Non-Interactive Zero-Knowledge from Vector Trapdoor Hash

Pedro Branco

Bocconi

Based on joint work with Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta and Akshayaram Srinivasan

Non-Interactive Zero-Knowledge from Vector Trapdoor Hash

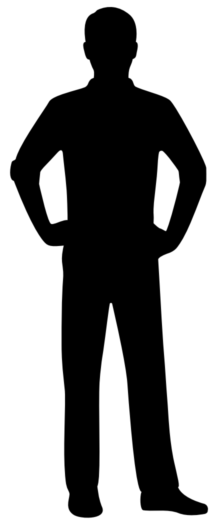
or
NIZK and Hidden-Bits Generator

Pedro Branco

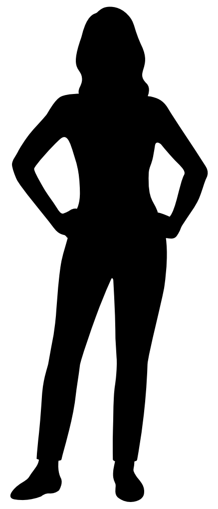
Bocconi

Based on joint work with Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta and Akshayaram Srinivasan

Zero-Knowledge Proofs [GMR85]

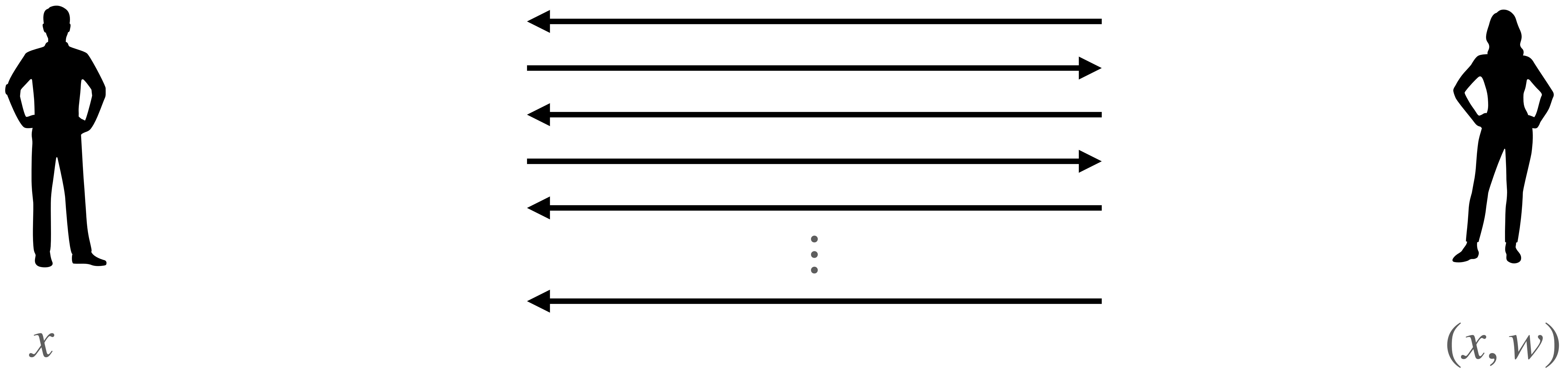


x

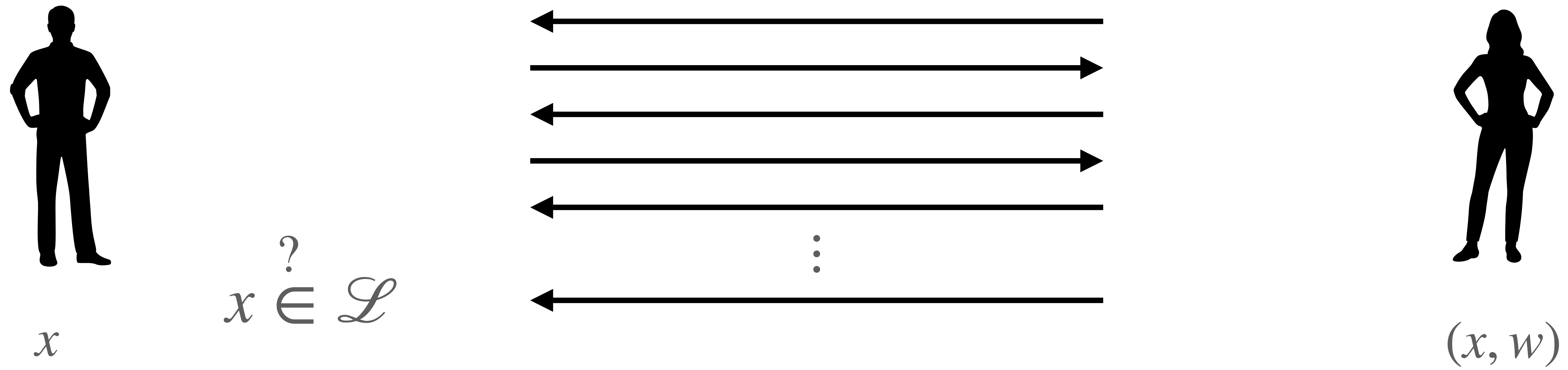


(x, w)

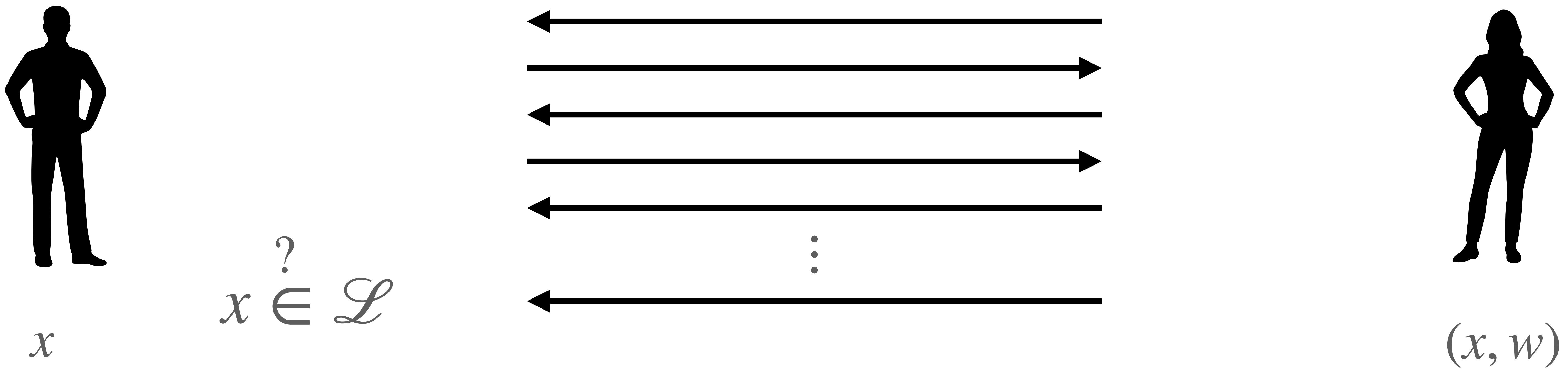
Zero-Knowledge Proofs [GMR85]



Zero-Knowledge Proofs [GMR85]



