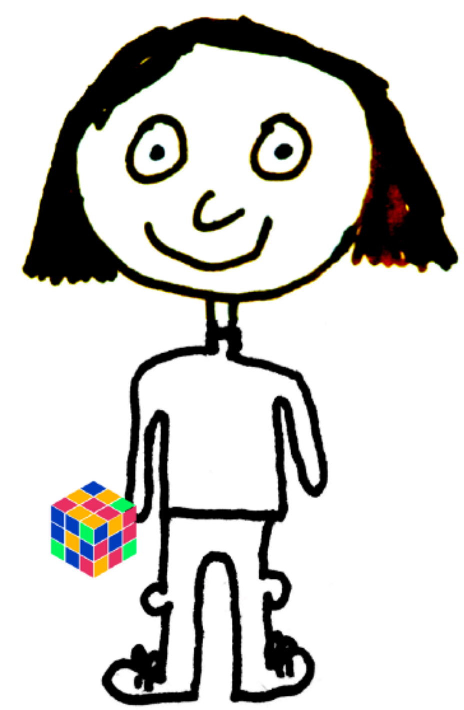
Verifier(x)
pk

How can the prover get its witness signed?

Homomorphic Signatures to NIZK?

$$\mathsf{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$
$$\mathsf{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

$\mathsf{Prover}(x, w)$
σ_k



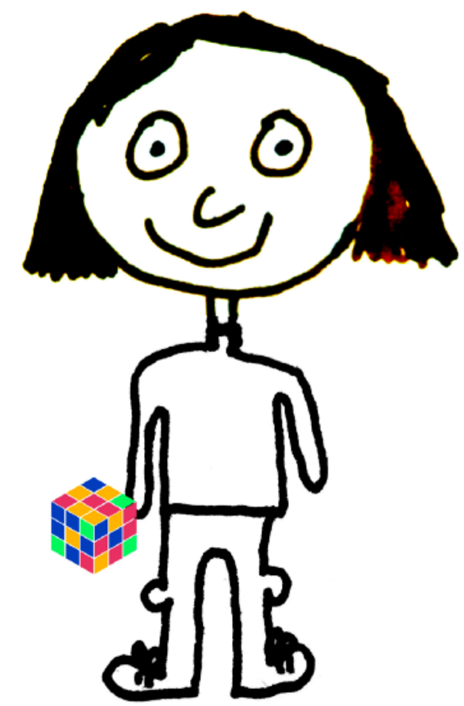
$\mathsf{Verifier}(x)$
pk

Homomorphic Signatures to NIZK?

$$\mathsf{Enc} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$$
$$\mathsf{Dec} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$$

$$\xrightarrow{\mathsf{ct} = \mathsf{Enc}(k, w)}$$

$$\xrightarrow{\sigma^*_{k, g_x, \mathsf{ct}}}$$



$\mathsf{Prover}(x, w)$
σ_k

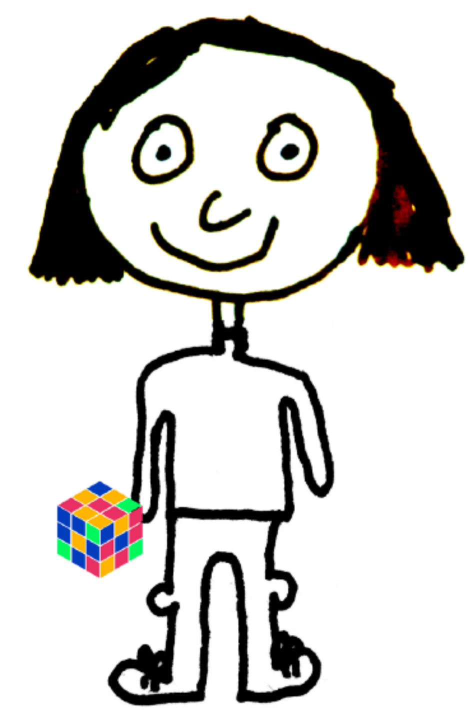
$\mathsf{Verifier}(x)$
pk

Homomorphic Signatures to NIZK?

$$g_{x,\mathsf{ct}}(k) = \mathcal{R}(x, \mathsf{Dec}(k, \mathsf{ct}))$$

$\mathsf{ct} = \mathsf{Enc}(k, w)$

$\sigma^*_{k, g_{x,\mathsf{ct}}}$



$\mathsf{Prover}(x, w)$
σ_k

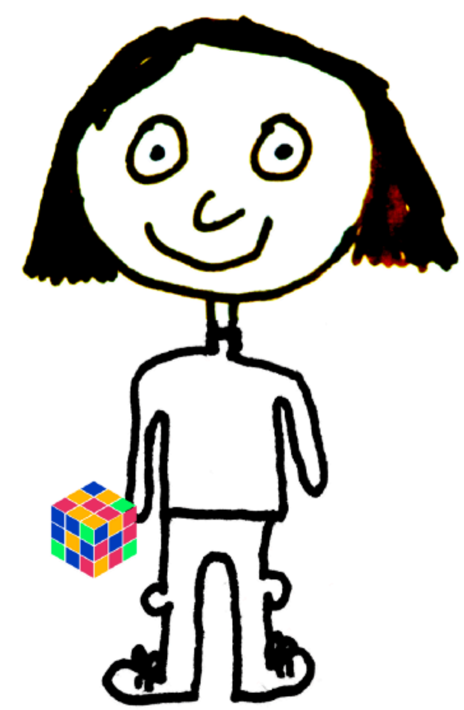
$\mathsf{Verifier}(x)$
pk

Homomorphic Signatures to NIZK?

$$g_{x,\mathsf{ct}}(k) = \mathcal{R}(x, \mathsf{Dec}(k, \mathsf{ct}))$$

$$\xrightarrow{\mathsf{ct} = \mathsf{Enc}(k, w)}$$

$$\xrightarrow{\sigma^*_{k, g_{x,\mathsf{ct}}}}$$



$\mathsf{Prover}(x, w)$

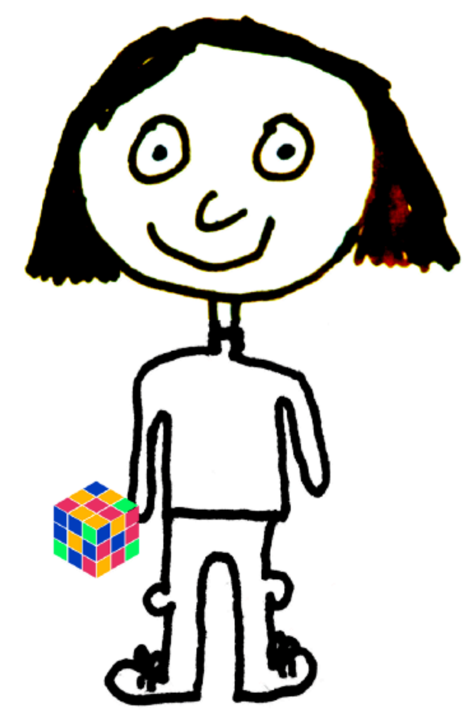
σ_k

$\mathsf{Verifier}(x)$

$\mathsf{Verify}_{\mathsf{pk}}(g_{x,\mathsf{ct}}, 1, \sigma^*_{k, g_{x,\mathsf{ct}}})$

