Adaptively Sound Zero-Knowledge SNARKs for UP

Surya Mathialagan
MIT

Spencer Peters
Cornell University

Vinod Vaikuntanathan
MIT

TL;DR

TL;DR

In this work, we

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

   - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

    - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!

3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u> for UP**.

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

   - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!

3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u>** for **UP**.

   - Corollary: **Adaptively sound** dv-SNAR**<u>K</u>** for UP from **falsifiable assumptions.**

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

    - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!

3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u>** for **UP**.

    - Corollary: **Adaptively sound** dv-SNAR**<u>K</u>** for UP from **falsifiable assumptions.**

All of the above constructions/transformations also **satisfy/preserve zero-knowledge!**

<u>S</u>uccinct <u>N</u>on-interactive <u>ARG</u>ument

Fix NP language L

$\mathscr{P}$

$\mathscr{V}$

Instance x, witness w

Succinct Non-interactive ARGument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Instance x, witness w

<u>S</u>uccinct <u>N</u>on-interactive <u>ARG</u>ument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Instance x, witness w

$\pi = \mathscr{P}(\text{crs}, x, w)$

<u>S</u>uccinct **<u>N</u>**on-interactive **<u>ARG</u>**ument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Instance x, witness w

$\pi = \mathscr{P}(\text{crs}, x, w)$

x, π

Succinct Non-interactive ARGument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Succinct:
$|\pi| \ll |w|$

Instance x, witness w

$\pi = \mathscr{P}(\mathsf{crs}, x, w)$

$\xrightarrow{x,\ \pi}$

Succinct **N**on-interactive **ARG**ument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Succinct:
$|\pi| \ll |w|$

Instance x, witness w

$\pi = \mathscr{P}(\text{crs}, x, w)$

$$\xrightarrow{x, \pi}$$

$\mathscr{V}(\text{crs}, x, \pi)$

Succinct Non-interactive ARGument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Succinct:
$|\pi| \ll |w|$

Instance x, witness w

$\pi = \mathscr{P}(\text{crs}, x, w)$

$$\xrightarrow{x,\ \pi}$$

$\mathscr{V}(\text{crs}, x, \pi)$

- **Completeness:** If $R(x, w) = 1$, then $\mathscr{V}(\text{crs}, \mathscr{P}(\text{crs}, x, w)) = 1$.

Succinct Non-interactive ARGument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}$ $\mathscr{V}$

Succinct: $|\pi| \ll |w|$

Instance x, witness w

$\pi = \mathscr{P}(\text{crs}, x, w)$

$\xrightarrow{x, \pi}$

$\mathscr{V}(\text{crs}, x, \pi)$

- **Completeness:** If $R(x, w) = 1$, then $\mathscr{V}(\text{crs}, \mathscr{P}(\text{crs}, x, w)) = 1$.
- **Soundness:** For all ppt $\mathscr{P}^*$, hard to come up with cheating proof for $x^* \notin L$:

Succinct Non-interactive ARGument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}*$

$\mathscr{V}$

Succinct:
$|\pi| \ll |w|$

Instance x, witness w

$\pi = \mathscr{P}(\text{crs}, x, w)$

x, π

$\mathscr{V}(\text{crs}, x, \pi)$

- **Completeness:** If $R(x, w) = 1$, then $\mathscr{V}(\text{crs}, \mathscr{P}(\text{crs}, x, w)) = 1$.
- **Soundness:** For all ppt $\mathscr{P}*$, hard to come up with cheating proof for $x^* \notin L$:

Succinct Non-interactive ARGument

Fix NP language L

Common Reference String (crs)

$\mathscr{P}^*$

Succinct:
$|\pi| \ll |w|$

$\mathscr{V}$

Instance x, witness w

$\pi = \mathscr{P}(\mathsf{crs}, x, w)$

x, π

$\mathscr{V}(\mathsf{crs}, x, \pi)$

- **Completeness:** If $R(x, w) = 1$, then $\mathscr{V}(\mathsf{crs}, \mathscr{P}(\mathsf{crs}, x, w)) = 1$.
- **Soundness:** For all ppt $\mathscr{P}^*$, hard to come up with cheating proof for $x^* \notin L$:

$$\Pr_{\mathsf{crs}}[(x^*, \pi^*) \leftarrow \mathscr{P}^*(\mathsf{crs}) \land x^* \notin L \land \mathscr{V}(\mathsf{crs}, x^*, \pi^*) = 1] \leq \mathsf{negl}(\lambda)$$

Designated-Verifier SNARG

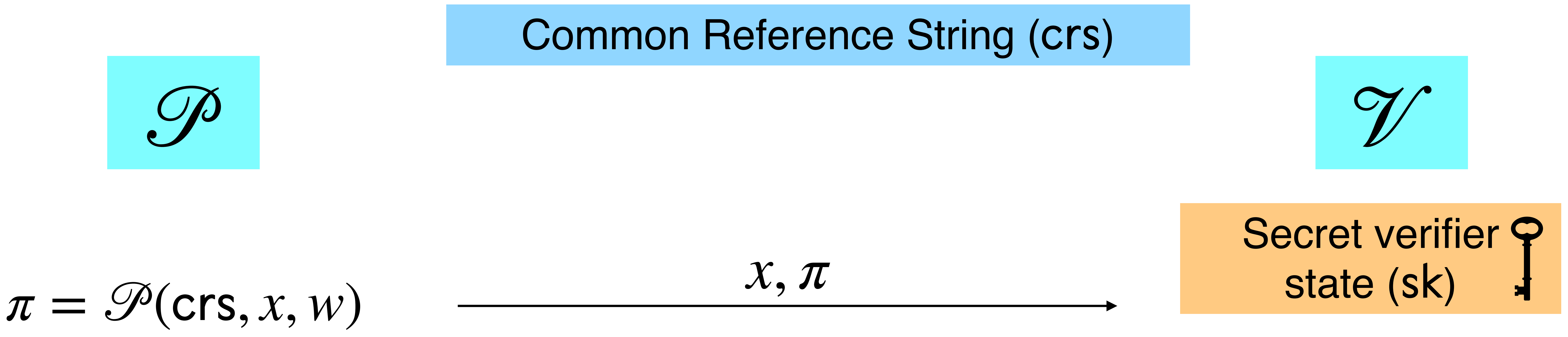
Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

$$\pi = \mathscr{P}(\mathsf{crs}, x, w) \xrightarrow{x,\ \pi}$$

Designated-Verifier SNARG



<u>D</u>esignated-<u>V</u>erifier SNARG

Common Reference String (crs)

$\mathcal{P}$

$\mathcal{V}$

Secret verifier state (sk)

$\pi = \mathcal{P}(\mathsf{crs}, x, w)$ $\xrightarrow{x,\ \pi}$

$\mathcal{V}(\mathsf{sk}, x, \pi)$

Designated-Verifier SNARG

Common Reference String (crs)

$\mathscr{P}$

$\mathscr{V}$

Secret verifier state (sk)

$\pi = \mathscr{P}(\mathsf{crs}, x, w)$

x, π

$\mathscr{V}(\mathsf{sk}, x, \pi)$

Reusable soundness:
Soundness holds even when $\mathscr{P}^*$ has black-box access to $\mathscr{V}$

Designated-Verifier SNARG

Common Reference String (crs)

$\mathscr{P}^*$

$\mathscr{V}$

Secret verifier state (sk)

$\pi = \mathscr{P}(\mathsf{crs}, x, w)$

x, π

$\mathscr{V}(\mathsf{sk}, x, \pi)$

Reusable soundness:
Soundness holds even when $\mathscr{P}^*$ has black-box access to $\mathscr{V}$

Designated-Verifier SNARG

Common Reference String (crs)

$\mathcal{P}^*$

$\mathcal{V}$

Secret verifier state (sk)

$\pi = \mathcal{P}(\mathsf{crs}, x, w)$

x, π

Accept/Reject

$\mathcal{V}(\mathsf{sk}, x, \pi)$

Reusable soundness:
Soundness holds even when $\mathcal{P}^*$ has black-box access to $\mathcal{V}$

Designated-Verifier SNARG

Common Reference String (crs)

$\mathscr{P}^*$

$\mathscr{V}$

Secret verifier state (sk)

$\pi = \mathscr{P}(\mathsf{crs}, x, w)$

x, π

Accept/Reject

$\mathscr{V}(\mathsf{sk}, x, \pi)$

x_2, π_2

Accept/Reject

Reusable soundness:
Soundness holds even when $\mathscr{P}^*$ has black-box access to $\mathscr{V}$

Designated-Verifier SNARG

Common Reference String (crs)

$\mathscr{P}^*$

$\mathscr{V}$

Secret verifier state (sk)

$\pi = \mathscr{P}(\mathsf{crs}, x, w)$

x, π →

← Accept/Reject

$\mathscr{V}(\mathsf{sk}, x, \pi)$

x_2, π_2 →

← Accept/Reject

x_3, π_3 →

← Accept/Reject

Reusable soundness:
Soundness holds even when $\mathscr{P}^*$ has black-box access to $\mathscr{V}$

State of SNARGs

State of SNARGs

ROM/Knowledge Assumptions	**NP** [Micali94], [Groth10], [DFH11], [BCIOP12], [BCCT13], [BCCGLR14] and many many more!

State of SNARGs

ROM/Knowledge Assumptions	**NP** [Micali94], [Groth10], [DFH11], [BCIOP12], [BCCT13], [BCCGLR14] and many many more!
Obfuscation	**NP** [SW14], [JJ22], [WW24], [WZ24]

State of SNARGs

ROM/Knowledge Assumptions	**NP** [Micali94], [Groth10], [DFH11], [BCIOP12], [BCCT13], [BCCGLR14] and many many more!
Obfuscation	**NP** [SW14], [JJ22], [WW24], [WZ24]
LWE/Other Polynomially Falsifiable Assumptions	**Batch-NP** [CJJ21, WW22], **P** [KRR14, KPY19, CJJ21], **NTISP** [KVZ22], **Monotone-Policy Batch-NP** [BBKLP23], Some of **NP** ∩ **CoNP** [JKLV24], many more!

State of SNARGs

ROM/Knowledge Assumptions	**NP** [Micali94], [Groth10], [DFH11], [BCIOP12], [BCCT13], [BCCGLR14] and many many more!
Obfuscation	**NP** [SW14], [JJ22], [WW24], [WZ24]
LWE/Other Polynomially Falsifiable Assumptions	**Batch-NP** [CJJ21, WW22], **P** [KRR14, KPY19, CJJ21], **NTISP** [KVZ22], **Monotone-Policy Batch-NP** [BBKLP23], Some of **NP** ∩ **CoNP** [JKLV24], many more!

NP!

State of SNARGs

ROM/Knowledge Assumptions	**NP** [Micali94], [Groth10], [DFH11], [BCIOP12], [BCCT13], [BCCGLR14] and many many more!	NP!
Obfuscation	**NP** [SW14], [JJ22], [WW24], [WZ24]	NP!
LWE/Other Polynomially Falsifiable Assumptions	**Batch-NP** [CJJ21, WW22], **P** [KRR14, KPY19, CJJ21], **NTISP** [KVZ22], **Monotone-Policy Batch-NP** [BBKLP23], Some of **NP** ∩ **CoNP** [JKLV24], many more!	

State of SNARGs

ROM/Knowledge Assumptions	**NP** [Micali94], [Groth10], [DFH11], [BCIOP12], [BCCT13], [BCCGLR14] and many many more!	NP!
Obfuscation	**NP** [SW14], [JJ22], [WW24], [WZ24]	NP!
LWE/Other Polynomially Falsifiable Assumptions	**Batch-NP** [CJJ21, WW22], **P** [KRR14, KPY19, CJJ21], **NTISP** [KVZ22], **Monotone-Policy Batch-NP** [BBKLP23], Some of **NP** ∩ **CoNP** [JKLV24], many more!	Not yet at NP, even in dv setting

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

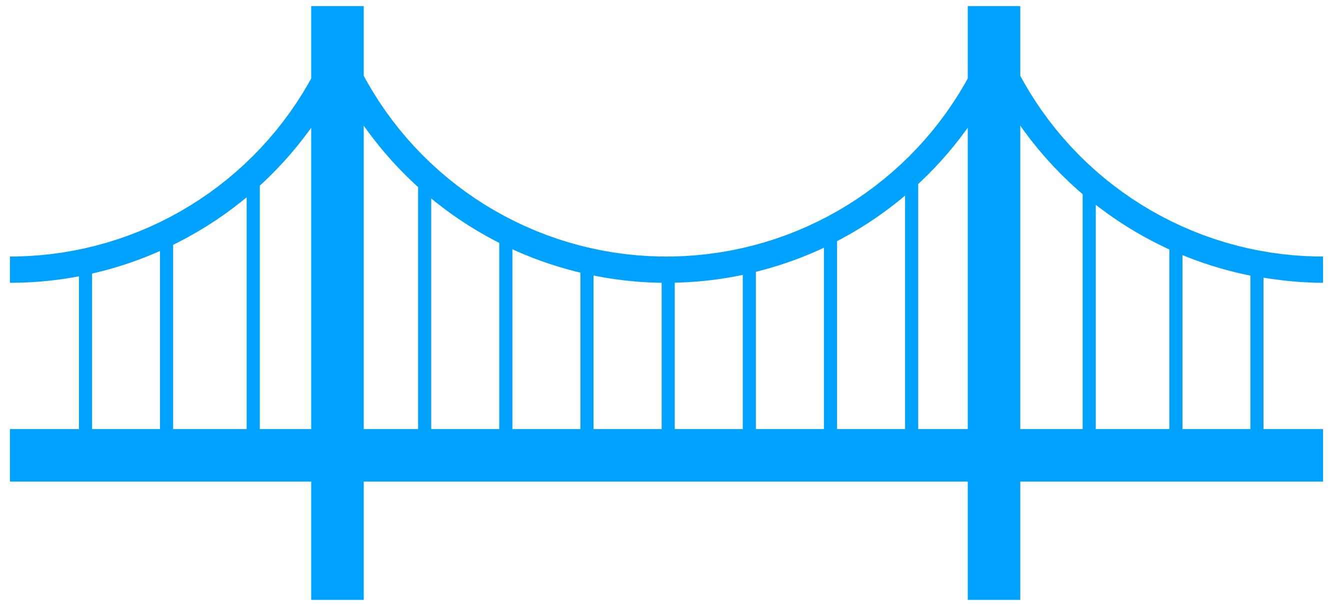
Big Picture:

LWE **Obfuscation**

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE  **Obfuscation**

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Obfuscation

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Obfuscation

Many new constructions due to evasive LWE, previously only known from obfuscation:

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Obfuscation

Many new constructions due to evasive LWE, previously only known from obfuscation:

- Optimal Broadcast Encryption and CP-ABE [Wee22]

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Obfuscation

Many new constructions due to evasive LWE, previously only known from obfuscation:

- Optimal Broadcast Encryption and CP-ABE [Wee22]
- Witness Encryption [Tsabary22, VWW22], Null-iO [VWW22]

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Obfuscation

Many new constructions due to evasive LWE, previously only known from obfuscation:

- Optimal Broadcast Encryption and CP-ABE [Wee22]
- Witness Encryption [Tsabary22, VWW22], Null-iO [VWW22]
- Multi-Authority ABE [WWW22]

Our Results

Theorem 1: We construct a dvSNARG for **UP** from **LWE** and **evasive LWE.**

Big Picture:

LWE

Obfuscation

Many new constructions due to evasive LWE, previously only known from obfuscation:

- Optimal Broadcast Encryption and CP-ABE [Wee22]
- Witness Encryption [Tsabary22, VWW22], Null-iO [VWW22]
- Multi-Authority ABE [WWW22]
- ABE for unbounded depth circuits [HLL23]

UP (or “Unique” P)

UP (or "Unique" P)

- **UP =** NP language which has a relation R such that if $x \in L$, there exists exactly **one** witness w such that $R(x, w) = 1$.

UP (or "Unique" P)

- **UP =** NP language which has a relation R such that if $x \in L$, there exists exactly **one** witness w such that $R(x, w) = 1$.

Examples of languages:

UP (or "Unique" P)

- **UP =** NP language which has a relation R such that if $x \in L$, there exists exactly **one** witness w such that $R(x, w) = 1$.

Examples of languages:

- $\text{ImF} = \{x \mid \exists y \text{ s.t. } F(y) = x\}$ where F is an injective function

UP (or "Unique" P)

- **UP =** NP language which has a relation R such that if $x \in L$, there exists exactly **one** witness w such that $R(x, w) = 1$.

Examples of languages:

- $\text{ImF} = \{x \mid \exists y \text{ s.t. } F(y) = x\}$ where F is an injective function
- $\text{Factor} = \{N \mid \exists \text{ primes } p \leq q \text{ s.t. } N = pq\}$

UP (or “Unique” P)

- **UP =** NP language which has a relation R such that if $x \in L$, there exists exactly **one** witness w such that $R(x, w) = 1$.

Examples of languages:

- ImF $= \{x \mid \exists y$ s.t. $F(y) = x\}$ where F is an injective function
- Factor $= \{N \mid \exists$ primes $p \leq q$ s.t. $N = pq\}$
- DDH $= \{(g, a, b, c) \mid \exists x, y$ s.t. $a = g^x, b = g^y, c = g^{xy}\}$

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$ (possibly correlated).

if $$(\utilde{\mathbf{SB}}, \utilde{\mathbf{SP}}) \approx_c (\mathscr{U}, \mathscr{U})$$

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$ (possibly correlated).

if $$(\utilde{\mathbf{SB}}, \utilde{\mathbf{SP}}) \approx_c (\mathcal{U},$$

Throughout this talk, squiggly lines indicate **noise**

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$ (possibly correlated).

if $$(\utilde{\mathbf{SB}}, \utilde{\mathbf{SP}}) \approx_c (\mathscr{U}, \mathscr{U})$$

Tool: Evasive LWE

Proposed by Wee (Eurocrypt ’22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$ (possibly correlated).

if $$(\mathbf{SB}, \mathbf{SP}) \approx_c (\mathscr{U}, \mathscr{U})$$

then $$(\mathbf{SB}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathscr{U}, \mathbf{B}^{-1}(\mathbf{P}))$$

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$ (possibly correlated).

$\mathbf{B}^{-1}(\mathbf{P})$ is a Gaussian pre-image sample such that $\mathbf{B} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$

if $$(\mathbf{SB}, \mathbf{SP}) \approx_c (\mathscr{U}, \mathscr{U})$$

then $$(\mathbf{SB}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathscr{U}, \mathbf{B}^{-1}(\mathbf{P}))$$

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}$, $\mathbf{B}$ and $\mathbf{P}$ (possibly correlated).

$\mathbf{B}^{-1}(\mathbf{P})$ is a Gaussian pre-image sample such that $\mathbf{B} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$

if $$(\mathbf{SB}, \mathbf{SP}) \approx_c (\mathscr{U}, \mathscr{U})$$

then $$(\mathbf{SB}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathscr{U}, \mathbf{B}^{-1}(\mathbf{P}))$$

Intuition: Given $\mathbf{SB}$ and $\mathbf{B}^{-1}(\mathbf{P})$, can compute $\mathbf{SB} \cdot \mathbf{B}^{-1}(\mathbf{P}) \approx \mathbf{SP}$, and not much else.

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}$, $\mathbf{B}$ and $\mathbf{P}$.

$\mathbf{B}^{-1}(\mathbf{P})$ is a Gaussian pre-image sample such that $\mathbf{B} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$

if $$(\mathbf{SB}, \mathbf{SP}) \approx_c (\mathscr{U}, \mathscr{U})$$

then $$(\mathbf{SB}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathscr{U}, \mathbf{B}^{-1}(\mathbf{P}))$$

Tool: Evasive LWE

Proposed by Wee (Eurocrypt '22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$.

$\mathbf{B}^{-1}(\mathbf{P})$ is a Gaussian pre-image sample such that $\mathbf{B} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$

if $$(\underset{\sim}{\mathbf{SB}}, \underset{\sim}{\mathbf{SP}}) \approx_c (\mathscr{U}, \mathscr{U})$$

then $$(\underset{\sim}{\mathbf{SB}}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathscr{U}, \mathbf{B}^{-1}(\mathbf{P}))$$

- **Motivation:** Many attacks on lattice-inspired obfuscation schemes rely on the so-called **"zeroizing regime".** Evasive LWE seems to avoid this.

Tool: Evasive LWE

Proposed by Wee (Eurocrypt ’22).

Fix distributions $\mathbf{S}, \mathbf{B}$ and $\mathbf{P}$.

if $$(\underset{\sim}{\mathbf{SB}}, \underset{\sim}{\mathbf{SP}}) \approx_c (\mathscr{U}, \mathscr{U})$$

$\mathbf{B}^{-1}(\mathbf{P})$ is a Gaussian pre-image sample such that $\mathbf{B} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$

then $$(\underset{\sim}{\mathbf{SB}}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathscr{U}, \mathbf{B}^{-1}(\mathbf{P}))$$

- **Motivation:** Many attacks on lattice-inspired obfuscation schemes rely on the so-called **“zeroizing regime”.** Evasive LWE seems to avoid this.
  - **Idea:** Collect many equations on **low-norm secrets** over low-norm constants. Solve over integers!

Zeroizing attacks

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{S}\mathbf{P} = \mathbf{0}$. Then, given $\mathbf{S}\mathbf{B} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})$$

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})$$

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})$$

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})$$

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})$$

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \boxed{\mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})}$$

- Now, we can solve for $\mathbf{E}$ over **integers**, because everything on RHS has **low-norm**.

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})$$

- Now, we can solve for $\mathbf{E}$ over **integers**, because everything on RHS has **low-norm**.

- With $\mathbf{E}$ in the clear, no more LWE guarantees on $\mathbf{SB} + \mathbf{E}$!

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \boxed{\mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})}$$

- Now, we can solve for $\mathbf{E}$ over **integers**, because everything on RHS has **low-norm**.

- With $\mathbf{E}$ in the clear, no more LWE guarantees on $\mathbf{SB} + \mathbf{E}$!

- Similar attack works for $\mathbf{SP}$ with correlated rows.

Zeroizing attacks

- **Extreme example:** Suppose $\mathbf{SP} = \mathbf{0}$. Then, given $\mathbf{SB} + \mathbf{E}$ and $\mathbf{B}^{-1}(\mathbf{P})$, one can compute the product:

$$(\mathbf{SB} + \mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{SP} + \mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P}) = \boxed{\mathbf{E} \cdot \mathbf{B}^{-1}(\mathbf{P})}$$

- Now, we can solve for $\mathbf{E}$ over **<u>integers</u>**, because everything on RHS has **<u>low-norm</u>**.
- With $\mathbf{E}$ in the clear, no more LWE guarantees on $\mathbf{SB} + \mathbf{E}$!
- Similar attack works for $\mathbf{SP}$ with correlated rows.
- **Evasive LWE:** This is the only attack! Doesn't work if $\mathbf{SP}$ were uniform.

Main Tool

Main Tool

- Using evasive LWE, we construct a new **“average-case obfuscation”** $\mathcal{O}$ for *“matrix programs”* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

Main Tool

- Using evasive LWE, we construct a new **“average-case obfuscation”** $\mathcal{O}$ for *“matrix programs”* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{\underset{\sim}{F_k(x)}\}_x, \mathsf{aux}) \approx (\mathcal{U}, \mathsf{aux})$$

Main Tool

- Using evasive LWE, we construct a new **“average-case obfuscation”** $\mathcal{O}$ for *“matrix programs”* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{\underset{\sim\sim\sim}{F_k(x)}\}_x, \mathsf{aux}) \approx (\mathscr{U}, \mathsf{aux})$$

(i.e. the function is a “very secure PRF”)

Main Tool

- Using evasive LWE, we construct a new **"average-case obfuscation"** $\mathcal{O}$ for *"matrix programs"* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{F_k(x)\}_x, \mathsf{aux}) \approx (\mathcal{U}, \mathsf{aux})$$

(i.e. the function is a "very secure PRF")

$$\text{then} \quad (\mathcal{O}(F_k), \mathsf{aux}) \approx (\mathcal{D}, \mathsf{aux})$$

Main Tool

- Using evasive LWE, we construct a new **"average-case obfuscation"** $\mathcal{O}$ for *"matrix programs"* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{F_k(x)\}_x, \mathsf{aux}) \approx (\mathcal{U}, \mathsf{aux})$$

(i.e. the function is a "very secure PRF")

$$\text{then} \quad (\mathcal{O}(F_k), \mathsf{aux}) \approx (\mathcal{D}, \mathsf{aux})$$

(i.e. the obfuscation leaks nothing more than the outputs)

Main Tool

- Using evasive LWE, we construct a new **"average-case obfuscation"** $\mathcal{O}$ for *"matrix programs"* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{\underset{\sim\sim\sim}{F_k(x)}\}_x, \mathsf{aux}) \approx (\mathcal{U}, \mathsf{aux})$$

(i.e. the function is a "very secure PRF")

$$\text{then} \quad (\mathcal{O}(F_k), \mathsf{aux}) \approx (\mathcal{D}, \mathsf{aux})$$

(i.e. the obfuscation leaks nothing more than the outputs)

- Follows techniques of [GGH15] and generalises [VWW22].

Main Tool

- Using evasive LWE, we construct a new **"average-case obfuscation"** $\mathcal{O}$ for *"matrix programs"* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{F_k(x)\}_x, \mathsf{aux}) \approx (\mathcal{U}, \mathsf{aux})$$

(i.e. the function is a "very secure PRF")

$$\text{then} \quad (\mathcal{O}(F_k), \mathsf{aux}) \approx (\mathcal{D}, \mathsf{aux})$$

(i.e. the obfuscation leaks nothing more than the outputs)

- Follows techniques of [GGH15] and generalises [VWW22].

- Useful notion that immediately implies: Constrained PRFs, shift-hiding PRFs, etc

Main Tool

- Using evasive LWE, we construct a new **"average-case obfuscation"** $\mathcal{O}$ for *"matrix programs"* $\{F_k\}_{k \in K}$ with roughly the following guarantee (over $k \leftarrow K$):

$$\text{if} \quad (\{F_k(x)\}_x, \mathsf{aux}) \approx (\mathcal{U}, \mathsf{aux})$$

(i.e. the function is a "very secure PRF")

$$\text{then} \quad (\mathcal{O}(F_k), \mathsf{aux}) \approx (\mathcal{D}, \mathsf{aux})$$

(i.e. the obfuscation leaks nothing more than the outputs)

- Follows techniques of [GGH15] and generalises [VWW22].
- Useful notion that immediately implies: Constrained PRFs, shift-hiding PRFs, etc
- Use this obfuscation to instantiate a "Sahai-Waters"-like SNARG. More details later!

TL;DR

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

    - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!

3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u>** for **UP**.

    - Corollary: **Adaptively sound** dv-SNAR<u>K</u> for UP from **falsifiable assumptions.**

All of the above constructions/transformations also **satisfy/preserve zero-knowledge!**

SNARGs vs. SNARKs

SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge

SNARGs vs. SNARKs

- **<u>S</u>**uccinct **<u>N</u>**on-Interactive **<u>Ar</u>**gument of **<u>K</u>**nowledge
    - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.

SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.

$\mathscr{P}$

SNARGs vs. SNARKs

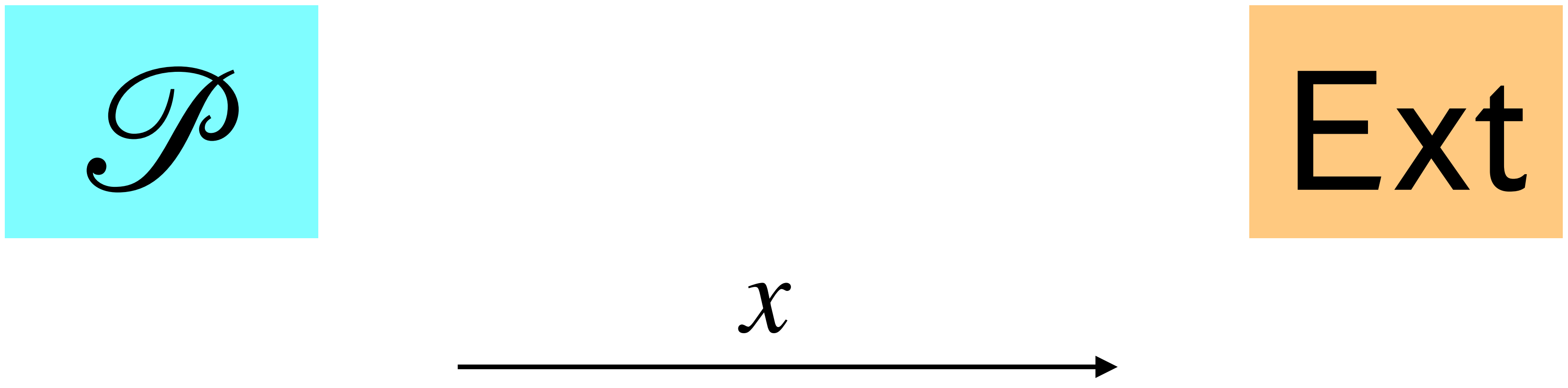
- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.

$\mathcal{P}$

Ext

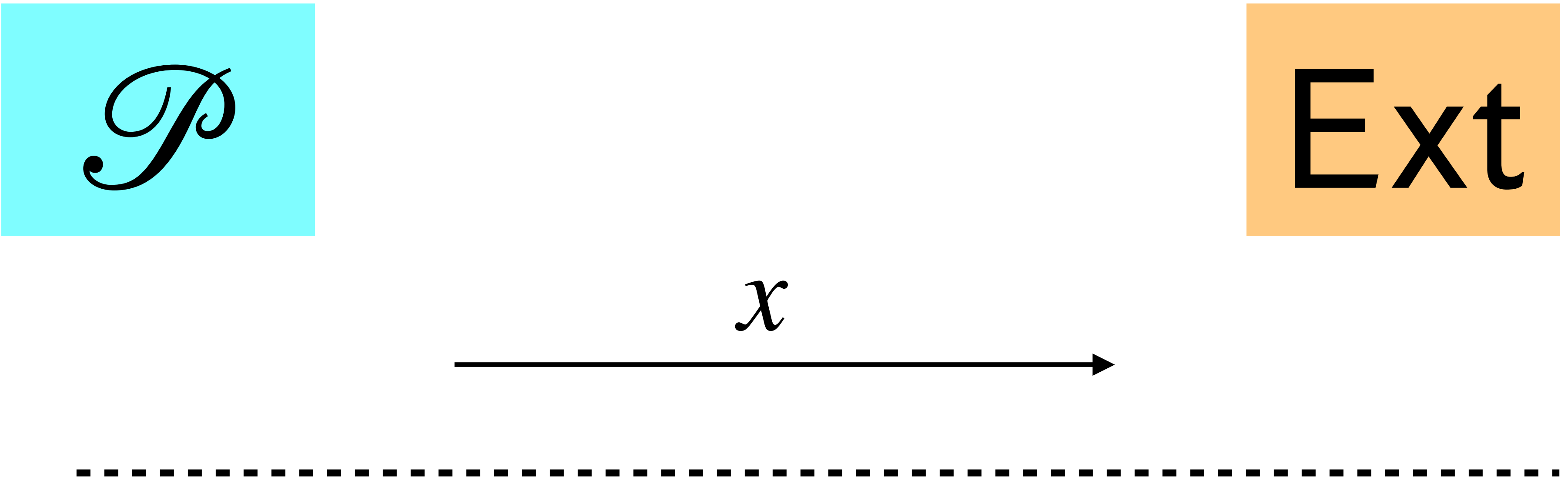
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



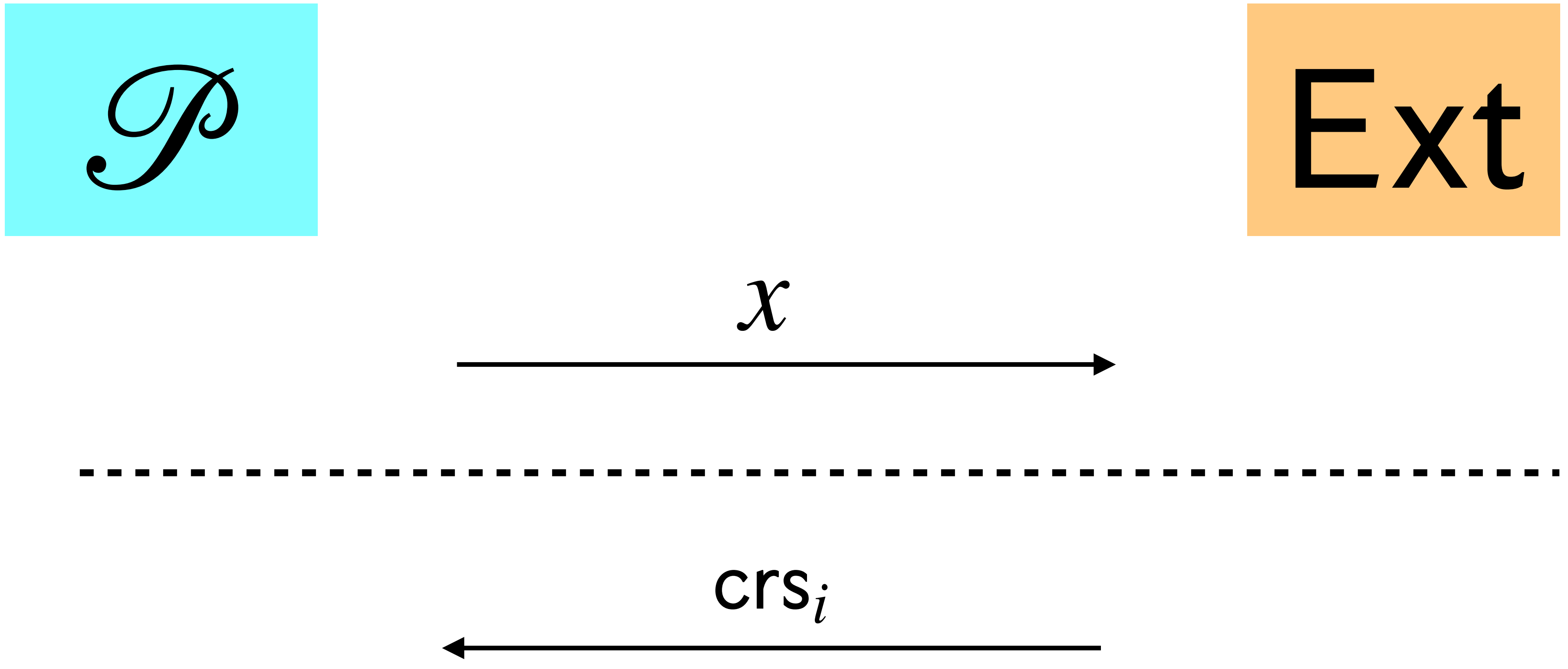
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



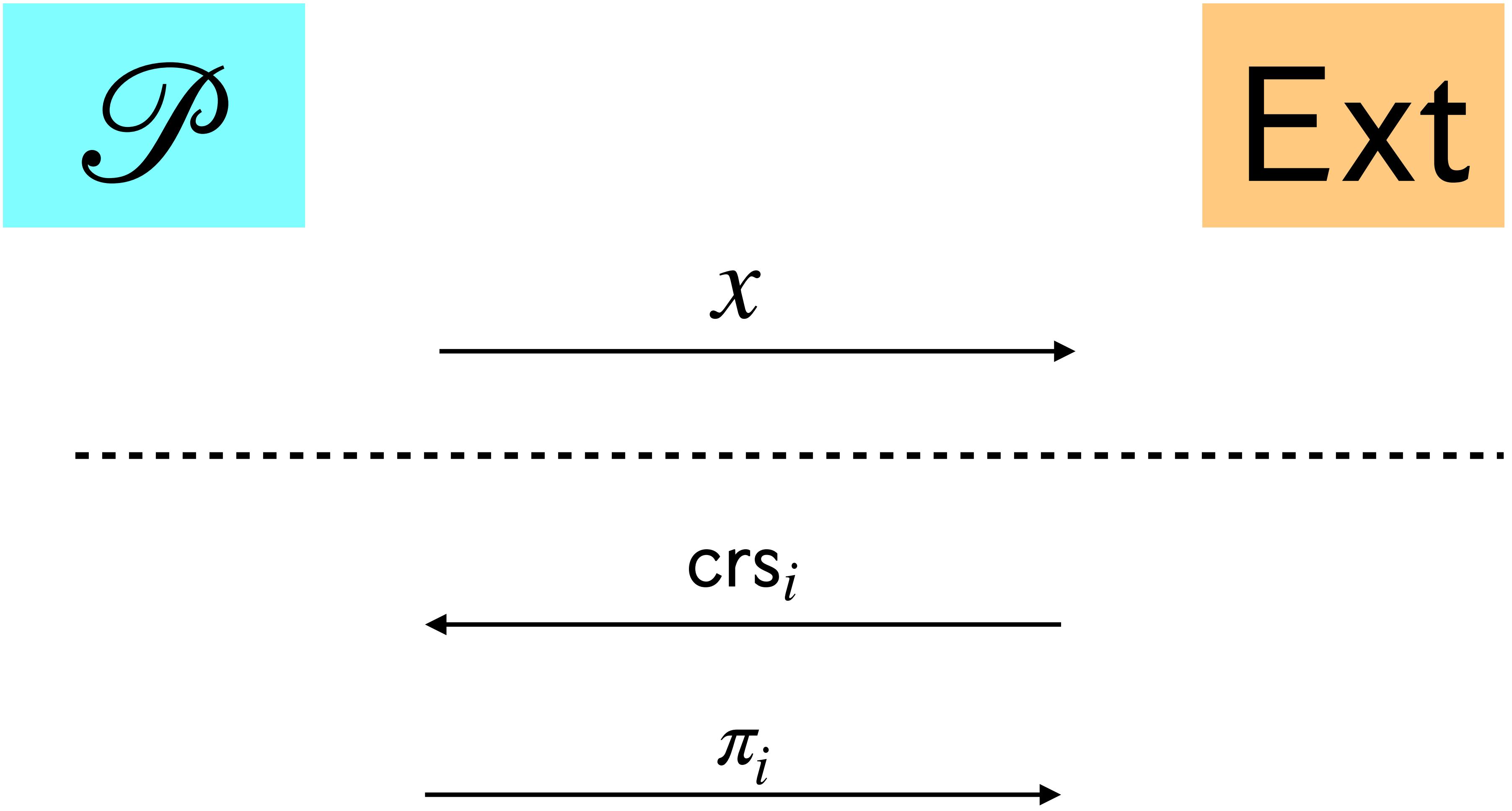
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



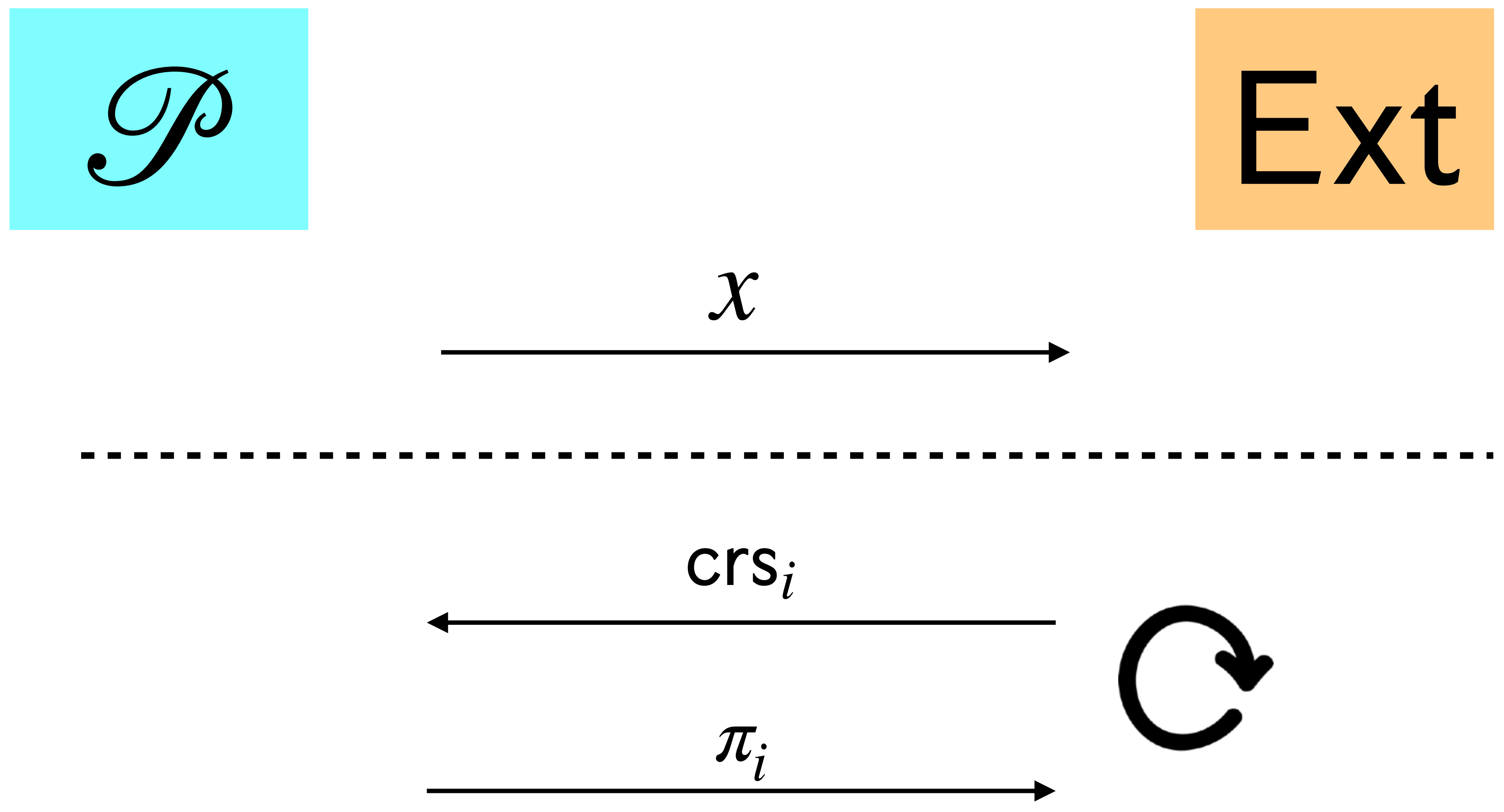
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



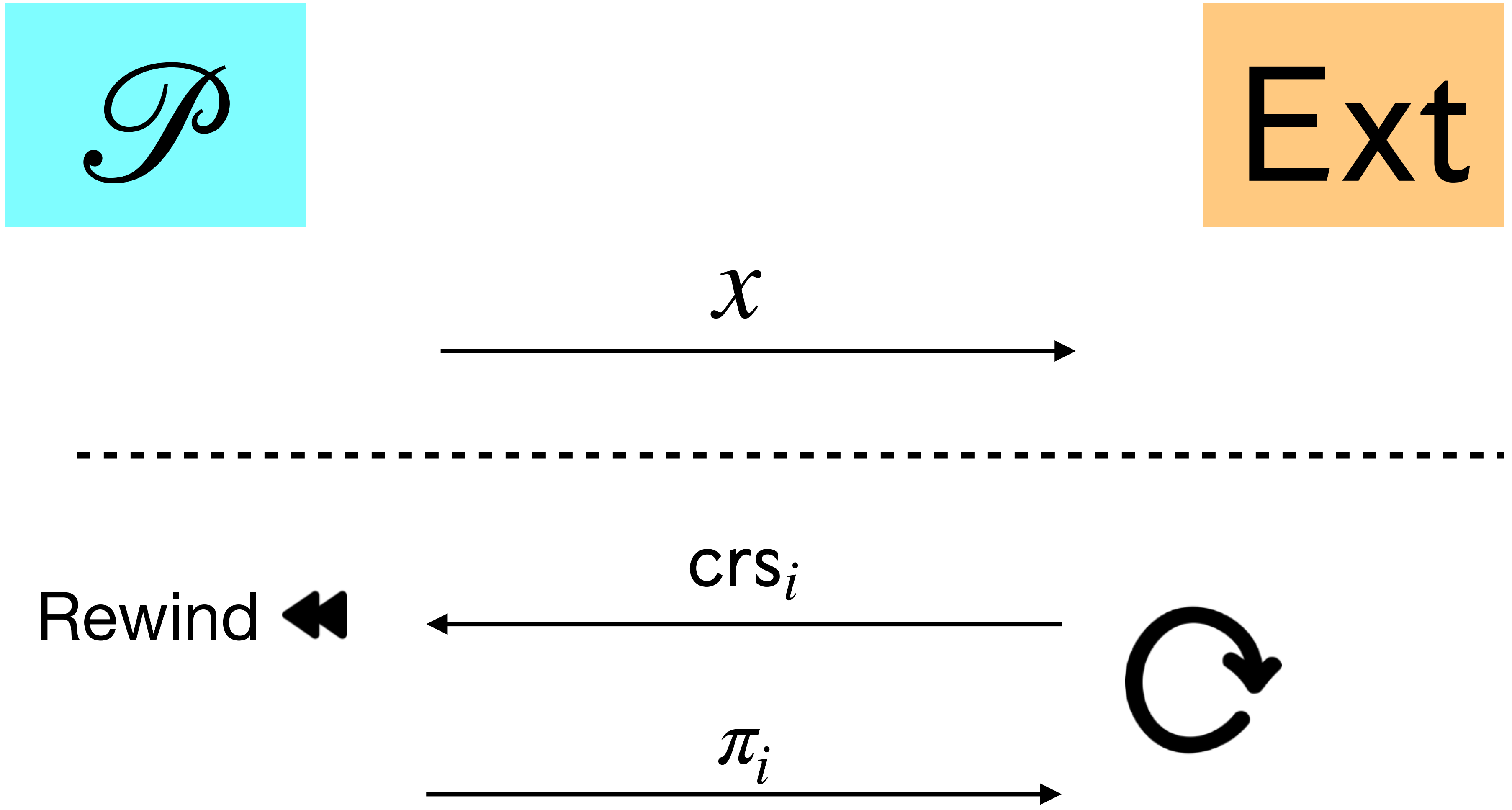
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must "know" the witness.