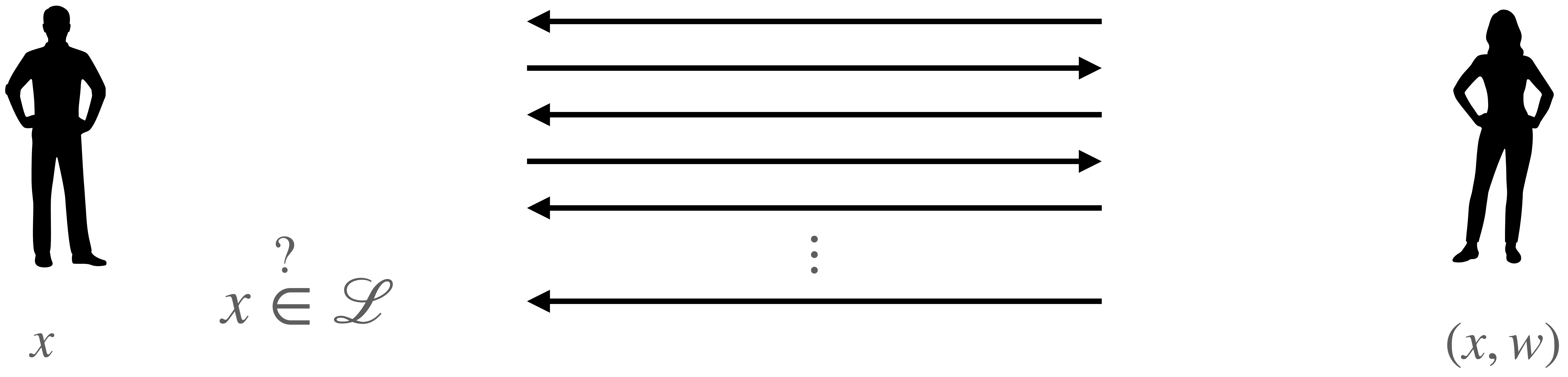
Zero-Knowledge Proofs [GMR85]



Soundness:

$$\Pr\left[x \notin \mathscr{L} \wedge \text{V accepts}\right] = \text{negl}(\lambda)$$

Zero-Knowledge Proofs [GMR85]



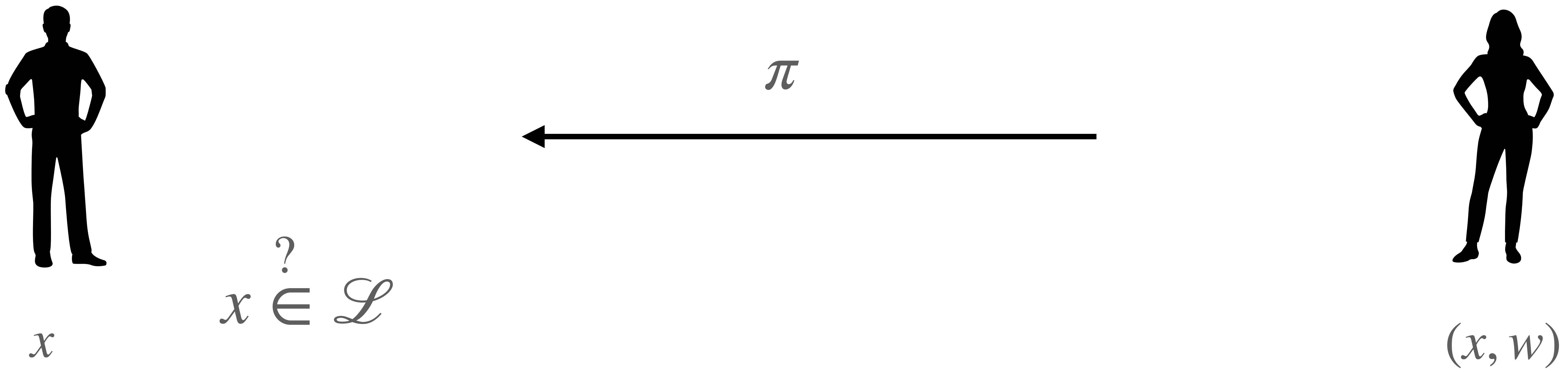
Soundness:

$$\Pr\left[x \notin \mathscr{L} \wedge \mathsf{V}\text{ accepts}\right] = \mathsf{negl}(\lambda)$$

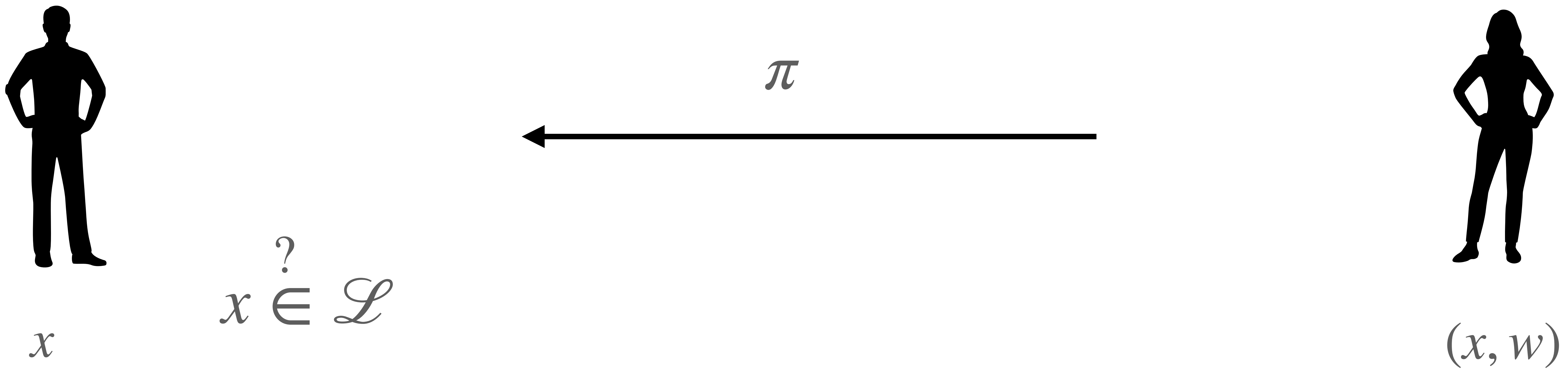
Zero-Knowledge:

$$\{\mathsf{T} \leftarrow \mathsf{Sim}(x)\} \approx \{\mathsf{T} \leftarrow (\mathsf{V}(x) \leftrightarrow \mathsf{P}(x, w))\}$$

Non-Interactive Zero-Knowledge Proofs [DMP88]



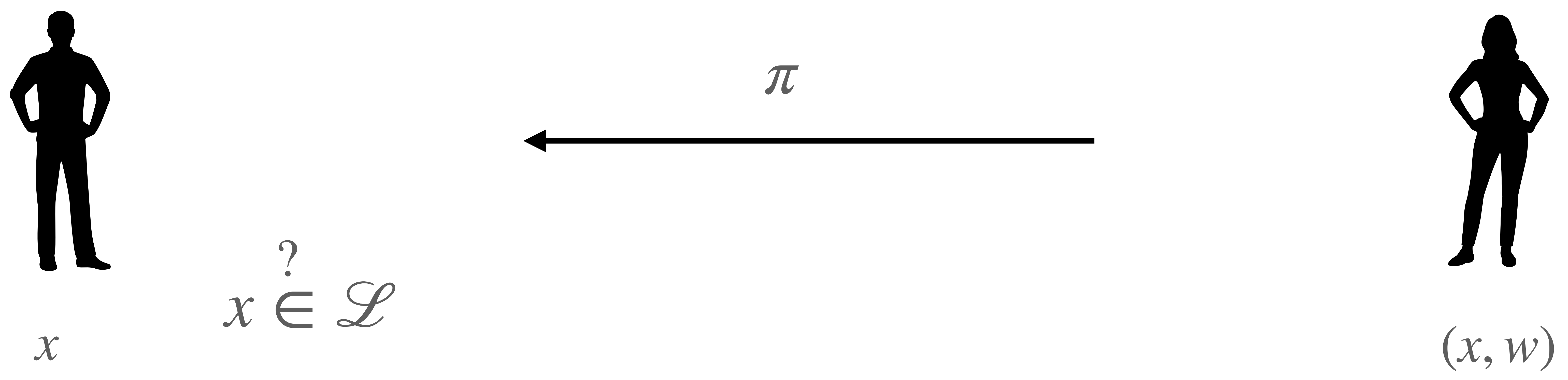
Non-Interactive Zero-Knowledge Proofs [DMP88]



Soundness:

$$\Pr\left[x \notin \mathscr{L} \land 1 \leftarrow \mathsf{V}(x, \pi)\right] = \mathsf{negl}(\lambda)$$

Non-Interactive Zero-Knowledge Proofs [DMP88]



Soundness:

$$\Pr\left[x \notin \mathscr{L} \wedge 1 \leftarrow \mathsf{V}(x, \pi)\right] = \mathsf{negl}(\lambda)$$

Zero-Knowledge:

$$\{\pi \leftarrow \mathsf{Sim}(x)\} \approx \{\pi \rightarrow \mathsf{P}(x, w)\}$$

Why NIZKs?