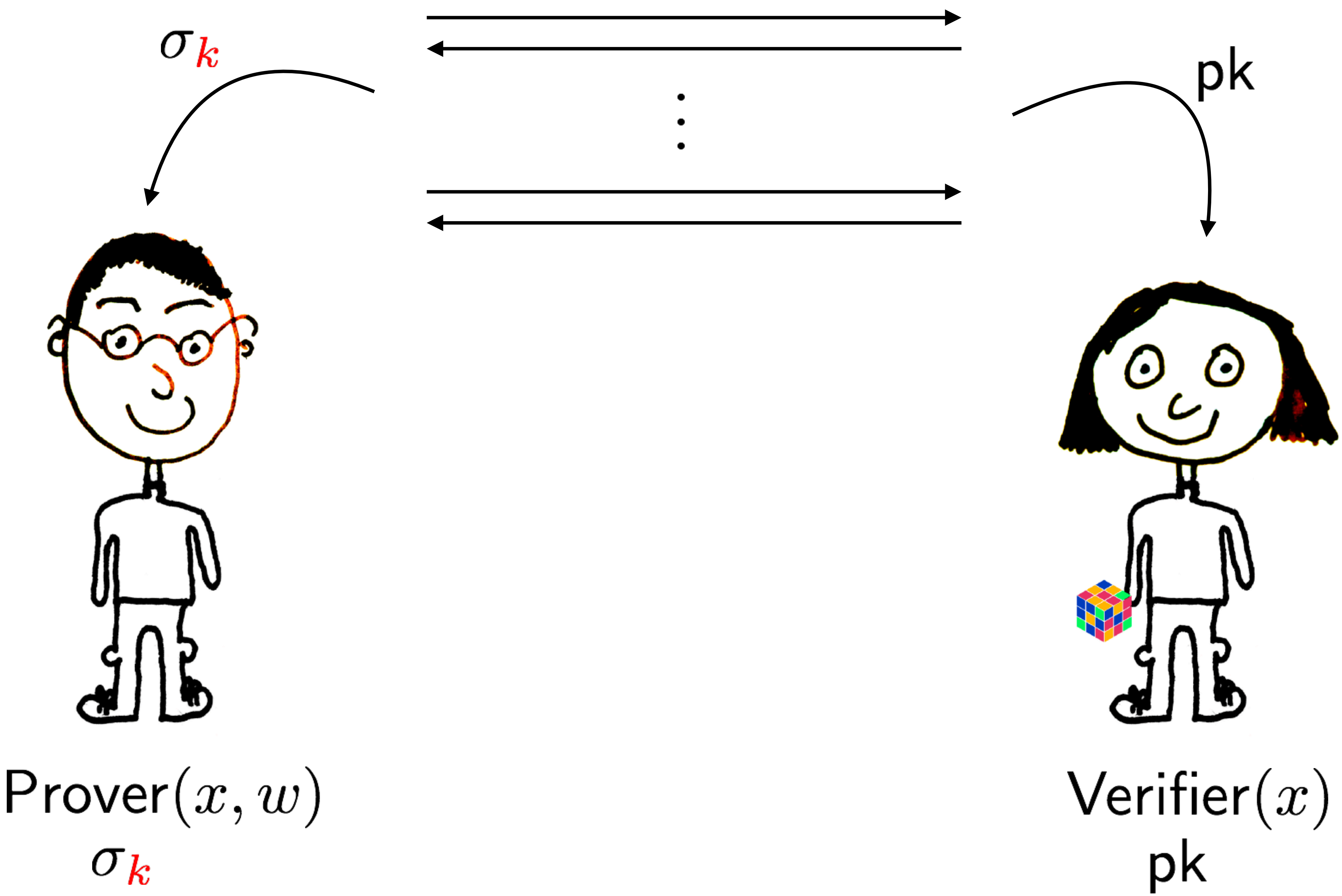
Homomorphic Signatures to NIZK?

Prover(x, w)
σ_k

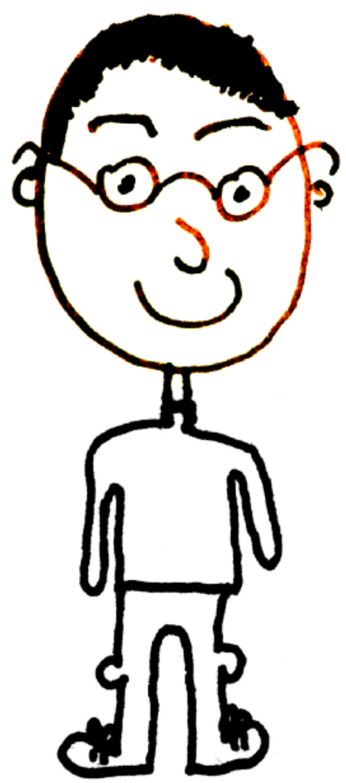


Verifier(x)
pk

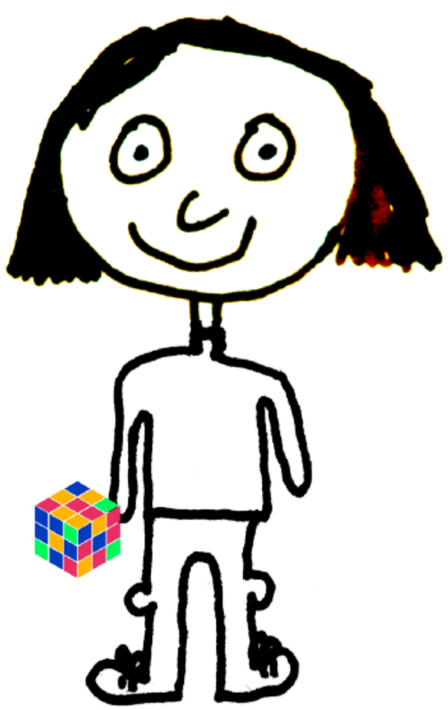
Homomorphic Signatures to NIZK?



Homomorphic Signatures to NIZK?



Prover(x, w)
σ_k

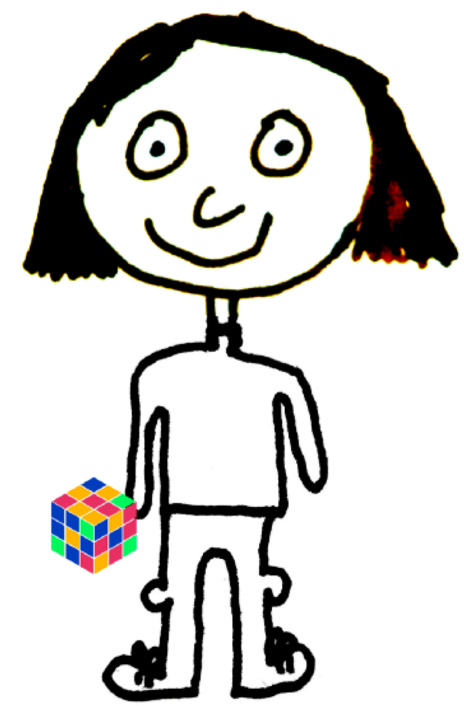


Verifier(x)
pk

Homomorphic Signatures to NIZK?