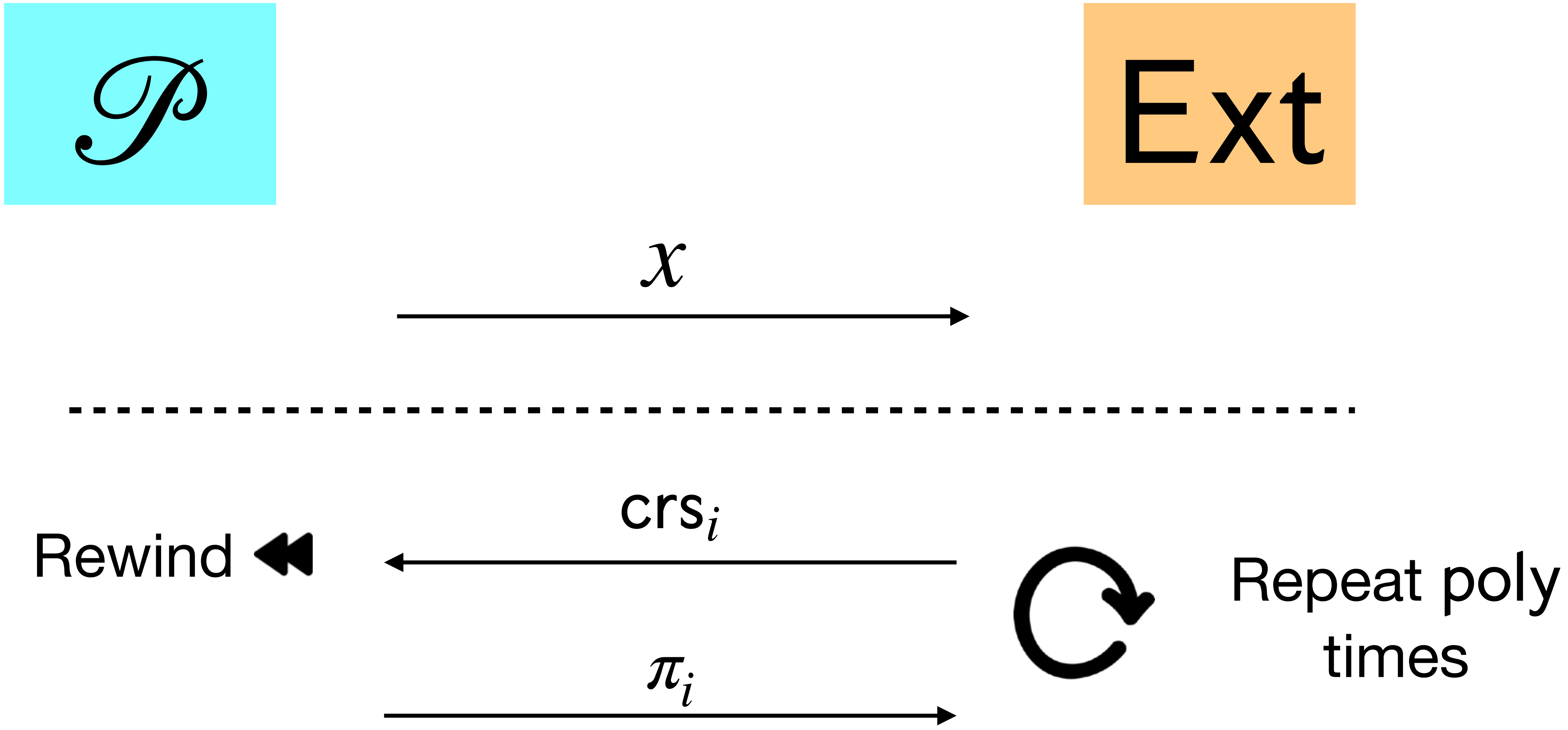
SNARGs vs. SNARKs

- **<u>S</u>**uccinct **<u>N</u>**on-Interactive **<u>Ar</u>**gument of **<u>K</u>**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



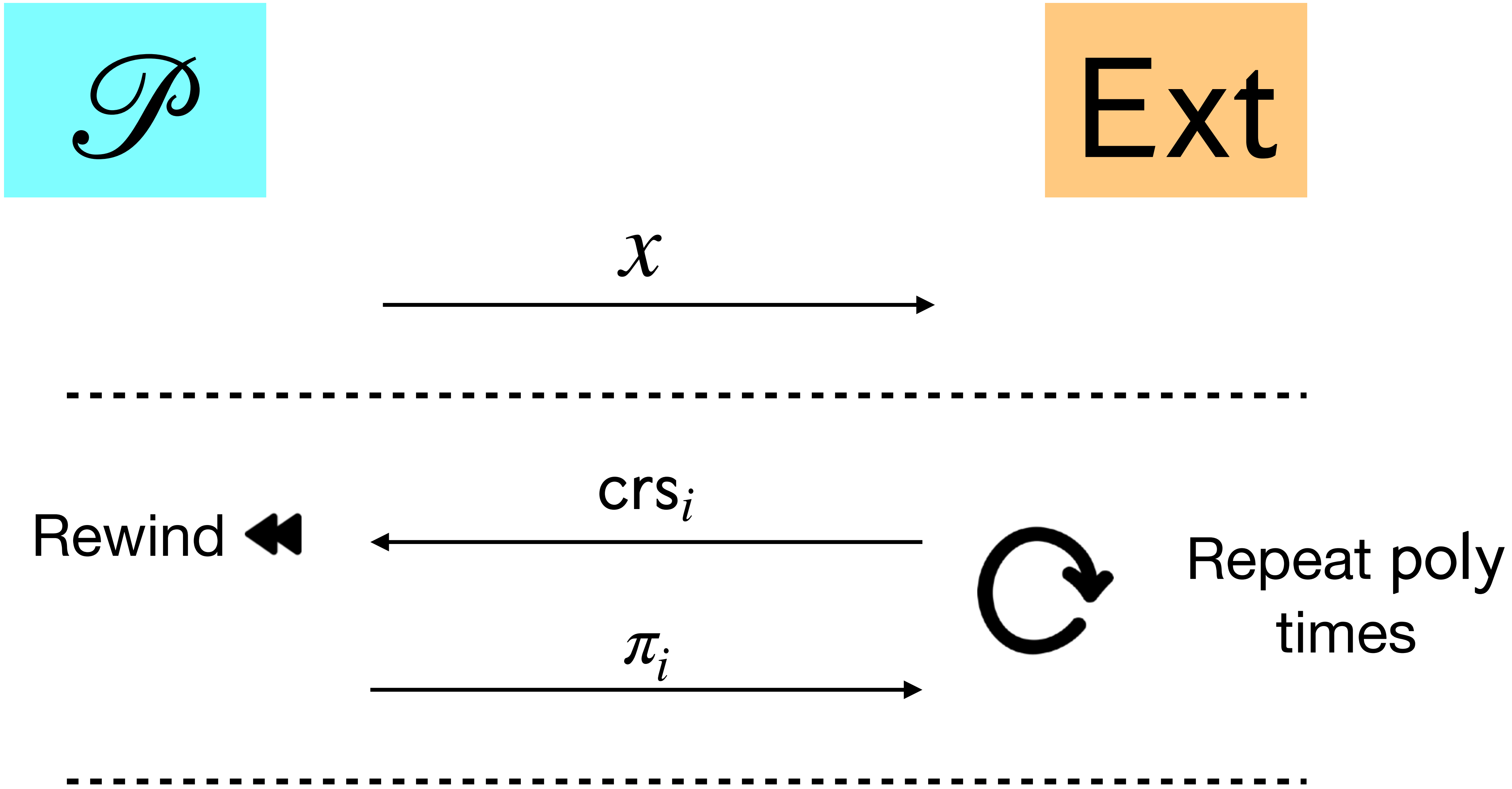
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



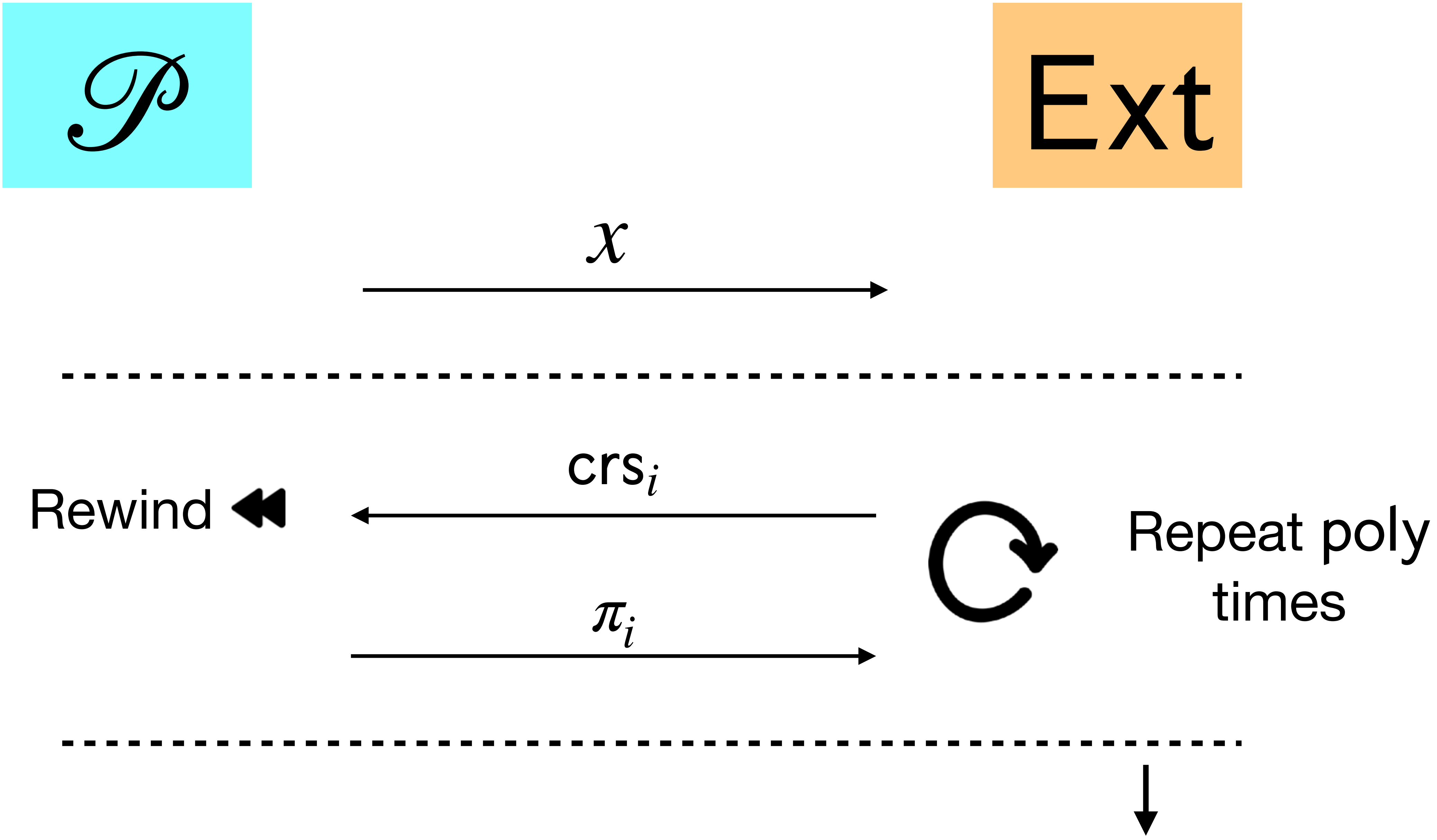
SNARGs vs. SNARKs

- **<u>S</u>**uccinct **<u>N</u>**on-Interactive **<u>Ar</u>**gument of **<u>K</u>**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must "know" the witness.



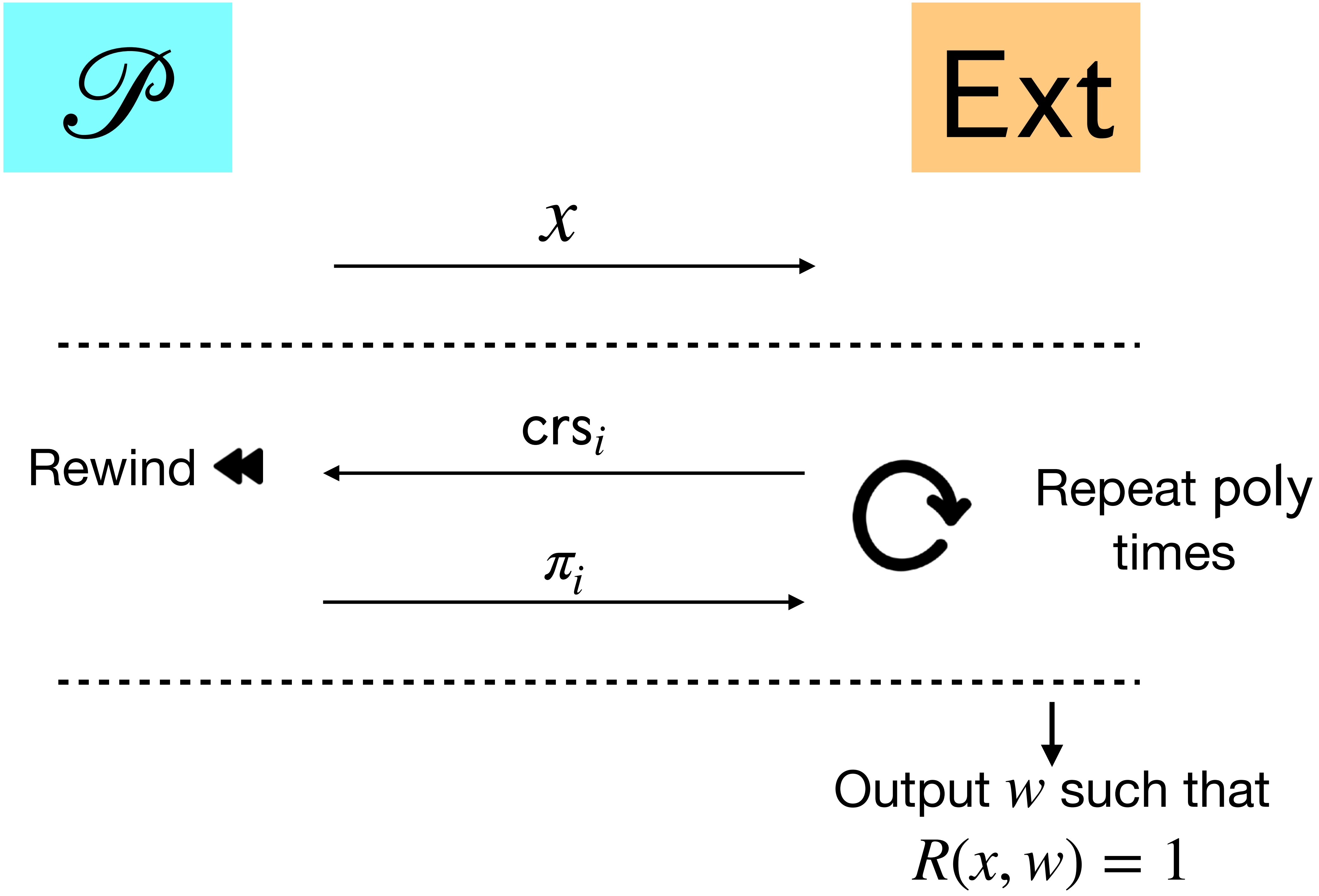
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
    - One can *extract* a witness from accepting proofs, i.e. the prover must "know" the witness.



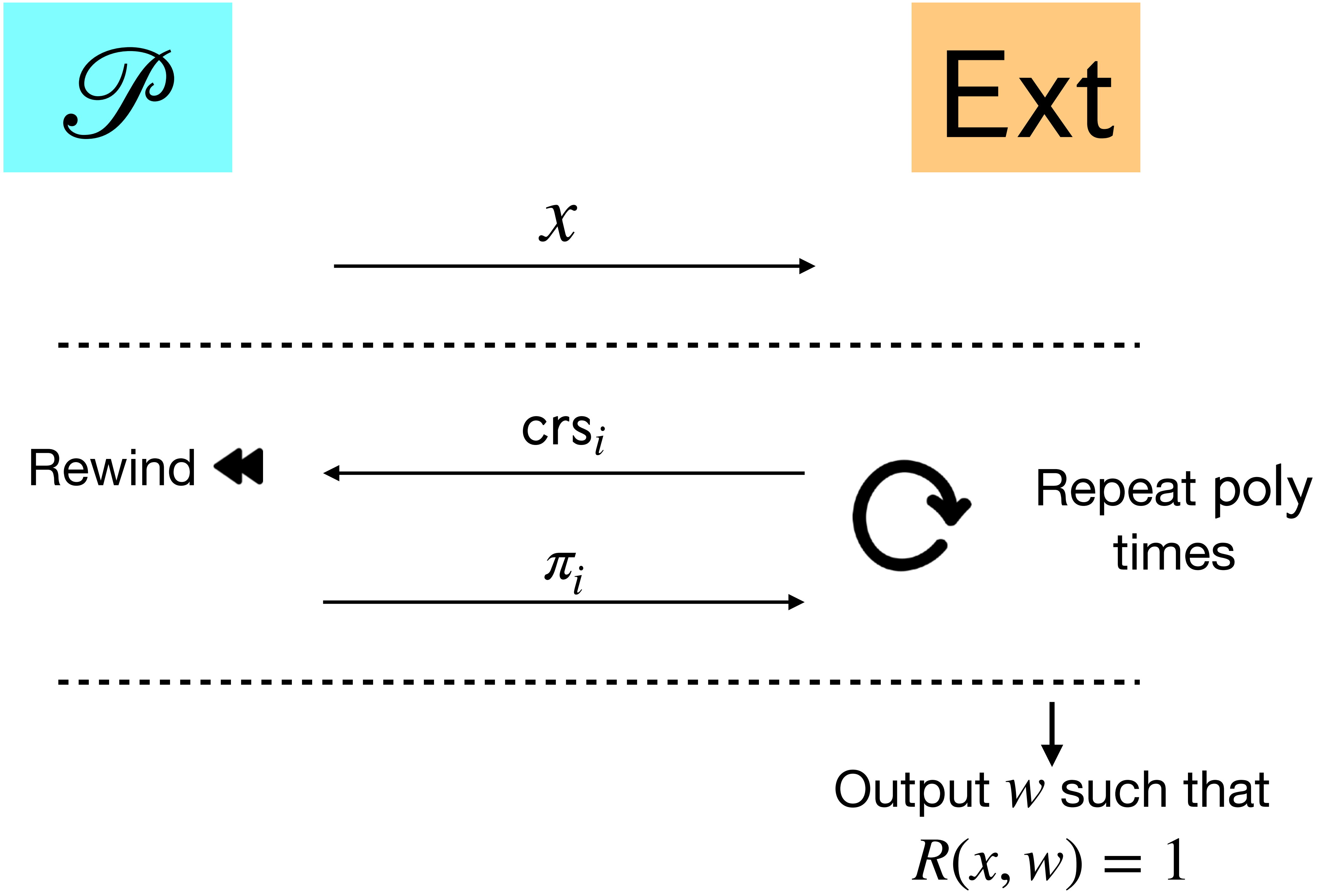
SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



SNARGs vs. SNARKs

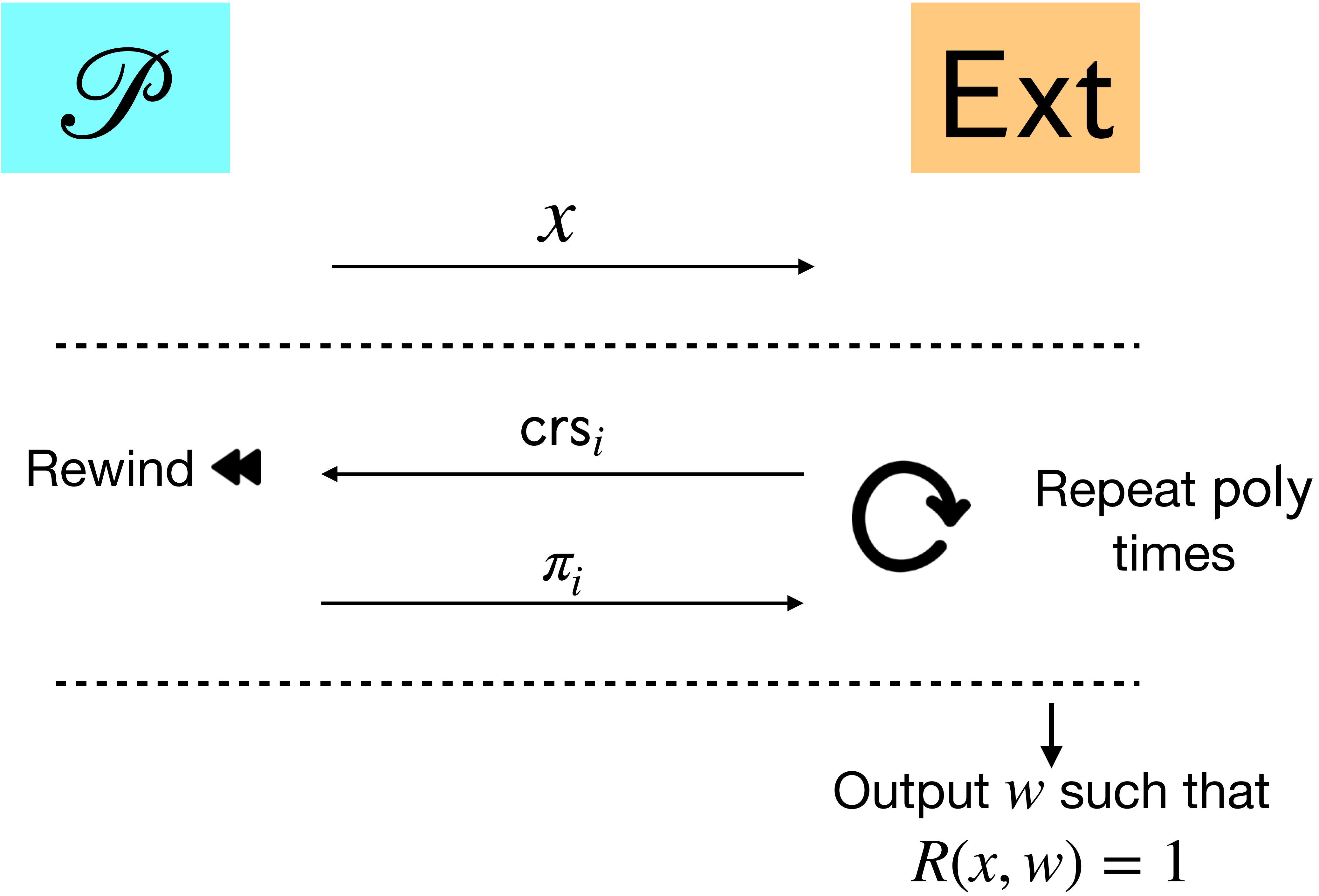
- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
    - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.



Knowledge soundness: $\exists$Ext, if $\mathscr{P}$ can create accepting proofs for x with $1/\text{poly}(n)$ probability, $\text{Ext}^{\mathscr{P}(x)}$ outputs a w.

SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
    - One can *extract* a witness from accepting proofs, i.e. the prover must "know" the witness.

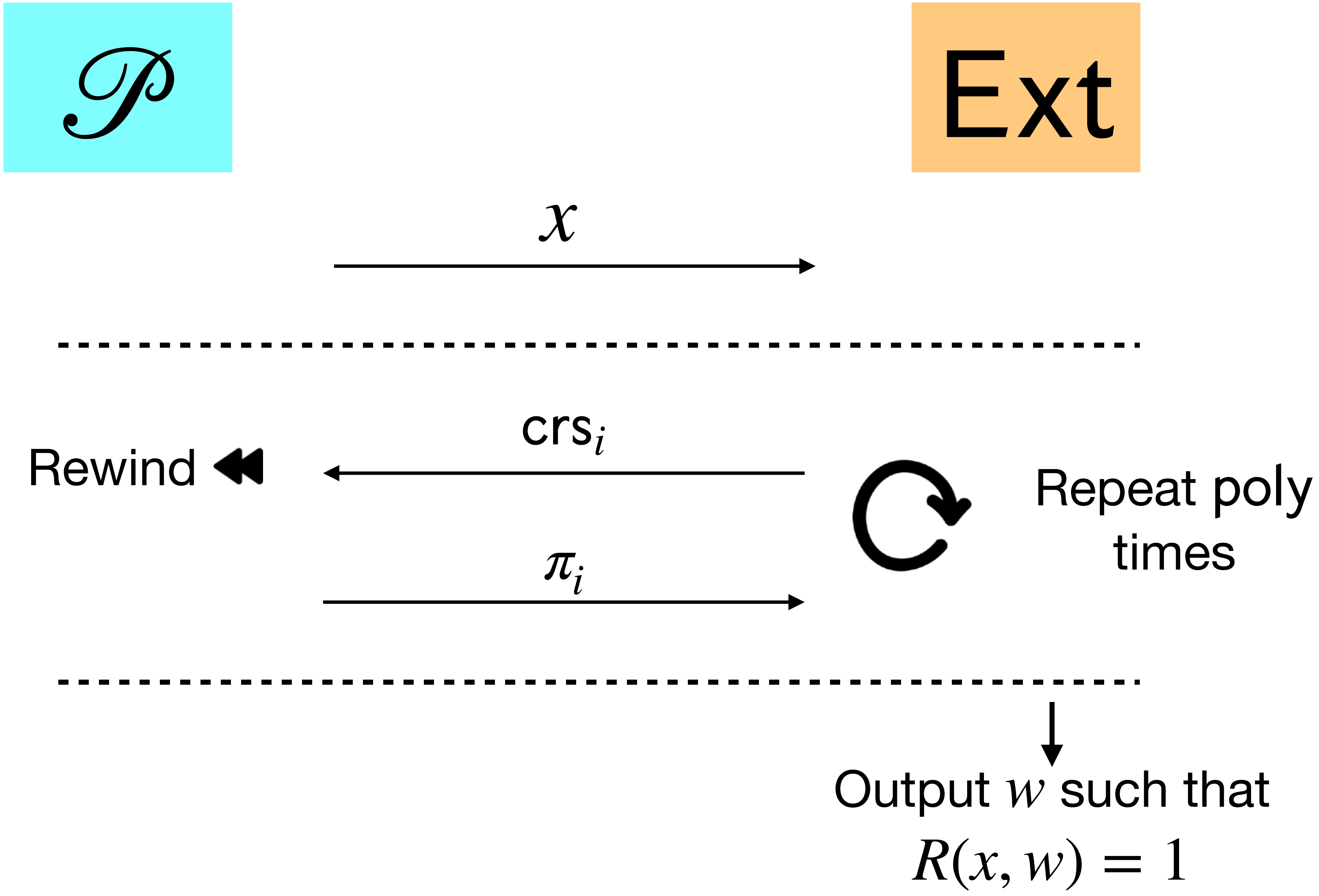


Knowledge soundness: $\exists$Ext, if $\mathscr{P}$ can create accepting proofs for x with $1/\mathsf{poly}(n)$ probability, $\mathsf{Ext}^{\mathscr{P}(x)}$ outputs a w.

*Note that this definition is non-adaptive. This is the best one can hope for from falsifiable assumptions.

SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must "know" the witness.
  - **Recall:** Regular SNARG definition has no soundness guarantees if a **prover does not know a witness** for $x \in L$.

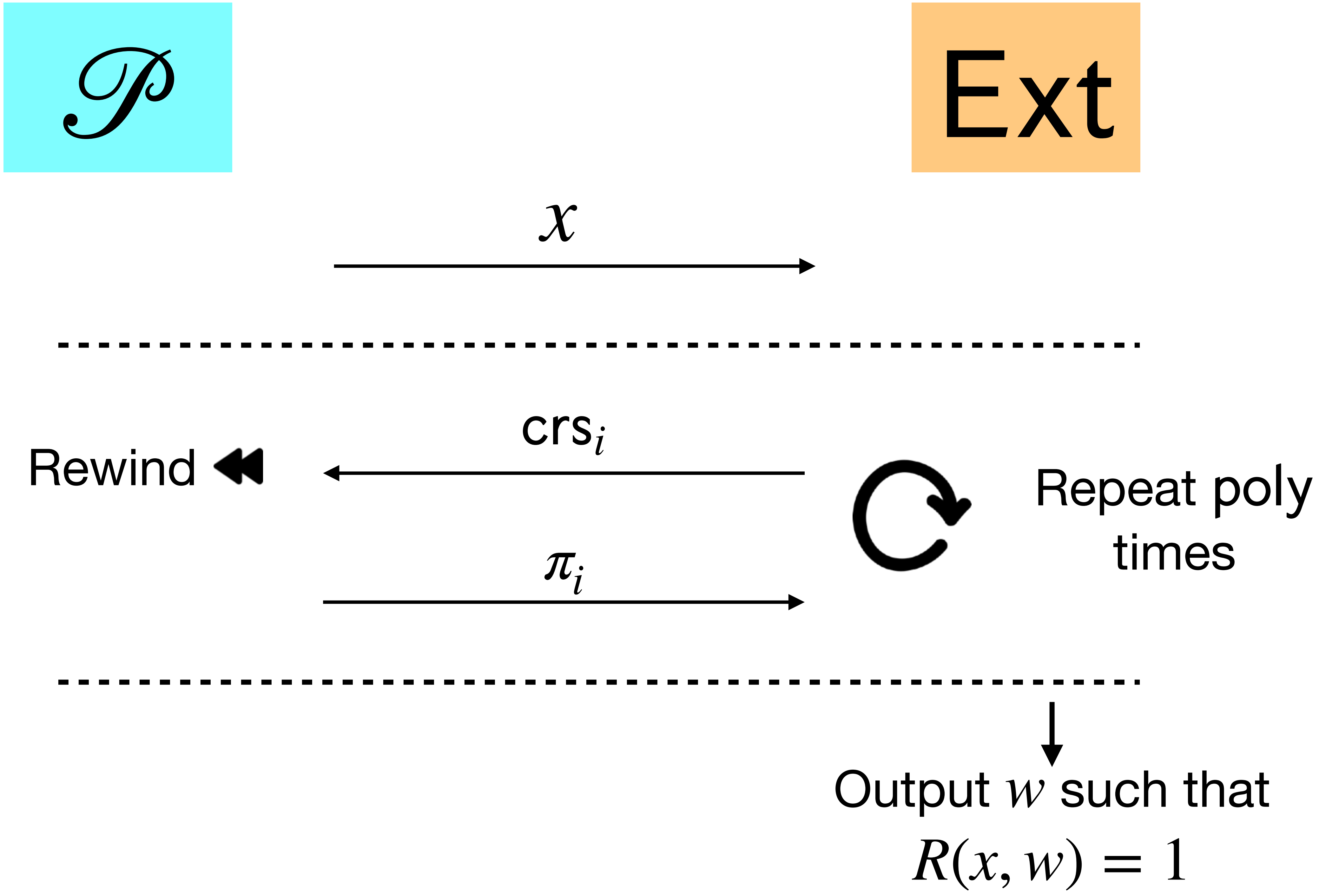


Knowledge soundness: $\exists$Ext, if $\mathscr{P}$ can create accepting proofs for x with $1/\mathrm{poly}(n)$ probability, $\mathrm{Ext}^{\mathscr{P}(x)}$ outputs a w.

*Note that this definition is non-adaptive. This is the best one can hope for from falsifiable assumptions.

SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.
  - **Recall:** Regular SNARG definition has no soundness guarantees if a **prover does not know a witness** for $x \in L$.
- SNARKs “**compose better**” than SNARGs with cryptographic objects

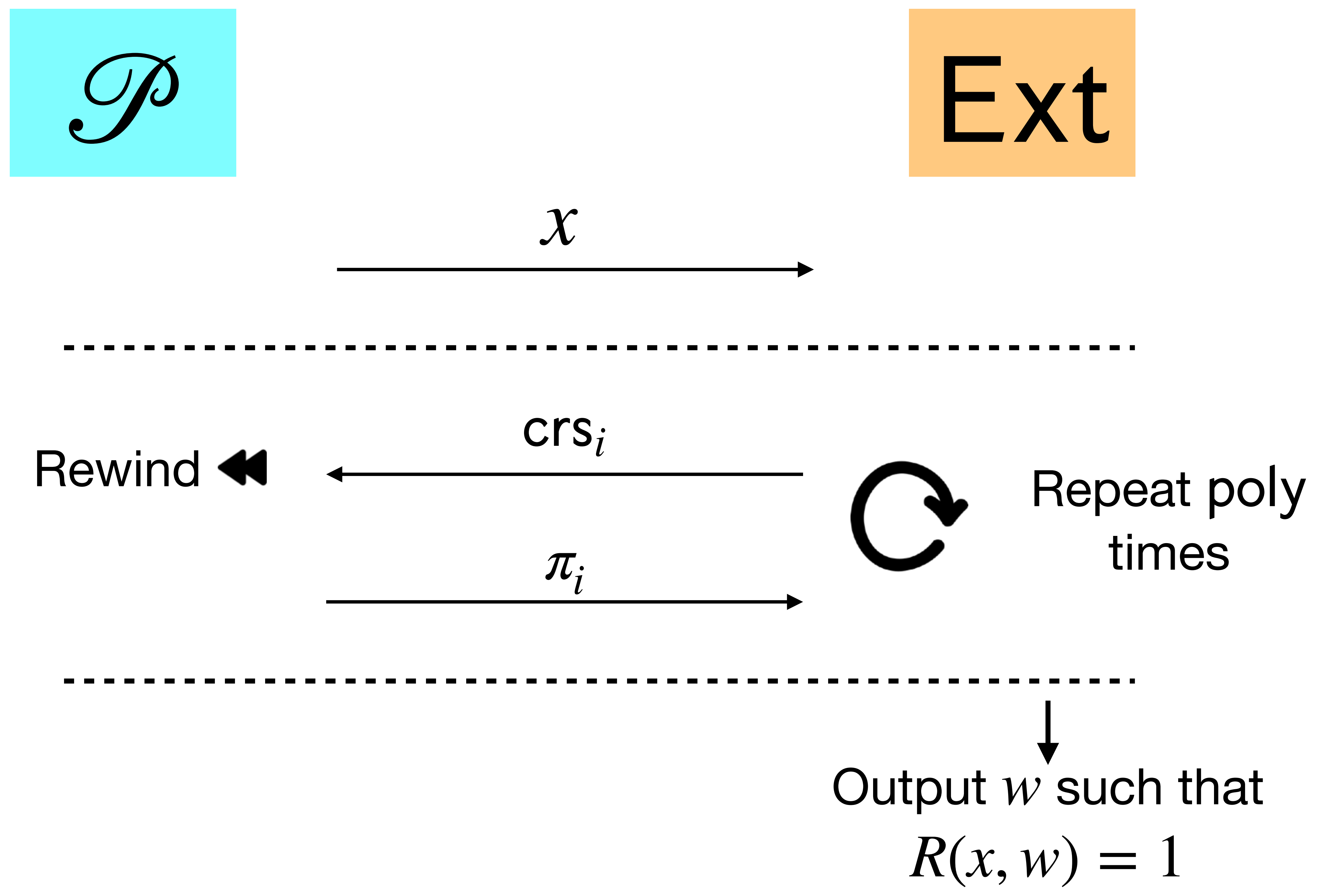


Knowledge soundness: $\exists$Ext, if $\mathscr{P}$ can create accepting proofs for x with $1/\mathrm{poly}(n)$ probability, $\mathrm{Ext}^{\mathscr{P}(x)}$ outputs a w.