Theory:

- Minimal round complexity
- Minimal assumptions

Why NIZKs?

Theory:

- Minimal round complexity
- Minimal assumptions

Applications:

- CCA security
- Signatures
- Blockchains

⋮

Random Oracle Vs Standard model

Random Oracle

- Fiat-Shamir

Random Oracle Vs Standard model

Random Oracle

- Fiat-Shamir

Problem: RO don’t exist!

Random Oracle Vs Standard model

Random Oracle

- Fiat-Shamir

Problem: RO don't exist!

Standard Model

- Impossible!
- Need at least 4 rounds

Random Oracle Vs Standard model

Random Oracle

- Fiat-Shamir

Problem: RO don't exist!

Standard Model

- Impossible!
- Need at least 4 rounds

Assume CRS

Random Oracle Vs Standard model

Random Oracle

- Fiat-Shamir

Problem: RO don’t exist!

Standard Model

- Impossible!
- Need at least 4 rounds

Assume CRS

This talk: NIZK = NIZK for all NP in the CRS model

NIZKs Constructions

[GOS'06]

- Pairings

NIZKs Constructions

[GOS'06]

- Pairings

Correlation Intractability Hash

- iO [CCRR18,HL18]
- FHE/LWE [CCH+19,PS19]
- DDH + LPN [BKM20]
- Sub-exp DDH [JJ21]
- MQ + LPN [DJJ24]

NIZKs Constructions

[GOS'06]

- Pairings

Correlation Intractability Hash

- iO [CCRR18,HL18]
- FHE/LWE [CCH+19,PS19]
- DDH + LPN [BKM20]
- Sub-exp DDH [JJ21]
- MQ + LPN [DJJ24]

Hidden-Bits Generator

- Trapdoor permutations [FLS90]
- LWE (super-poly mod to noise) [Wat24]

NIZKs Constructions

[GOS'06]

- Pairings

Correlation Intractability Hash

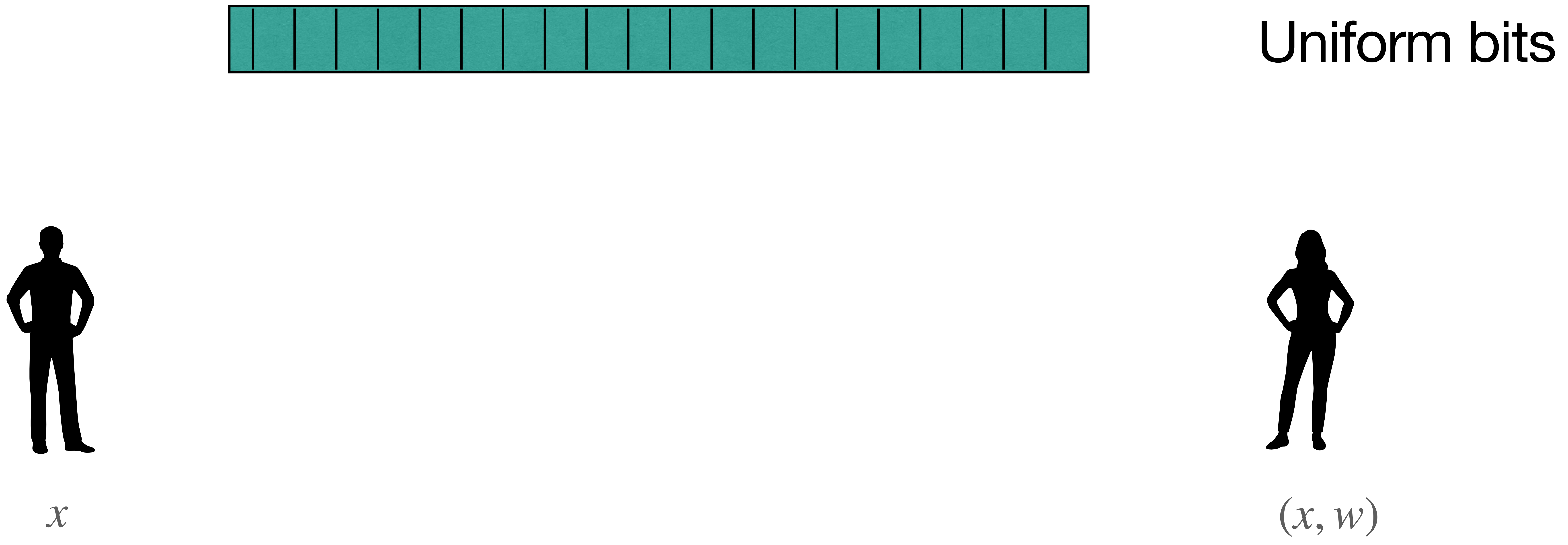
- iO [CCRR18,HL18]
- FHE/LWE [CCH+19,PS19]
- DDH + LPN [BKM20]
- Sub-exp DDH [JJ21]
- MQ + LPN [DJJ24]

Hidden-Bits Generator

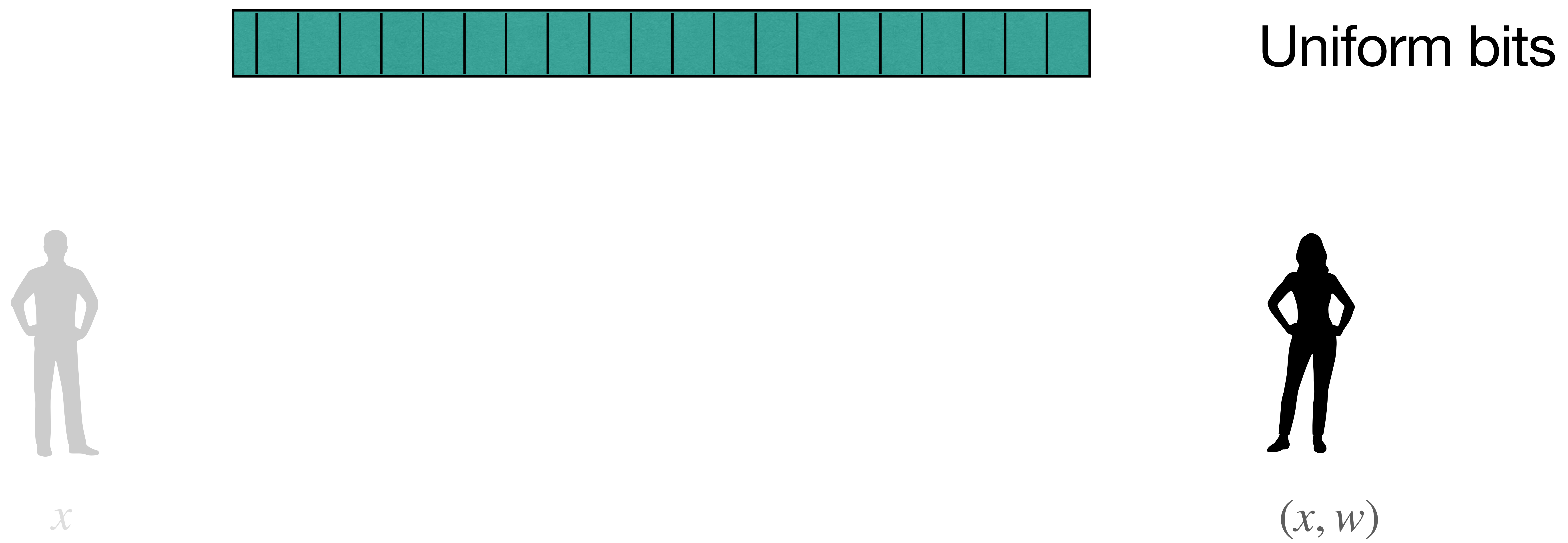
- Trapdoor permutations [FLS90]
- LWE (super-poly mod to noise) [Wat24]

?