Prover(x, w)
σ_{k}

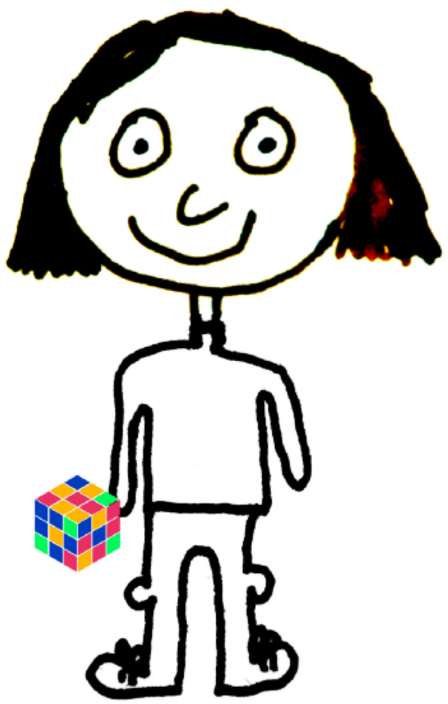
$$\pi_1 = (\mathsf{ct}_1, \sigma_1^*) \longrightarrow$$



Verifier(x)
pk

Homomorphic Signatures to NIZK?

$$\pi_2 = (\mathsf{ct}_2, \sigma_2^*) \longrightarrow$$



$\mathsf{Prover}(x, w)$
σ_{k}

$\mathsf{Verifier}(x)$
pk

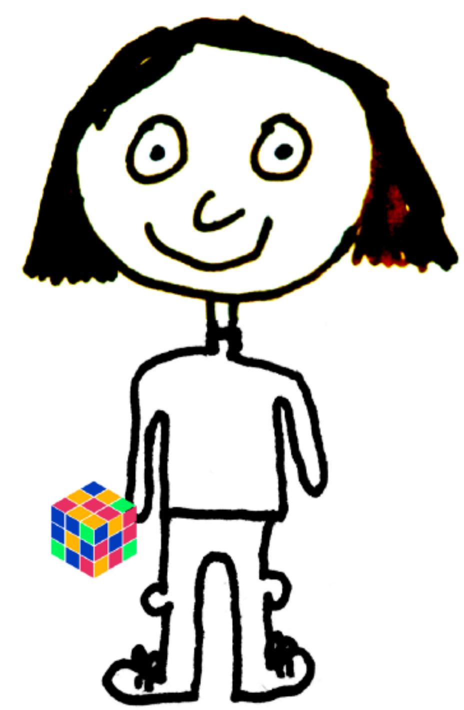
Homomorphic Signatures to NIZK?

$\mathsf{Prover}(x, w)$

σ_{k}

$$\xrightarrow{\pi_3 = (\mathsf{ct}_3, \sigma_3^*)}$$

$$\vdots$$



$\mathsf{Verifier}(x)$

pk

How should we do <u>Preprocessing</u>?

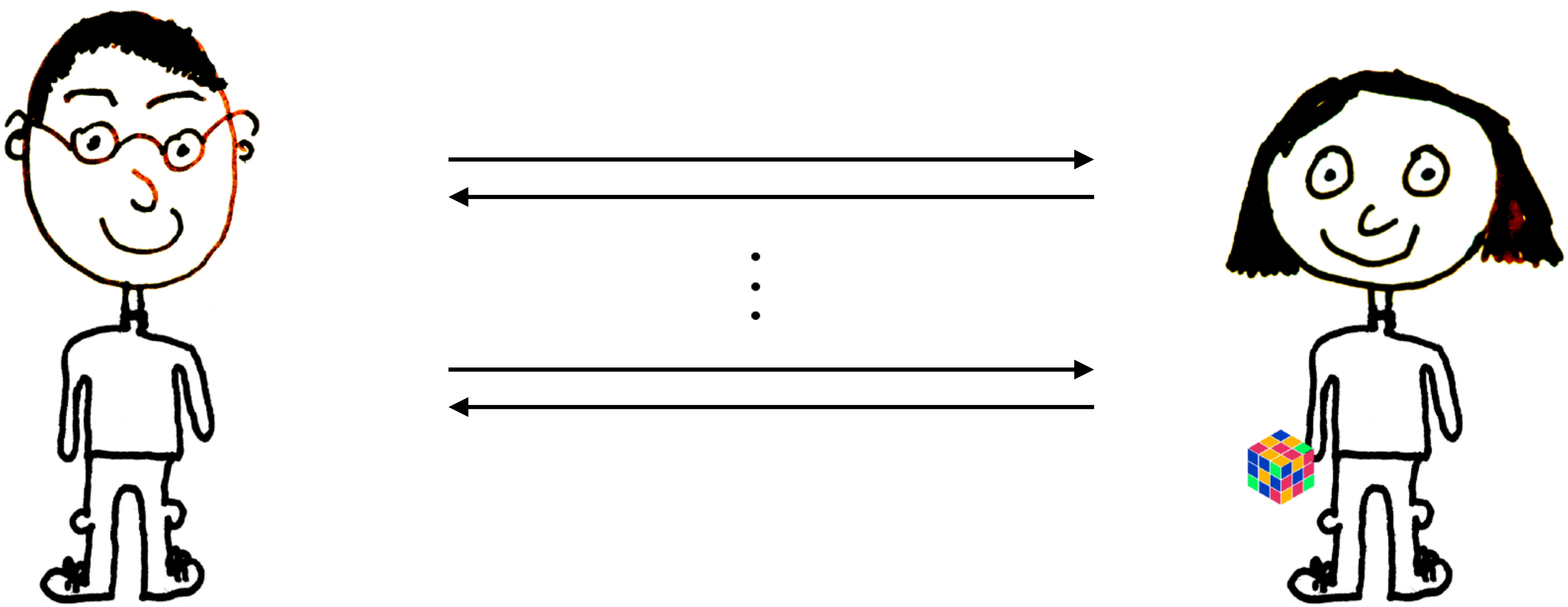
Generically: Either have to use many rounds or non-black use (costly)

How should we do <u>**Preprocessing**</u>?

Generically: Either have to use many rounds or non-black use (costly)

Let's just construct directly!