*Note that this definition is non-adaptive. This is the best one can hope for from falsifiable assumptions.

SNARGs vs. SNARKs

- **S**uccinct **N**on-Interactive **Ar**gument of **K**nowledge
  - One can *extract* a witness from accepting proofs, i.e. the prover must “know” the witness.
  - **Recall:** Regular SNARG definition has no soundness guarantees if a **prover does not know a witness** for $x \in L$.
- SNARKs “**compose better**” than SNARGs with cryptographic objects
  - E.g. Somewhere extractable BARGs



Knowledge soundness: $\exists$Ext, if $\mathscr{P}$ can create accepting proofs for x with $1/\text{poly}(n)$ probability, $\text{Ext}^{\mathscr{P}(x)}$ outputs a w.

*Note that this definition is non-adaptive. This is the best one can hope for from falsifiable assumptions.

Barrier to SNARKs for NP

Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]

Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.

Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!

Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
    - Each proof is a **"<u>small leakage</u>"** on a witness.
    - The prover might **use a different witness** each time!

$\mathscr{P}$

Barrier to SNARKs for NP

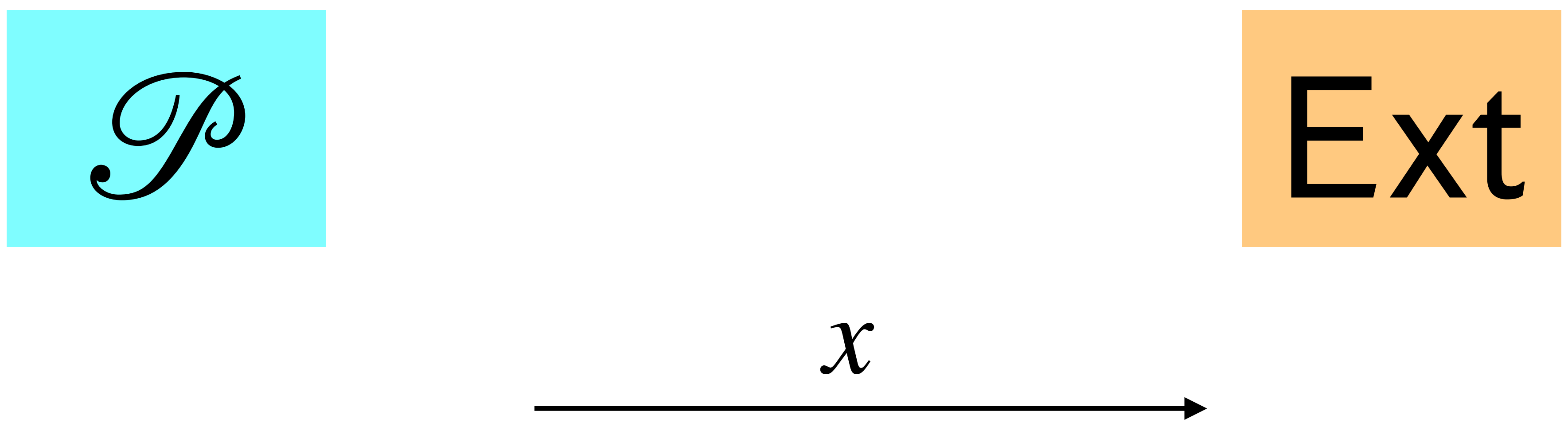
$\mathcal{P}$

Ext

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
    - Each proof is a **"<u>small leakage</u>"** on a witness.
    - The prover might **use a different witness** each time!

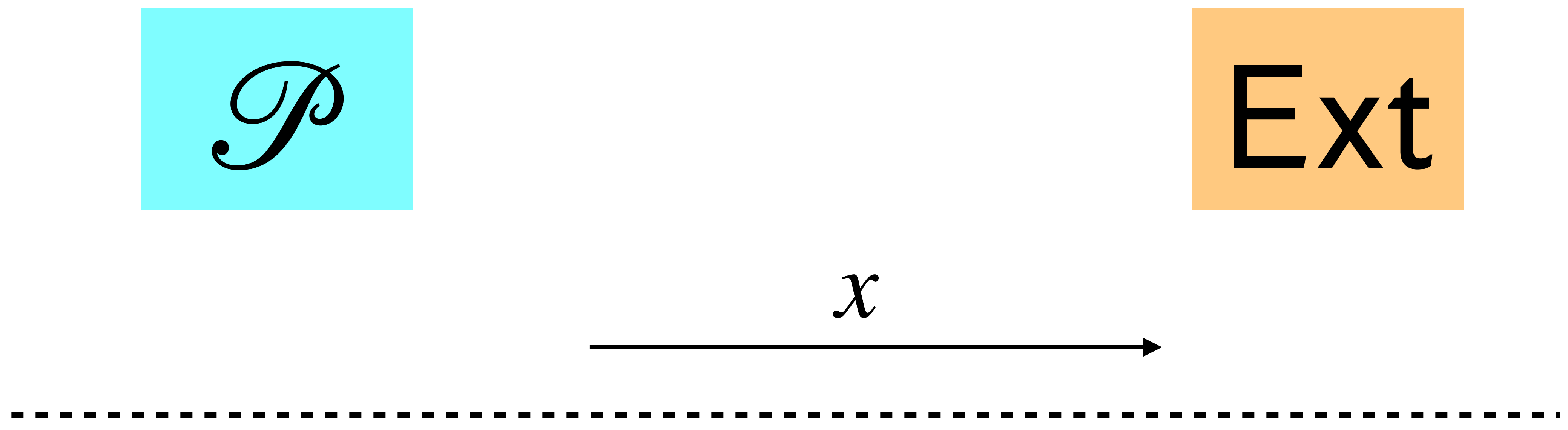
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
    - Each proof is a **"<u>small leakage</u>"** on a witness.
    - The prover might **use a different witness** each time!



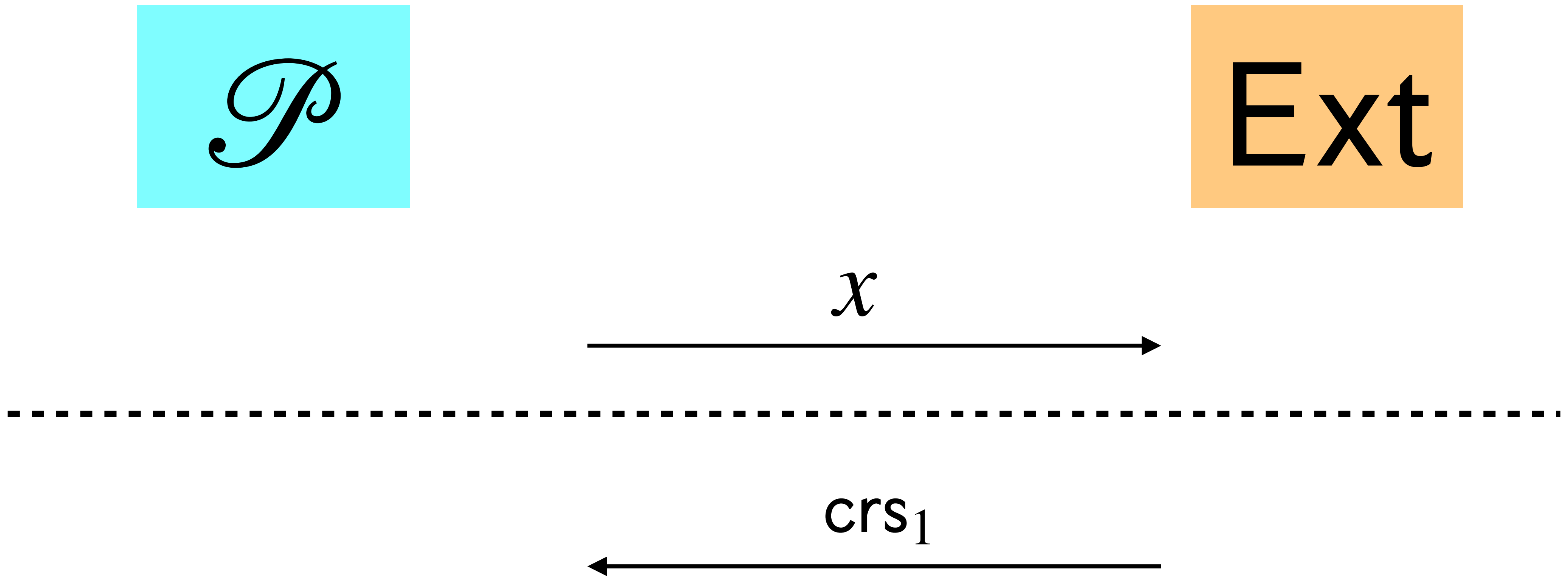
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!



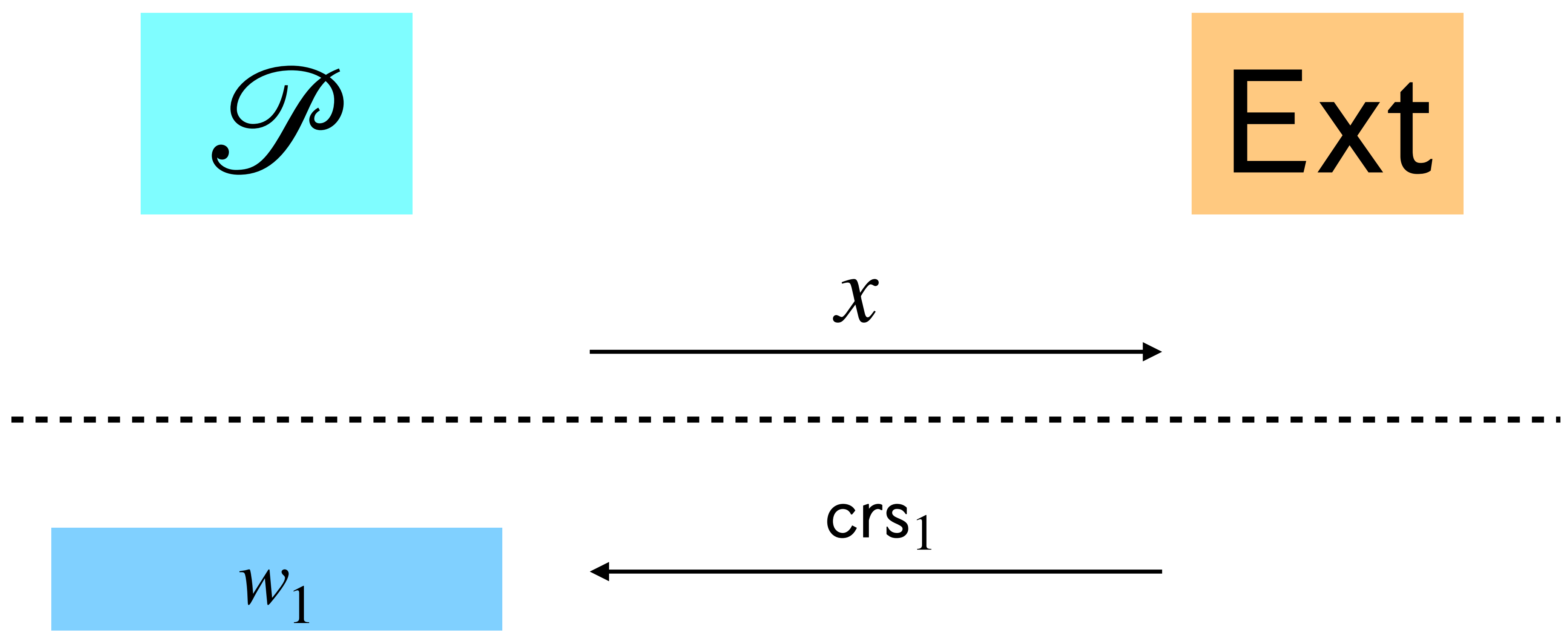
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!



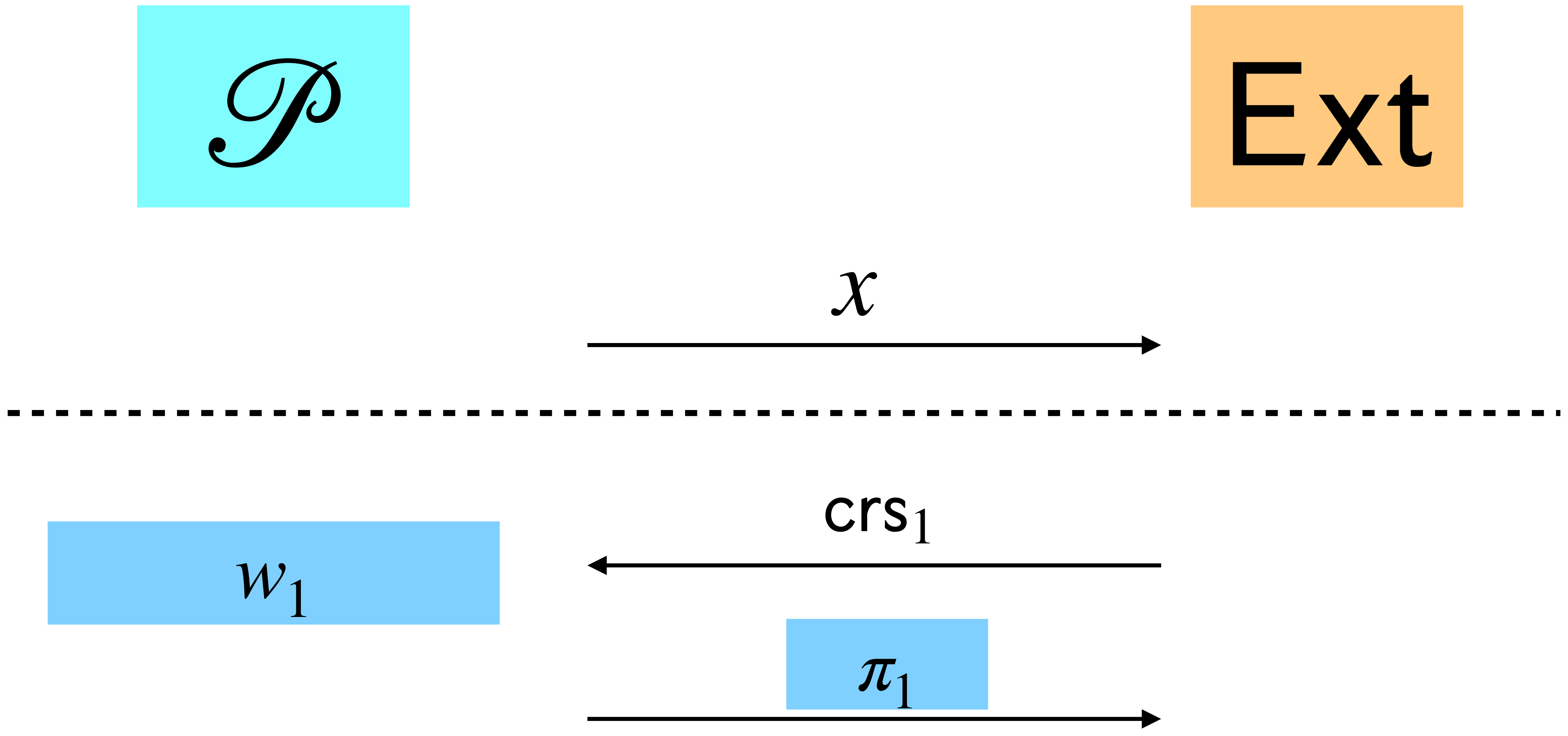
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **“<u>small leakage</u>”** on a witness.
  - The prover might **use a different witness** each time!



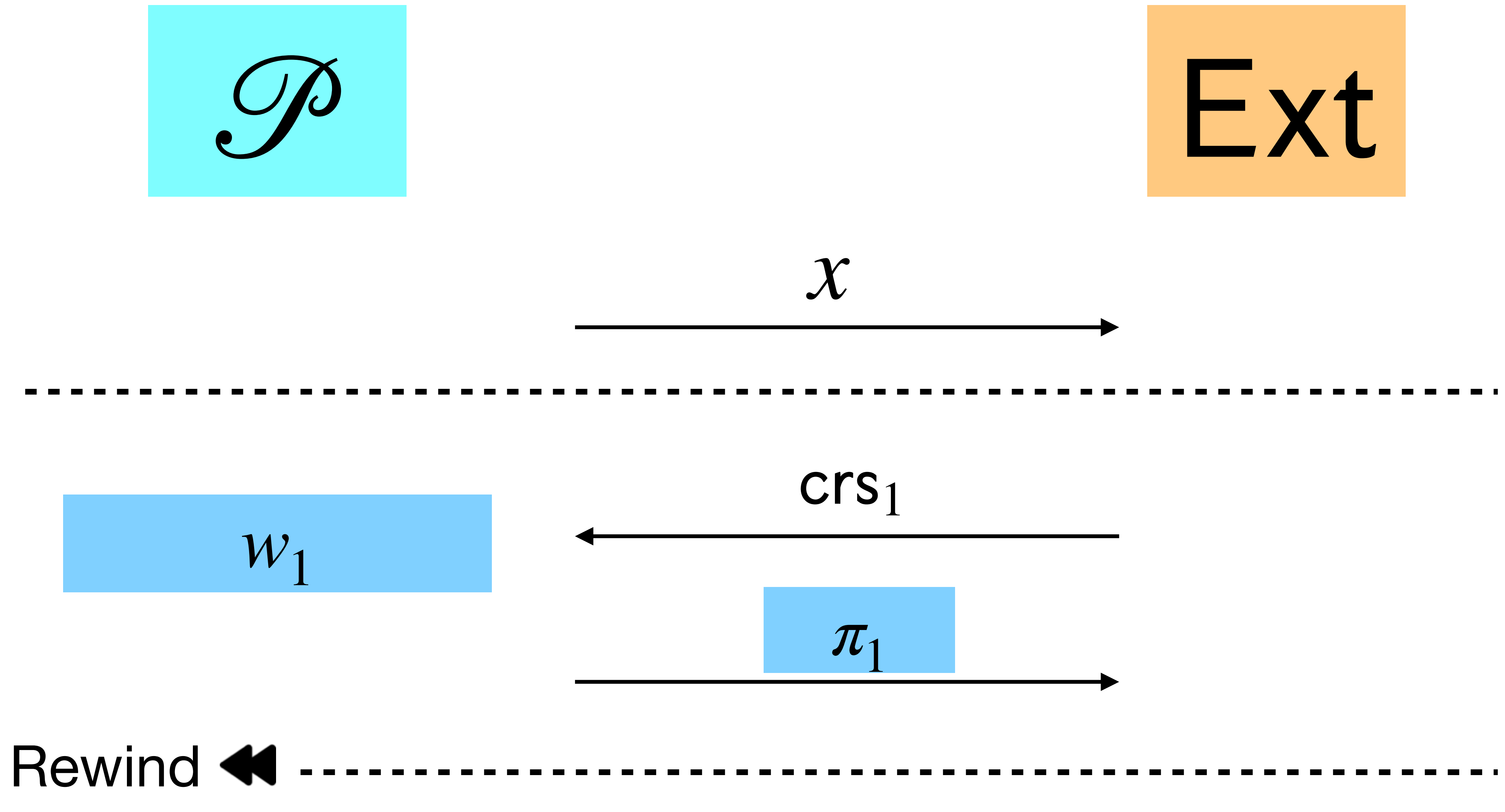
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!



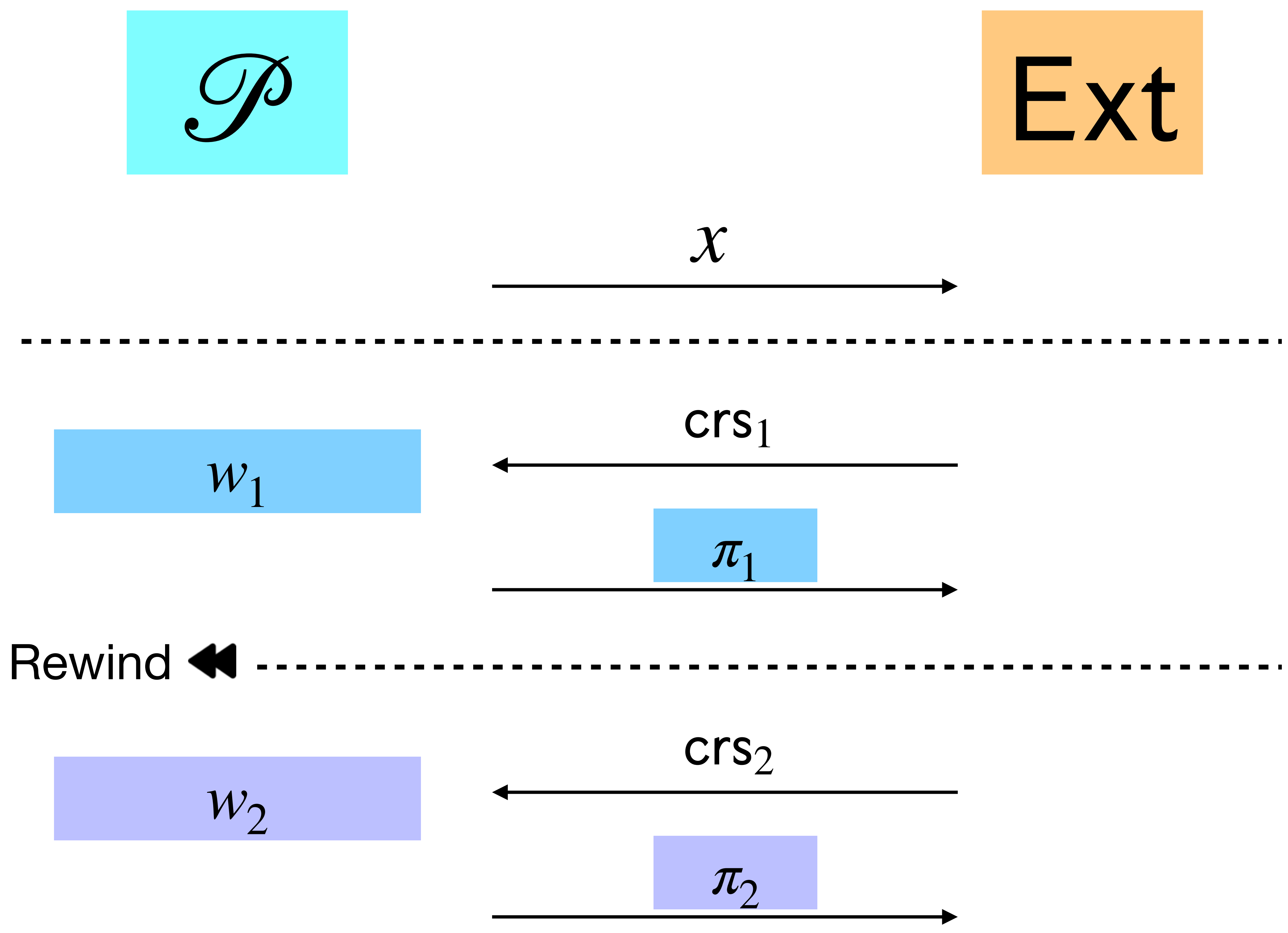
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!



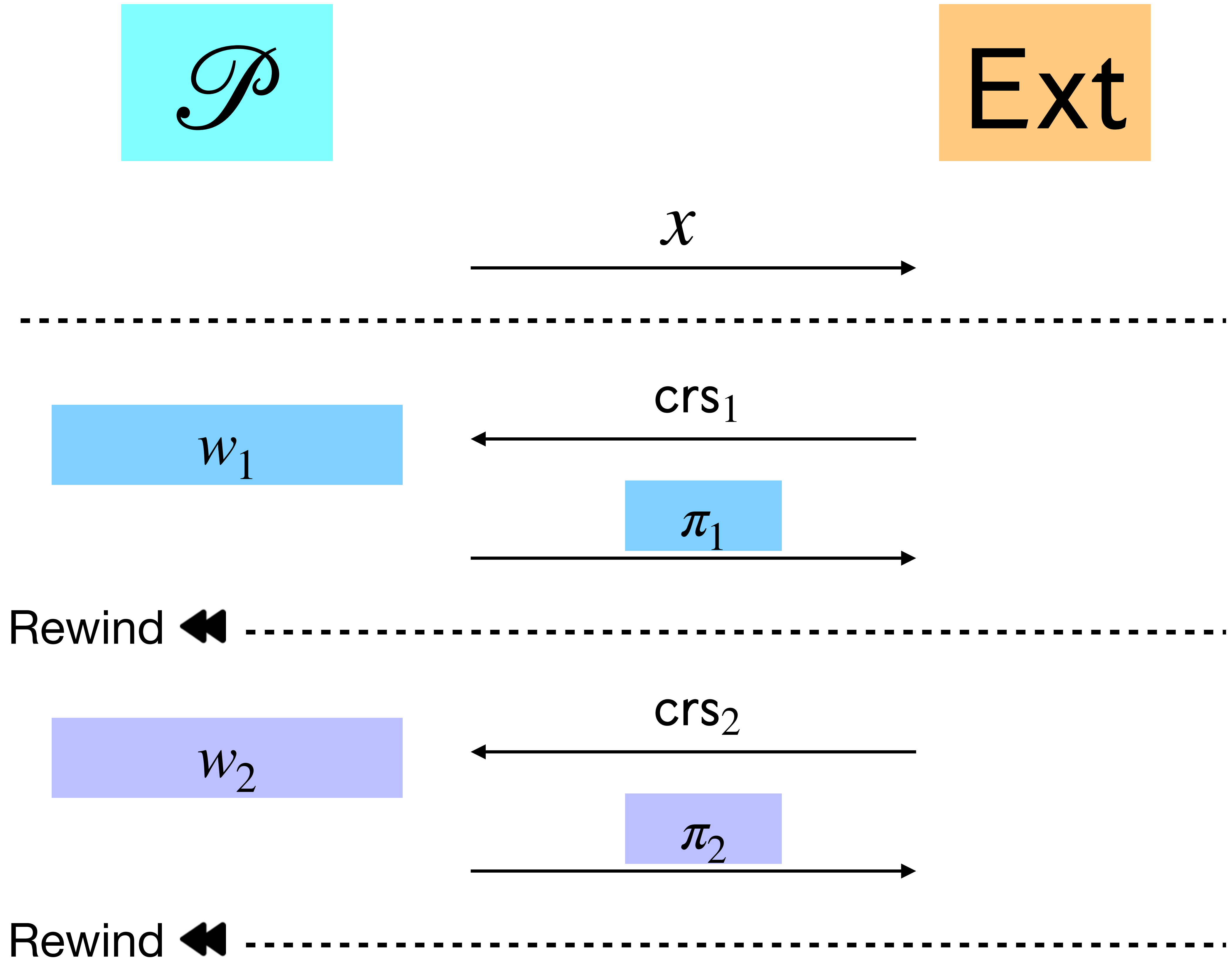
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
    - Each proof is a **“<u>small leakage</u>”** on a witness.
    - The prover might **use a different witness** each time!



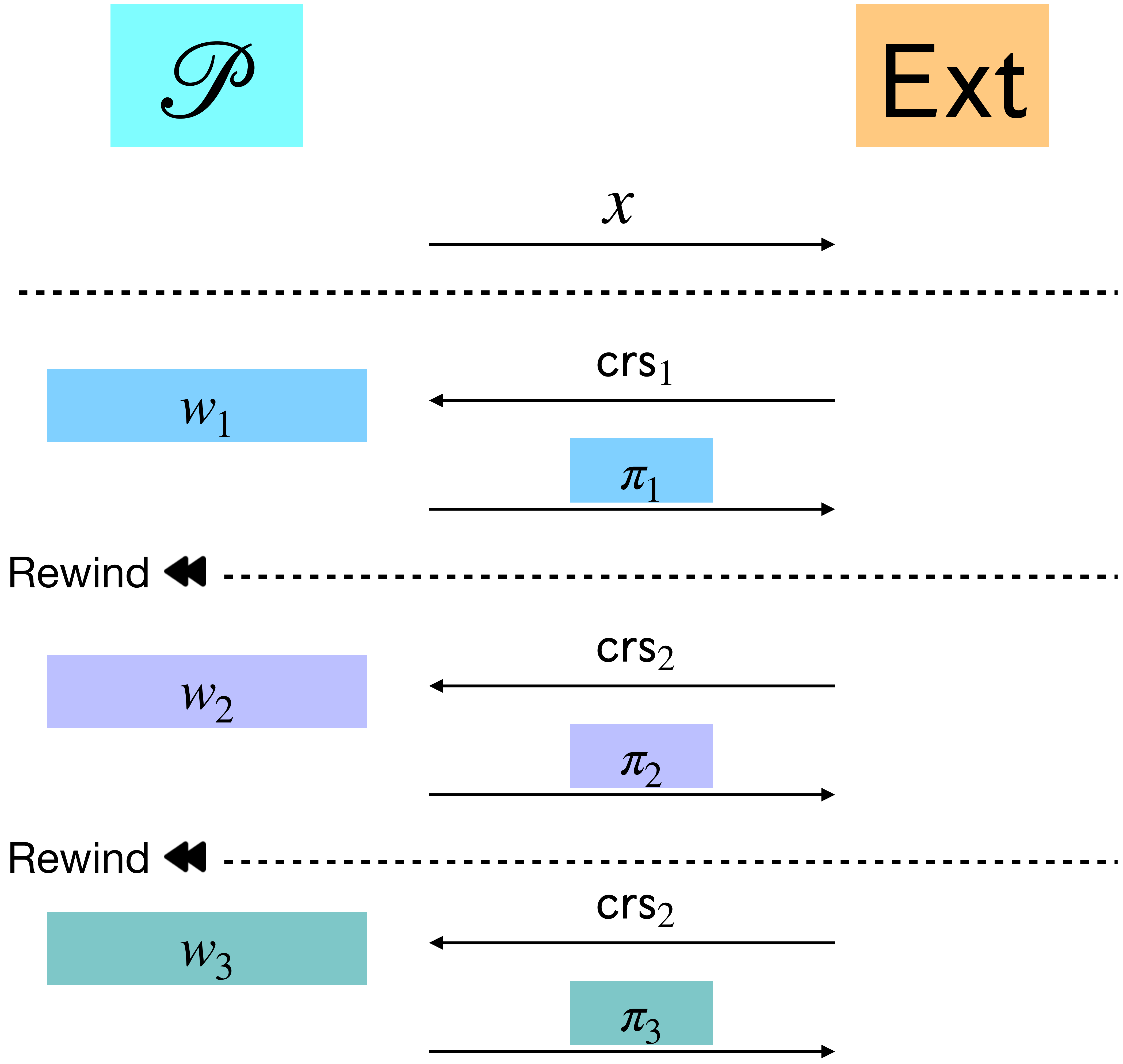
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!



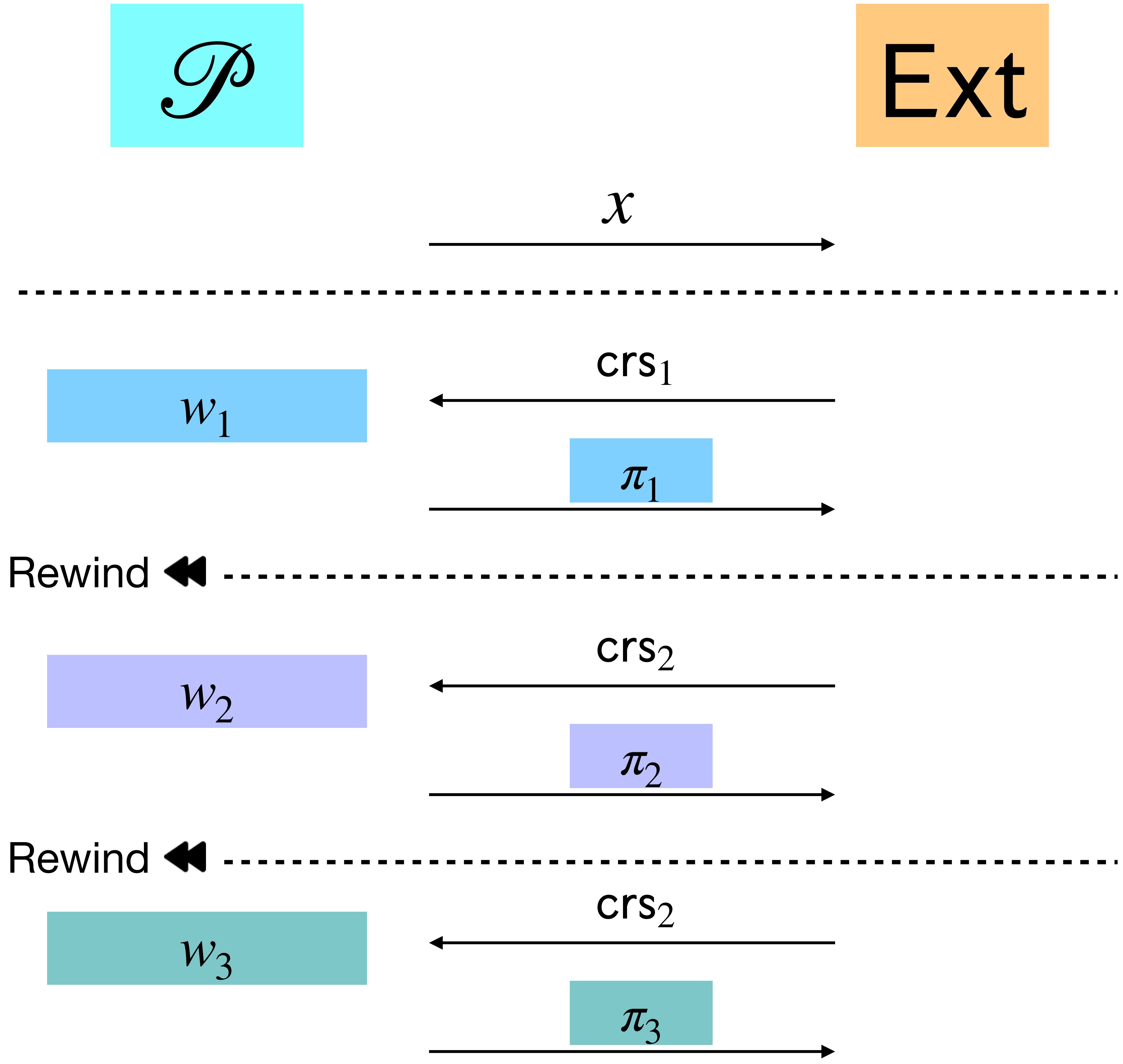
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"<u>small leakage</u>"** on a witness.
  - The prover might **use a different witness** each time!



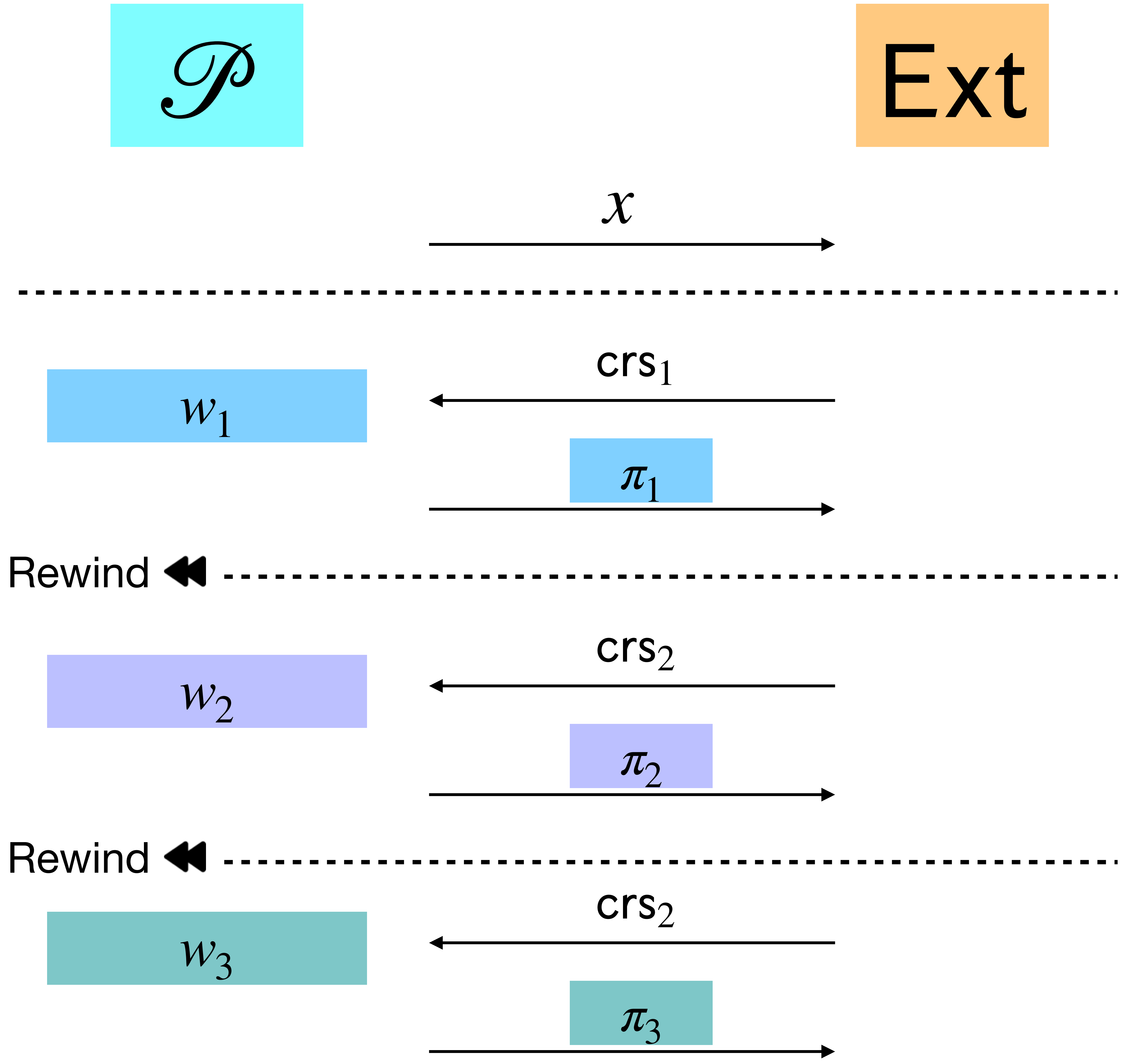
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a **"small leakage"** on a witness.
  - The prover might **use a different witness** each time!
  - Hard to piece together **a single witness**! Can be formalised in terms of **leakage resilience.**



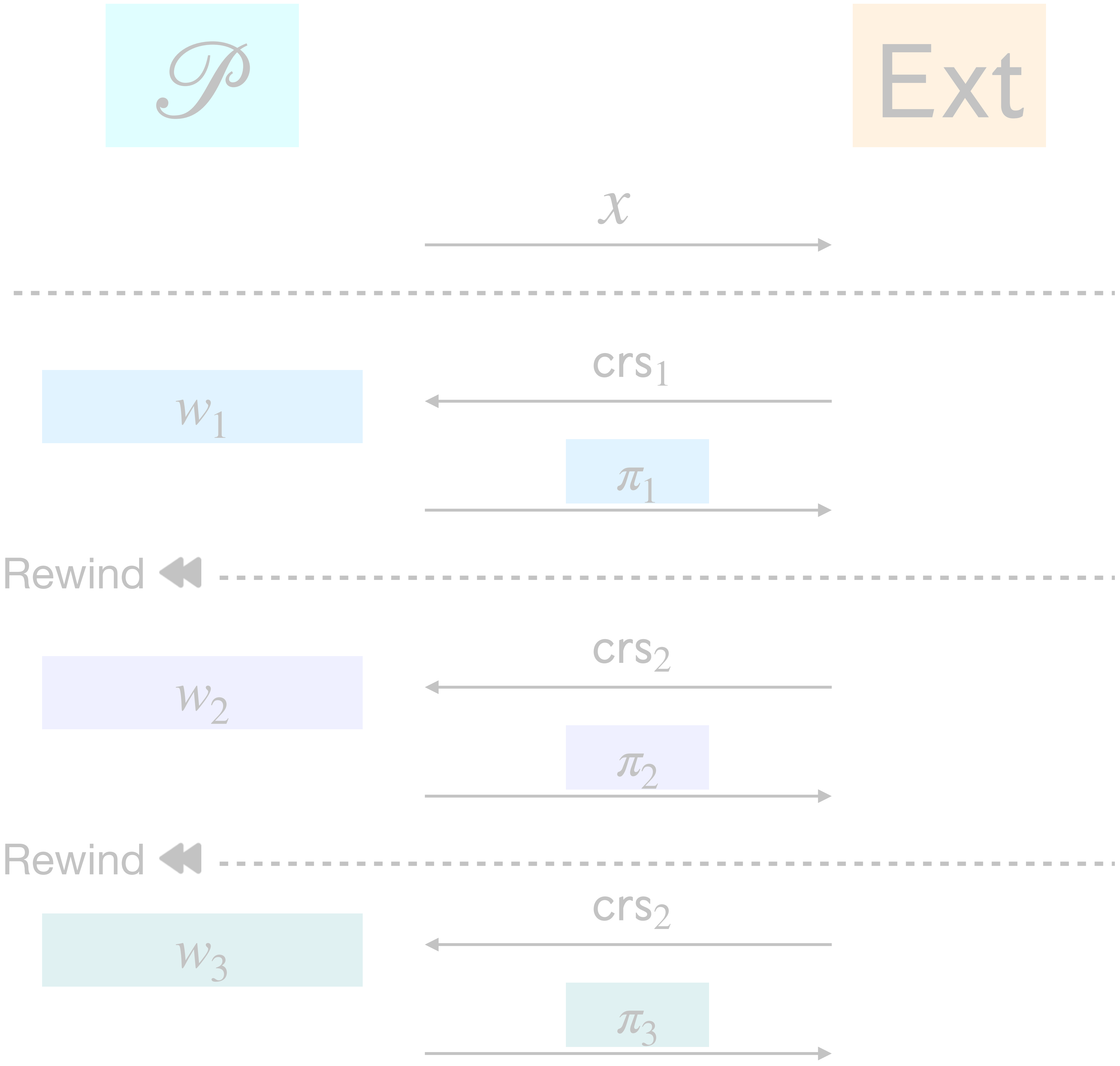
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
    - Each proof is a **"<u>small leakage</u>"** on a witness.
    - The prover might **use a different witness** each time!
    - Hard to piece together **a single witness**! Can be formalised in terms of **leakage resilience.**
- Impossibility doesn't hold for UP!