Hidden-Bits Model [FLS90]

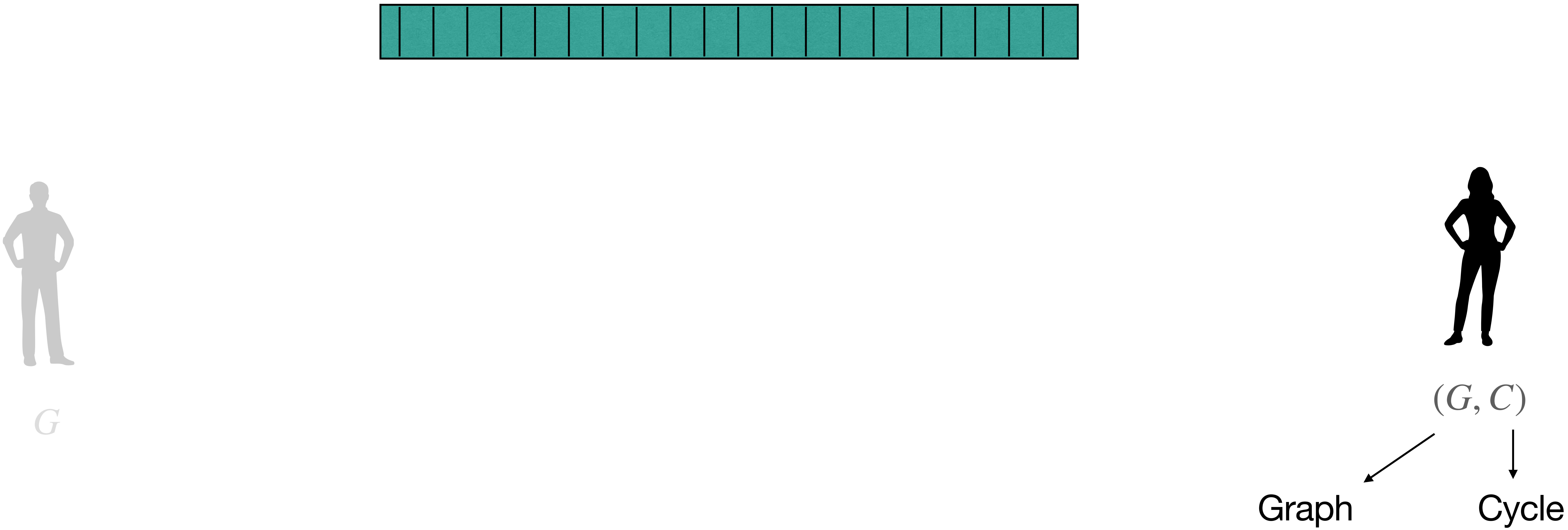


Hidden-Bits Model [FLS90]

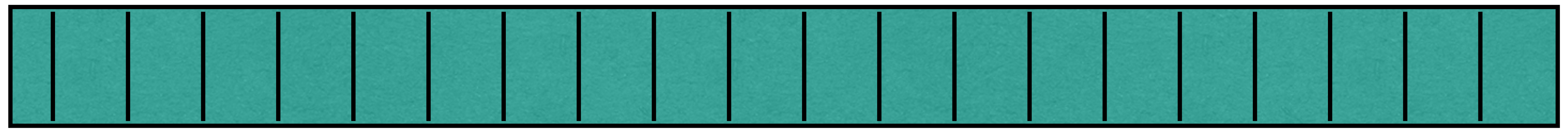


Hidden-Bits Model [FLS90]

NIZK in the Hidden-Bits Model [FLS90]

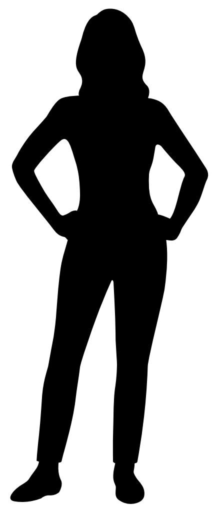


NIZK in the Hidden-Bits Model [FLS90]



Graph H with an Hamiltonian cycle C'

G

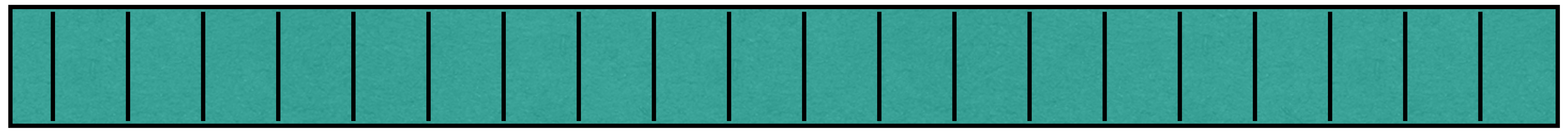


(G, C)

Graph

Cycle

NIZK in the Hidden-Bits Model [FLS90]

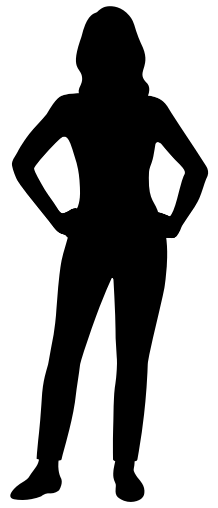


Graph H with an Hamiltonian cycle C'

G

δ mapping C to C'

If e non-edge of G,
Reveal $\delta(e)$ non-edge of H

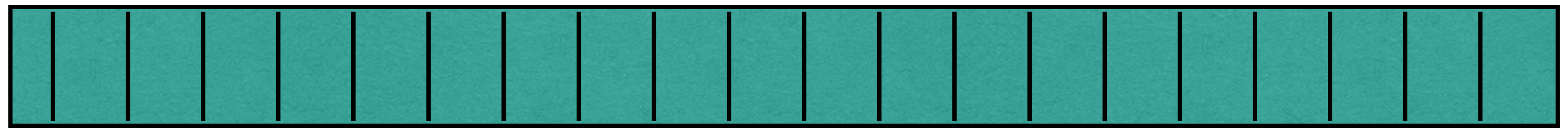


(G, C)

Graph

Cycle

NIZK in the Hidden-Bits Model [FLS90]



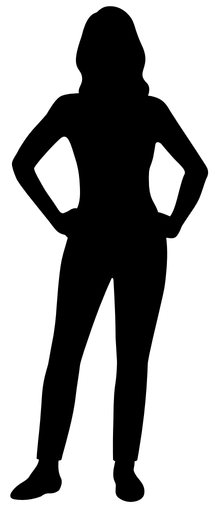
Graph H with an Hamiltonian cycle C'

G

$\pi = \delta$

δ mapping C to C'

If e non-edge of G,
Reveal $\delta(e)$ non-edge of H

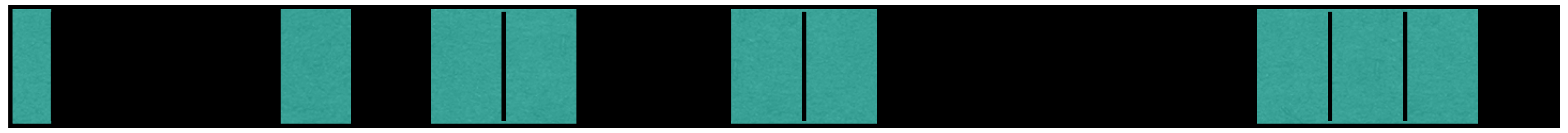


(G, C)