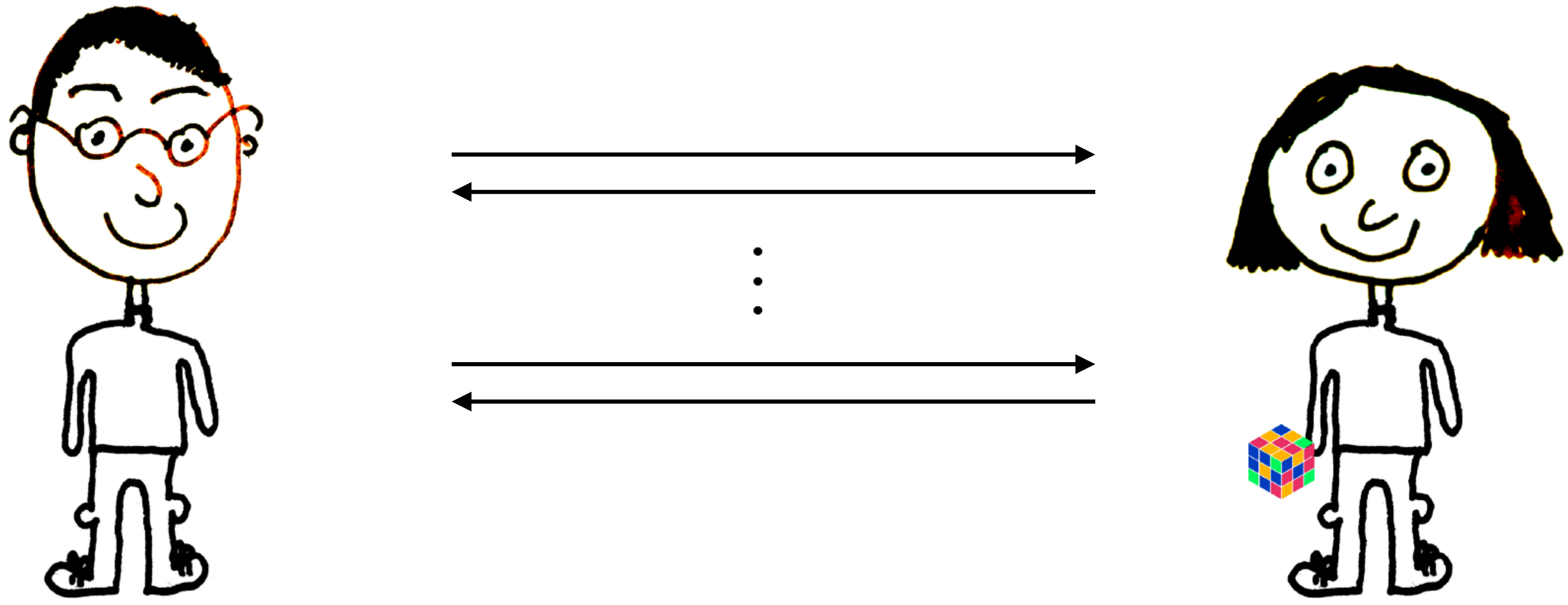
How should we do **Preprocessing**?



Prover(x, w)
σ_{k}

Verifier(x)
pk

How should we do **Preprocessing**?



Receiver

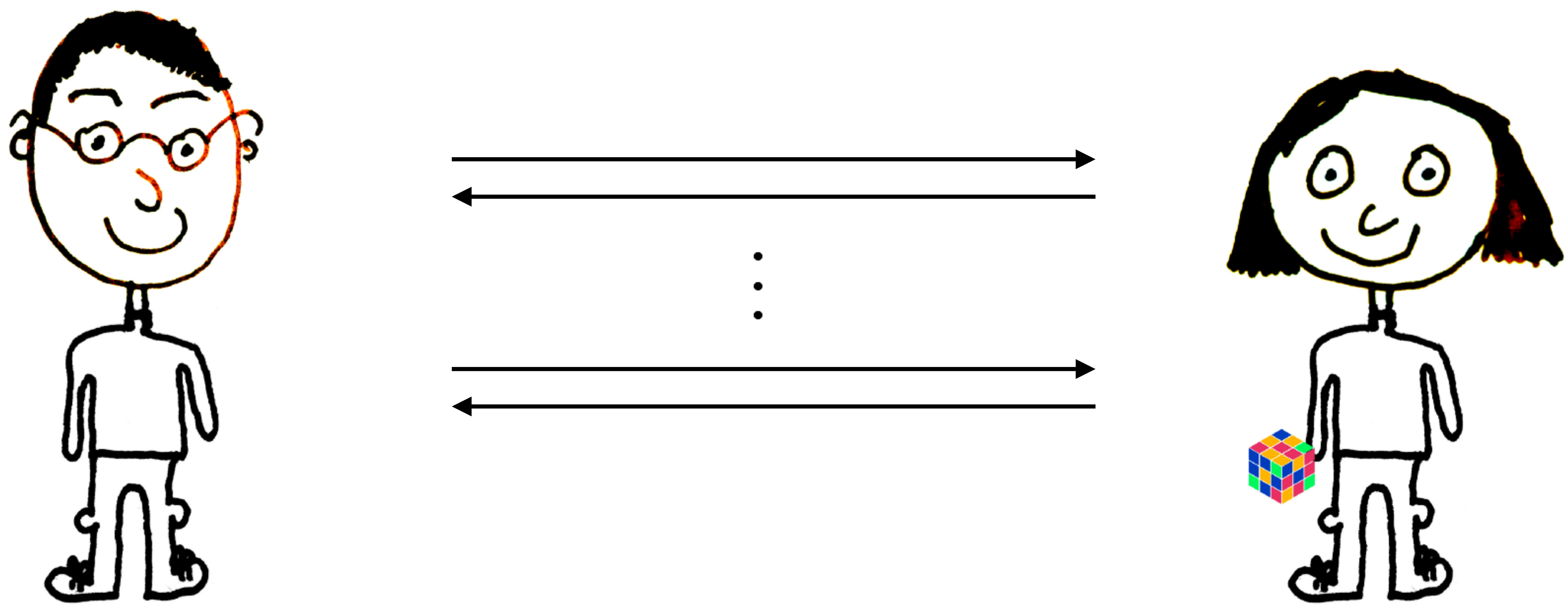
σ_m

Signer

$(\mathsf{sk}, \mathsf{pk})$

How should we do **Preprocessing**?

Blind Signatures?