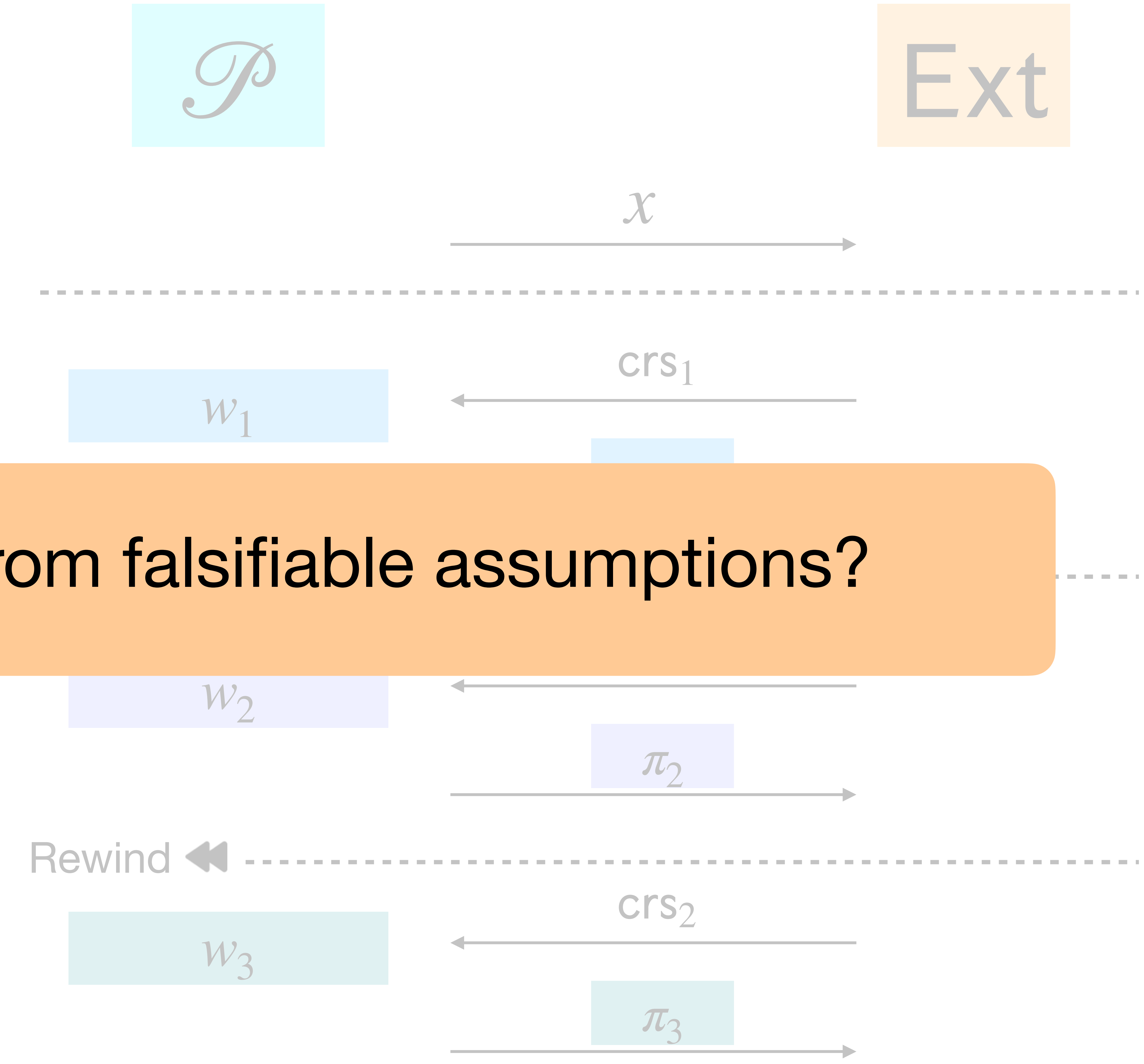
Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
    - Each proof is a **"small leakage"** on a witness.
    - The prover might **use a different witness** each time!
    - Hard to piece together **a single witness**! Can be formalised in terms of **leakage resilience.**
- Impossibility doesn't hold for UP!



Barrier to SNARKs for NP

- Black-box extraction is impossible for all of NP from falsifiable assumptions! [CGKS23]
  - Each proof is a "**small leakage**" on a witness.
  - Hard to piece together **a single witness**! Can be formalised in terms of **leakage resilience.**
- Impossibility doesn't hold for UP!



Qn: Can we build SNAR**K**s for UP from falsifiable assumptions?

SNARG to SNARK for UP

Theorem 2: Assuming polynomial hardness of LWE, an **adaptively sound SNARG** for **UP** can be used to construct an adaptively sound **SNARK** for **UP**, while preserving **zero-knowledge**.

SNARG to SNARK for UP

Theorem 2: Assuming polynomial hardness of LWE, an **adaptively sound SNARG** for **UP** can be used to construct an adaptively sound **SNARK** for **UP**, while preserving **zero-knowledge**.

- Our transformation follows [CGKS23] who show a similar transformation from SNARG for **NP** to SNARK for **UP**.

SNARG to SNARK for UP

Theorem 2: Assuming polynomial hardness of LWE, an **adaptively sound SNARG** for **UP** can be used to construct an adaptively sound **SNAR<u>K</u>** for **UP**, while preserving **zero-knowledge**.

- Our transformation follows [CGKS23] who show a similar transformation from SNARG for **NP** to SNARK for **UP**.
- We also correct some issues in their work:

SNARG to SNARK for UP

Theorem 2: Assuming polynomial hardness of LWE, an **adaptively sound SNARG** for **UP** can be used to construct an adaptively sound **SNARK** for **UP**, while preserving **zero-knowledge**.

- Our transformation follows [CGKS23] who show a similar transformation from SNARG for **NP** to SNARK for **UP**.
- We also correct some issues in their work:
  - Their transformation (as is) is not zero-knowledge and requires **adaptive** SNARGs for NP.

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathcal{P}^*$

$\mathcal{V}$

Non-Adaptive Soundness

$\mathcal{P}^*$

$\mathcal{V}$

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathcal{P}^*$

$\mathcal{V}$

Common Reference String (crs)

Non-Adaptive Soundness

$\mathcal{P}^*$

$\mathcal{V}$

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathscr{P}^*$

$\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{x^*, \pi^*}$

Non-Adaptive Soundness

$\mathscr{P}^*$

$\mathscr{V}$

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathscr{P}^*$ $\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{x^*, \pi^*}$

Non-Adaptive Soundness

$\mathscr{P}^*$ $\xrightarrow{x^* \notin L}$ $\mathscr{V}$

Adaptive vs. Non-Adaptive Soundness

<u>*Adaptive Soundness*</u>

$\mathscr{P}^*$

$\mathscr{V}$

Common Reference String (crs)

$$\xrightarrow{x^*,\ \pi^*}$$

<u>*Non-Adaptive Soundness*</u>

$\mathscr{P}^*$

$$\xrightarrow{x^* \notin L}$$

$\mathscr{V}$

Common Reference String (crs)

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathscr{P}^*$ $\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{x^*,\ \pi^*}$

Non-Adaptive Soundness

$\mathscr{P}^*$ $\xrightarrow{x^* \notin L}$ $\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{\pi^*}$

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathscr{P}^*$

$\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{x^*, \pi^*}$

Non-Adaptive Soundness

$\mathscr{P}^*$

$\xrightarrow{x^* \notin L}$

$\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{\pi^*}$

(Informal) Gentry-Wichs Barrier: If a language takes time 2^{n^δ} to "decide", there is no 2^{n^ϵ} for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

Adaptive vs. Non-Adaptive Soundness

Adaptive Soundness

$\mathscr{P}^*$ $\qquad$ $\mathscr{V}$

Common Reference String (crs)

$\xrightarrow{x^*,\ \pi^*}$

Intuition: How can you tell if soundness was broken? Need to decide if $x^* \in L$!

Non-Adaptive Soundness

$\mathscr{P}^* \xrightarrow{x^* \notin L} \mathscr{V}$

Common Reference String (crs)

$\xrightarrow{\pi^*}$

(Informal) Gentry-Wichs Barrier: If a language takes time 2^{n^δ} to "decide", there is no 2^{n^ϵ} for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

Falsifiable Assumptions

Falsifiable Assumptions

- An assumption is falsifiable if there exists an **efficient** challenger $\mathscr{C}$ that can decide if an adversary $\mathscr{A}$ "won" the game.

Falsifiable Assumptions

- An assumption is falsifiable if there exists an **efficient** challenger $\mathscr{C}$ that can decide if an adversary $\mathscr{A}$ "won" the game.

- An assumption usually is associated with a parameter $c \in [0,1]$ s.t. the assumption is considered "broken" if $\Pr[\mathscr{A} \text{ wins}] \geq c + \mathsf{negl}(\lambda)$.

Falsifiable Assumptions

- An assumption is falsifiable if there exists an **efficient** challenger $\mathscr{C}$ that can decide if an adversary $\mathscr{A}$ “won” the game.

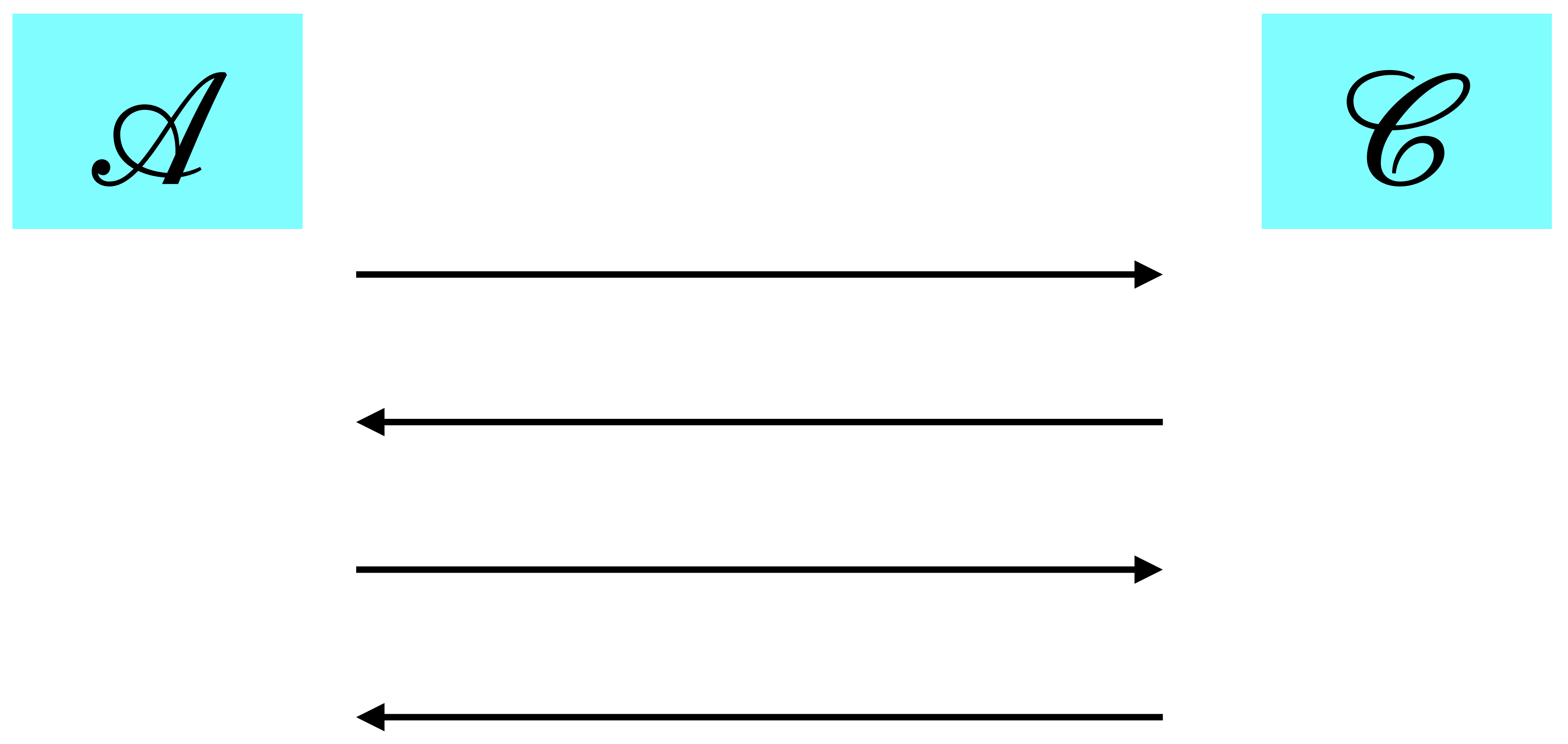
- An assumption usually is associated with a parameter $c \in [0,1]$ s.t. the assumption is considered “broken” if $\Pr[\mathscr{A} \text{ wins}] \geq c + \mathsf{negl}(\lambda)$.

$\mathscr{A}$ $\mathscr{C}$

Falsifiable Assumptions

- An assumption is falsifiable if there exists an **efficient** challenger $\mathscr{C}$ that can decide if an adversary $\mathscr{A}$ "won" the game.

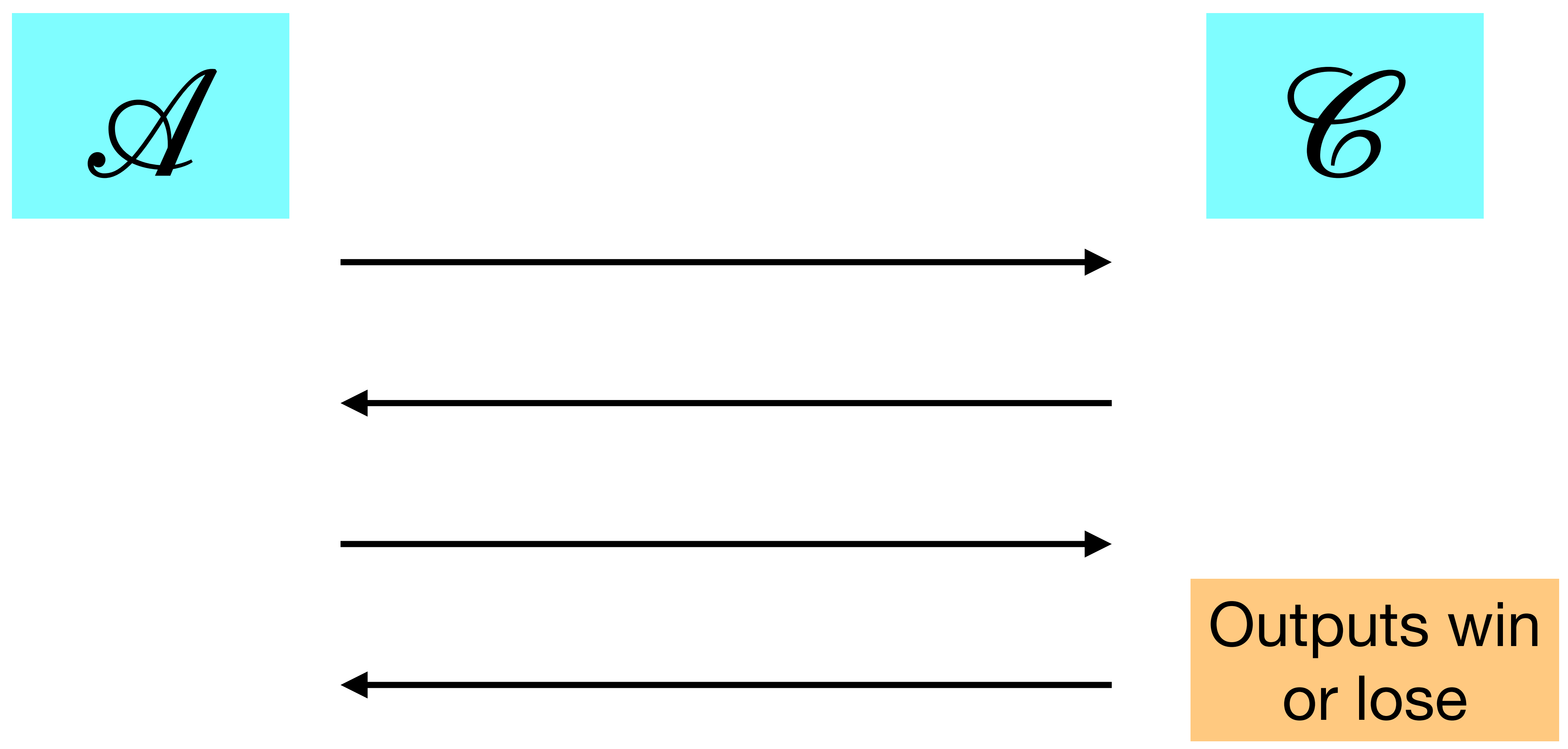
- An assumption usually is associated with a parameter $c \in [0,1]$ s.t. the assumption is considered "broken" if $\Pr[\mathscr{A} \text{ wins}] \geq c + \mathsf{negl}(\lambda)$.



Falsifiable Assumptions

- An assumption is falsifiable if there exists an **efficient** challenger $\mathscr{C}$ that can decide if an adversary $\mathscr{A}$ "won" the game.

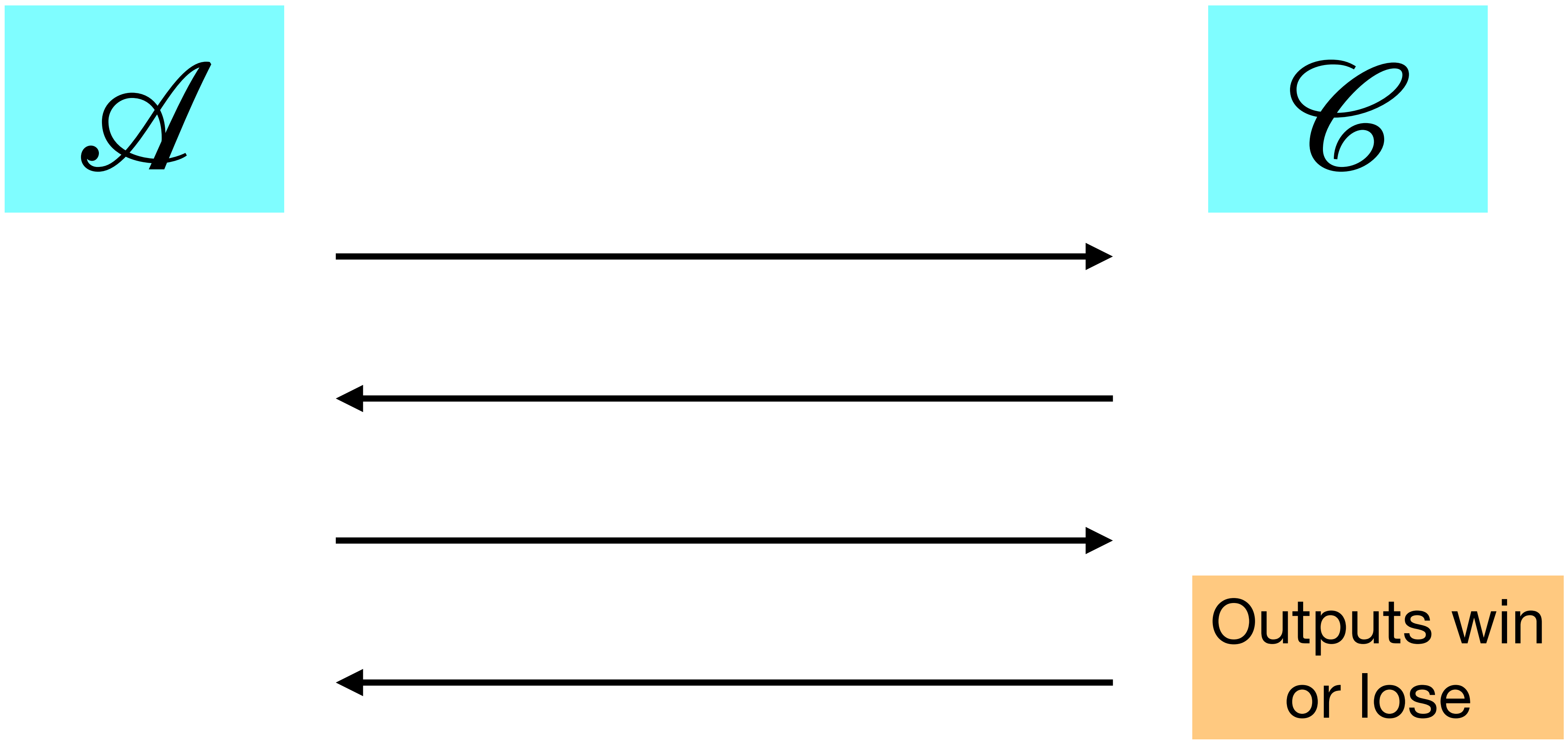
- An assumption usually is associated with a parameter $c \in [0,1]$ s.t. the assumption is considered "broken" if $\Pr[\mathscr{A} \text{ wins}] \geq c + \mathsf{negl}(\lambda)$.



Falsifiable Assumptions

- An assumption is falsifiable if there exists an **efficient** challenger $\mathscr{C}$ that can decide if an adversary $\mathscr{A}$ “won” the game.

- An assumption usually is associated with a parameter $c \in [0,1]$ s.t. the assumption is considered “broken” if $\Pr[\mathscr{A} \text{ wins}] \geq c + \mathsf{negl}(\lambda)$.



- E.g. Decision problems like DDH and LWE have parameter $c = 1/2$
- E.g. Search problems like OWF, DLOG have parameter $c = 0$

(Non-)Examples of Falsifiable Assumptions

(Non-)Examples of Falsifiable Assumptions

- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.

(Non-)Examples of Falsifiable Assumptions

- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.
- **"Eg" 2:** For all $C_1 \equiv C_2$, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.

(Non-)Examples of Falsifiable Assumptions

- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.

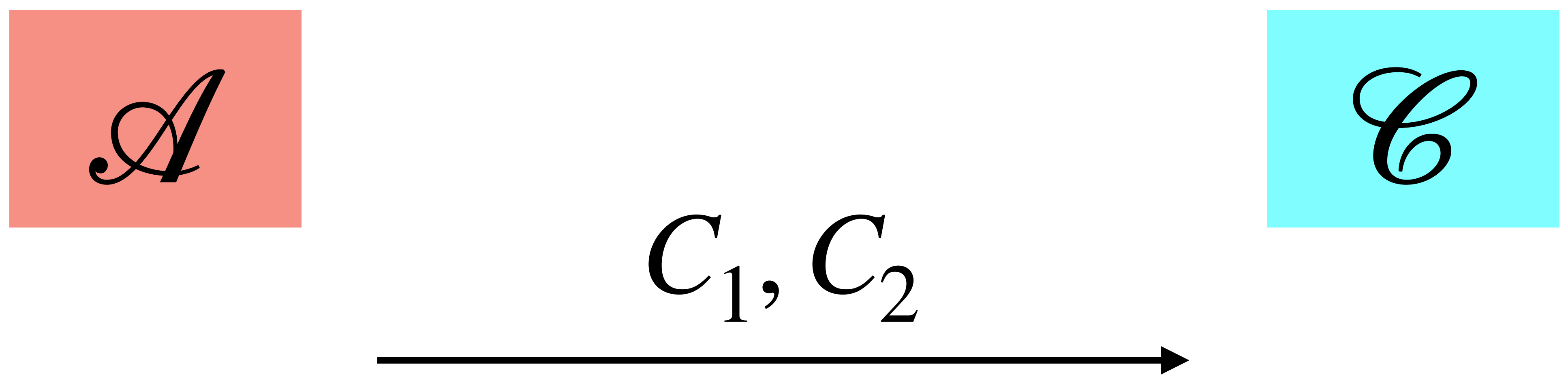
- **“Eg” 2:** <u>For all $C_1 \equiv C_2$</u>, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.

$\mathscr{A}$ $\mathscr{C}$

(Non-)Examples of Falsifiable Assumptions

- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.

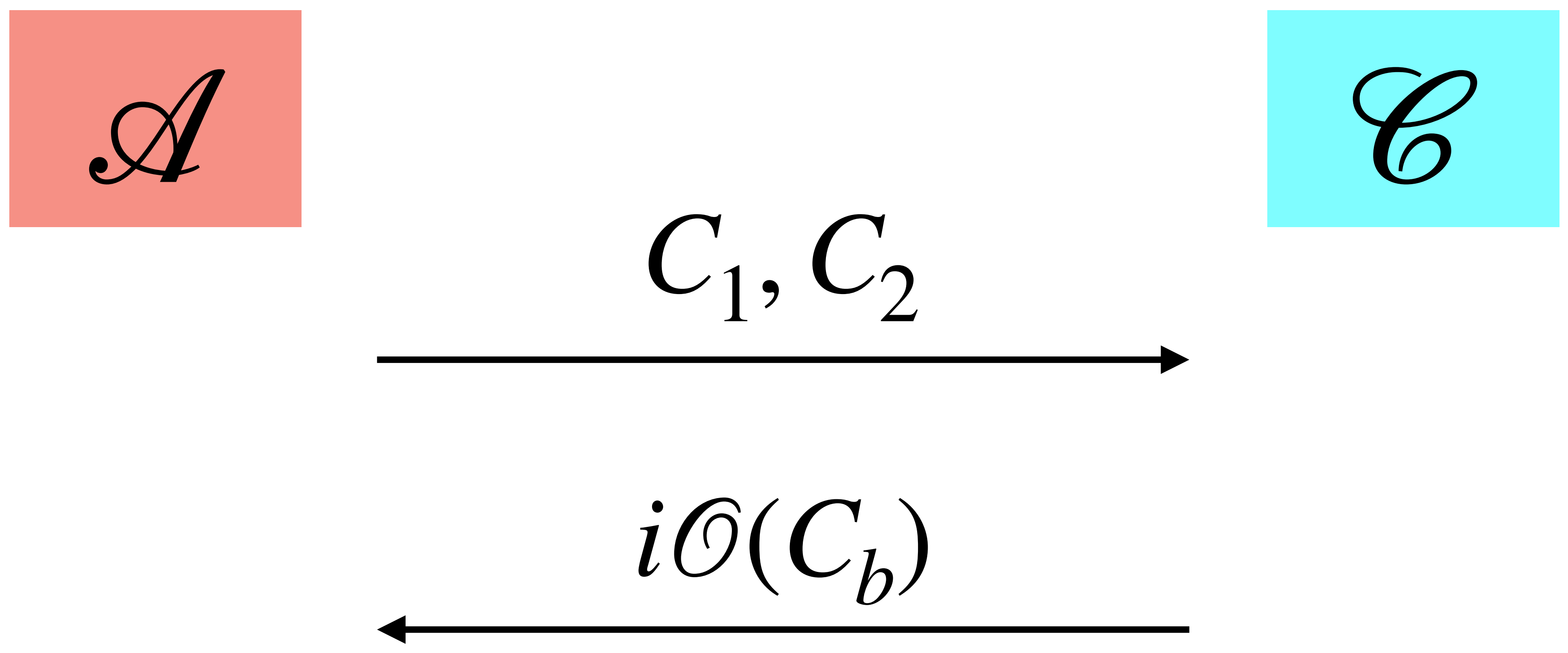
- **“Eg” 2:** ~~For all $C_1 \equiv C_2$~~, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.



(Non-)Examples of Falsifiable Assumptions

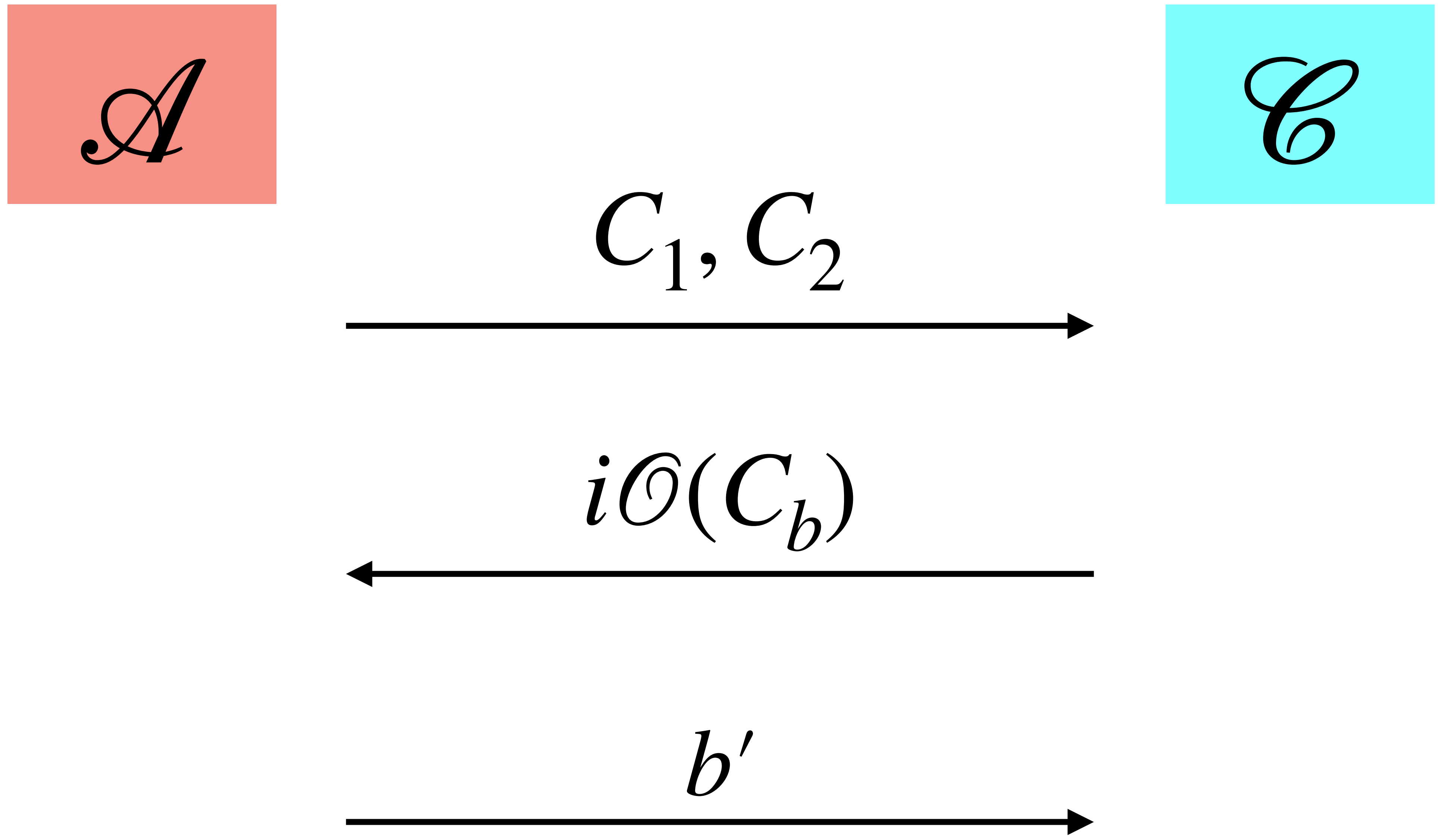
- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.

- **“Eg” 2:** ~~For all $C_1 \equiv C_2$~~, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.



(Non-)Examples of Falsifiable Assumptions

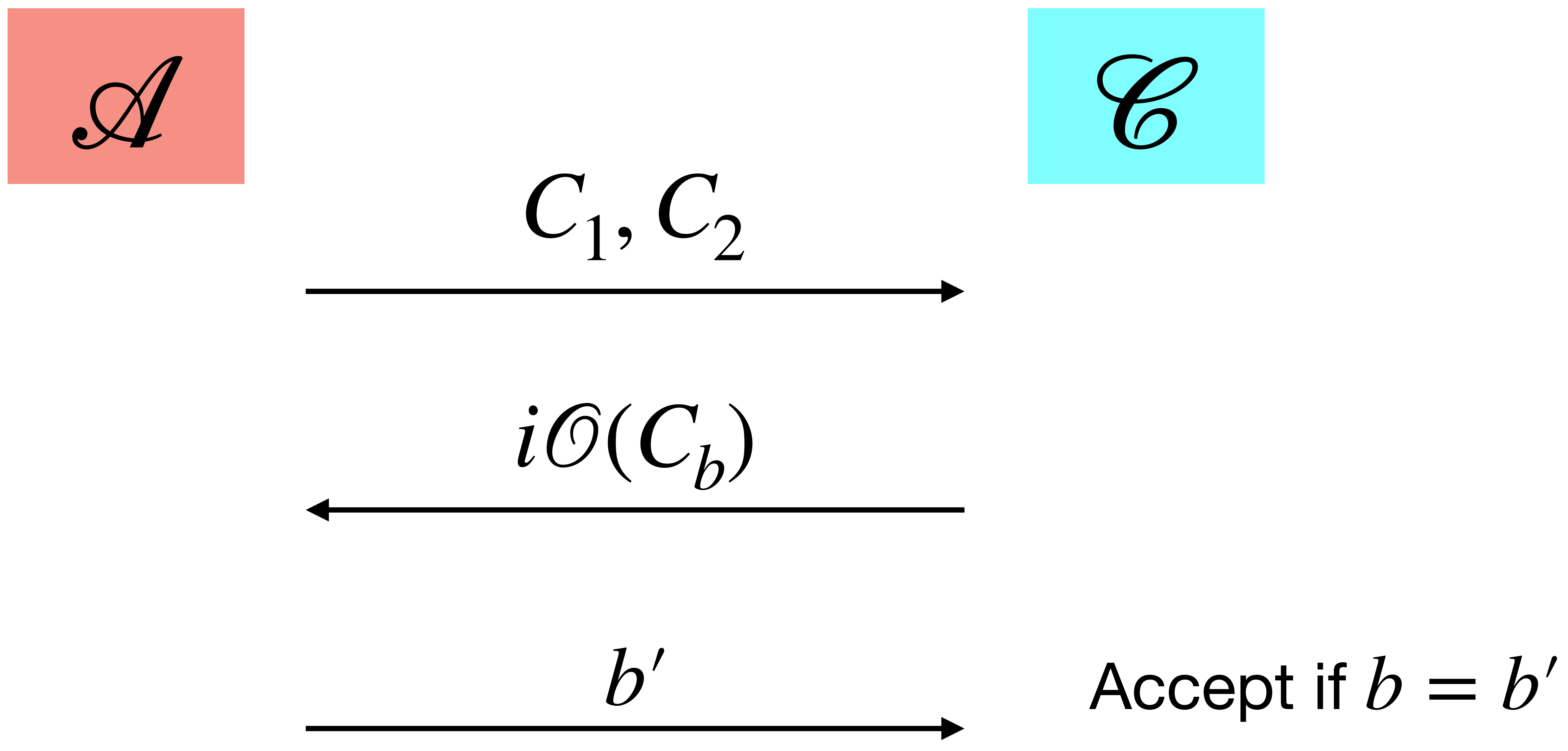
- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.
- **“Eg” 2:** ~~For all $C_1 \equiv C_2$~~, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.



(Non-)Examples of Falsifiable Assumptions

- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.

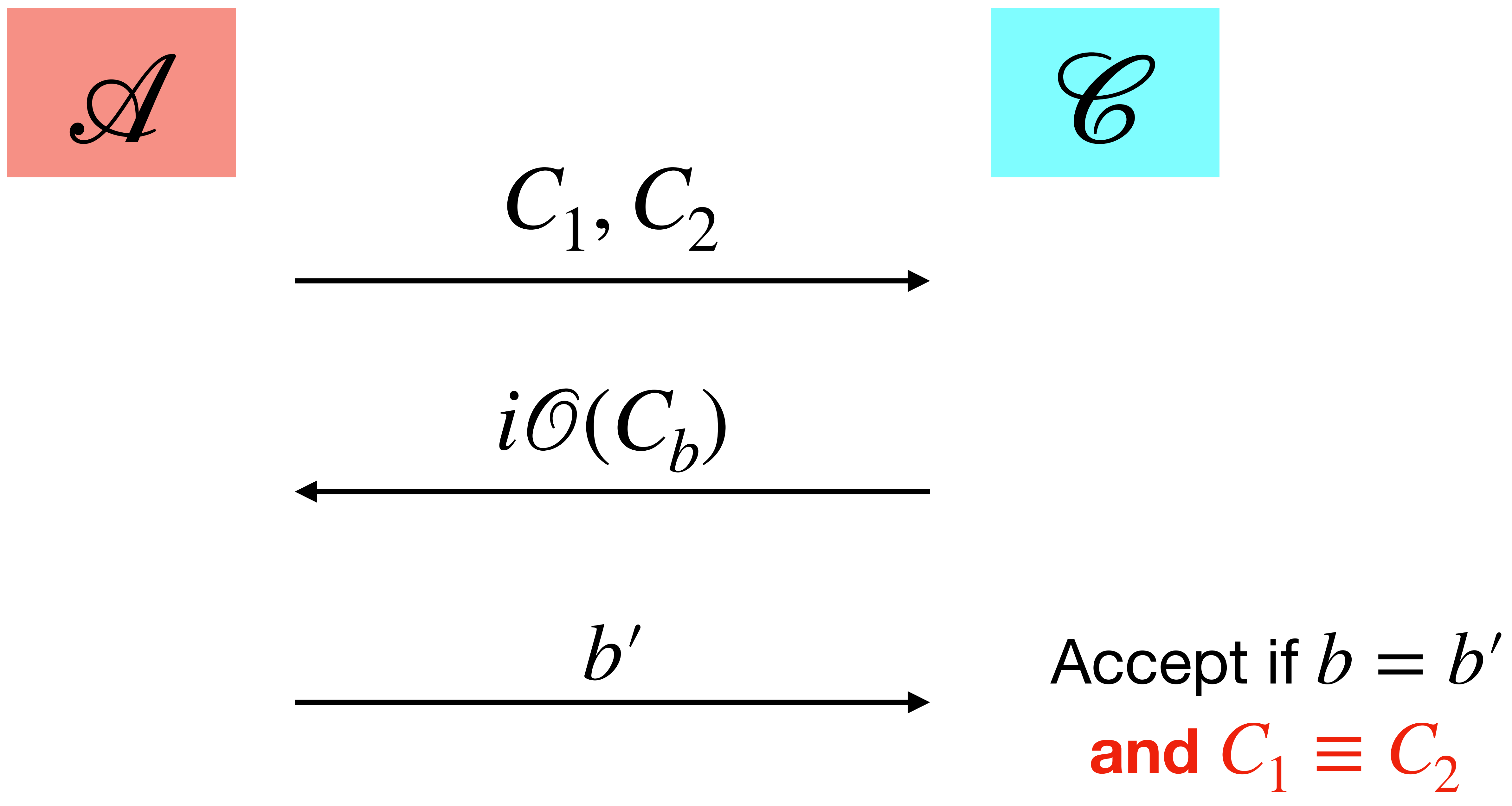
- **"Eg" 2:** ~~For all $C_1 \equiv C_2$~~, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.