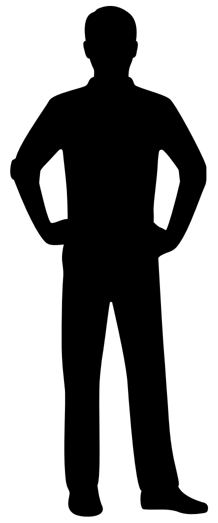
Graph

Cycle

NIZK in the Hidden-Bits Model [FLS90]



Graph H with an Hamiltonian cycle C'



G

For all non-edges e,
Check if $\delta(e)$ is non-edge

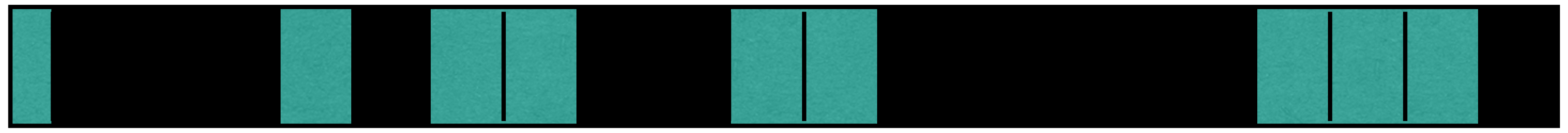
$\pi = \delta$

δ mapping C to C'

If e non-edge of G,
Reveal $\delta(e)$ non-edge of H

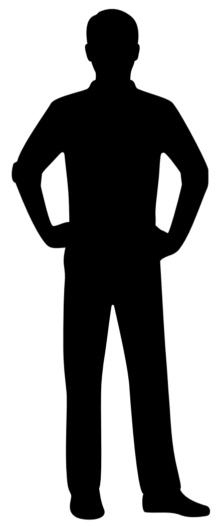
(G, C)

NIZK in the Hidden-Bits Model [FLS90]



Graph H with an
Hamiltonian cycle C'

$\delta(e)$



G

For all non-edges e,
Check if $\delta(e)$ is non-edge

$\pi = \delta$

δ mapping C to C'

If e non-edge of G,
Reveal $\delta(e)$ non-edge of H

(G, C)

Soundness

- H has a cycle C'

Soundness

- H has a cycle C'
- To prove:

If $\delta(e)$ is an edge in H then e is an edge in G

Soundness

- H has a cycle C'
- To prove:

If $\delta(e)$ is an edge in H then e is an edge in G

$$\Leftrightarrow$$

If e is an non-edge in G then $\delta(e)$ is an non-edge in H

Zero-Knowledge

Description of Sim:

- Choose a random permutation δ.

Zero-Knowledge

Description of Sim:

- Choose a random permutation δ.
- For all non-edges e of G, reveal a non-edge $\delta(e)$ of H

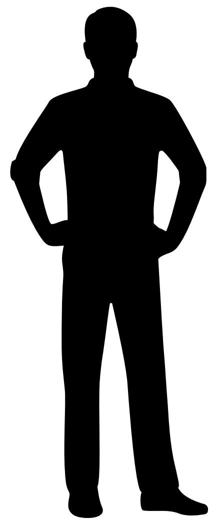
Zero-Knowledge

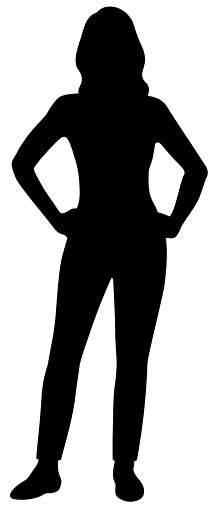
Description of Sim:

- Choose a random permutation δ.
- For all non-edges e of G, reveal a non-edge $\delta(e)$ of H
- (Sim has full control over the hidden-bits)

From Hidden-Bits Model to CRS Model:
Hidden-Bits Generator [QRW19,KMY20]

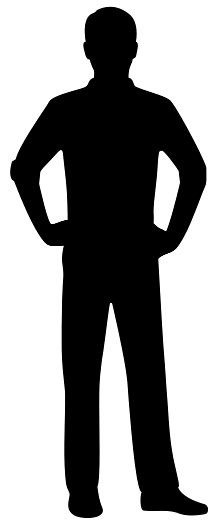
$\mathsf{crs} \leftarrow \mathsf{Setup}$



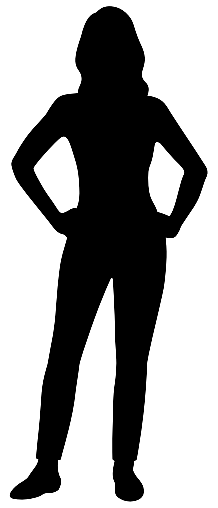


From Hidden-Bits Model to CRS Model: Hidden-Bits Generator [QRW19,KMY20]

$\mathsf{crs} \leftarrow \mathsf{Setup}$



$\mathsf{com}, \{\alpha_i, \pi_i\}_{i \in S}$ ←

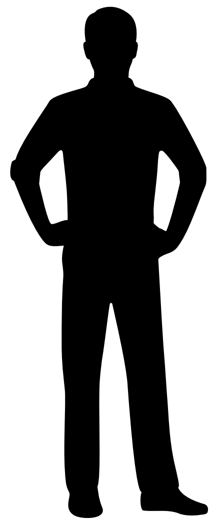


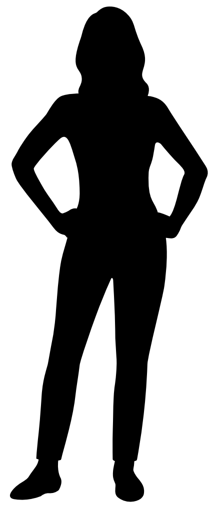
$(\mathsf{com}, (\alpha_1, \ldots, \alpha_k), (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}$

From Hidden-Bits Model to CRS Model:
Hidden-Bits Generator [QRW19,KMY20]

$\mathsf{crs} \leftarrow \mathsf{Setup}$

$\mathsf{com}, \{\alpha_i, \pi_i\}_{i \in S}$





$\mathsf{Ver}(\mathsf{com}, \tilde{\alpha}, i, \pi_i)$

$(\mathsf{com}, (\alpha_1, \ldots, \alpha_k), (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}$

Hidden-Bits Generator [QRW19,KMY20]

Completeness:

Honest proofs are accepted

Hidden-Bits Generator [QRW19,KMY20]

Completeness:

Honest proofs are accepted

Output sparsity:

Given crs, the number of strings that a prover can open is small

Hidden-Bits Generator [QRW19,KMY20]

Completeness:

Honest proofs are accepted

Output sparsity:

Given crs, the number of strings that a prover can open is small

Statistical Binding:

com statistically determines a string

Hidden-Bits Generator [QRW19,KMY20]

Completeness:

Honest proofs are accepted

Output sparsity:

Given crs, the number of strings that a prover can open is small

Statistical Binding:

com statistically determines a string

Hiding:

Given com, $\{\alpha_i, \pi_i\}_{i \neq i^*}$, α_{i^*} remains hidden

Hidden-Bits Generator to NIZKs

Theorem [QRW19,KMY20]:

We can build a NIZK in the CRS model, given a HBG in the CRS model and a NIZK in the hidden-bits model

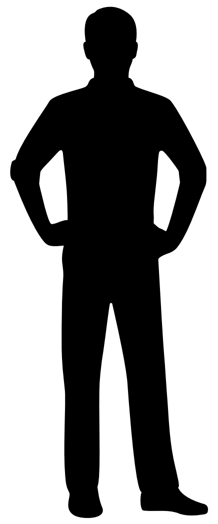
Hidden-Bits Generator to NIZKs

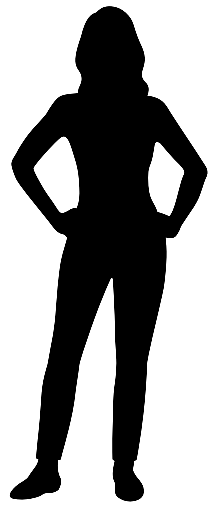
Theorem [QRW19,KMY20]:
We can build a NIZK in the CRS model, given a HBG in the CRS model and a NIZK in the hidden-bits model

Problem:

- Build HBG.
- Previous works: TDP and LWE (with super-poly mod-to-noise).