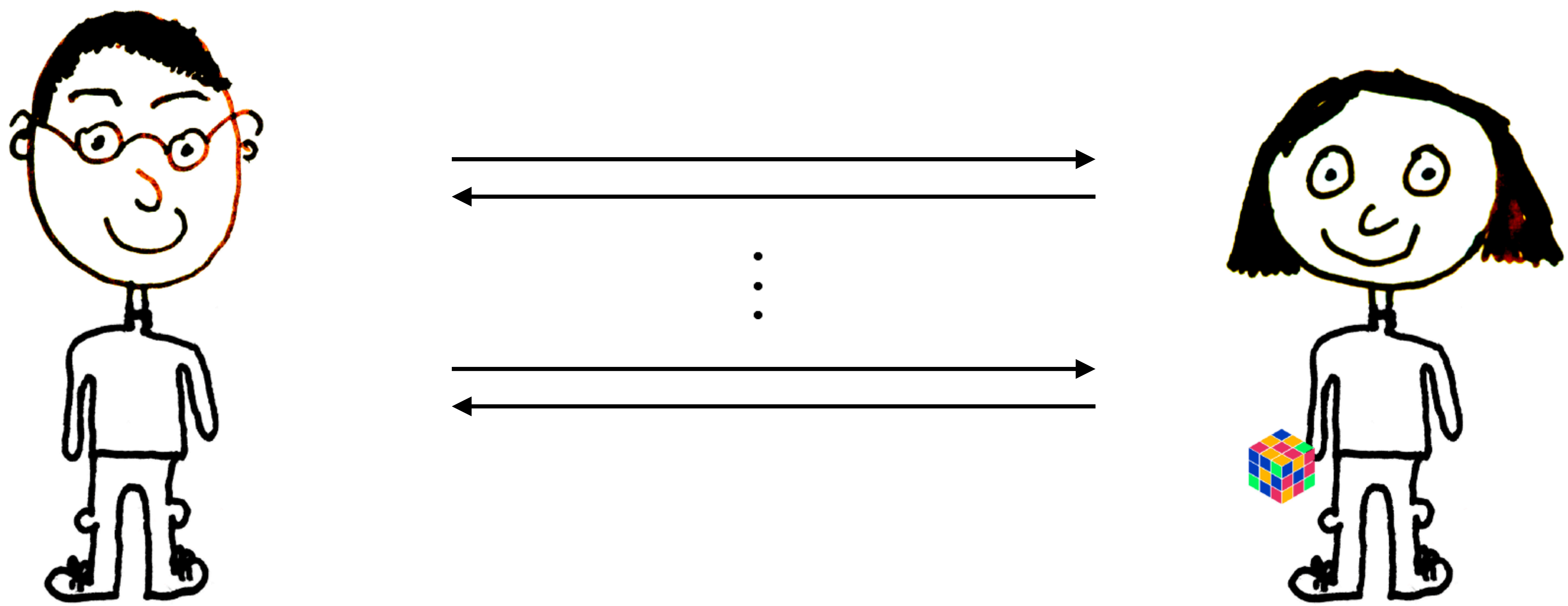
Receiver σ_m

Signer $(\mathsf{sk}, \mathsf{pk})$

How should we do **Preprocessing**?

Blind Signatures?

Blind Homomorphic Signatures (BHS)



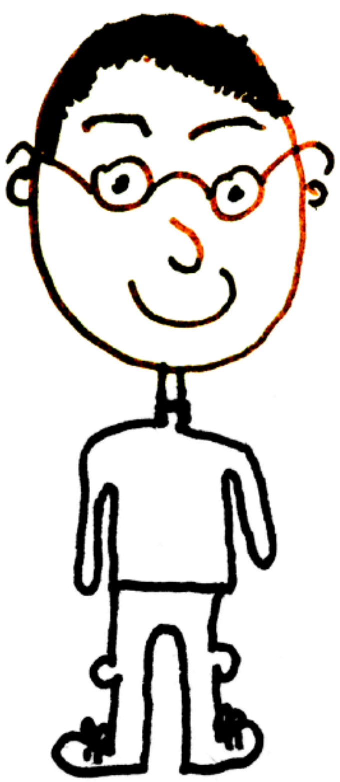
Receiver

σ_m

Signer

(sk, pk)

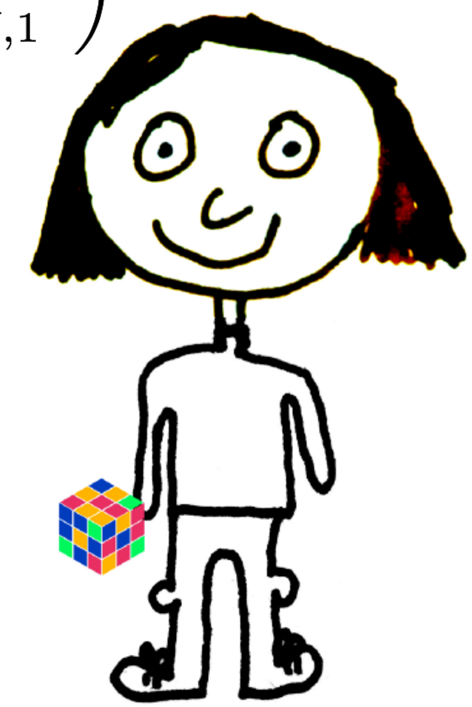
Blind Homomorphic Signatures



Receiver

m

$$\begin{pmatrix} \sigma_{1,0} & \sigma_{2,0} & \cdots & \sigma_{N,0} \\ \sigma_{1,1} & \sigma_{2,1} & \cdots & \sigma_{N,1} \end{pmatrix}$$



Signer

Blind Homomorphic Signatures

Receiver

m

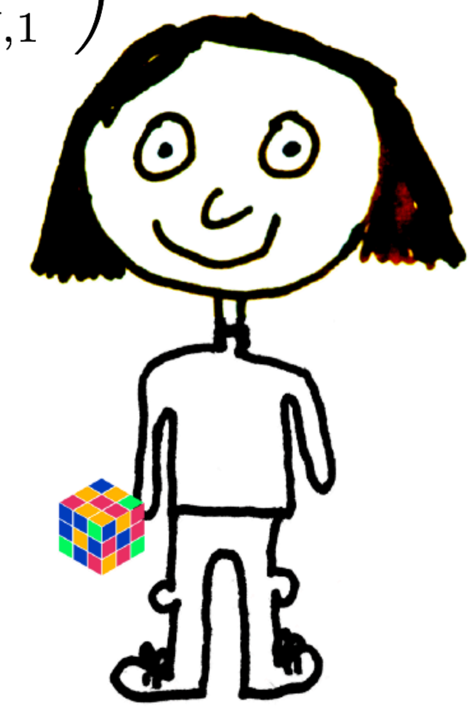
$$\begin{pmatrix} \sigma_{1,0} & \sigma_{2,0} & \cdots & \sigma_{N,0} \\ \sigma_{1,1} & \sigma_{2,1} & \cdots & \sigma_{N,1} \end{pmatrix}$$