(Non-)Examples of Falsifiable Assumptions

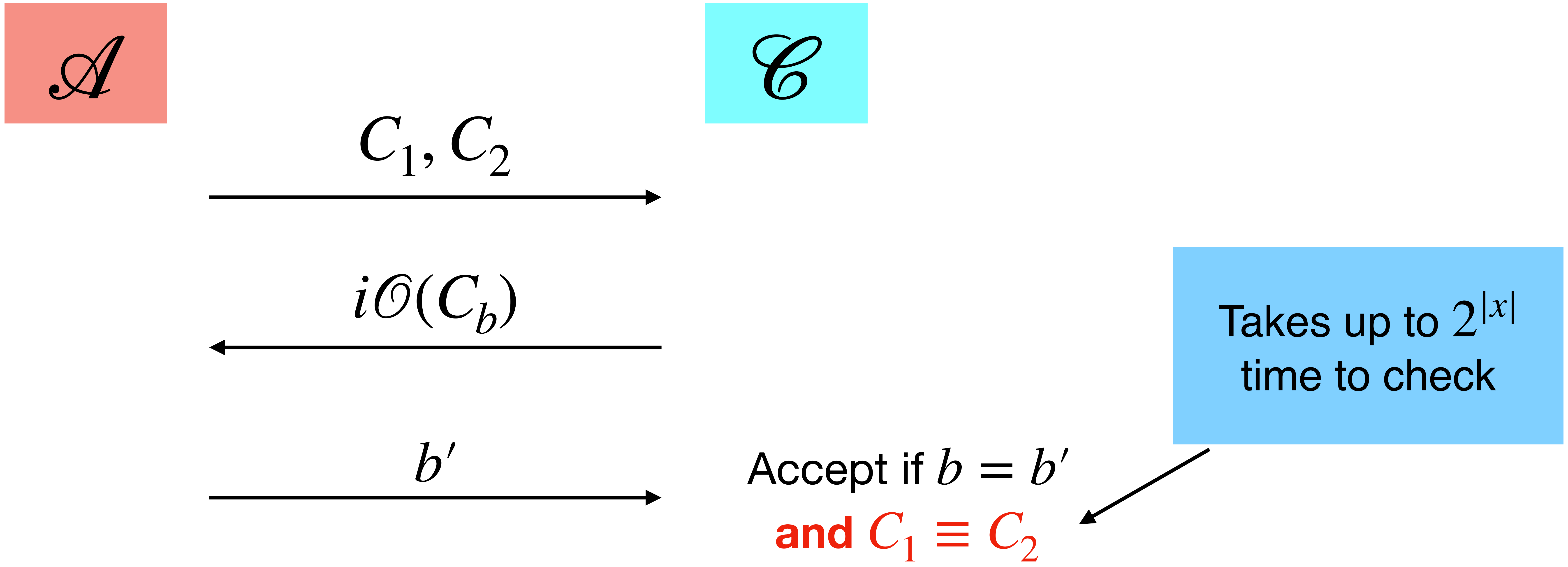
- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.

- **“Eg” 2:** ~~For all~~ $\cancel{C_1 \equiv C_2}$, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.

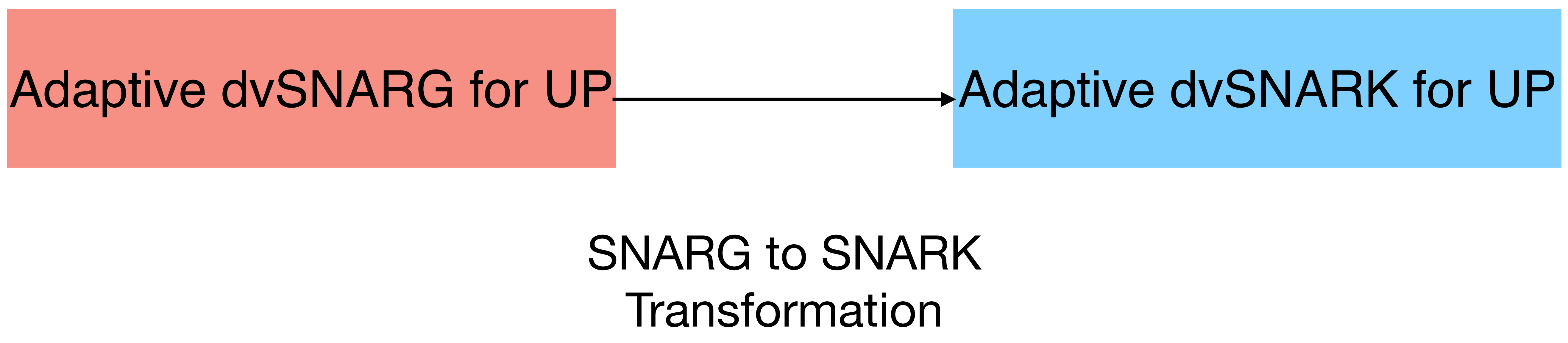


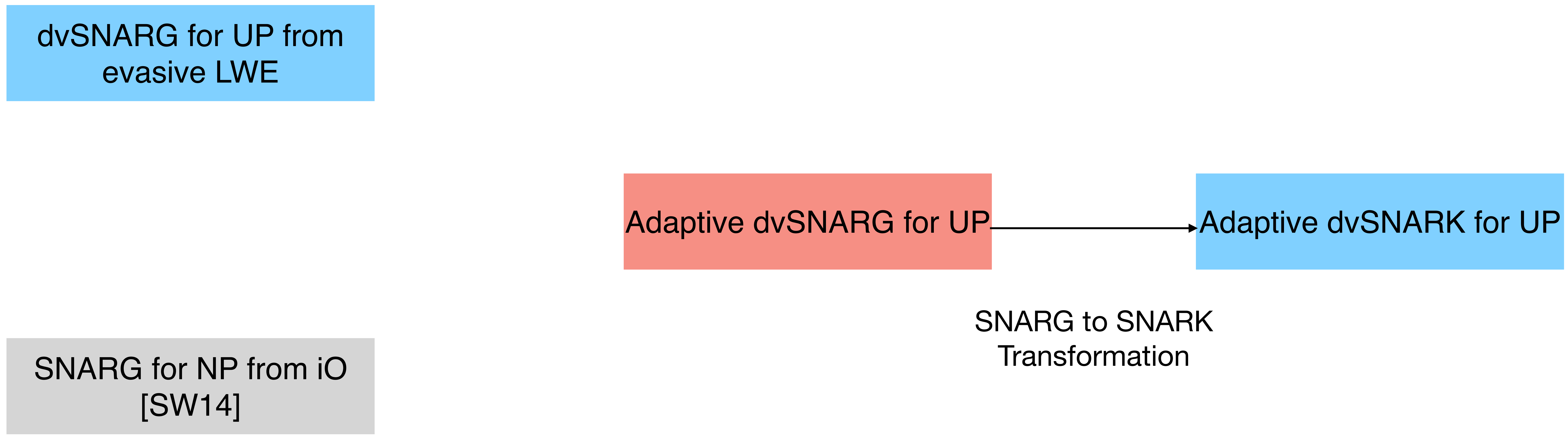
(Non-)Examples of Falsifiable Assumptions

- **Eg 1:** Decisional LWE: $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$.
- **“Eg” 2:** For all $C_1 \equiv C_2$, $i\mathcal{O}(C_1) \approx_c i\mathcal{O}(C_2)$.



Adaptive dvSNARK for UP


Adaptive dvSNARG for UP
Adaptive dvSNARK for UP
SNARG to SNARK
Transformation



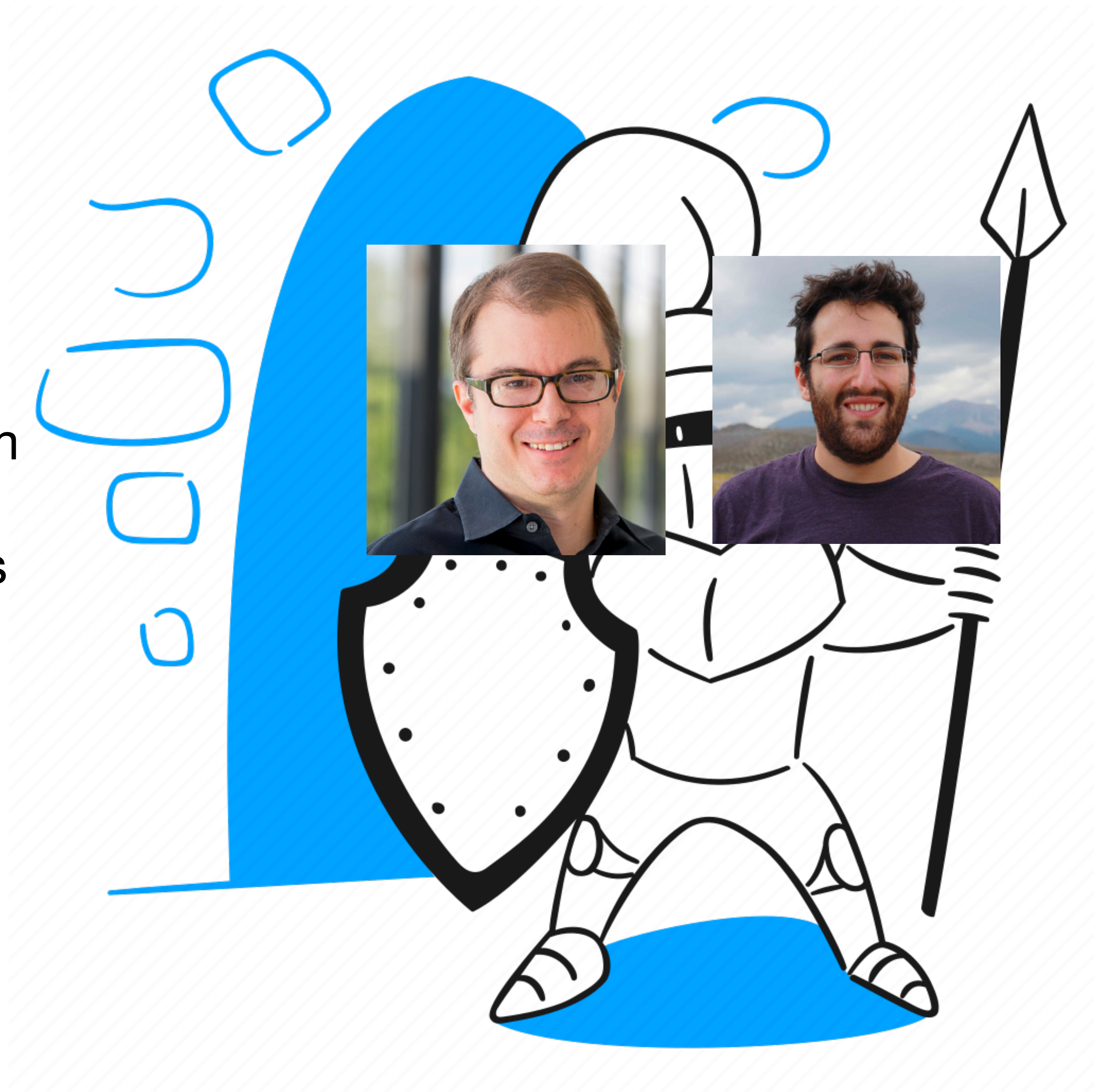
dvSNARG for UP from evasive LWE
Adaptive dvSNARG for UP
Adaptive dvSNARK for UP
SNARG to SNARK Transformation
SNARG for NP from iO [SW14]

Gentry-Wichs Barrier

SNARGs from
Falsifiable
Assumptions

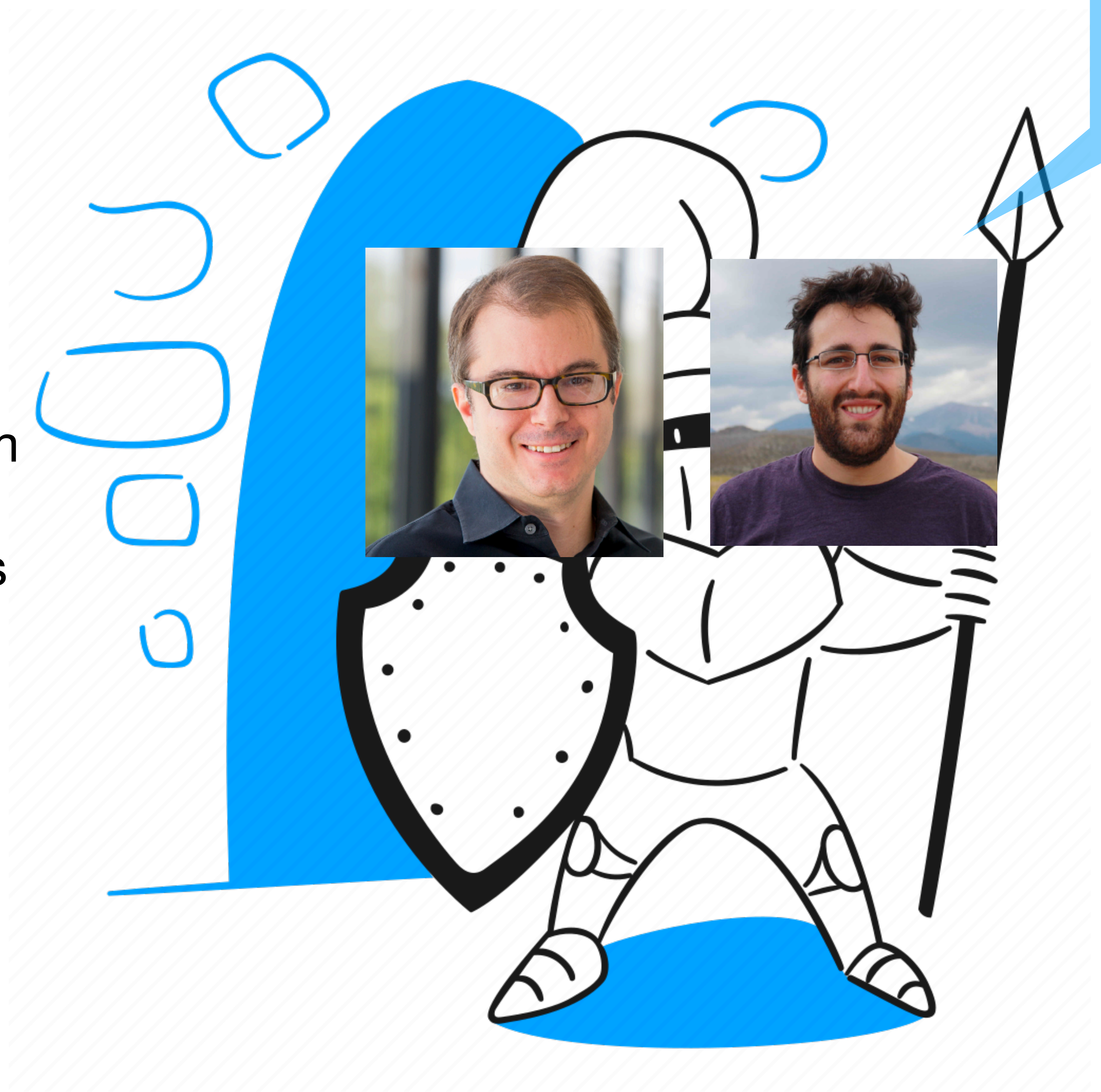
Gentry-Wichs Barrier

SNARGs from Falsifiable Assumptions



Gentry-Wichs Barrier

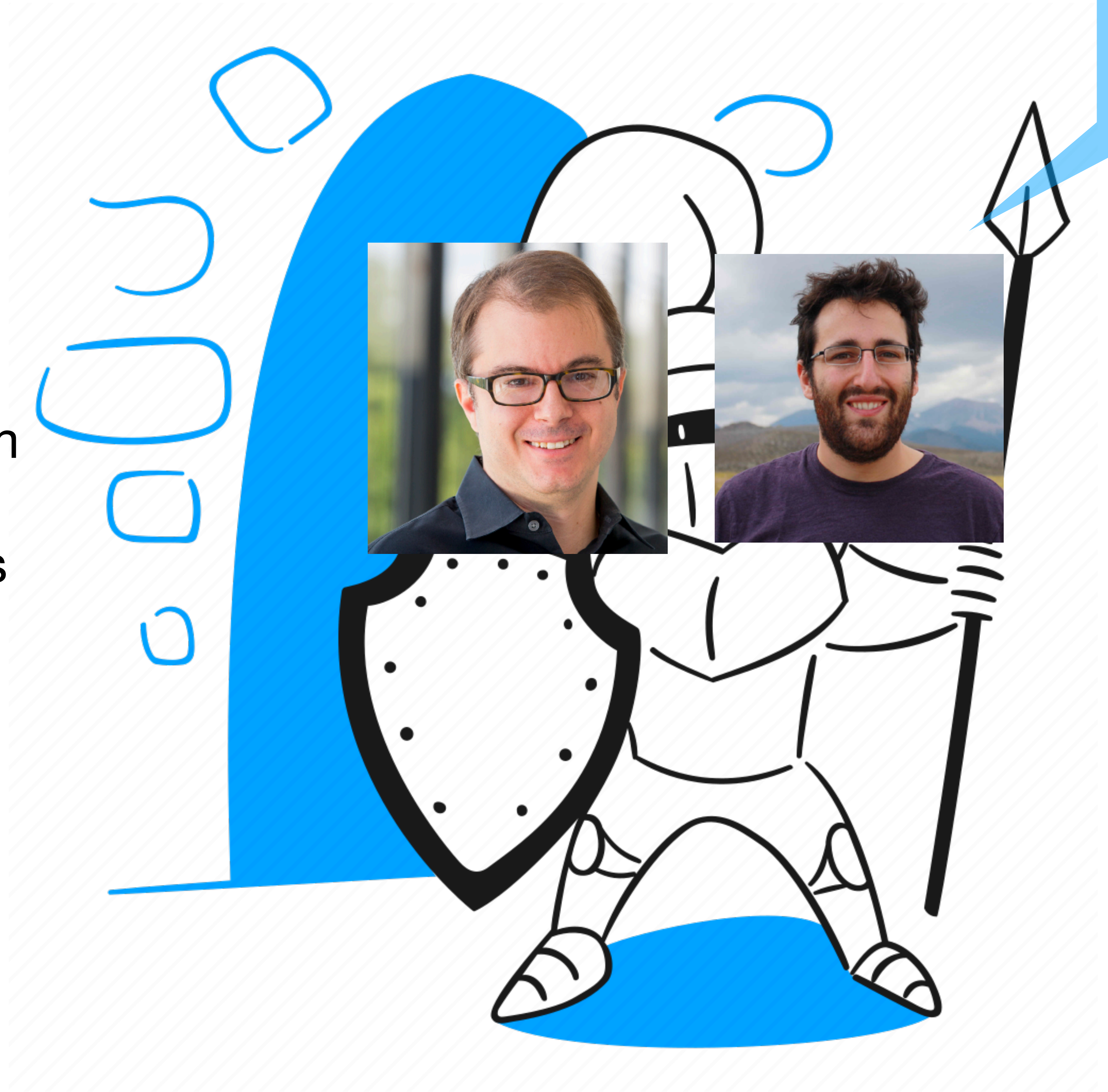
SNARGs from Falsifiable Assumptions



What is the password?

Gentry-Wichs Barrier

SNARGs from Falsifiable Assumptions

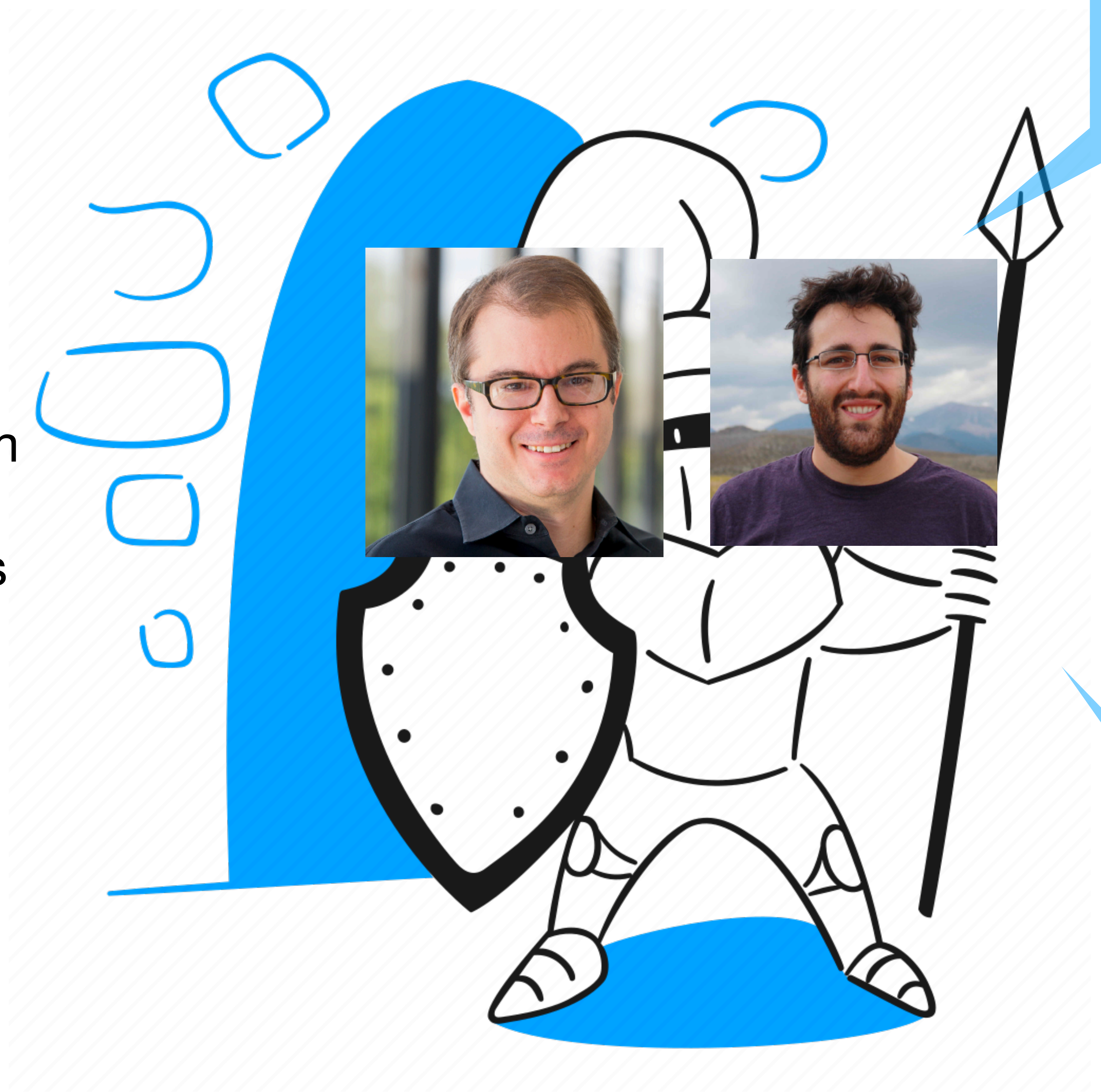


What is the password?

Exponential time reduction

Gentry-Wichs Barrier

SNARGs from Falsifiable Assumptions



What is the password?

Exponential time reduction

You may proceed…
with caution

Gentry-Wichs Barrier

Gentry-Wichs Barrier

(Informal) Gentry-Wichs Barrier: If a language takes time 2^{n^δ} to "decide", there is no 2^{n^ϵ} for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

Gentry-Wichs Barrier

(Informal) Gentry-Wichs Barrier: If a language takes time 2^{n^δ} to "decide", there is no 2^{n^ϵ} for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

- **One interpretation:** One has to rely on sub-exponential hardness assumptions to obtain adaptive soundness.

Gentry-Wichs Barrier

(Informal) Gentry-Wichs Barrier: If a language takes time 2^{n^δ} to "decide", there is no 2^{n^ϵ} for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

- **One interpretation:** One has to rely on sub-exponential hardness assumptions to obtain adaptive soundness.

- **Issue:** It is not clear that one can maintain succinctness while doing this.

Gentry-Wichs Barrier

(Informal) Gentry-Wichs Barrier: If a language takes time 2^{n^δ} to “decide”, there is no 2^{n^ϵ} for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

- **One interpretation:** One has to rely on <u>sub-exponential hardness assumptions</u> to obtain adaptive soundness.

- **Issue:** It is not clear that one can maintain <u>succinctness</u> while doing this.

  - Eg. Directly applying complexity-leveraging to the Sahai-Waters SNARG does not maintain succinctness.

Gentry-Wichs Barrier

(Informal) Gentry-Wichs Barrier: If a language takes time $2^{n^{\delta}}$ to "decide", there is no $2^{n^{\epsilon}}$ for $\epsilon < \delta$ black-box reduction to *falsifiable assumptions* that shows **adaptive soundness.**

- **One interpretation:** One has to rely on sub-exponential hardness assumptions to obtain adaptive soundness.
- **Issue:** It is not clear that one can maintain succinctness while doing this.
  - Eg. Directly applying complexity-leveraging to the Sahai-Waters SNARG does not maintain succinctness.
- No known constructions of adaptively sound SNARGs from **falsifiable assumptions** (prior to Feb 2024*).

*Feb 2024: [WW24], [**M**PV24], [WZ24]

Our work

Our work

- **Theorem 3.** We show that our dvSNARG for UP is **adaptively sound**.

Our work

- **Theorem 3.** We show that our dvSNARG for UP is <u>**adaptively sound**</u>.

- **Theorem 4.** Any “Sahai-Waters”-like sub-exponentially sound SNARG can be made <u>**adaptively sound**</u> in the <u>**designated verifier**</u> setting with no additional assumptions.

Our work

- **Theorem 3.** We show that our dvSNARG for UP is **adaptively sound**.

- **Theorem 4.** Any “Sahai-Waters”-like sub-exponentially sound SNARG can be made **adaptively sound** in the **designated verifier** setting with no additional assumptions.

- **Corollary:** Adaptively sound dv-zkSNAR**K**s for UP from either

Our work

- **Theorem 3.** We show that our dvSNARG for UP is **adaptively sound**.
- **Theorem 4.** Any "Sahai-Waters"-like sub-exponentially sound SNARG can be made **adaptively sound** in the **designated verifier** setting with no additional assumptions.
- **Corollary:** Adaptively sound dv-zkSNAR**K**s for UP from either
  - LWE and evasive LWE

Our work

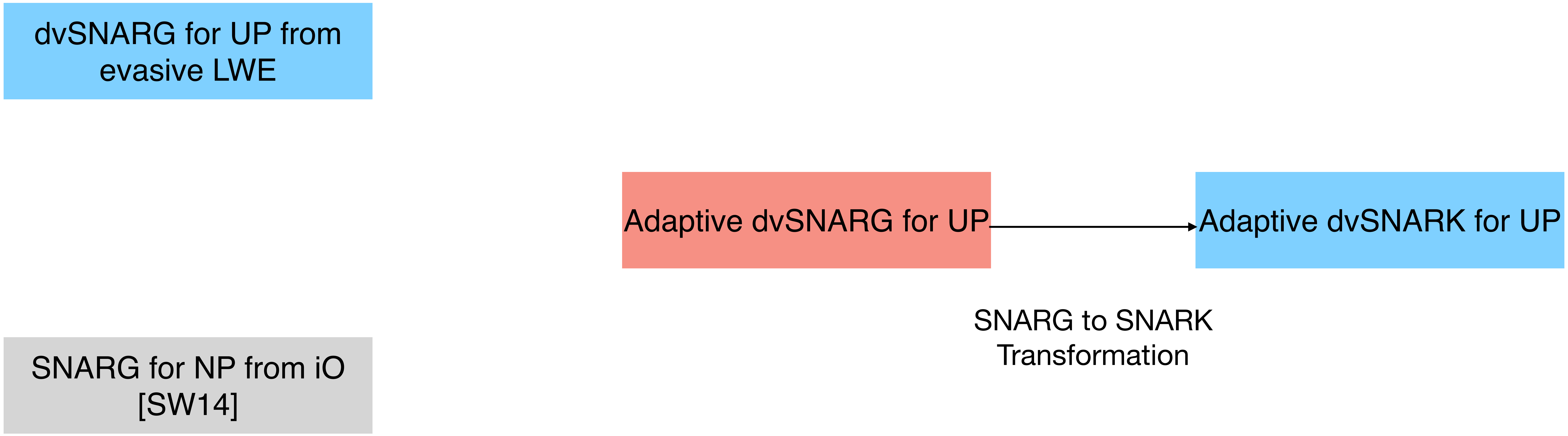
- **Theorem 3.** We show that our dvSNARG for UP is **adaptively sound**.

- **Theorem 4.** Any "Sahai-Waters"-like sub-exponentially sound SNARG can be made **adaptively sound** in the **designated verifier** setting with no additional assumptions.

- **Corollary:** Adaptively sound dv-zkSNAR**K**s for UP from either

  - LWE and evasive LWE

  - LWE, subexponentially-secure iO, subexponentially-secure OWF


dvSNARG for UP from evasive LWE
Adaptive dvSNARG for UP
Adaptive dvSNARK for UP
SNARG to SNARK Transformation
SNARG for NP from iO [SW14]

dvSNARG for UP from evasive LWE

SNARG for NP from iO [SW14]

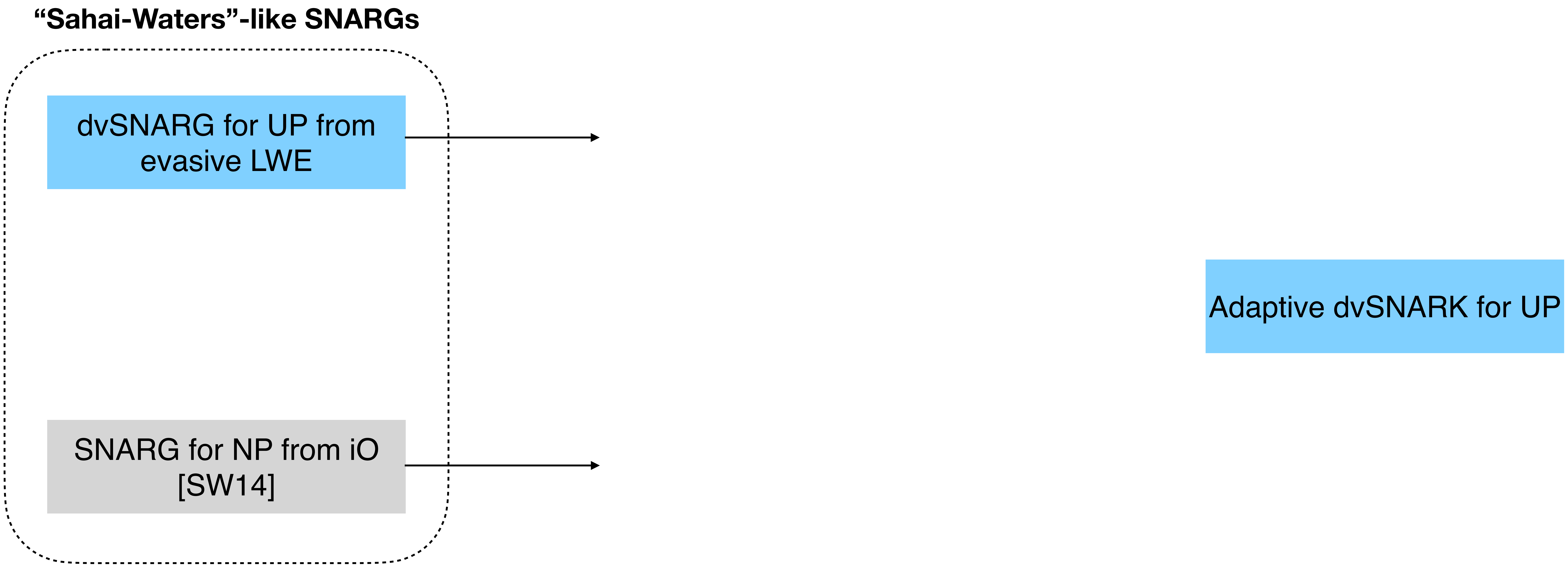
Adaptive dvSNARK for UP

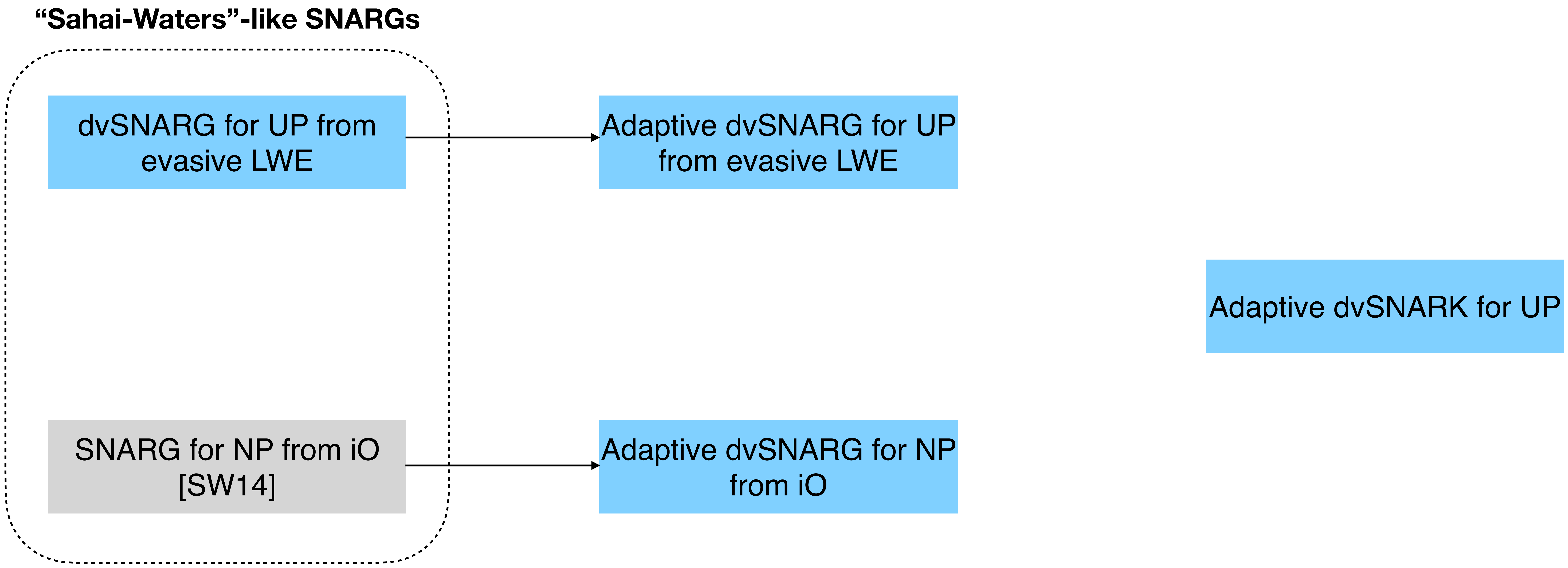
“Sahai-Waters”-like SNARGs

dvSNARG for UP from evasive LWE

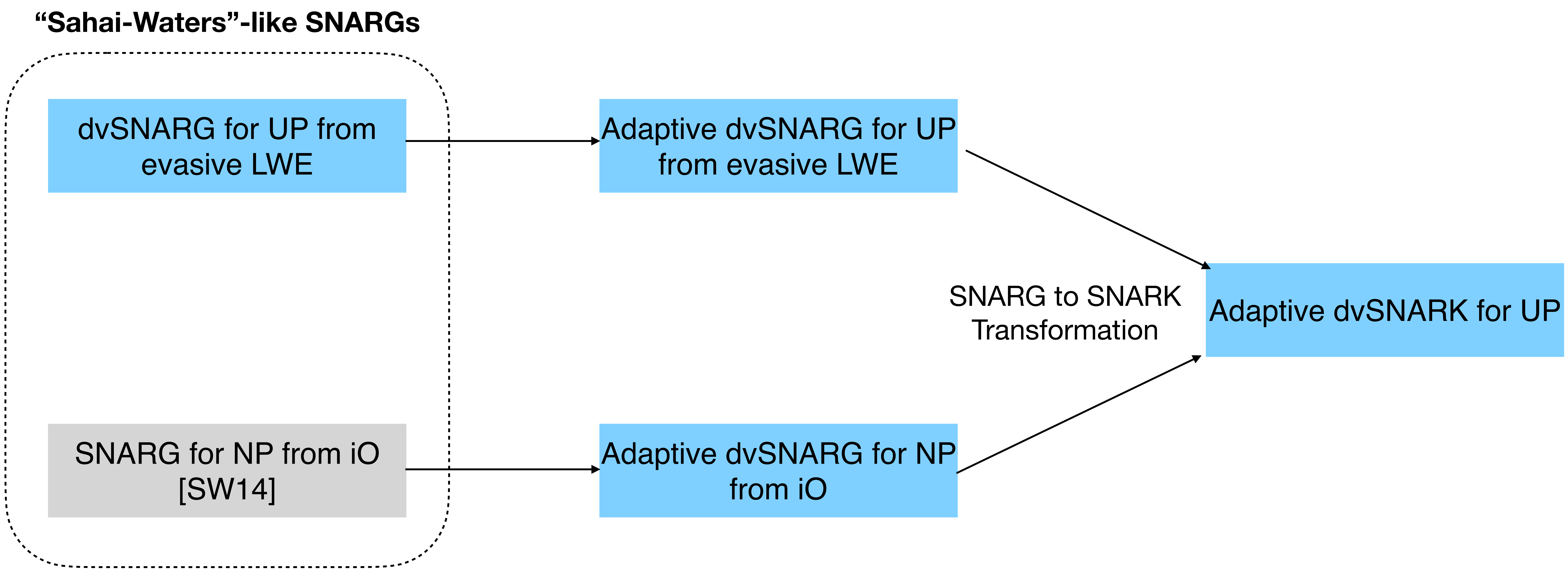
SNARG for NP from iO [SW14]

Adaptive dvSNARK for UP

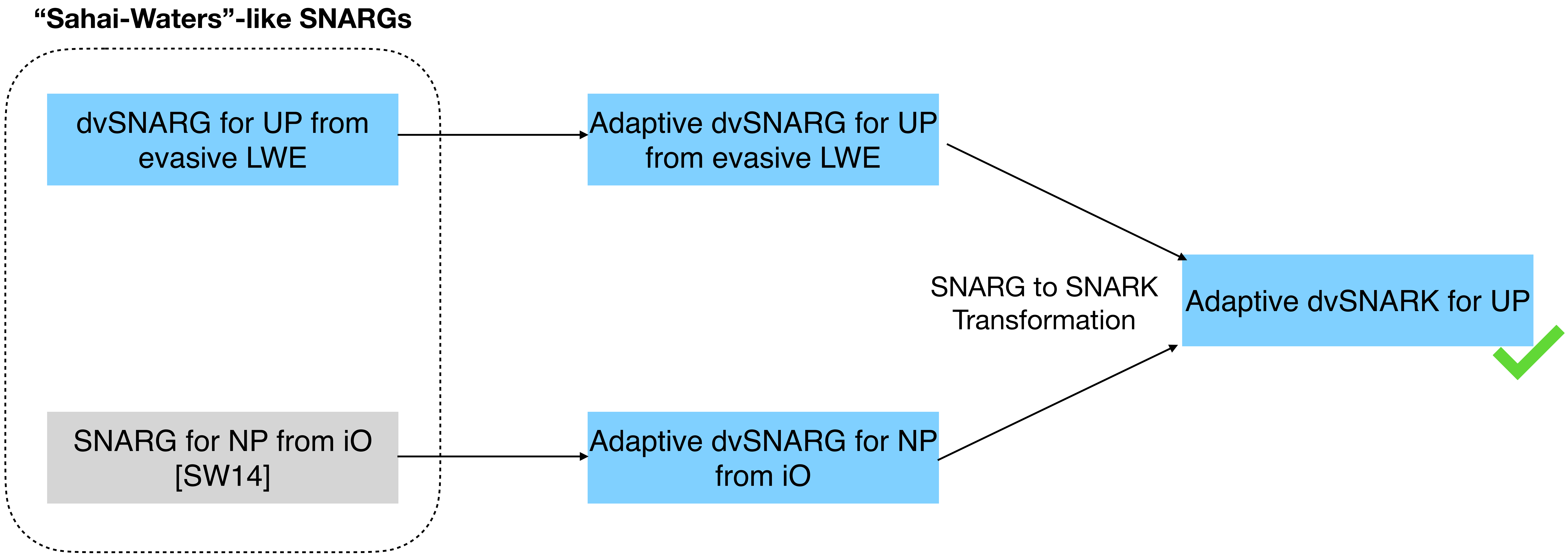

"Sahai-Waters"-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARK for UP



“Sahai-Waters”-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
Adaptive dvSNARK for UP



“Sahai-Waters”-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
SNARG to SNARK Transformation
Adaptive dvSNARK for UP


“Sahai-Waters”-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
SNARG to SNARK Transformation
Adaptive dvSNARK for UP

Concurrent works

Concurrent works

- Beautiful concurrent works [WW24, WZ24] construct adaptively secure **publicly verifiable** SNARGs for NP.