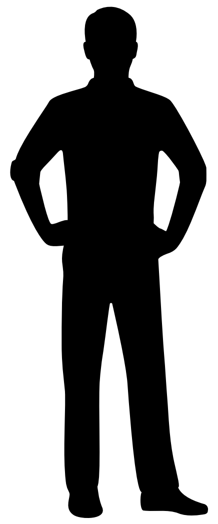
New Primitive: Vector Trapdoor Hash

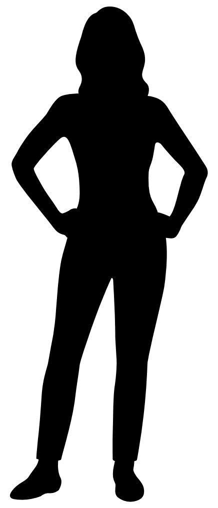




New Primitive: Vector Trapdoor Hash

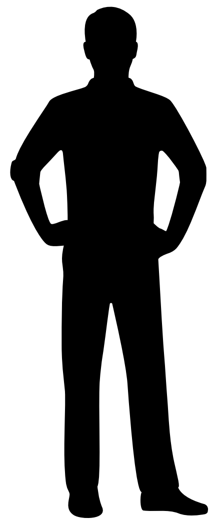
$$\left(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}\right) \leftarrow \mathsf{Setup}$$





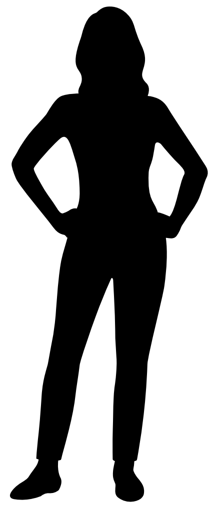
New Primitive: Vector Trapdoor Hash

$$\left(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}\right) \leftarrow \mathsf{Setup}$$



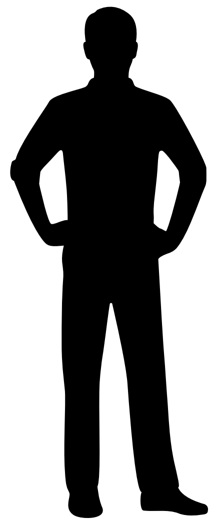
$\left(\mathbf{h}, \{\pi_i\}_{i\in[k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$



New Primitive: Vector Trapdoor Hash

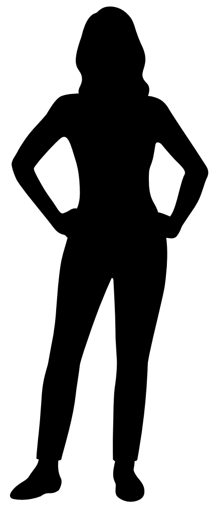
$$\left(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}\right) \leftarrow \mathsf{Setup}$$



$d_i \leftarrow \mathsf{Dec}(\mathsf{td}_i, \mathbf{h})$

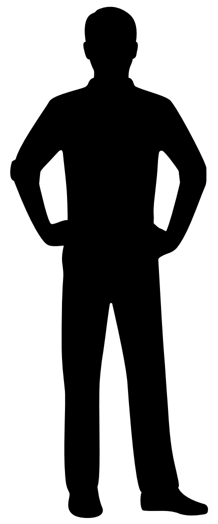
$\left(\mathbf{h}, \{\pi_i\}_{i\in[k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$



New Primitive: Vector Trapdoor Hash

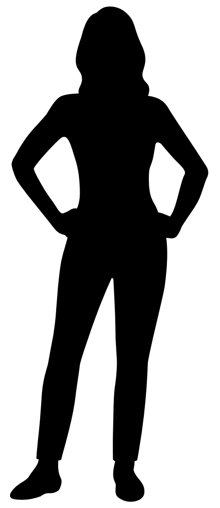
$$\left(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}\right) \leftarrow \mathsf{Setup}$$



$d_i \leftarrow \mathsf{Dec}(\mathsf{td}_i, \mathbf{h})$

$\left(\mathbf{h}, \{\pi_i\}_{i\in[k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$

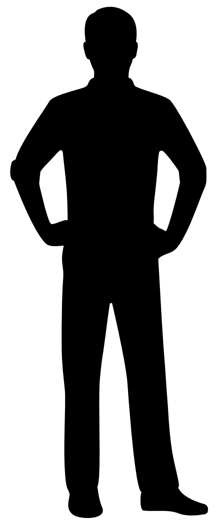


Local opening:

π_i is a local opening for $\mathbf{x}_i$

New Primitive: Vector Trapdoor Hash

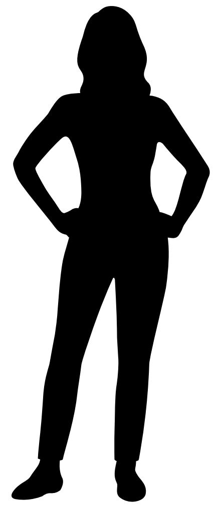
$$\left(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}\right) \leftarrow \mathsf{Setup}$$



$d_i \leftarrow \mathsf{Dec}(\mathsf{td}_i, \mathbf{h})$

$\left(\mathbf{h}, \{\pi_i\}_{i\in[k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$



Local opening:

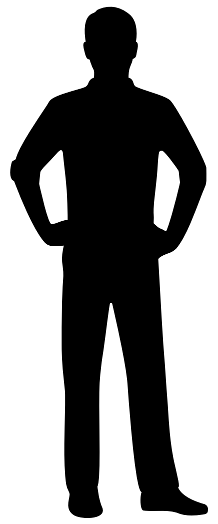
π_i is a local opening for $\mathbf{x}_i$

Statistical binding:

$e_i = d_i$ for almost all $i \in [k]$

New Primitive: Vector Trapdoor Hash

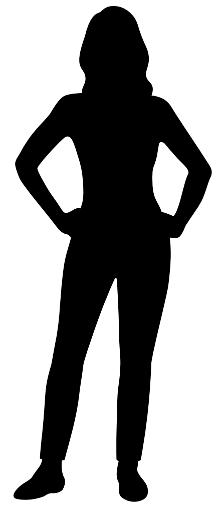
$$\left(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}\right) \leftarrow \mathsf{Setup}$$



$d_i \leftarrow \mathsf{Dec}(\mathsf{td}_i, \mathbf{h})$

$\left(\mathbf{h}, \{\pi_i\}_{i\in[k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$



Local opening:

π_i is a local opening for $\mathbf{x}_i$

Statistical binding:

$e_i = d_i$ for almost all $i \in [k]$

Hiding:

e_{i^*} is uniform, given $\{e_i, \pi_i\}_{i \neq i^*}$

VTDH to Hidden-Bits Generator

Theorem:

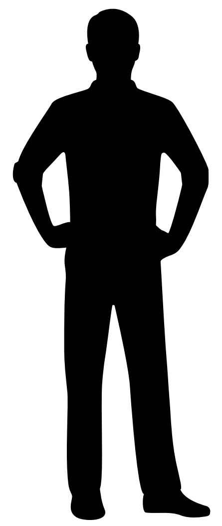
VTDH implies HBG

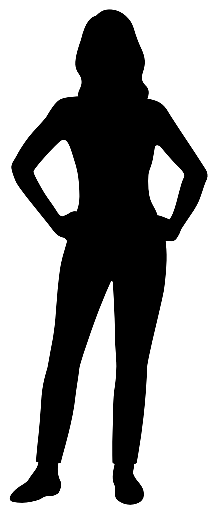
VTDH to Hidden-Bits Generator

Theorem:

VTDH implies HBG

$\mathsf{hk}, \{\mathsf{ek}_i\}_{i \in [k]}$

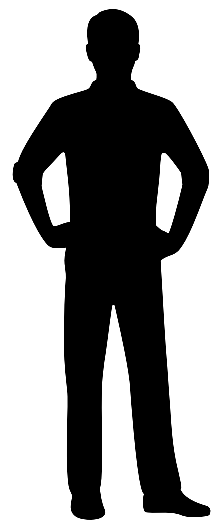




VTDH to Hidden-Bits Generator

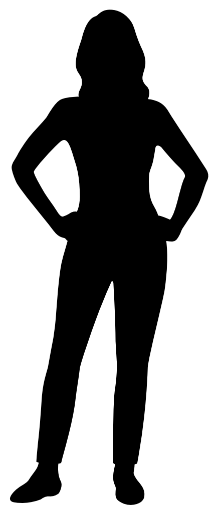
Theorem:
VTDH implies HBG

$\mathsf{hk}, \{\mathsf{ek}_i\}_{i \in [k]}$



$\left(\mathbf{h}, \{\pi_i\}_{i \in [k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

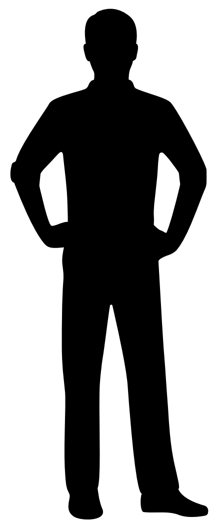
$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$



VTDH to Hidden-Bits Generator

Theorem:
VTDH implies HBG

$\mathsf{hk}, \{\mathsf{ek}_i\}_{i \in [k]}$



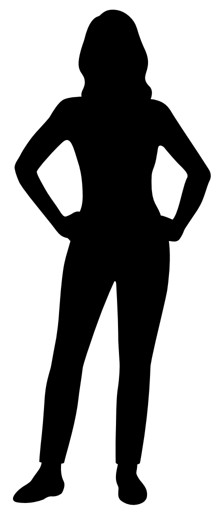
$\left(\mathbf{h}, \{\pi_i\}_{i \in [k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$

$\mathsf{com} = \mathbf{h}$

$\alpha_i = e_i$

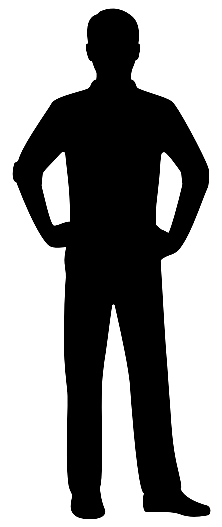
$\pi'_i = (\pi_i, \mathbf{x}_i)$



VTDH to Hidden-Bits Generator

Theorem:
VTDH implies HBG

$\mathsf{hk}, \{\mathsf{ek}_i\}_{i \in [k]}$



$\mathsf{com}, \{\alpha_i, \pi_i'\}_{i \in S}$ ←

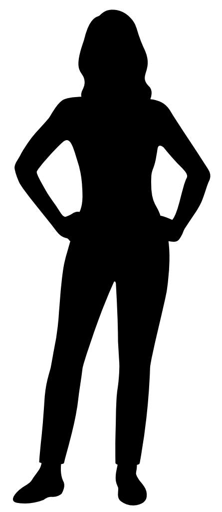
$\left(\mathbf{h}, \{\pi_i\}_{i \in [k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$

$\mathsf{com} = \mathbf{h}$

$\alpha_i = e_i$

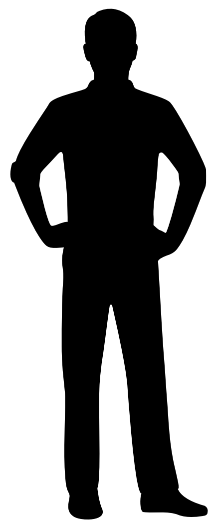
$\pi_i' = (\pi_i, \mathbf{x}_i)$



VTDH to Hidden-Bits Generator

Theorem:
VTDH implies HBG

$\mathsf{hk}, \{\mathsf{ek}_i\}_{i \in [k]}$



$\mathsf{com}, \{\alpha_i, \pi'_i\}_{i \in S}$ ←

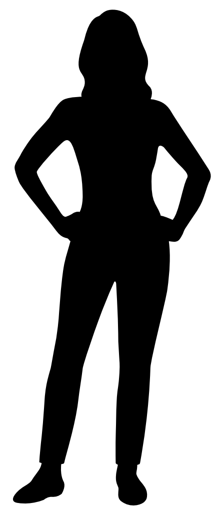
$\left(\mathbf{h}, \{\pi_i\}_{i \in [k]}\right) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$

$e_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \pi_i)$

$\mathsf{com} = \mathbf{h}$

$\alpha_i = e_i$

$\pi'_i = (\pi_i, \mathbf{x}_i)$



$\mathsf{LocVer}(\mathsf{com}, \pi_i, \mathbf{x}_i) = 1$

$\mathsf{Enc}(\mathsf{ek}_i, \pi_i) = e_i$

Our Results

Theorem:

VTDH from: i) LWE with polynomial mod-to-noise ratio

ii) DDH + LPN

Our Results

Theorem:

VTDH from: i) LWE with polynomial mod-to-noise ratio

ii) DDH + LPN

Corollaries:

- (Dual-mode) NIZK from LWE.
- NIZK from DDH + LPN with statistical soundness.

Our Results

Theorem:

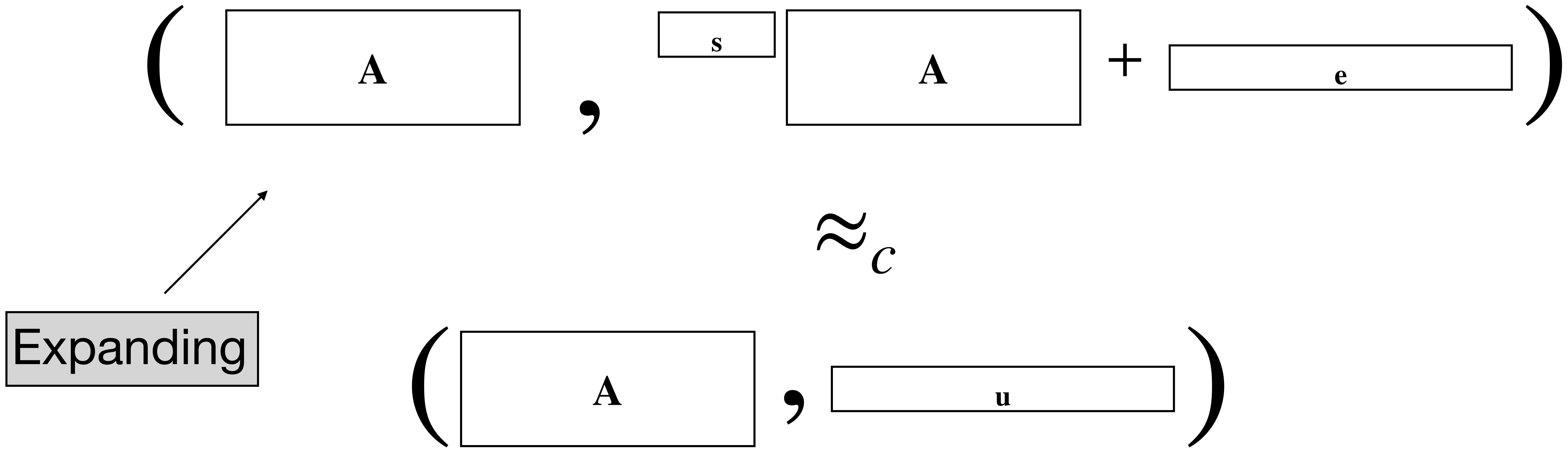
VTDH from: i) LWE with polynomial mod-to-noise ratio

ii) DDH + LPN

Corollaries:

- (Dual-mode) NIZK from LWE.
- NIZK from DDH + LPN with statistical soundness.