$$\xleftrightarrow{\text{2-out-1 OT}}$$

$$\vdots$$

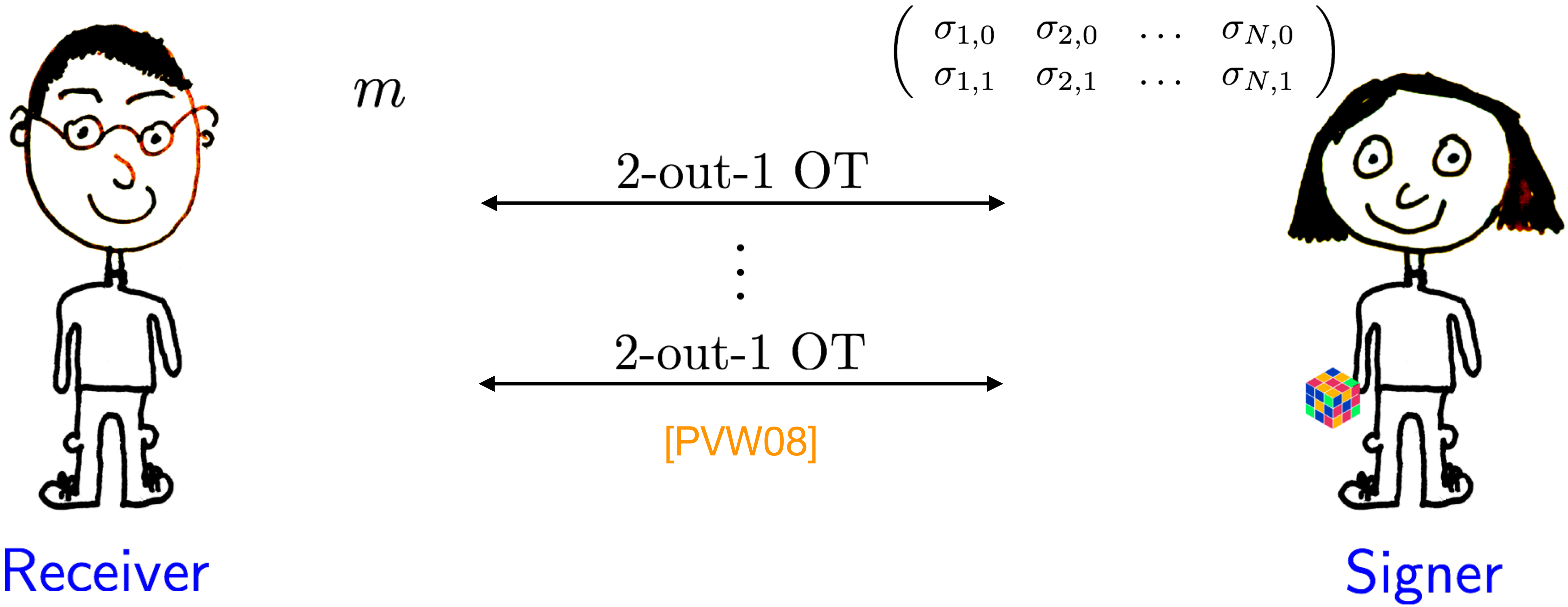
$$\xleftrightarrow{\text{2-out-1 OT}}$$

[PVW08]



Signer

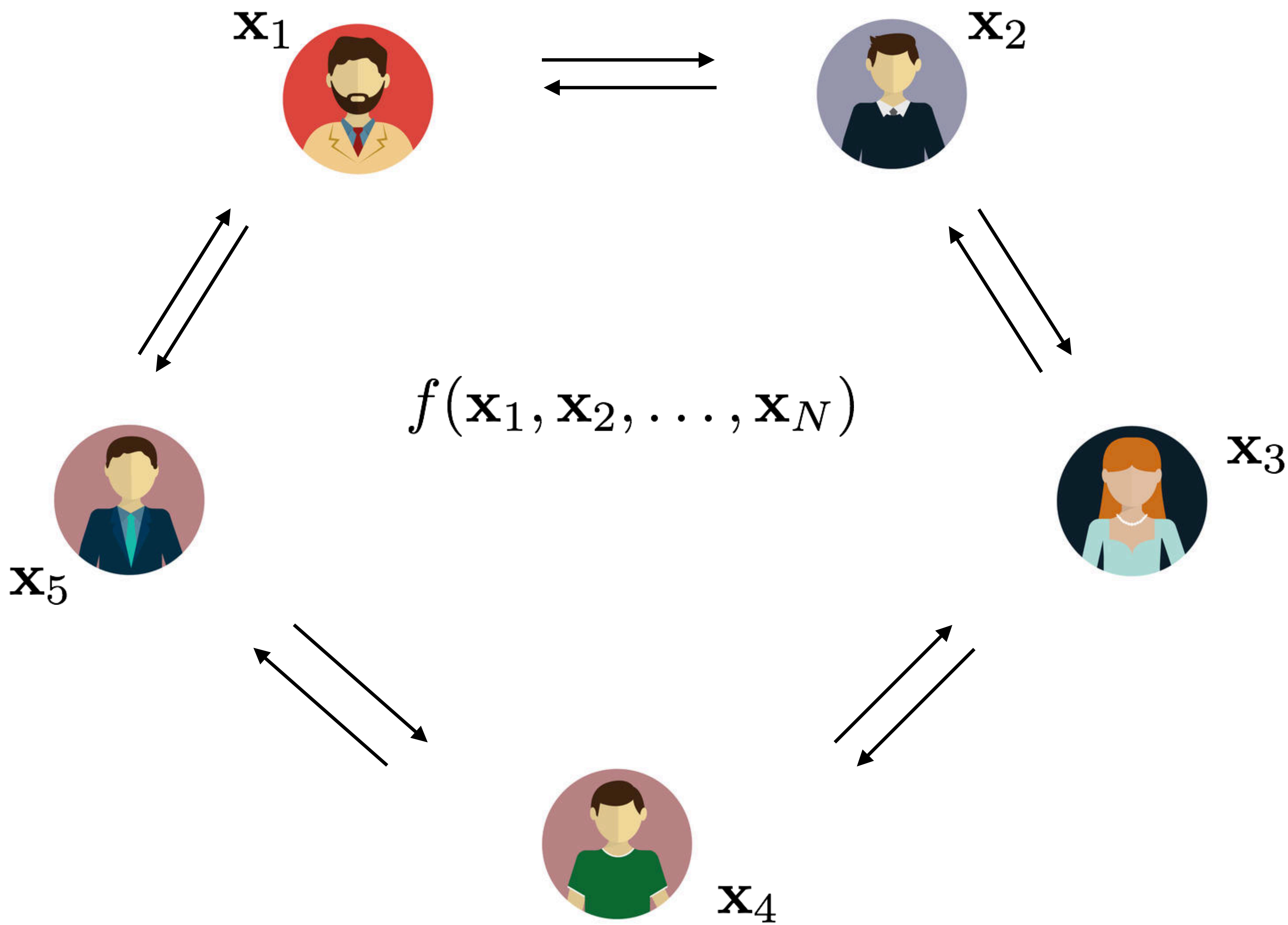
Blind Homomorphic Signatures



Must account for:

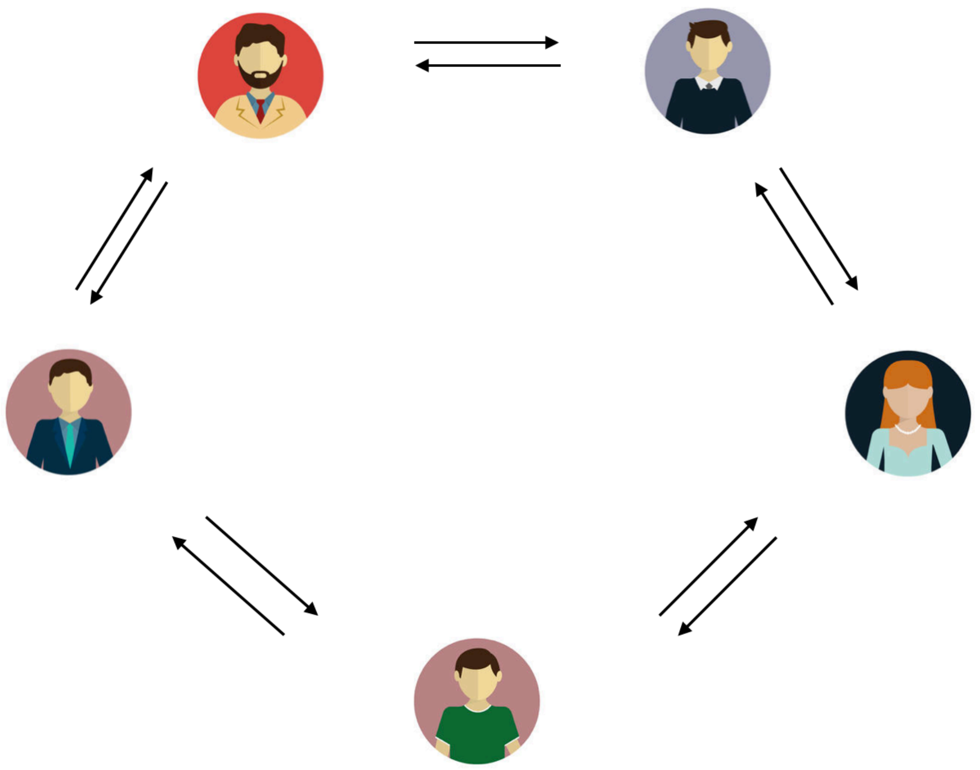
- Arbitrary abort attacks
- Maliciously generated signatures
- Guarantee context-hiding even when signer has signing key
- …

Security against malicious adversaries in MPC



Security against malicious adversaries in MPC

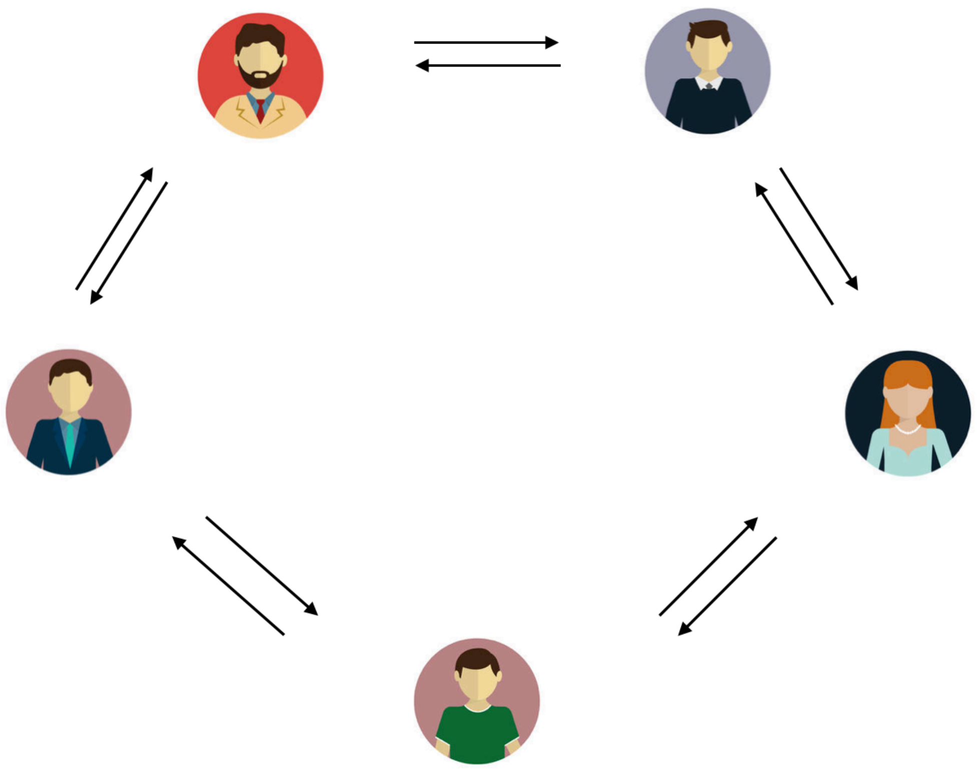
GMW Compiler:
Semi-Honest to Malicious security



Security against malicious adversaries in MPC

GMW Compiler:
Semi-Honest to Malicious security

At every step, each party proves that they are following protocol.

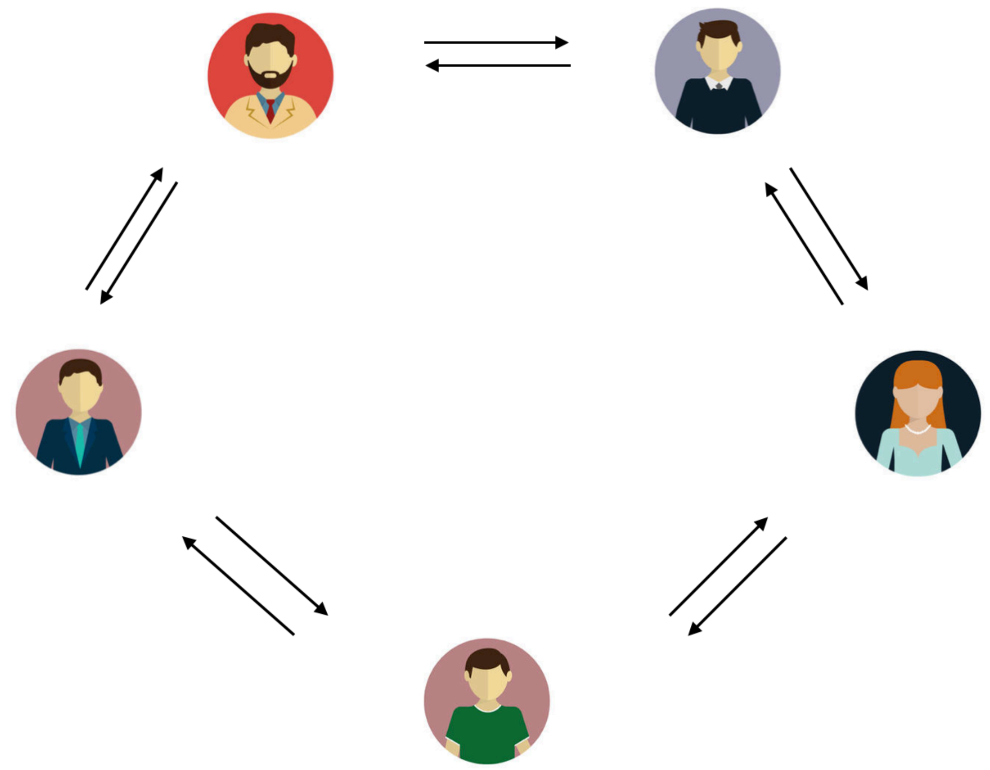


Security against malicious adversaries in MPC

GMW Compiler:
Semi-Honest to Malicious security

At every step, each party proves that they are following protocol.