Concurrent works

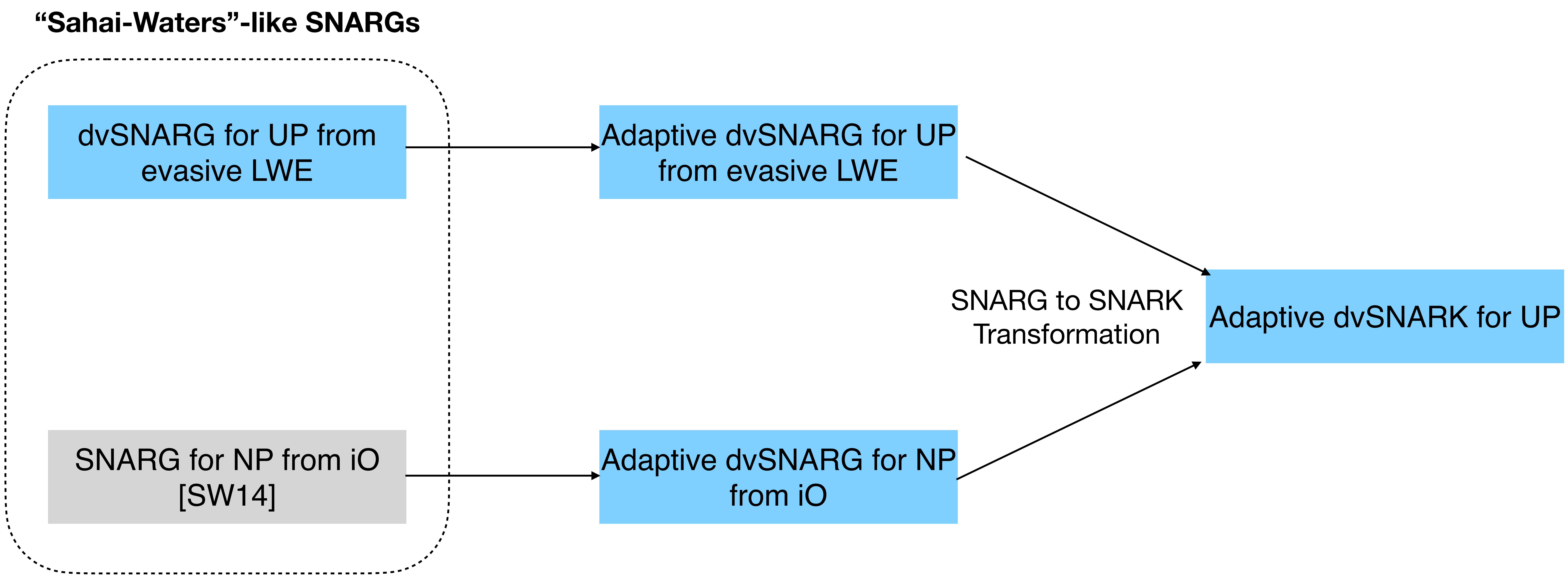
- Beautiful concurrent works [WW24, WZ24] construct adaptively secure **publicly verifiable** SNARGs for NP.
    - [WW24] Sub-exponential iO + OWF, hardness of factoring/discrete log.

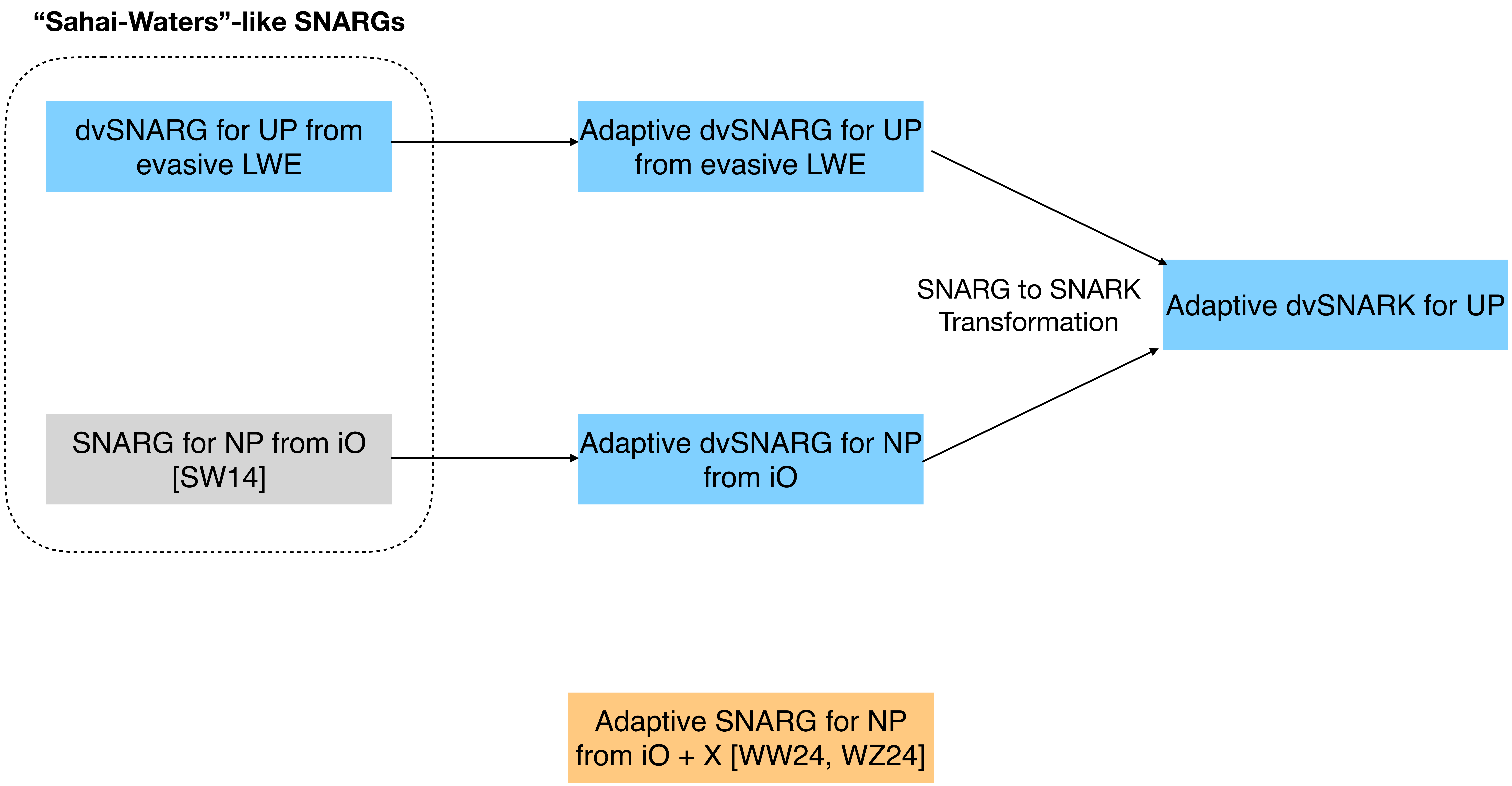
Concurrent works

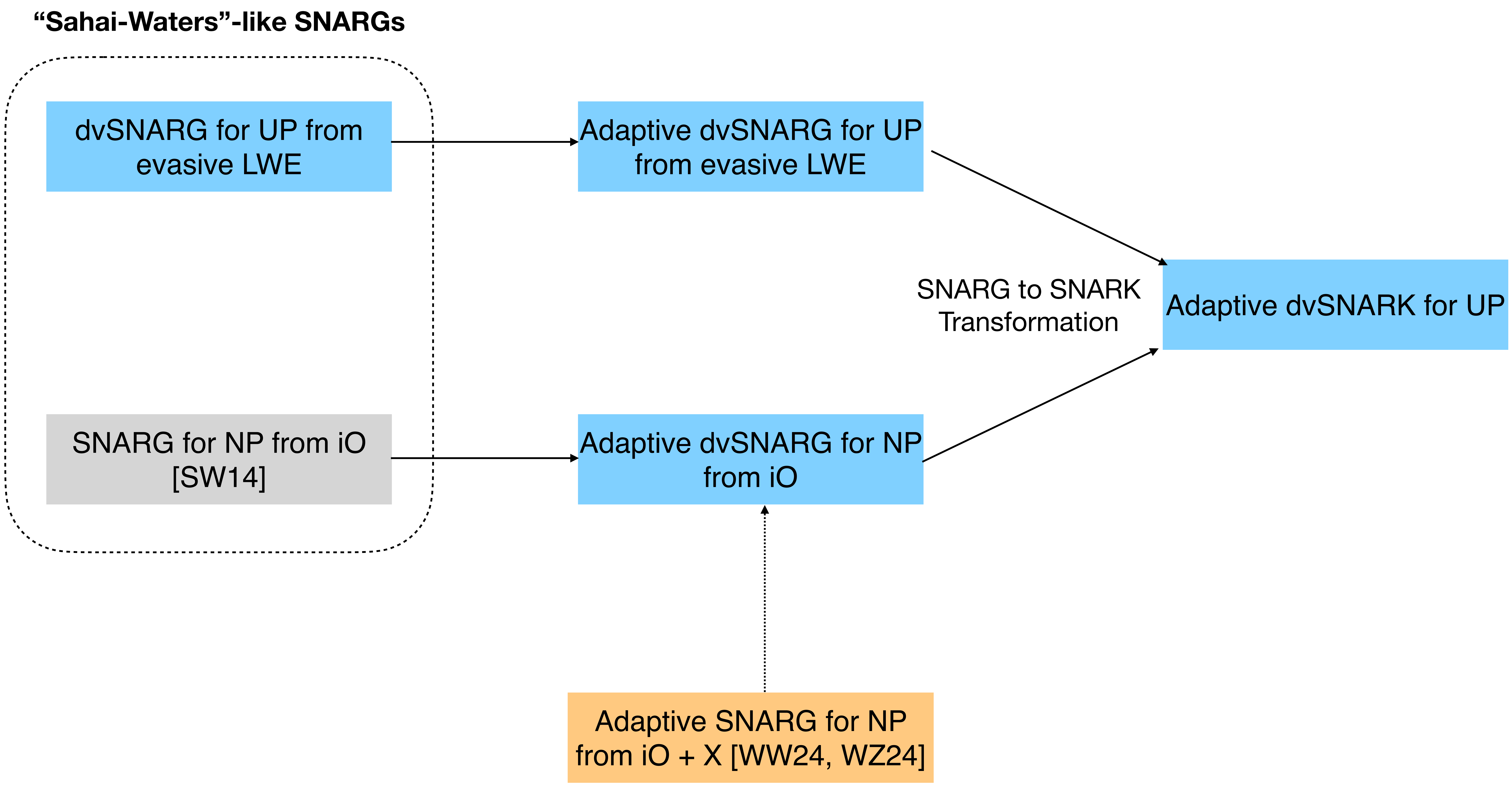
- Beautiful concurrent works [WW24, WZ24] construct adaptively secure **publicly verifiable** SNARGs for NP.
  - [WW24] Sub-exponential iO + OWF, hardness of factoring/discrete log.
  - [WZ24] Sub-exponential iO + OWF, LWE.

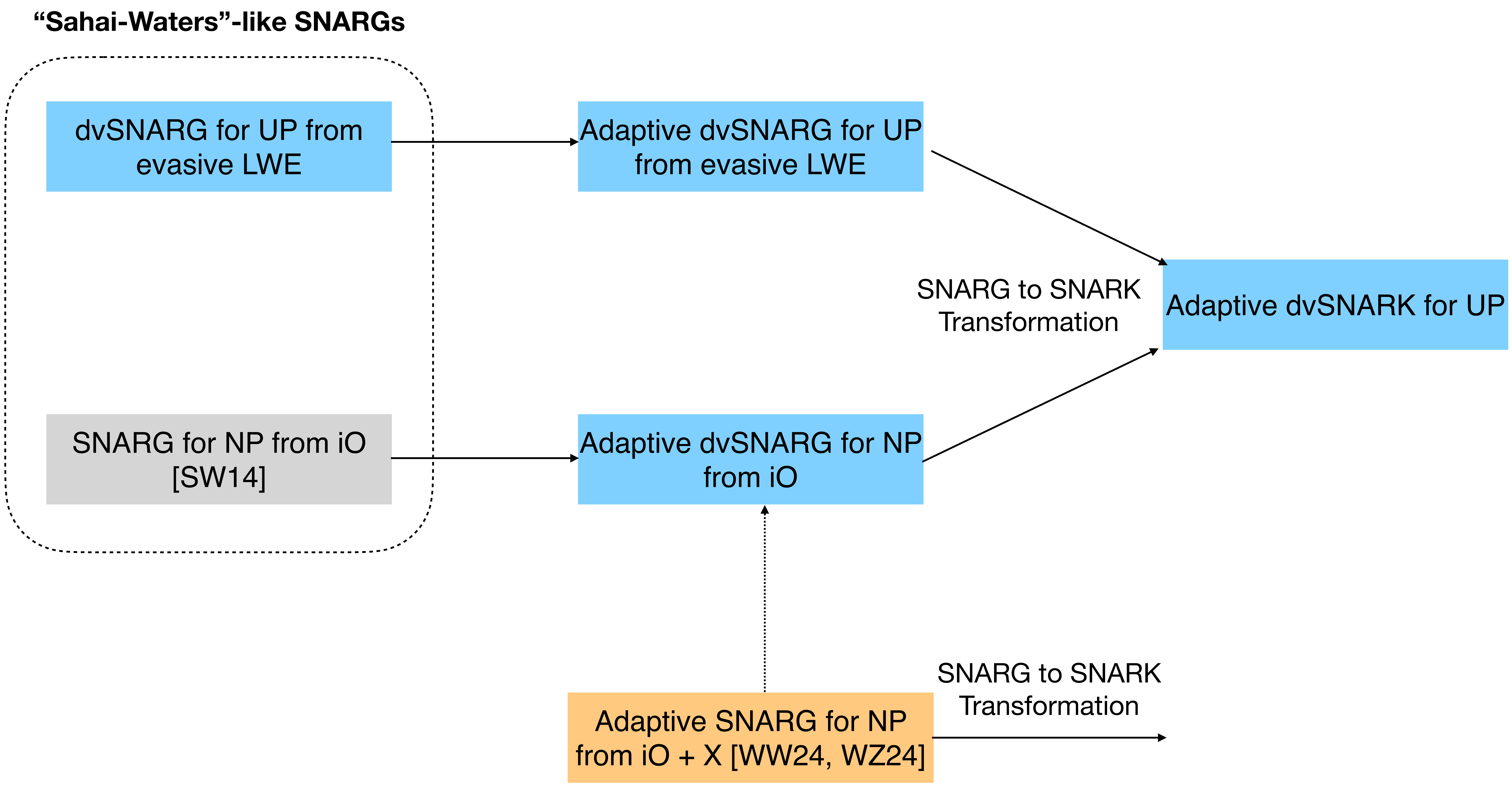
Concurrent works

- Beautiful concurrent works [WW24, WZ24] construct adaptively secure **publicly verifiable** SNARGs for NP.
  - [WW24] Sub-exponential iO + OWF, hardness of factoring/discrete log.
  - [WZ24] Sub-exponential iO + OWF, LWE.
- **Corollary:** <u>Publicly verifiable</u> SNAR<u>**K**</u>s for UP using our/[CGKS23] compiler.

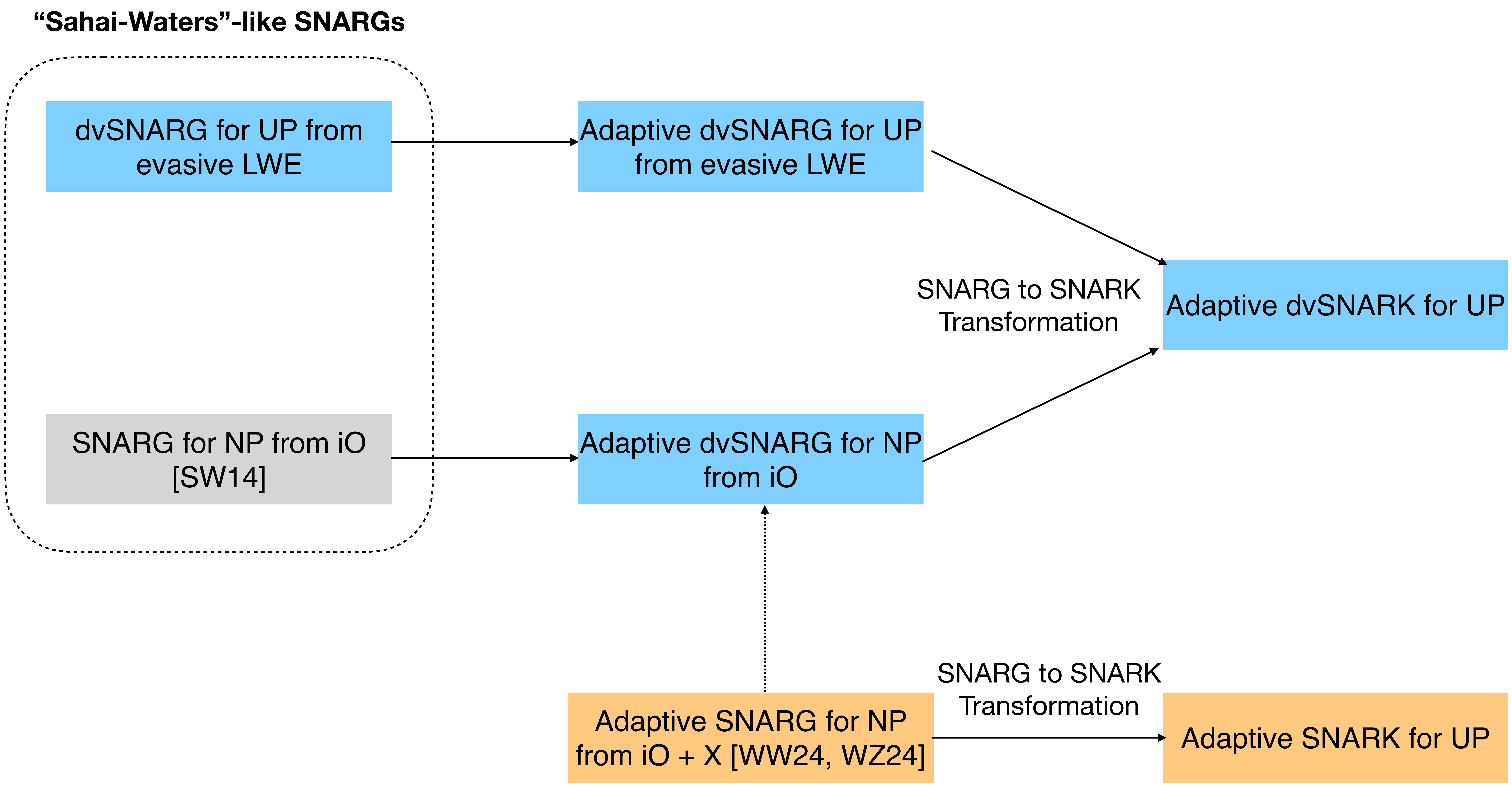

"Sahai-Waters"-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
SNARG to SNARK Transformation
Adaptive dvSNARK for UP


"Sahai-Waters"-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
SNARG to SNARK Transformation
Adaptive dvSNARK for UP
Adaptive SNARG for NP from iO + X [WW24, WZ24]


“Sahai-Waters”-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
Adaptive SNARG for NP from iO + X [WW24, WZ24]
SNARG to SNARK Transformation
Adaptive dvSNARK for UP



“Sahai-Waters”-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
SNARG to SNARK Transformation
Adaptive dvSNARK for UP
Adaptive SNARG for NP from iO + X [WW24, WZ24]
SNARG to SNARK Transformation


"Sahai-Waters"-like SNARGs
dvSNARG for UP from evasive LWE
SNARG for NP from iO [SW14]
Adaptive dvSNARG for UP from evasive LWE
Adaptive dvSNARG for NP from iO
SNARG to SNARK Transformation
Adaptive dvSNARK for UP
Adaptive SNARG for NP from iO + X [WW24, WZ24]
SNARG to SNARK Transformation
Adaptive SNARK for UP

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE
2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.
   - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!
3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u>** for **UP**.
   - Corollary: **Adaptively sound** dv-SNAR<u>K</u> for UP from **falsifiable assumptions.**

All of the above constructions/transformations also **satisfy/preserve zero-knowledge!**

Witness PRF [Zhandry 16]

Witness PRF [Zhandry 16]

- We view the “Sahai-Waters” SNARG in the designated verifier setting as a special case of **witness PRF**.

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, $(\mathsf{pk}, F_{\mathsf{sk}}(x)) \approx_c (\mathsf{pk}, r)$ where r is a random string.

Witness PRF [Zhandry 16]

- We view the “Sahai-Waters” SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is ... F).
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, $(\mathsf{pk}, F_{\mathsf{sk}}(x)) \approx_c (\mathsf{pk}, r)$ where r is a random string.

Hybrid between witness encryption and constrained PRFs

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, $(\mathsf{pk}, F_{\mathsf{sk}}(x)) \approx_c (\mathsf{pk}, r)$ where r is a random string.

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF.Gen(R)}$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, $(\mathsf{pk}, F_{\mathsf{sk}}(x)) \approx_c (\mathsf{pk}, r)$ where r is a random string.

Sahai-Waters: Non-adaptive witness PRF for NP from iO + OWF.

Witness PRF [Zhandry 16]

- We view the "Sahai-Waters" SNARG in the designated verifier setting as a special case of **witness PRF**.
- Fix an NP relation R. Witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, F)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, $(\mathsf{pk}, F_{\mathsf{sk}}(x)) \approx_c (\mathsf{pk}, r)$ where r is a random string.

Sahai-Waters: Non-adaptive witness PRF for NP from iO + OWF.

Our UP SNARG: Adaptive witness PRF for UP from evasive LWE.

Sahai-Waters Witness PRF

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a puncturable PRF. Key is the obfuscation of this:

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a <u>puncturable</u> PRF. Key is the obfuscation of this:

IndObf:

Prove$_k(x, w)$:
- If $R(x, w) = 1$, output PRF$_k(x)$
- Else, output $\perp$

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a <u>puncturable</u> PRF. Key is the obfuscation of this:

IndObf:

$\text{Prove}_k(x, w)$:
- If $R(x, w) = 1$, output $\text{PRF}_k(x)$
- Else, output $\bot$

- **Secret key:** PRF Key k.

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a puncturable PRF. Key is the obfuscation of this:

IndObf:

Prove$_k(x, w)$:
- If $R(x, w) = 1$, output PRF$_k(x)$
- Else, output $\perp$

- **Secret key:** PRF Key k.

- **Correctness:** If $R(x, w) = 1$, then obfuscation outputs PRF$_k(x)$.

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a puncturable PRF. Key is the obfuscation of this:

IndObf:

Prove$_k(x, w)$:
- If $R(x, w) = 1$, output PRF$_k(x)$
- Else, output $\perp$

- **Secret key:** PRF Key k.

- **Correctness:** If $R(x, w) = 1$, then obfuscation outputs PRF$_k(x)$.

- **Non-adaptive security:** If $x^* \notin L$, replace k in obfuscation with punctured key $k\{x^*\}$.

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a puncturable PRF. Key is the obfuscation of this:

IndObf:

Prove$_k(x, w)$:
- If $R(x, w) = 1$, output PRF$_k(x)$
- Else, output $\perp$

- **Secret key:** PRF Key k.

- **Correctness:** If $R(x, w) = 1$, then obfuscation outputs PRF$_k(x)$.
- **Non-adaptive security:** If $x^* \notin L$, replace k in obfuscation with punctured key $k\{x^*\}$.

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Sahai-Waters Witness PRF

- **Public key:** Let PRF be a <u>puncturable</u> PRF. Key is the obfuscation of this:

IndObf:

Prove$_k(x, w)$:
- If $R(x, w) = 1$, output PRF$_k(x)$
- Else, output $\bot$

- **Secret key:** PRF Key k.

- **Correctness:** If $R(x, w) = 1$, then obfuscation outputs PRF$_k(x)$.

- **Non-adaptive security:** If $x^* \notin L$, replace k in obfuscation with <u>punctured key $k\{x^*\}$</u>.

$$(\mathrm{pk}, \mathrm{PRF}_k(x)) \approx_c (\mathrm{pk}', r)$$

Disclaimer: This slide is targeted at those of you familiar with Sahai-Waters SNARG.

Witness PRF to SNARG Template

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$

$\mathscr{V}$

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF.Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF.Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$ crs = pk $\mathscr{V}$

State = sk

Witness PRF for R

- (pk, sk) $\leftarrow$ wPRF . Gen(R).
- **Correctness:** If $R(x, w) = 1$, $\text{Eval}_{\text{pk}}(x, w) = F_{\text{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\text{sk}}(x)$, i.e. $F_{\text{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w)$

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$ | crs = pk | $\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w) \xrightarrow{x,\ \pi}$

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF.Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$ | crs = pk | $\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w)$ $\xrightarrow{x,\, \pi}$ Accept if $\pi = F_{\mathsf{sk}}(x)$

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF.Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w) \xrightarrow{x,\, \pi}$ Accept if $\pi = F_{\mathsf{sk}}(x)$

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF.Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w)$ $\xrightarrow{x,\ \pi}$ Accept if $\pi = F_{\mathsf{sk}}(x)$

Claim: For $x^* \notin L$,
$(\mathsf{crs}, F_{\mathsf{sk}}(x^*)) \approx_c (\mathsf{crs}, r)$.
Moreover, this transformation preserves **adaptiveness.**

Witness PRF for R

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{wPRF}.\mathsf{Gen}(\mathsf{R})$.
- **Correctness:** If $R(x, w) = 1$, $\mathsf{Eval}_{\mathsf{pk}}(x, w) = F_{\mathsf{sk}}(x)$.
- **Security:** If $x \notin L$, pk hides the value of $F_{\mathsf{sk}}(x)$, i.e. $F_{\mathsf{sk}}(x)$ looks *random.*

Complexity Leveraging the Witness PRF

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.

- Take a **non-adaptive witness PRF** construction <u>with $2^{-(|x|+\lambda)}$ security.</u>

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Non-adaptive Challenger $\mathscr{C}$

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Adaptive Adversary $\mathscr{A}'$

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Adaptive Adversary $\mathscr{A}'$

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Choose random
$x^* \notin L$

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.

- Take a **non-adaptive witness PRF** construction <u>with $2^{-(|x|+\lambda)}$ security.</u>

- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is <u>polynomially secure</u>!

Adaptive Adversary $\mathscr{A}'$

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Choose random $x^* \notin L$

$\xrightarrow{x^*}$

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Adaptive Adversary $\mathscr{A}'$

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Choose random $x^* \notin L$

$\xrightarrow{x^*}$

$y_0 = F_{\mathsf{sk}}(x^*), y_1 = r$

Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.

- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.

- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Adaptive Adversary $\mathscr{A}'$

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Choose random $x^* \notin L$

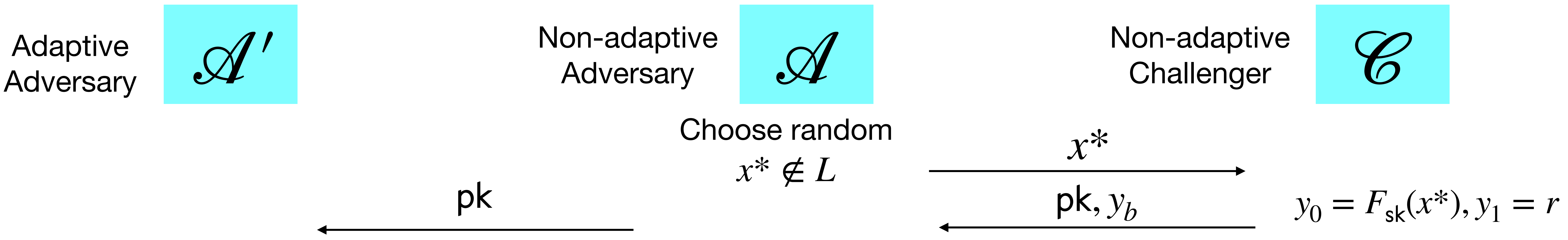
x^* →

← pk, y_b

$y_0 = F_{\mathsf{sk}}(x^*), y_1 = r$

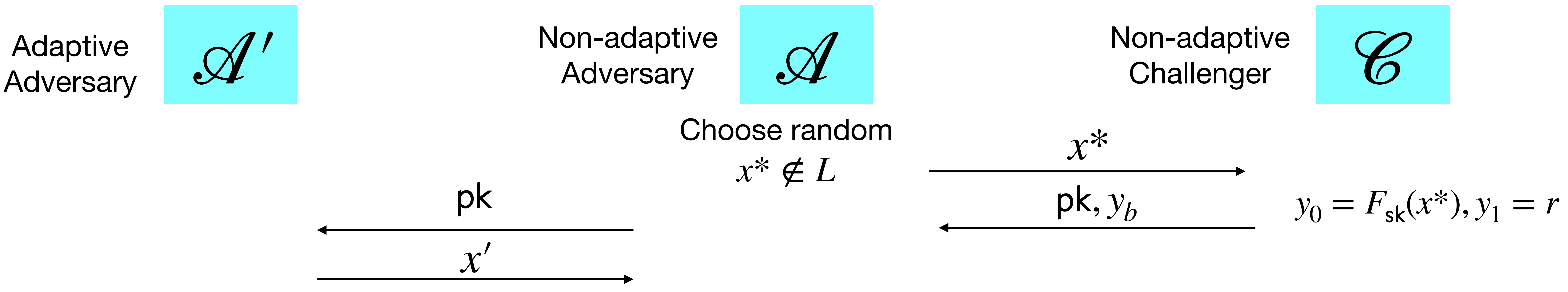
Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction <u>with $2^{-(|x|+\lambda)}$ security.</u>
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is <u>polynomially secure</u>!



Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction <u>with $2^{-(|x|+\lambda)}$ security.</u>
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is <u>polynomially secure</u>!



Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!

Adaptive Adversary $\mathscr{A}'$

Non-adaptive Adversary $\mathscr{A}$

Non-adaptive Challenger $\mathscr{C}$

Choose random $x^* \notin L$

x^*

pk, y_b

$y_0 = F_{\mathsf{sk}}(x^*), y_1 = r$

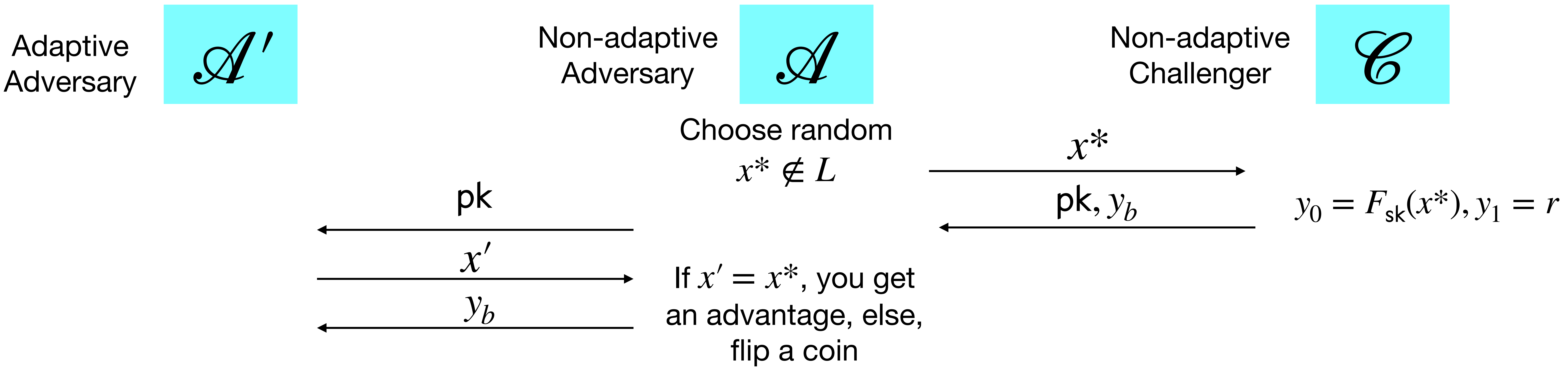
pk

x'

If $x' = x^*$, you get an advantage, else, flip a coin

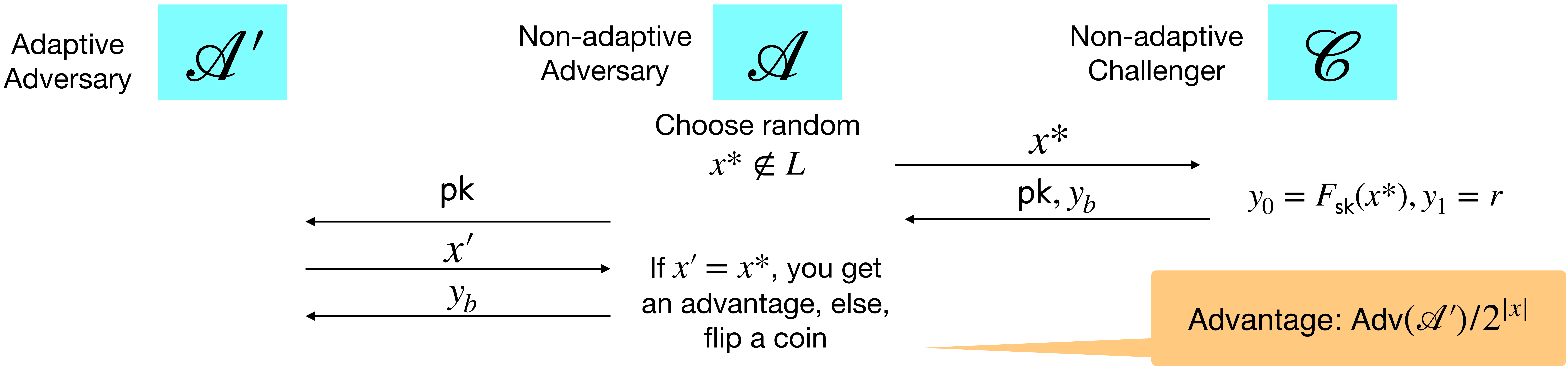
Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!



Complexity Leveraging the Witness PRF

- We argued that the resulting SNARG is **adaptively sound** if the witness PRF is **adaptively sound**.
- Take a **non-adaptive witness PRF** construction with $2^{-(|x|+\lambda)}$ security.
- Complexity leverage the witness PRF to obtain an **adaptive** witness PRF which is polynomially secure!



Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w) \xrightarrow{x, \pi}$ Accept if $\pi = F_{\mathsf{sk}}(x)$

Claim: For $x^* \notin L$,
$(\mathsf{crs}, F_{\mathsf{sk}}(x^*)) \approx_c (\mathsf{crs}, r)$.
Moreover, this transformation preserves **adaptiveness.**

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w)$ $\xrightarrow{x,\, \pi}$ Accept if $\pi = F_{\mathsf{sk}}(x)$

- The length of π is depends only on **security parameter** of the SNARG!

Claim: For $x^* \notin L$,
$(\mathsf{crs}, F_{\mathsf{sk}}(x^*)) \approx_c (\mathsf{crs}, r)$.
Moreover, this transformation preserves **adaptiveness.**

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \mathsf{Eval}_{\mathsf{pk}}(x, w)$ $\xrightarrow{x, \pi}$ Accept if $\pi = F_{\mathsf{sk}}(x)$

- The length of π is depends only on **security parameter** of the SNARG!
- Can decouple the **wPRF security <u>indistinguishability</u> parameter** from **proof <u>search</u> size.**

Claim: For $x^* \notin L$,
$(\mathsf{crs}, F_{\mathsf{sk}}(x^*)) \approx_c (\mathsf{crs}, r)$.
Moreover, this transformation preserves **adaptiveness.**

Witness PRF to SNARG Template

$\mathscr{P}$

crs = pk

$\mathscr{V}$

State = sk

$\pi = \text{Eval}_{\text{pk}}(x, w)$ $\xrightarrow{x, \pi}$ Accept if $\pi = F_{\text{sk}}(x)$

Claim: For $x^* \notin L$,
$(\text{crs}, F_{\text{sk}}(x^*)) \approx_c (\text{crs}, r)$.
Moreover, this transformation preserves **adaptiveness.**

- The length of π is depends only on **security parameter** of the SNARG!
- Can decouple the **wPRF security indistinguishability parameter** from **proof search size.**
- We can choose proof size $\sim \lambda$ for $2^{-\lambda}$ soundness!

TL;DR

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE

2. Show our dvSNARG, and any “Sahai-Waters”-like dvSNARG can be made **adaptively sound**.

   - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!

3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u>** for **UP**.

   - Corollary: **Adaptively sound** dv-SNAR<u>K</u> for UP from **falsifiable assumptions.**

All of the above constructions/transformations also **satisfy/preserve zero-knowledge!**

TL;DR

Build a witness PRF for UP from evasive LWE

In this work, we

1. Build a designated-verifier **SNARG for UP** from LWE and evasive LWE
2. Show our dvSNARG, and any "Sahai-Waters"-like dvSNARG can be made **adaptively sound**.
   - **Adaptively sound** SNARGs from **falsifiable assumptions** ([JLS20] iO + OWF)!
3. Transformation from **SNAR<u>G</u> for UP** to **SNAR<u>K</u>** for **UP**.
   - Corollary: **Adaptively sound** dv-SNAR<u>K</u> for UP from **falsifiable assumptions.**

All of the above constructions/transformations also **satisfy/preserve zero-knowledge!**

Witness PRF from Evasive LWE

Witness PRF from Evasive LWE

- **Step 1: “Average-case obfuscation”** for functions with **pseudorandom outputs** from evasive LWE.

Witness PRF from Evasive LWE

- **Step 1: "Average-case obfuscation"** for functions with **pseudorandom outputs** from evasive LWE.

- **Step 2:** Consider the following function constructed from PRFs F_{K_1} and G_{K_2}.

Witness PRF from Evasive LWE

- **Step 1: "Average-case obfuscation"** for functions with **pseudorandom outputs** from evasive LWE.

- **Step 2:** Consider the following function constructed from PRFs F_{K_1} and G_{K_2}.

$$W_{K_1,K_2}(x,w) = \begin{cases} F_{K_1}(x) & \text{if } R(x,w) = 1 \\ G_{K_2}(x,w) & \text{otherwise} \end{cases}$$

Witness PRF from Evasive LWE

- **Step 1: "Average-case obfuscation"** for functions with **<u>pseudorandom outputs</u>** from evasive LWE.

- **Step 2:** Consider the following function constructed from PRFs F_{K_1} and G_{K_2}.

$$W_{K_1,K_2}(x, w) = \begin{cases} F_{K_1}(x) & \text{if } R(x, w) = 1 \\ G_{K_2}(x, w) & \text{otherwise} \end{cases}$$

This is a PRF if R is a UP relation. If not UP, then this might not hold!

Witness PRF from Evasive LWE

- **Step 1: "Average-case obfuscation"** for functions with **pseudorandom outputs** from evasive LWE.

- **Step 2:** Consider the following function constructed from PRFs F_{K_1} and G_{K_2}.

$$W_{K_1,K_2}(x, w) = \begin{cases} F_{K_1}(x) & \text{if } R(x, w) = 1 \\ G_{K_2}(x, w) & \text{otherwise} \end{cases}$$

This is a PRF if R is a UP relation. If not UP, then this might not hold!

- If x has two witness w_1, w_2, then $W_{K_1,K_2}(x, w_1) = W_{K_1,K_2}(x, w_2)$ (i.e. **zeroizing regime!**)

Witness PRF from Evasive LWE

- **Step 1: "Average-case obfuscation"** for functions with **<u>pseudorandom outputs</u>** from evasive LWE.

- **Step 2:** Consider the following function constructed from PRFs F_{K_1} and G_{K_2}.

$$W_{K_1,K_2}(x,w) = \begin{cases} F_{K_1}(x) & \text{if } R(x,w) = 1 \\ G_{K_2}(x,w) & \text{otherwise} \end{cases}$$

This is a PRF if R is a UP relation. If not UP, then this might not hold!

  - If x has two witness w_1, w_2, then $W_{K_1,K_2}(x,w_1) = W_{K_1,K_2}(x,w_2)$ (i.e. **zeroizing regime!**)

- **Step 3:** Construct wPRF: pk $= \mathcal{O}(W)$, sk $= K_1$. The obfuscation guarantee, for $x^* \notin L$:

Witness PRF from Evasive LWE

- **Step 1: "Average-case obfuscation"** for functions with **<u>pseudorandom outputs</u>** from evasive LWE.