Learning with Errors



$\mathbf{A} \leftarrow \{0,1\}^{n\times m}$, $\mathbf{s} \leftarrow \{0,1\}^{n}$, $\mathbf{u} \leftarrow \{0,1\}^{m}$ and $\mathbf{e} \leftarrow \mathsf{DG}_{\sigma}^{m}$

VTDH from LWE: Hash, Encoding and Decoding

$$\mathsf{hk} = \mathbf{A}_1, \ldots, \mathbf{A}_k, \underbrace{\mathbf{W}_1, \ldots, \mathbf{W}_k}_{\text{binary}}$$

VTDH from LWE: Hash, Encoding and Decoding

$$\mathsf{hk} = \mathbf{A}_1, \ldots, \mathbf{A}_k, \underbrace{\mathbf{W}_1, \ldots, \mathbf{W}_k}_{\text{binary}} \text{ such that } \mathbf{A}_i \mathbf{W}_i = \mathbf{U}_i$$

VTDH from LWE: Hash, Encoding and Decoding

$\mathsf{hk} = \mathbf{A}_1, \ldots, \mathbf{A}_k, \underbrace{\mathbf{W}_1, \ldots, \mathbf{W}_k}_{\text{binary}}$ such that $\mathbf{A}_i \mathbf{W}_i = \mathbf{U}_i$

$\mathsf{ek}_i = (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathbf{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)$

VTDH from LWE: Commitment and Local Proofs

Pick binary $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$

VTDH from LWE: Commitment and Local Proofs

Pick binary $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$, compute $\mathbf{h} = \sum \mathbf{U}_i \mathbf{x}_i$

VTDH from LWE: Commitment and Local Proofs

Pick binary $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$, compute $\mathbf{h} = \sum \mathbf{U}_i \mathbf{x}_i$

$$\pi_i = (\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

VTDH from LWE: Commitment and Local Proofs

Pick binary $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$, compute $\mathbf{h} = \sum \mathbf{U}_i \mathbf{x}_i$

$$\pi_i = (\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

Local Verification:

$(\mathbf{A}_1, \ldots, \mathbf{U}_i, \ldots, , \mathbf{A}_k)\pi_i$

VTDH from LWE: Commitment and Local Proofs

Pick binary $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$, compute $\mathbf{h} = \sum \mathbf{U}_i \mathbf{x}_i$

$$\pi_i = (\mathbf{W}_1\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k\mathbf{x}_k)$$

Local Verification:

$$(\mathbf{A}_1, \ldots, \mathbf{U}_i, \ldots, , \mathbf{A}_k)\pi_i \quad = (\mathbf{A}_1, \ldots, \mathbf{U}_i, \ldots, , \mathbf{A}_k)(\mathbf{W}_1\mathbf{x}_1, \ldots, x_i, \ldots, \mathbf{W}_k\mathbf{x}_k)$$

VTDH from LWE: Commitment and Local Proofs

Pick binary $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$, compute $\mathbf{h} = \sum \mathrm{U}_i \mathbf{x}_i$

$$\pi_i = (\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

Local Verification:

$$(\mathbf{A}_1, \ldots, \mathbf{U}_i, \ldots, , \mathbf{A}_k)\pi_i \;= (\mathbf{A}_1, \ldots, \mathbf{U}_i, \ldots, , \mathbf{A}_k)(\mathbf{W}_1 \mathbf{x}_1, \ldots, x_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

$$= \sum \mathrm{U}_i \mathbf{x}_i = h$$

VTDH from LWE: Encoding and Local Proofs

Encoding: $e_i = \mathsf{ek}_i \pi_i$

VTDH from LWE: Encoding and Local Proofs

Encoding: $e_i = \mathsf{ek}_i \pi_i$

$$= (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathrm{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)(\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

VTDH from LWE: Encoding and Local Proofs

Encoding: $e_i = \mathsf{ek}_i \pi_i$

$$= (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathrm{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)(\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

$$= \mathbf{s}_i \left(\sum \mathrm{U}_i \mathbf{x}_i \right) + \tilde{e}$$

VTDH from LWE: Encoding and Local Proofs

Encoding: $e_i = \mathsf{ek}_i \pi_i$

$$= (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathrm{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)(\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

$$= \mathbf{s}_i \left(\sum \mathrm{U}_i \mathbf{x}_i \right) + \tilde{e}$$

Decoding: $d_i = \mathbf{s}_i^T \mathbf{h} = \mathbf{s}_i \left(\sum \mathbf{U}_i \mathbf{x}_i \right)$

VTDH from LWE: Encoding and Local Proofs

Encoding: $e_i = \mathsf{ek}_i \pi_i$

$$= (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathrm{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)(\mathbf{W}_1 \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{W}_k \mathbf{x}_k)$$

$$= \mathbf{s}_i \left(\sum \mathrm{U}_i \mathbf{x}_i \right) + \tilde{e}$$

Decoding: $d_i = \mathbf{s}_i^T \mathbf{h} = \mathbf{s}_i \left(\sum \mathbf{U}_i \mathbf{x}_i \right)$

$\mathsf{Round}(e_i) = \mathsf{Round}(d_i)$

$\Downarrow$

Statistical Binding

VTDH from LWE: Hiding for $i = 1$

To prove: $e_1 = \mathsf{ek}_1 \pi_1 \approx v \leftarrow \mathsf{Unif}$

VTDH from LWE: Hiding for $i = 1$

To prove: $e_1 = \mathsf{ek}_1 \pi_1 \approx v \leftarrow \mathsf{Unif}$

1st Step: $\mathsf{ek}_i = (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathbf{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)$

VTDH from LWE: Hiding for $i = 1$

To prove: $e_1 = \mathsf{ek}_1\pi_1 \approx v \leftarrow \mathsf{Unif}$

1st Step:

$$\mathsf{ek}_i = (\mathbf{s}_i^T\mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T\mathbf{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T\mathbf{A}_k + \mathbf{e}_k)$$

↓ LWE

$$\mathsf{ek}_i = (\mathbf{u}_1, \ldots, \mathbf{u}_i, \ldots, \mathbf{u}_k)$$

VTDH from LWE: Hiding for $i = 1$

To prove: $e_1 = \text{ek}_1 \pi_1 \approx v \leftarrow \text{Unif}$

1st Step:

$$\text{ek}_i = (\mathbf{s}_i^T \mathbf{A}_1 + \mathbf{e}_1, \ldots, \mathbf{s}_i^T \mathbf{U}_i + \mathbf{e}_i, \ldots, \mathbf{s}_i^T \mathbf{A}_k + \mathbf{e}_k)$$

↓ LWE

$$\text{ek}_i = (\mathbf{u}_1, \ldots, \mathbf{u}_i, \ldots, \mathbf{u}_k)$$

2nd Step:

$$(\text{ek}_1 \pi_1, \mathbf{W}_1 \mathbf{x}_1) \approx_s (v, \mathbf{W}_1 \mathbf{x}_1)$$

→ LHL

Recap

- LWE Result: Dual-mode NIZK from LWE .
- DDH + LPN Result: NIZK from (DDH + LPN) with statistical soundness.

Thanks!

Non-Interactive Zero-Knowledge from Vector Trapdoor Hash

Pedro Branco

Bocconi

Based on joint work with Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta and Akshayaram Srinivasan