Semi-honest + NIZK = Malicious
- Preserves round complexity



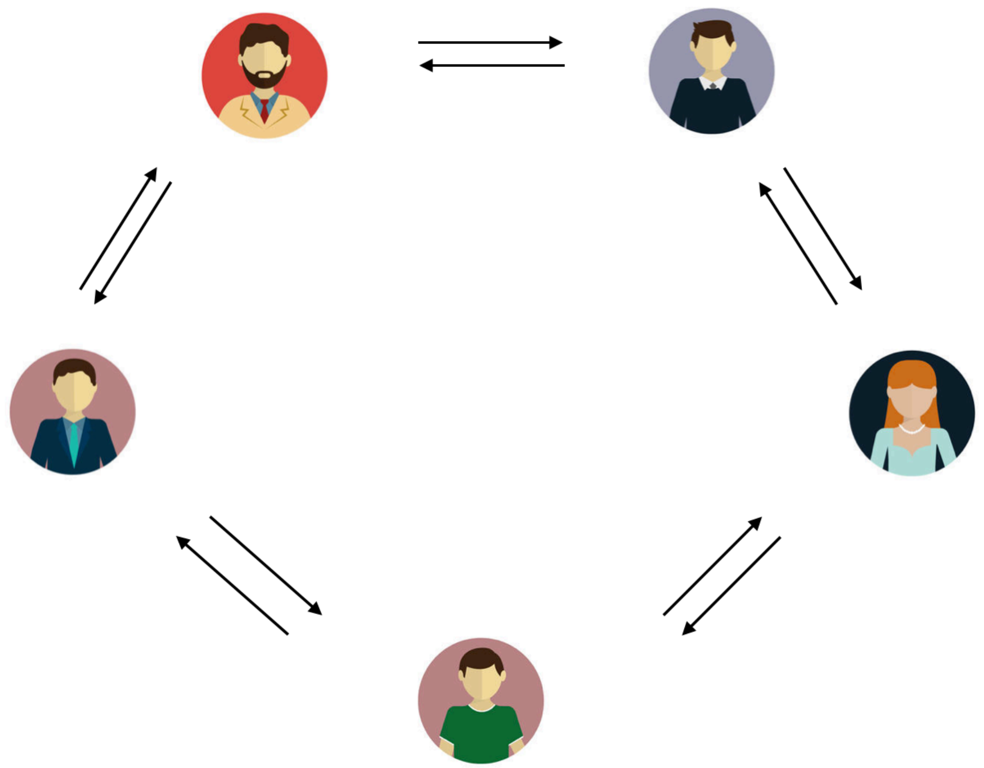
Security against malicious adversaries in MPC

GMW Compiler:
Semi-Honest to Malicious security

At every step, each party proves that they are following protocol.

Semi-honest + NIZK = Malicious
- Preserves round complexity

Semi-honest + NIZK w/preprocessing = Malicious
- Preprocessing step done once

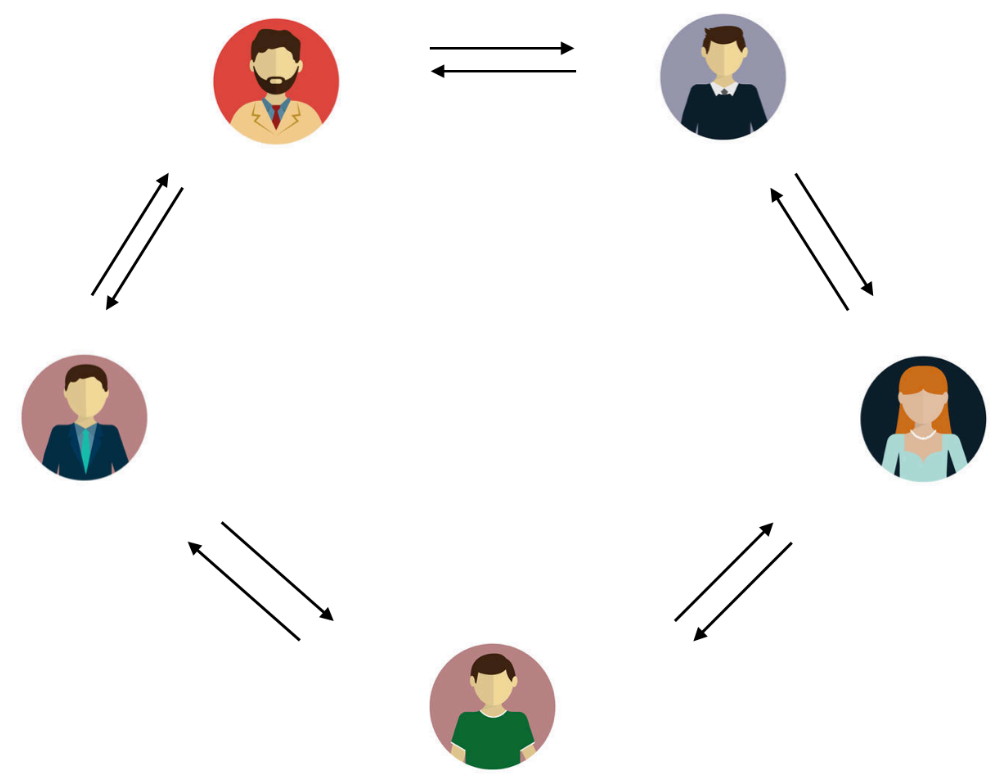


Security against malicious adversaries in MPC

GMW Compiler:
Semi-Honest to Malicious security

At every step, each party proves that they are following protocol.

Semi-honest + NIZK = Malicious
- Preserves round complexity



Semi-honest + NIZK w/preprocessing = Malicious
- Preprocessing step done once

Good: Rely on lattices + small communication size

To Summarize…

To Summarize…

1. LWE-based NIZK arguments in preprocessing (multi-theorem)

To Summarize…

1. LWE-based NIZK arguments in preprocessing (multi-theorem)

2. Use Blind Homomorphic Signatures to do preprocessing

To Summarize…

1. LWE-based NIZK arguments in preprocessing (multi-theorem)

2. Use Blind Homomorphic Signatures to do preprocessing

3. GMW compiler using NIZK with preprocessing

To Summarize…

1. LWE-based NIZK arguments in preprocessing (multi-theorem)

2. Use Blind Homomorphic Signatures to do preprocessing

3. GMW compiler using NIZK with preprocessing

Thanks!