- **Step 2:** Consider the following function constructed from PRFs F_{K_1} and G_{K_2}.

$$W_{K_1,K_2}(x,w) = \begin{cases} F_{K_1}(x) & \text{if } R(x,w) = 1 \\ G_{K_2}(x,w) & \text{otherwise} \end{cases}$$

  This is a PRF if R is a UP relation. If not UP, then this might not hold!

  - If x has two witness w_1, w_2, then $W_{K_1,K_2}(x, w_1) = W_{K_1,K_2}(x, w_2)$ (i.e. **zeroizing regime!**)

- **Step 3:** Construct wPRF: $\mathsf{pk} = \mathcal{O}(W)$, $\mathsf{sk} = K_1$. The obfuscation guarantee, for $x^* \notin L$:

$$(\mathsf{pk}, F_{K_1}(x^*)) \approx_c (\mathsf{pk}, r)$$

Summary

Summary

- We build **adaptively sound designated-verifier SNARGs** for:

Summary

- We build **adaptively sound designated-verifier SNARGs** for:
  - UP from LWE and evasive LWE

Summary

- We build **adaptively sound designated-verifier SNARGs** for:
  - UP from LWE and evasive LWE
  - NP from sub-exponential iO + OWF

Summary

- We build **adaptively sound designated-verifier SNARGs** for:
  - UP from LWE and evasive LWE
  - NP from sub-exponential iO + OWF
- We show that adaptively sound SNARGs for UP can be transformed into SNAR**K**s for UP assuming polynomially secure LWE.

Summary

- We build **adaptively sound designated-verifier SNARGs** for:
  - UP from LWE and evasive LWE
  - NP from sub-exponential iO + OWF
- We show that adaptively sound SNARGs for UP can be transformed into SNAR**K**s for UP assuming polynomially secure LWE.
  - We can build SNARKs from **falsifiable assumptions**!

Open Questions

Open Questions

- Can we construct witness PRFs **directly from LWE**?

Open Questions

- Can we construct witness PRFs **directly from LWE**?
    - Can we prove evasive LWE from LWE?

Open Questions

- Can we construct witness PRFs **directly from LWE**?
    - Can we prove evasive LWE from LWE?
- What else can we **prove from evasive LWE** that we can build from obfuscation?

Open Questions

- Can we construct witness PRFs **directly from LWE**?
  - Can we prove evasive LWE from LWE?
- What else can we **prove from evasive LWE** that we can build from obfuscation?
- Can we transform our SNARG into a **publicly verifiable SNARG**?

Open Questions

- Can we construct witness PRFs **directly from LWE**?
  - Can we prove evasive LWE from LWE?
- What else can we **prove from evasive LWE** that we can build from obfuscation?
- Can we transform our SNARG into a **publicly verifiable SNARG**?
  - Have to be very careful about **zeroizing attacks!**

Open Questions

- Can we construct witness PRFs **directly from LWE**?
  - Can we prove evasive LWE from LWE?
- What else can we **prove from evasive LWE** that we can build from obfuscation?
- Can we transform our SNARG into a **publicly verifiable SNARG**?
  - Have to be very careful about **zeroizing attacks!**
- Can we get a SNARG with a **smaller CRS**? Can we get a common *random/ transparent* string?

Thank you very much for your attention!

Bonus Slides

σ-PRF Obfuscation

σ-PRF Obfuscation

- Consider a matrix branching program given by $\mathbf{P} = \{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i\in[k],b\in\{0,1\}}, \mathbf{v}\}$. Then, suppose that:

σ-PRF Obfuscation

- Consider a matrix branching program given by $\mathbf{P} = \{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i \in [k], b \in \{0,1\}}, \mathbf{v}\}$. Then, suppose that:

$$\left\{ \mathbf{u} \left(\prod \mathbf{M}_{i,x_i} \right) \mathbf{v} \right\}_{x \in \{0,1\}^k}, \text{aux} \approx_c \{\mathscr{U}\}_{x \in \{0,1\}^k}, \text{aux}$$

σ-PRF Obfuscation

- Consider a matrix branching program given by $\mathbf{P} = \{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i\in[k],b\in\{0,1\}}, \mathbf{v}\}$. Then, suppose that:

$$\left\{\mathbf{u}\left(\prod \mathbf{M}_{i,x_i}\right)\mathbf{v}\right\}_{x\in\{0,1\}^k}, \text{aux} \approx_c \{\mathscr{U}\}_{x\in\{0,1\}^k}, \text{aux}$$

(i.e. the function is a “very secure PRF” **when noise is added**)

σ-PRF Obfuscation

- Consider a matrix branching program given by $\mathbf{P} = \{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i \in [k], b \in \{0,1\}}, \mathbf{v}\}$. Then, suppose that:

$$\left\{ \mathbf{u} \left(\prod \mathbf{M}_{i,x_i} \right) \mathbf{v} \right\}_{x \in \{0,1\}^k}, \text{aux} \approx_c \{\mathscr{U}\}_{x \in \{0,1\}^k}, \text{aux}$$

(i.e. the function is a "very secure PRF" **when noise is added**)

- Then, our obfuscation guarantees that $(\mathcal{O}(P), \text{aux}) \approx_c (\mathscr{D}, \text{aux})$.

σ-PRF Obfuscation

- Consider a matrix branching program given by $\mathbf{P} = \{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i \in [k], b \in \{0,1\}}, \mathbf{v}\}$. Then, suppose that:

$$\left\{ \mathbf{u} \left(\prod \mathbf{M}_{i,x_i} \right) \mathbf{v} \right\}_{x \in \{0,1\}^k}, \text{aux} \approx_c \{\mathscr{U}\}_{x \in \{0,1\}^k}, \text{aux}$$

(i.e. the function is a “very secure PRF” **when noise is added**)

- Then, our obfuscation guarantees that $(\mathscr{O}(P), \text{aux}) \approx_c (\mathscr{D}, \text{aux})$.

(i.e. the obfuscation leaks nothing more than the outputs)

Simplified Obfuscation Construction

Simplified Obfuscation Construction

- **Step 1:** Consider a read-once branching program PRF $F_k : \{0,1\}^h \to \mathcal{Y}$ given by$\mathbf{u}$, $\{M_{i,b}\}_{i\in[h],b\in\{0,1\}}$, $\mathbf{v}$ satisfying:

Simplified Obfuscation Construction

- **Step 1:** Consider a read-once branching program PRF $F_k : \{0,1\}^h \to \mathscr{Y}$ given by$\mathbf{u}$, $\{M_{i,b}\}_{i\in[h],b\in\{0,1\}}$, $\mathbf{v}$ satisfying:

$$F_k(\mathbf{x}) = \mathbf{u}\left(\prod_{i=1}^{h}\mathbf{M}_{i,x_i}\right)\mathbf{v}$$

Simplified Obfuscation Construction

- **Step 1:** Consider a read-once branching program PRF $F_k : \{0,1\}^h \to \mathscr{Y}$ given by$\mathbf{u}$, $\{M_{i,b}\}_{i\in[h],b\in\{0,1\}}$, $\mathbf{v}$ satisfying:

Take the subset product!

$$F_k(\mathbf{x}) = \mathbf{u}\left(\prod_{i=1}^{h}\mathbf{M}_{i,x_i}\right)\mathbf{v}$$

Simplified Obfuscation Construction

- **Step 1:** Consider a read-once branching program PRF $F_k : \{0,1\}^h \to \mathscr{Y}$ given by $\mathbf{u}$, $\{M_{i,b}\}_{i \in [h], b \in \{0,1\}}$, $\mathbf{v}$ satisfying:

Take the subset product!

Note: There are <u>**no**</u> read-once PRFs, but we assume this for simplicity.

$$F_k(\mathbf{x}) = \mathbf{u} \left(\prod_{i=1}^{h} \mathbf{M}_{i,x_i} \right) \mathbf{v}$$

Simplified Obfuscation Construction

- **Step 1:** Consider a read-once branching program PRF $F_k : \{0,1\}^h \to \mathscr{Y}$ given by$\mathbf{u}$, $\{M_{i,b}\}_{i\in[h],b\in\{0,1\}}$, $\mathbf{v}$ satisfying:

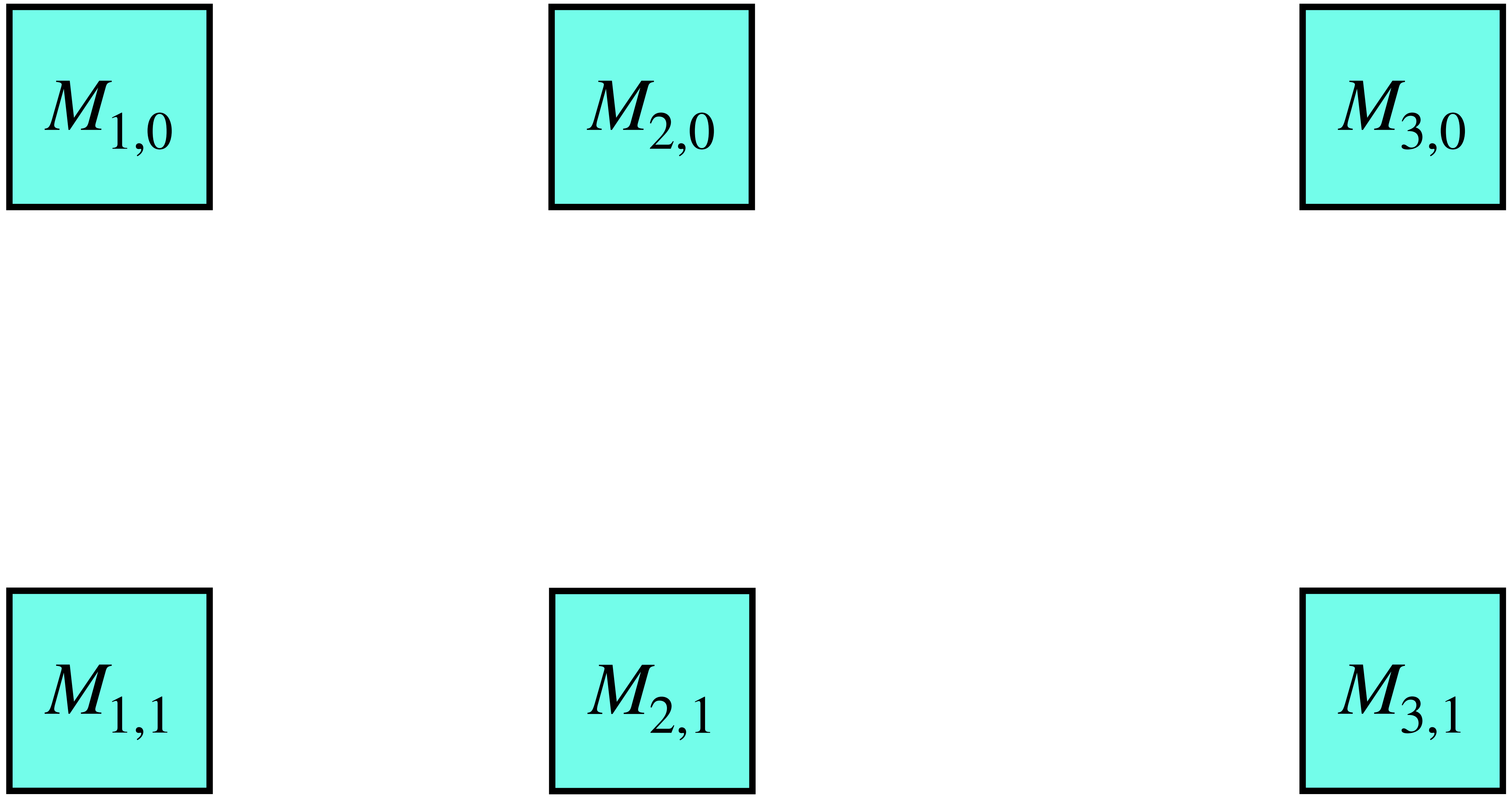
Take the subset product!

Note: There are <u>**no**</u> read-once PRFs, but we assume this for simplicity.

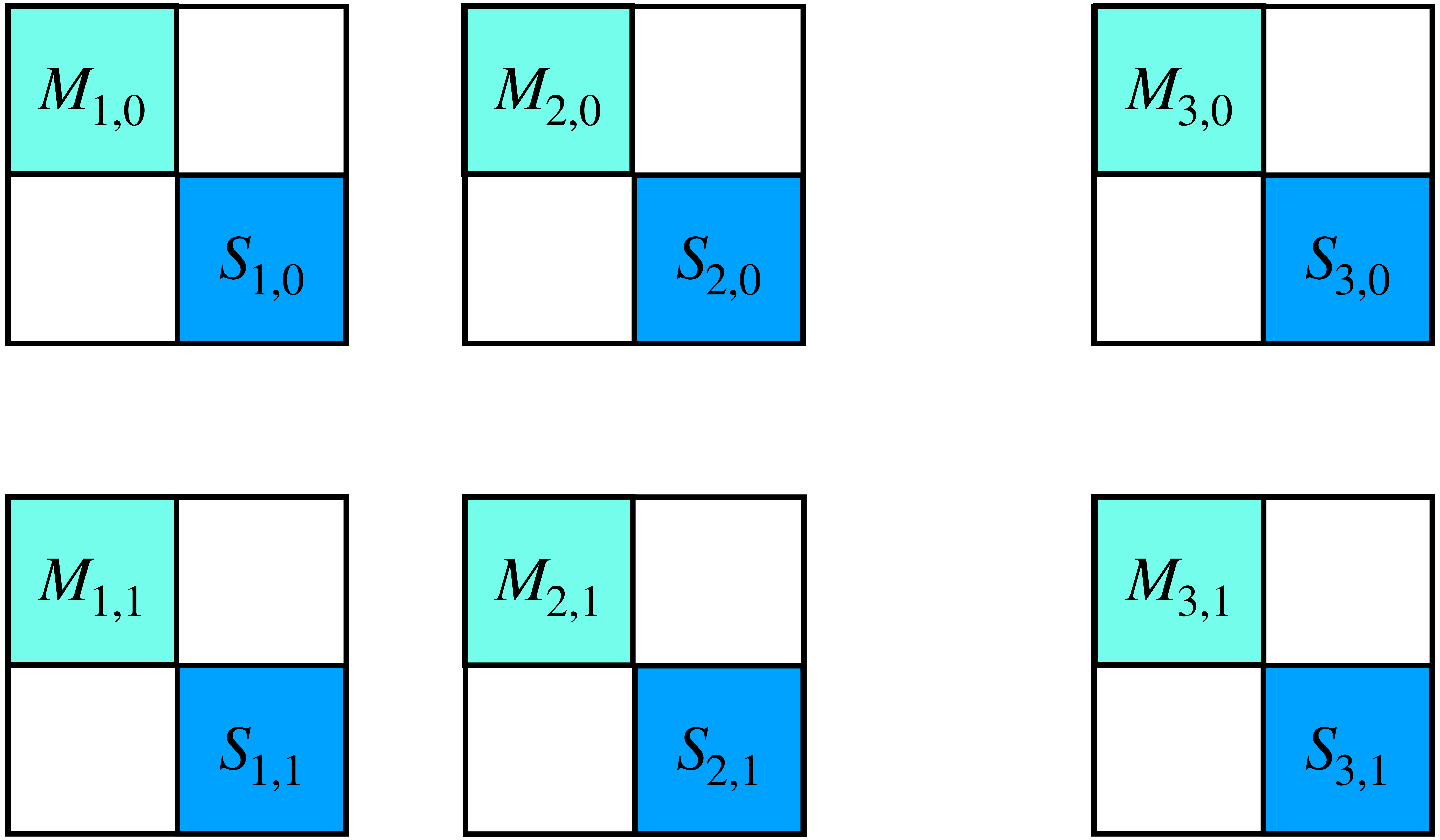
$$F_k(\mathbf{x}) = \mathbf{u}\left(\prod_{i=1}^{h}\mathbf{M}_{i,x_i}\right)\mathbf{v}$$

- **Step 2:** Perform GGH15 [Garg-Gentry-Halevi] encoding of the branching program.

GGH15 Encodings

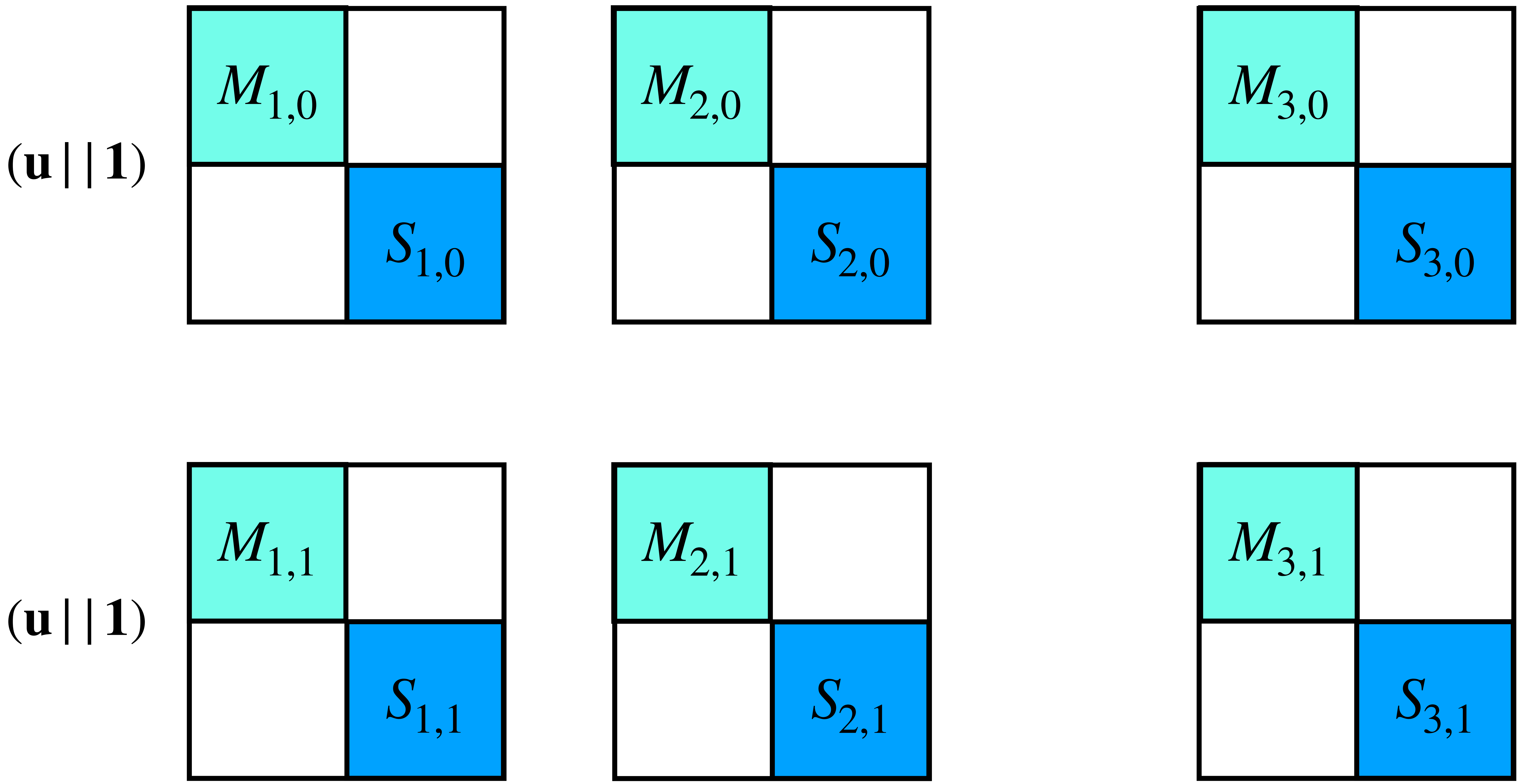


GGH15 Encodings



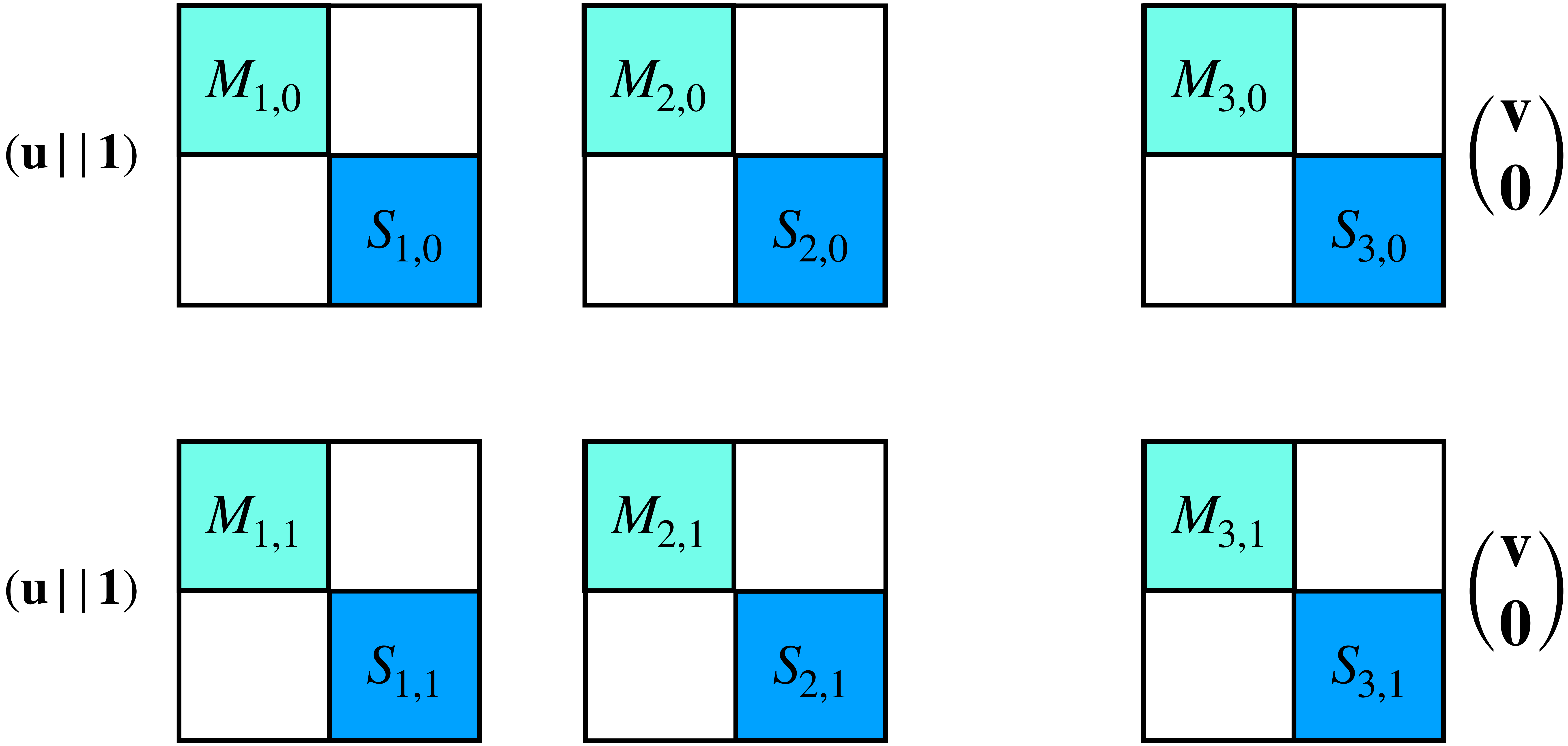
- Sample $S_{i,b} \leftarrow \chi^{c \times c}$ (i.e. small entries)

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)

GGH15 Encodings

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,0} & \\ & S_{1,0} \end{pmatrix} \begin{pmatrix} M_{2,0} & \\ & S_{2,0} \end{pmatrix} \begin{pmatrix} M_{3,0} & \\ & S_{3,0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,1} & \\ & S_{1,1} \end{pmatrix} \begin{pmatrix} M_{2,1} & \\ & S_{2,1} \end{pmatrix} \begin{pmatrix} M_{3,1} & \\ & S_{3,1} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)

Taking subset product still gives:

$$\mathbf{u}\left(\prod_{i=1}^{3} M_{i,x_i}\right)\mathbf{v} = F_k(\mathbf{x})$$

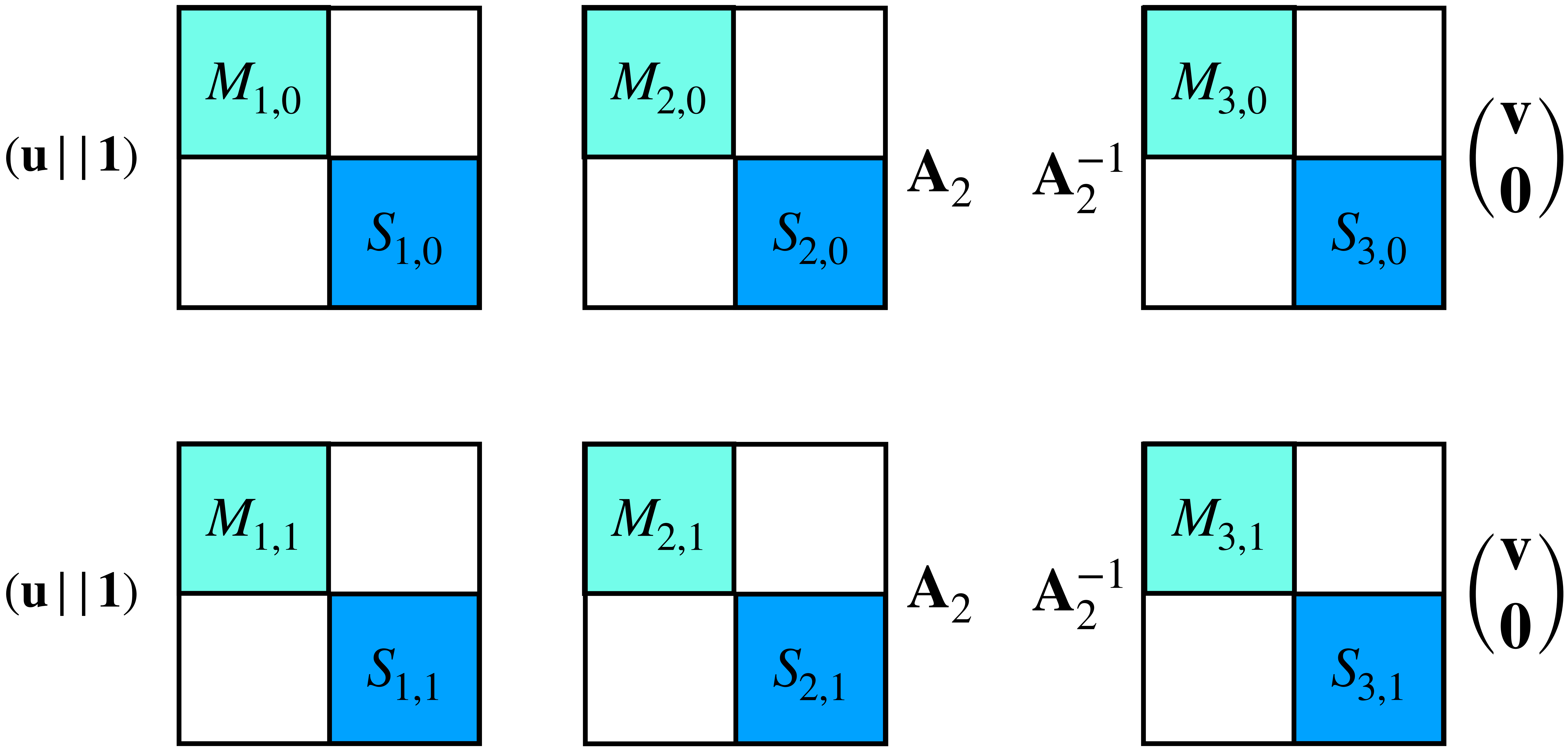
GGH15 Encodings

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,0} & \\ & S_{1,0} \end{pmatrix} \begin{pmatrix} M_{2,0} & \\ & S_{2,0} \end{pmatrix} \quad \begin{pmatrix} M_{3,0} & \\ & S_{3,0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,1} & \\ & S_{1,1} \end{pmatrix} \begin{pmatrix} M_{2,1} & \\ & S_{2,1} \end{pmatrix} \quad \begin{pmatrix} M_{3,1} & \\ & S_{3,1} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

GGH15 Encodings

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,0} & \\ & S_{1,0} \end{pmatrix} \begin{pmatrix} M_{2,0} & \\ & S_{2,0} \end{pmatrix} \mathbf{A}_2 \quad \mathbf{A}_2^{-1} \begin{pmatrix} M_{3,0} & \\ & S_{3,0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,1} & \\ & S_{1,1} \end{pmatrix} \begin{pmatrix} M_{2,1} & \\ & S_{2,1} \end{pmatrix} \mathbf{A}_2 \quad \mathbf{A}_2^{-1} \begin{pmatrix} M_{3,1} & \\ & S_{3,1} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

GGH15 Encodings

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,0} & \\ & S_{1,0} \end{pmatrix} \begin{pmatrix} M_{2,0} & \\ & S_{2,0} \end{pmatrix} \mathbf{A}_2 \quad \mathbf{A}_2^{-1} \begin{pmatrix} M_{3,0} & \\ & S_{3,0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

$$(\mathbf{u}\,||\,\mathbf{1}) \begin{pmatrix} M_{1,1} & \\ & S_{1,1} \end{pmatrix} \begin{pmatrix} M_{2,1} & \\ & S_{2,1} \end{pmatrix} \mathbf{A}_2 \quad \mathbf{A}_2^{-1} \begin{pmatrix} M_{3,1} & \\ & S_{3,1} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

GGH15 Encodings

$$(\mathbf{u}\,||\,\mathbf{1})\begin{pmatrix} M_{1,0} & \\ & S_{1,0} \end{pmatrix}\begin{pmatrix} M_{2,0} & \\ & S_{2,0} \end{pmatrix}\mathbf{A}_2 \quad \mathbf{A}_2^{-1}\begin{pmatrix} M_{3,0} & \\ & S_{3,0} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

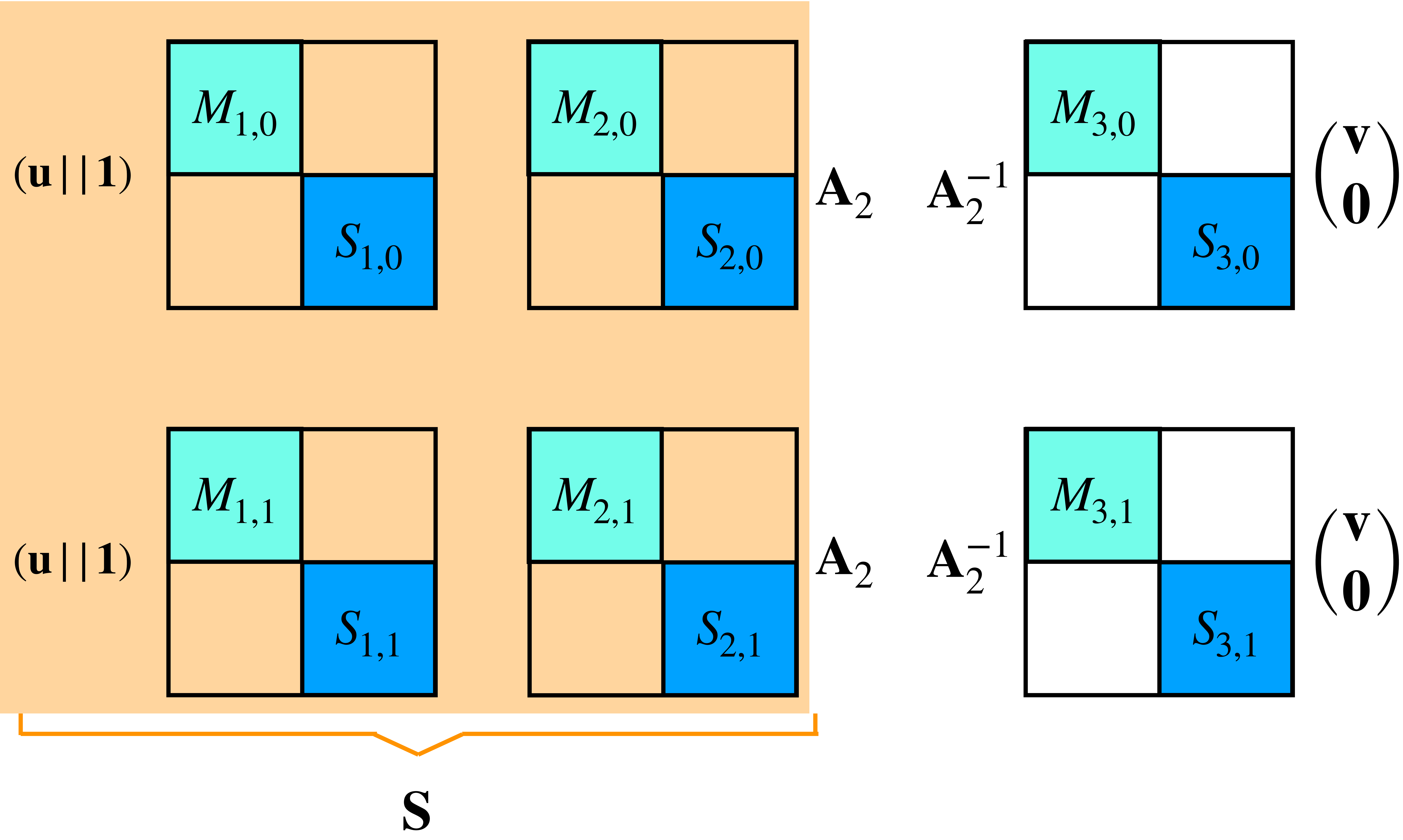
$$(\mathbf{u}\,||\,\mathbf{1})\begin{pmatrix} M_{1,1} & \\ & S_{1,1} \end{pmatrix}\begin{pmatrix} M_{2,1} & \\ & S_{2,1} \end{pmatrix}\mathbf{A}_2 \quad \mathbf{A}_2^{-1}\begin{pmatrix} M_{3,1} & \\ & S_{3,1} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix}$$

- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$

GGH15 Encodings



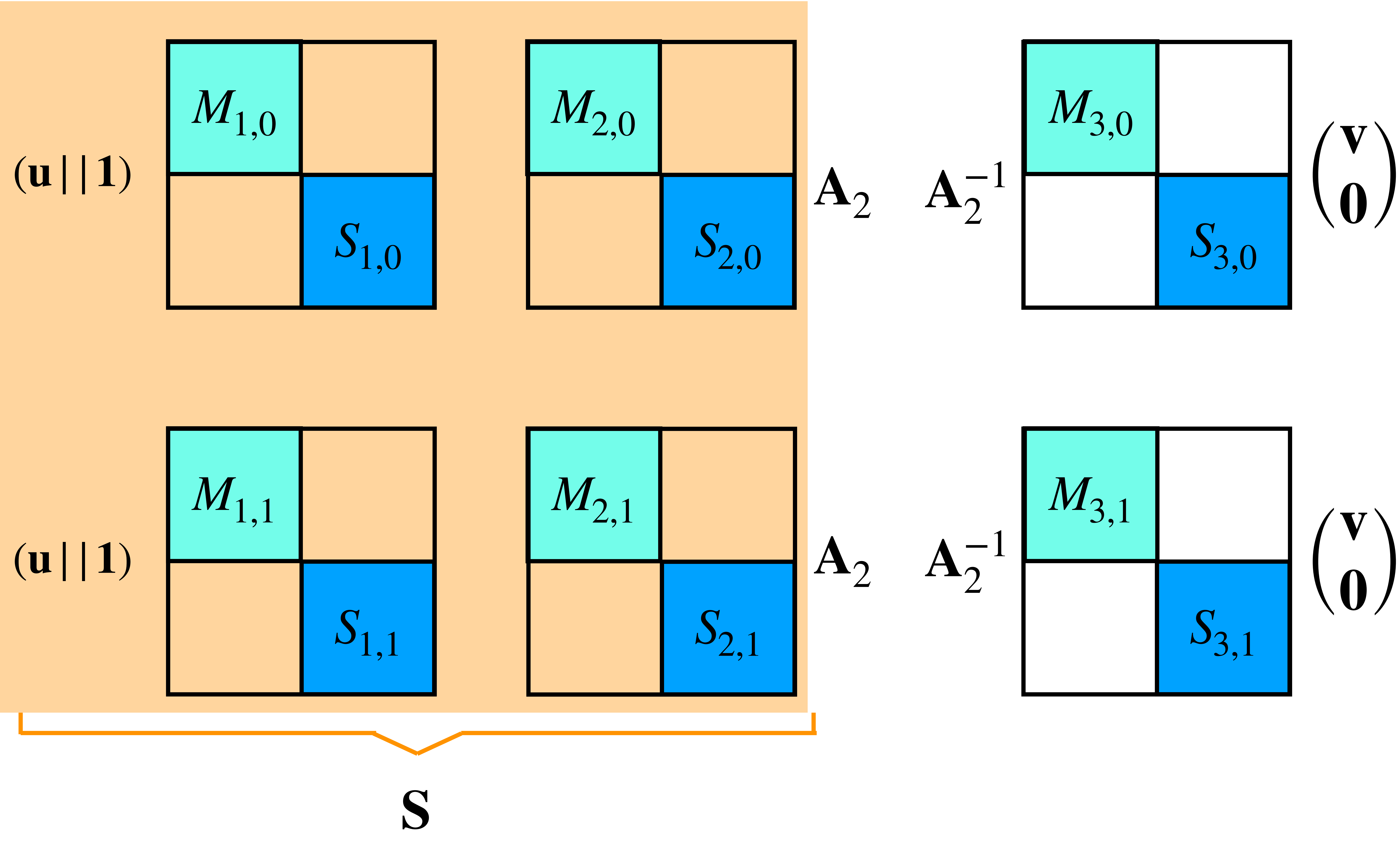
- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

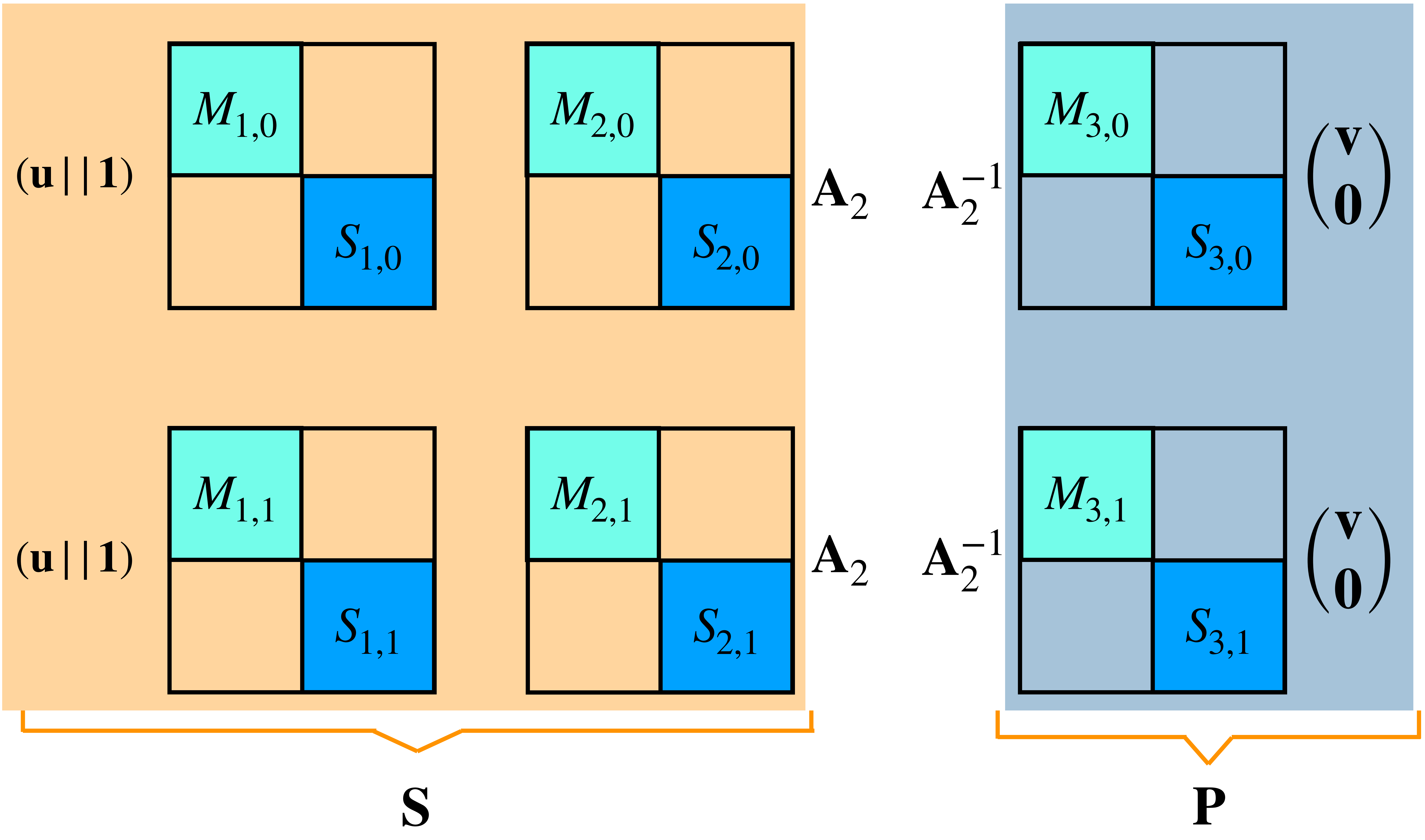
Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c \times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

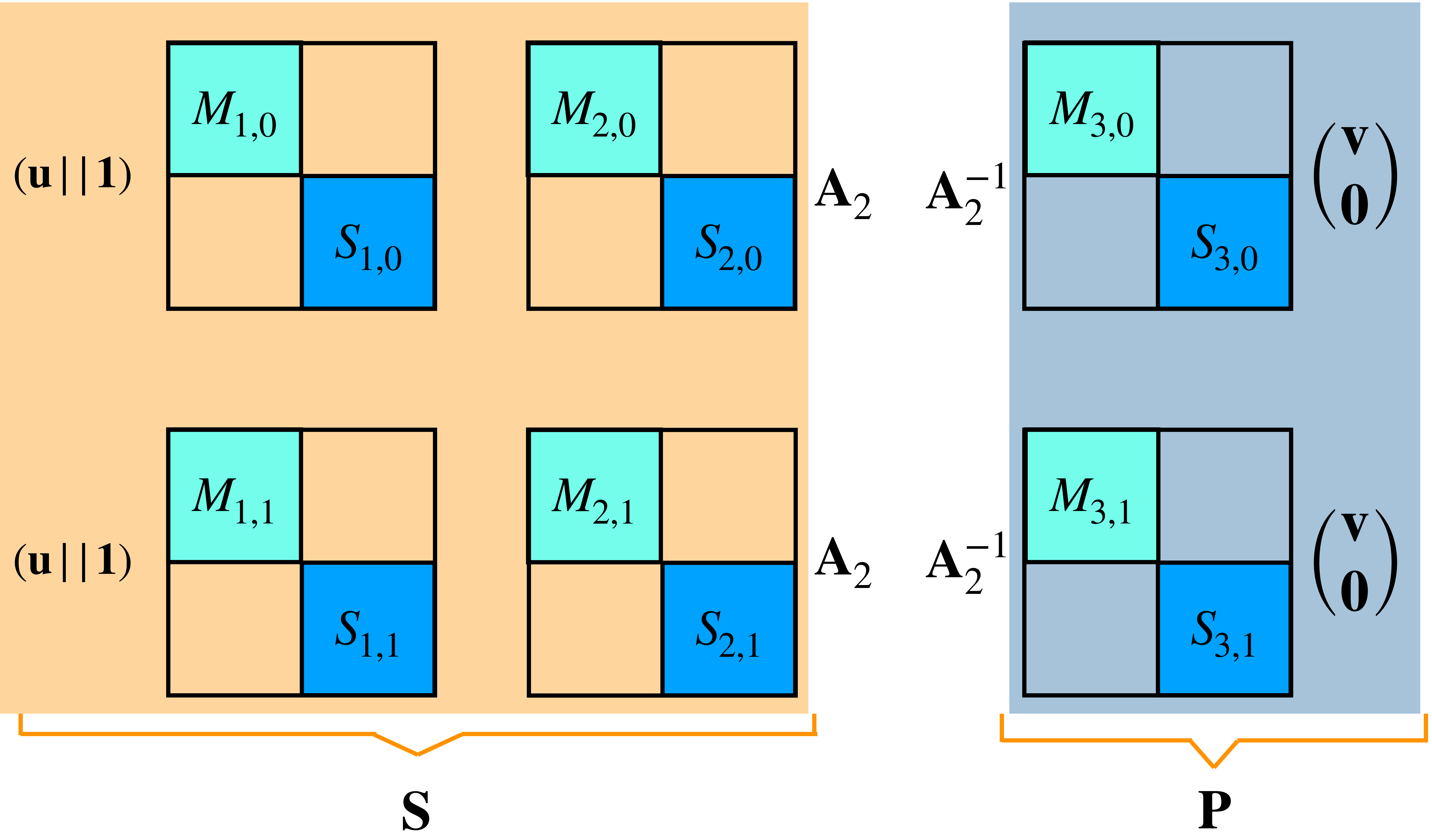
Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$ Two matrices

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c \times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

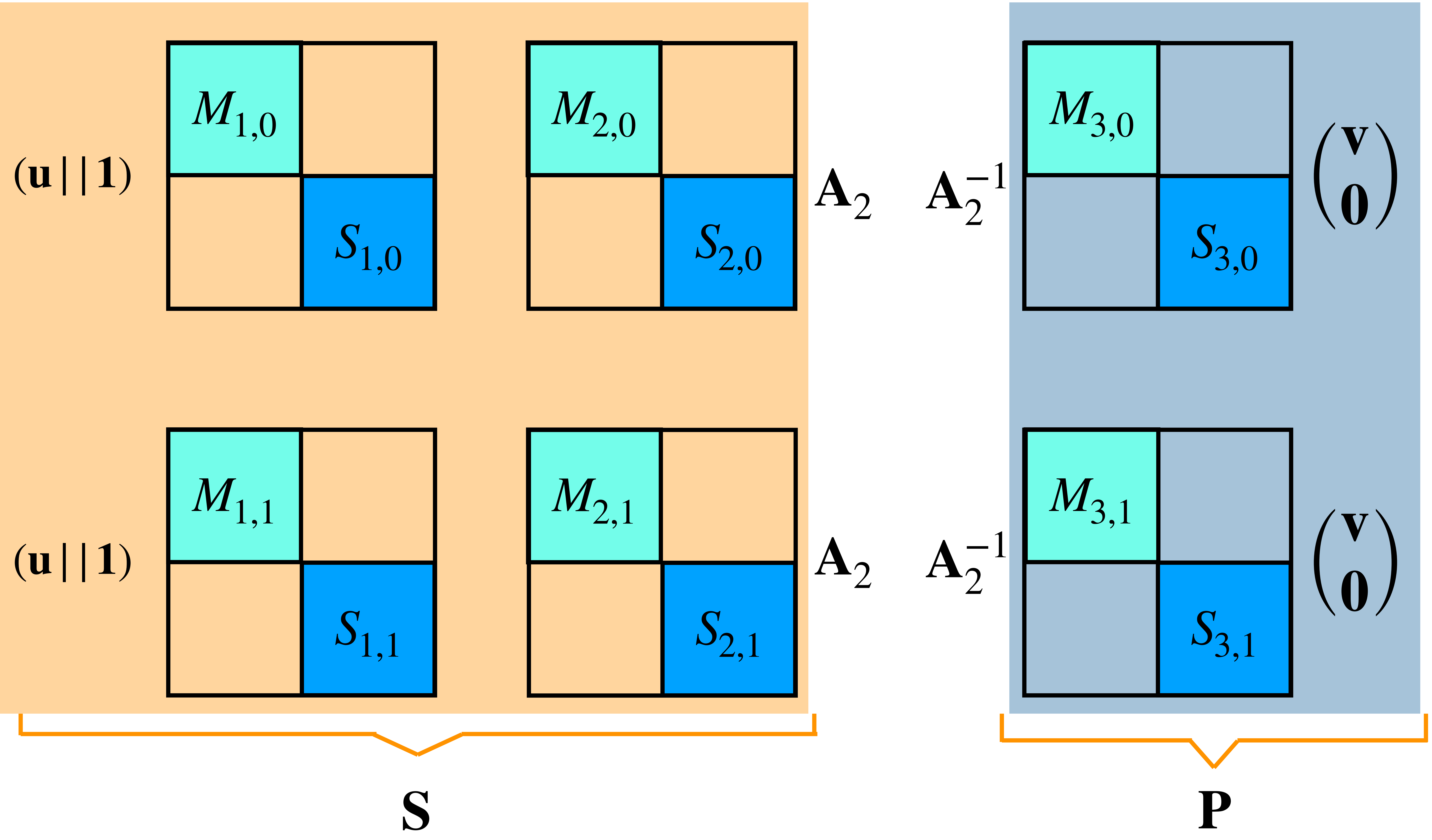
$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$ Two matrices

Then:

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c \times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

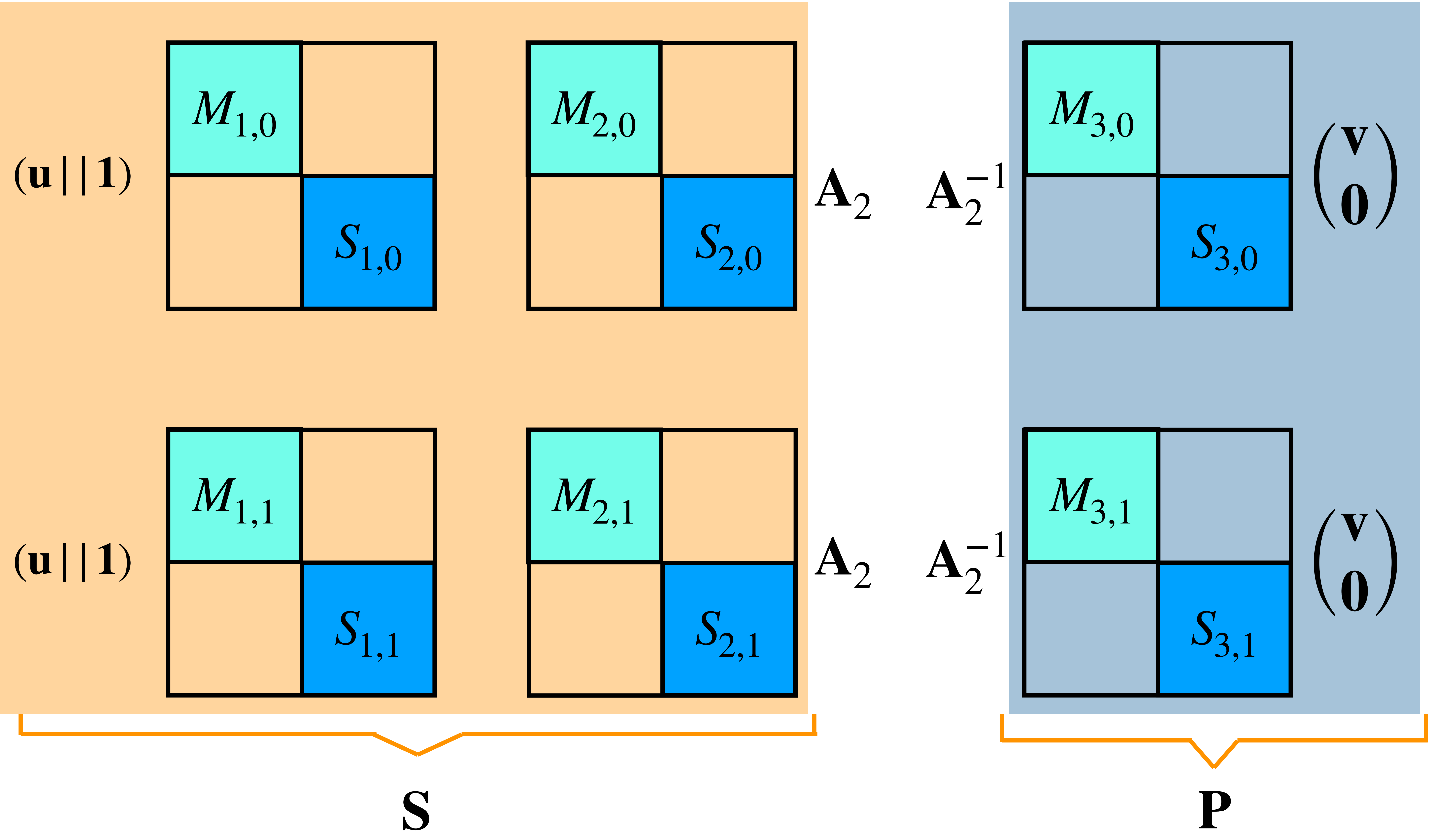
$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$ Two matrices

Then:

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

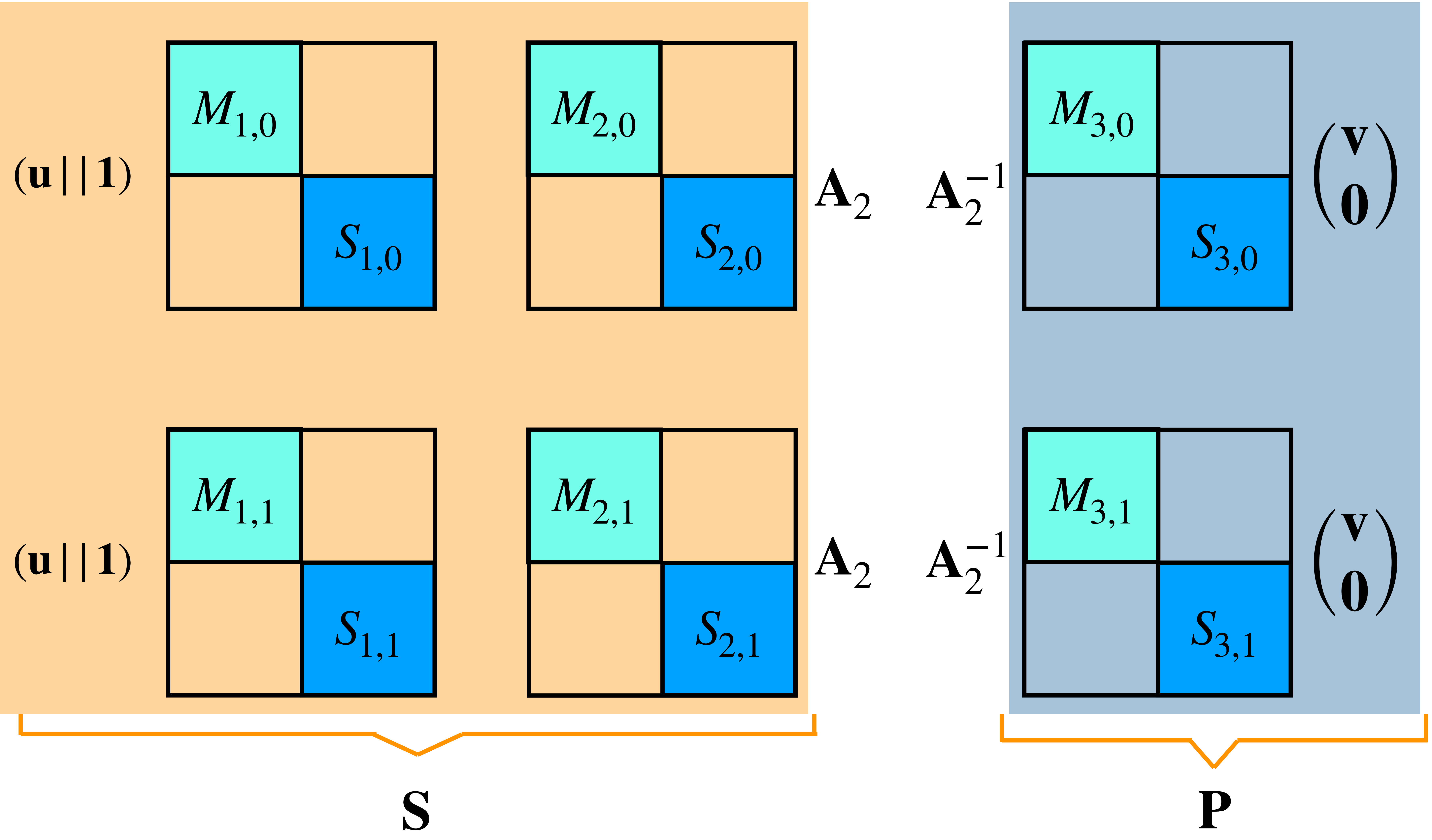
$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$ Two matrices

Then:

$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x}\in\{0,1\}^3} \approx \mathcal{U}$

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

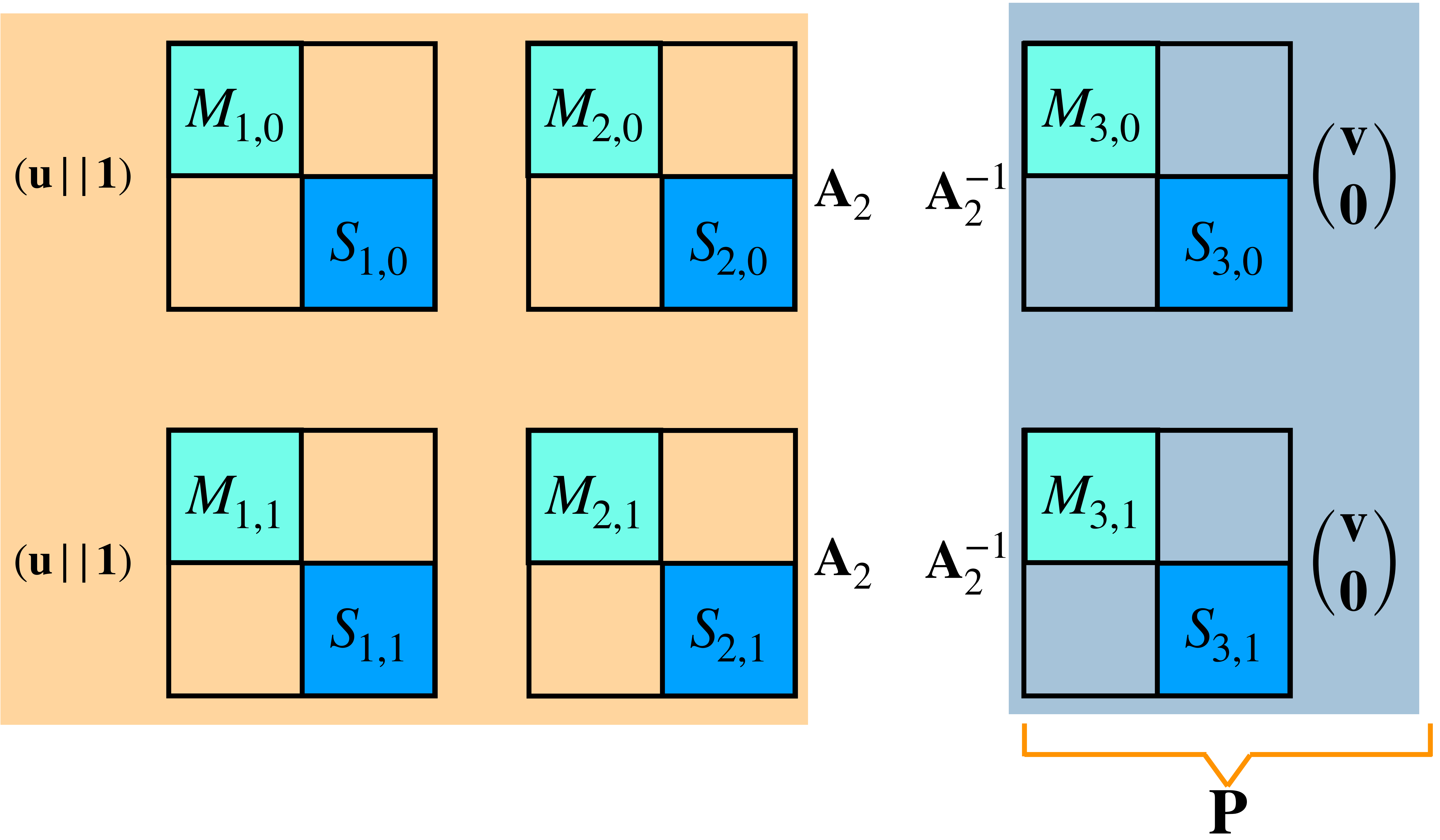
$\mathbf{P} =$ Two matrices

Then:

$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x}\in\{0,1\}^3} \approx \mathcal{U}$

Because F_k is a PRF

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c \times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

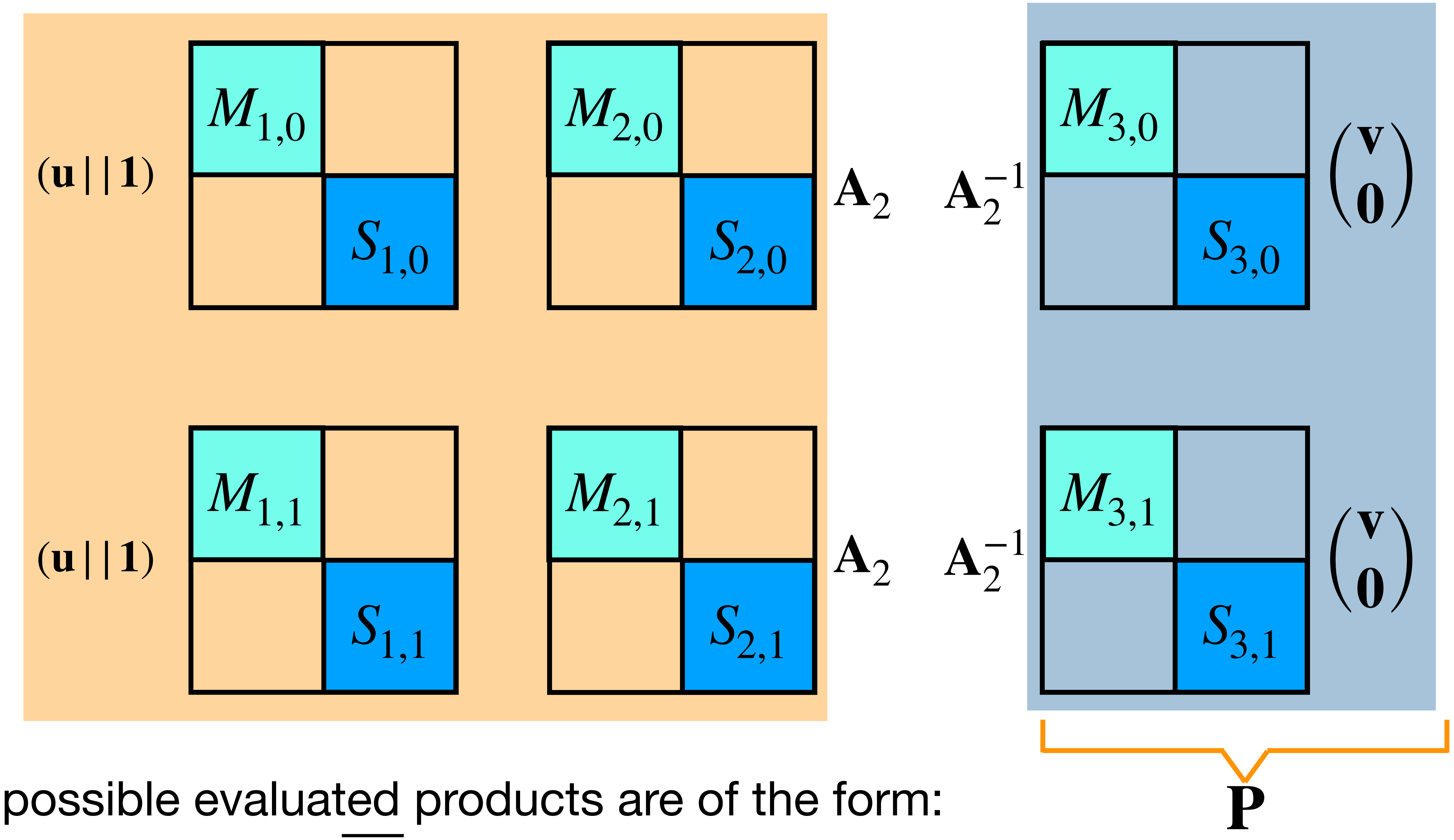
$\mathbf{P} =$ Two matrices

Then:

$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x} \in \{0,1\}^3} \approx \mathscr{U}$

Because F_k is a PRF

GGH15 Encodings



All possible evaluated products are of the form:

$$\mathbf{SB} = \{\mathbf{u}M_{1,x_1}M_{2,x_2}\overline{\mathbf{A}_2} + \mathbf{1}S_{1,x_1}S_{2,x_2}\underline{\mathbf{A}_2}\}_{x_1,x_2\in\{0,1\}}$$

- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

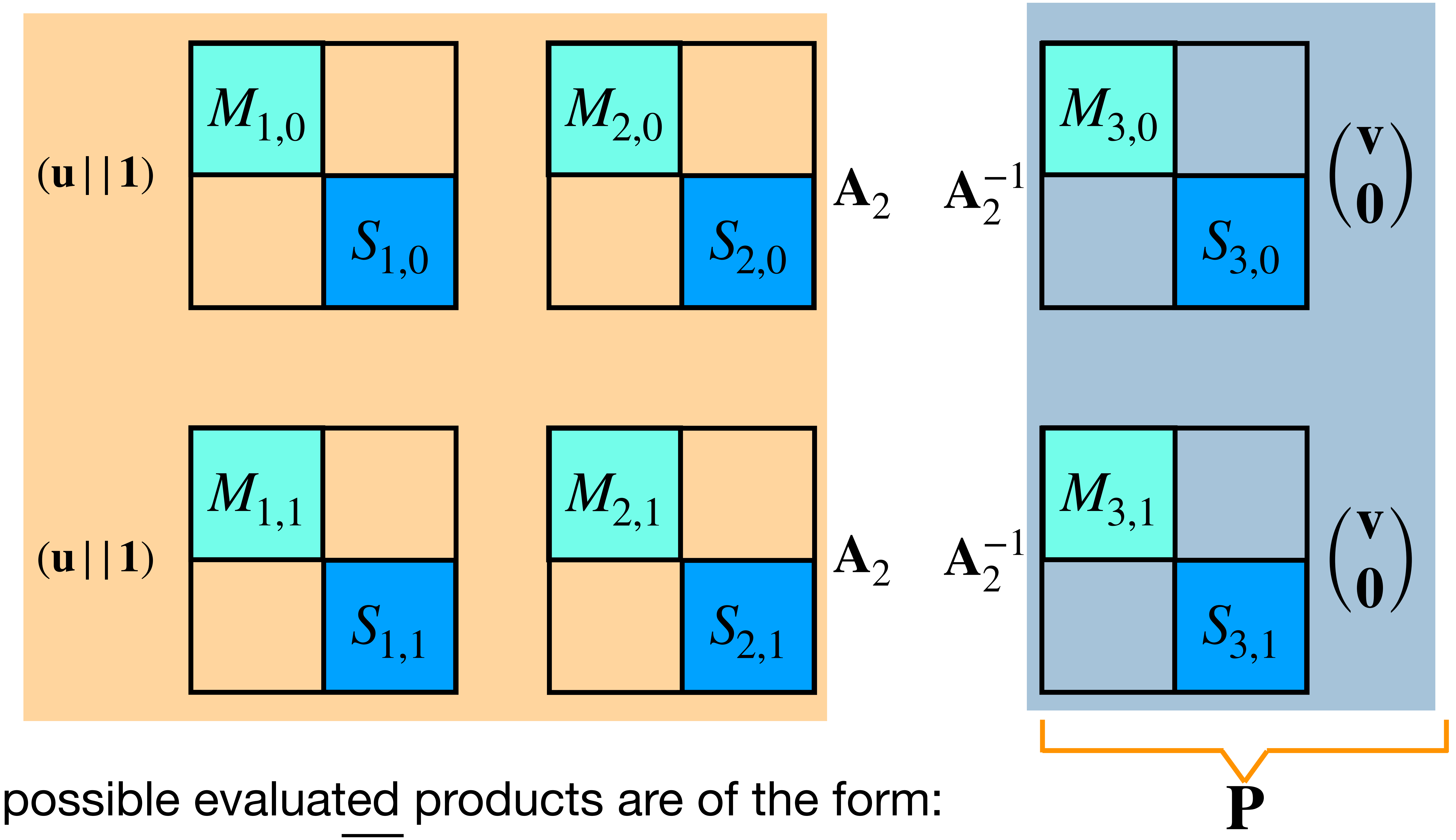
$\mathbf{P} =$ Two matrices

Then:

$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x}\in\{0,1\}^3} \approx \mathcal{U}$

Because F_k is a PRF

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

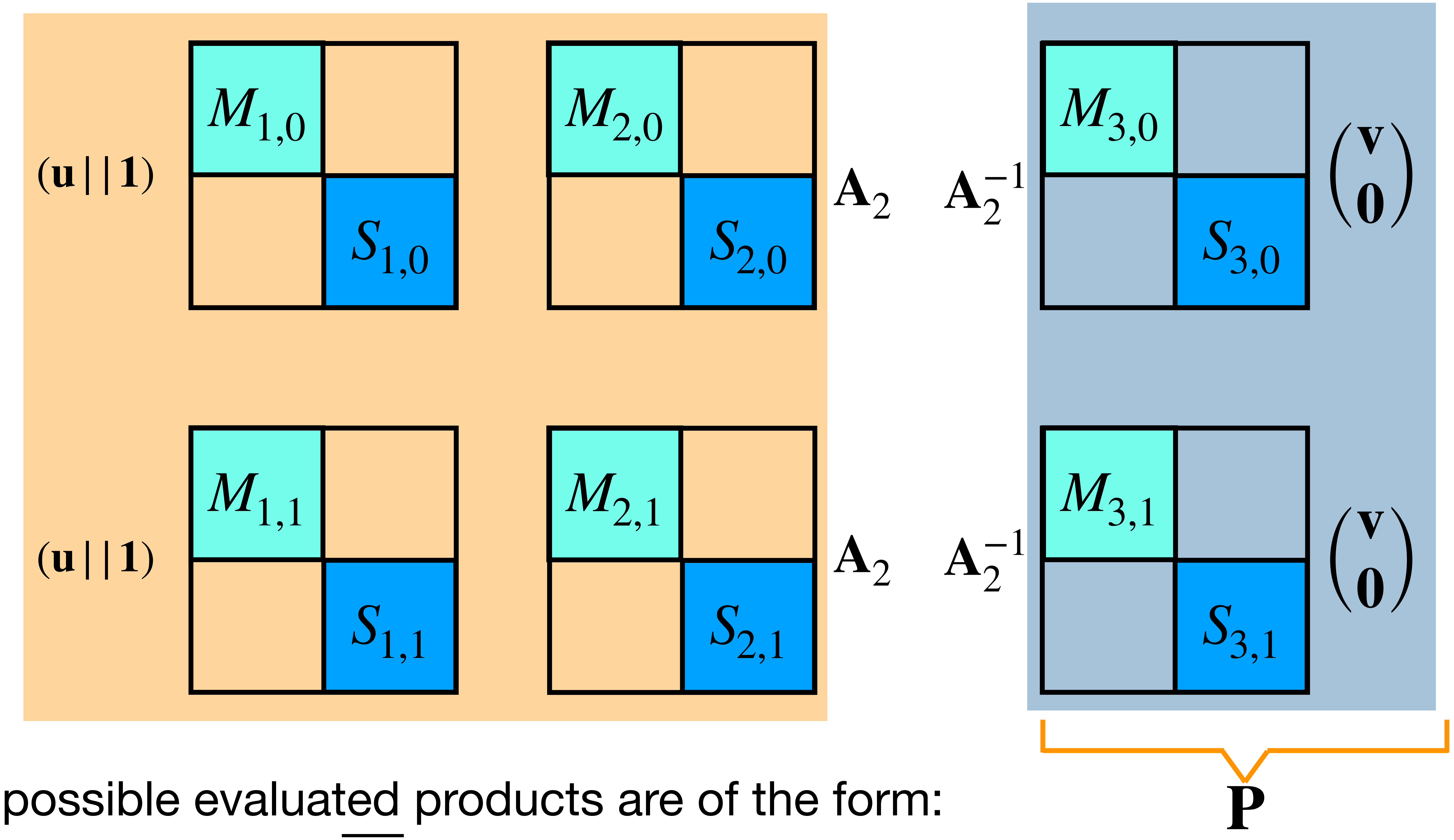
Set
$\mathbf{S} =$ All $2^2 = 4$ evaluations
$\mathbf{B} = \mathbf{A}_2$
$\mathbf{P} =$ Two matrices

Then:
$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x}\in\{0,1\}^3} \approx \mathscr{U}$
Because F_k is a PRF

All possible evaluated products are of the form:

$$\mathbf{SB} = \{\mathbf{u}M_{1,x_1}M_{2,x_2}\mathbf{A}_2 + \underbrace{\mathbf{1}S_{1,x_1}S_{2,x_2}\mathbf{A}_2}_{\text{pseudorandom (with noise) by LWE!}}\}_{x_1,x_2\in\{0,1\}}$$

GGH15 Encodings



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$ Two matrices

Then:

$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x}\in\{0,1\}^3} \approx \mathcal{U}$

Because F_k is a PRF

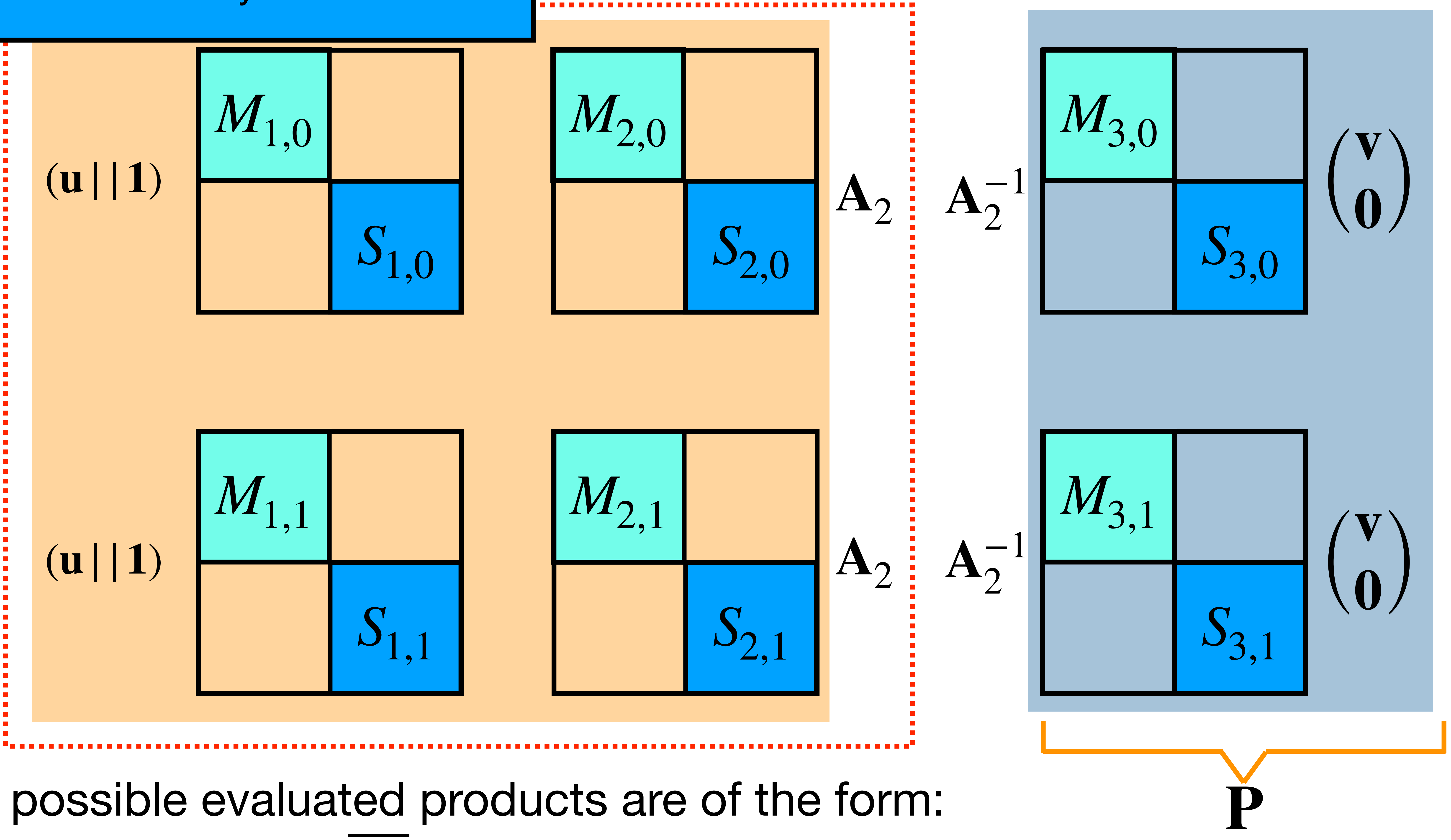
All possible evaluated products are of the form:

$$\mathbf{SB} = \{\mathbf{u}M_{1,x_1}M_{2,x_2}\mathbf{A}_2 + \underbrace{\mathbf{1}S_{1,x_1}S_{2,x_2}\mathbf{A}_2}_{\text{pseudorandom (with noise) by LWE!}}\}_{x_1,x_2\in\{0,1\}}$$

$$\boxed{\mathbf{SB},\mathbf{SP} \approx_c \mathcal{U},\mathcal{U} \Rightarrow \mathbf{SB},\mathbf{B}^{-1}(\mathbf{P}) \approx_c \mathcal{U},\mathbf{B}^{-1}(\mathbf{P})}$$

GGH15 Encodings

Pseudorandom by evasive LWE!



- Sample $S_{i,b} \leftarrow \chi^{c\times c}$ (i.e. small entries)
- Sample $\mathbf{A}_2$ with a trapdoor

Set

$\mathbf{S} =$ All $2^2 = 4$ evaluations

$\mathbf{B} = \mathbf{A}_2$

$\mathbf{P} =$ Two matrices

Then:

$\mathbf{SP} = \{F_k(\mathbf{x})\}_{\mathbf{x}\in\{0,1\}^3} \approx \mathcal{U}$

Because F_k is a PRF

All possible evaluated products are of the form:

$$\mathbf{SB} = \{\mathbf{u}M_{1,x_1}M_{2,x_2}\overline{\mathbf{A}_2} + \underbrace{\mathbf{1}S_{1,x_1}S_{2,x_2}\underline{\mathbf{A}_2}}_{\text{pseudorandom (with noise) by LWE!}}\}_{x_1,x_2\in\{0,1\}}$$

$$\boxed{\mathbf{SB}, \mathbf{SP} \approx_c \mathcal{U}, \mathcal{U} \Rightarrow \mathbf{SB}, \mathbf{B}^{-1}(\mathbf{P}) \approx_c \mathcal{U}, \mathbf{B}^{-1}(\mathbf{P})}$$

GGH15 Encodings

$$(\mathbf{u}||\mathbf{1}) \begin{pmatrix} M_{1,0} & \\ & S_{1,0} \end{pmatrix} \qquad \begin{pmatrix} M_{2,0} & \\ & S_{2,0} \end{pmatrix} \mathbf{A}_2 \quad \mathbf{A}_2^{-1} \left(\begin{pmatrix} M_{3,0} & \\ & S_{3,0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} \right)$$

$$(\mathbf{u}||\mathbf{1}) \begin{pmatrix} M_{1,1} & \\ & S_{1,1} \end{pmatrix} \qquad \begin{pmatrix} M_{2,1} & \\ & S_{2,1} \end{pmatrix} \mathbf{A}_2 \quad \mathbf{A}_2^{-1} \left(\begin{pmatrix} M_{3,1} & \\ & S_{3,1} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} \right)$$

- Repeatedly apply evasive LWE!
- Shrunk the size from 2^h evaluated products to size to $2h$ matrices.

GGH15 Encodings

$$\left((\mathbf{u}||\mathbf{1}) \begin{bmatrix} M_{1,0} & \\ & S_{1,0} \end{bmatrix} \mathbf{A}_1 \right) \mathbf{A}_1^{-1} \left(\begin{bmatrix} M_{2,0} & \\ & S_{2,0} \end{bmatrix} \mathbf{A}_2 \right) \mathbf{A}_2^{-1} \left(\begin{bmatrix} M_{3,0} & \\ & S_{3,0} \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} \right)$$

$$\left((\mathbf{u}||\mathbf{1}) \begin{bmatrix} M_{1,1} & \\ & S_{1,1} \end{bmatrix} \mathbf{A}_1 \right) \mathbf{A}_1^{-1} \left(\begin{bmatrix} M_{2,1} & \\ & S_{2,1} \end{bmatrix} \mathbf{A}_2 \right) \mathbf{A}_2^{-1} \left(\begin{bmatrix} M_{3,1} & \\ & S_{3,1} \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \end{pmatrix} \right)$$

- Repeatedly apply evasive LWE!
- Shrunk the size from 2^h evaluated products to size to $2h$ matrices.