SLAP 👋

Succinct Lattice-Based Polynomial Commitment Schemes from Standard Assumptions (2023/1469)

Giacomo Fenzi @ EPFL

Joint work with:
Martin Albrecht
Ngoc Khanh Nguyen

Oleksandra Lapiha

Motivation

SNARKs

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)

SNARKs

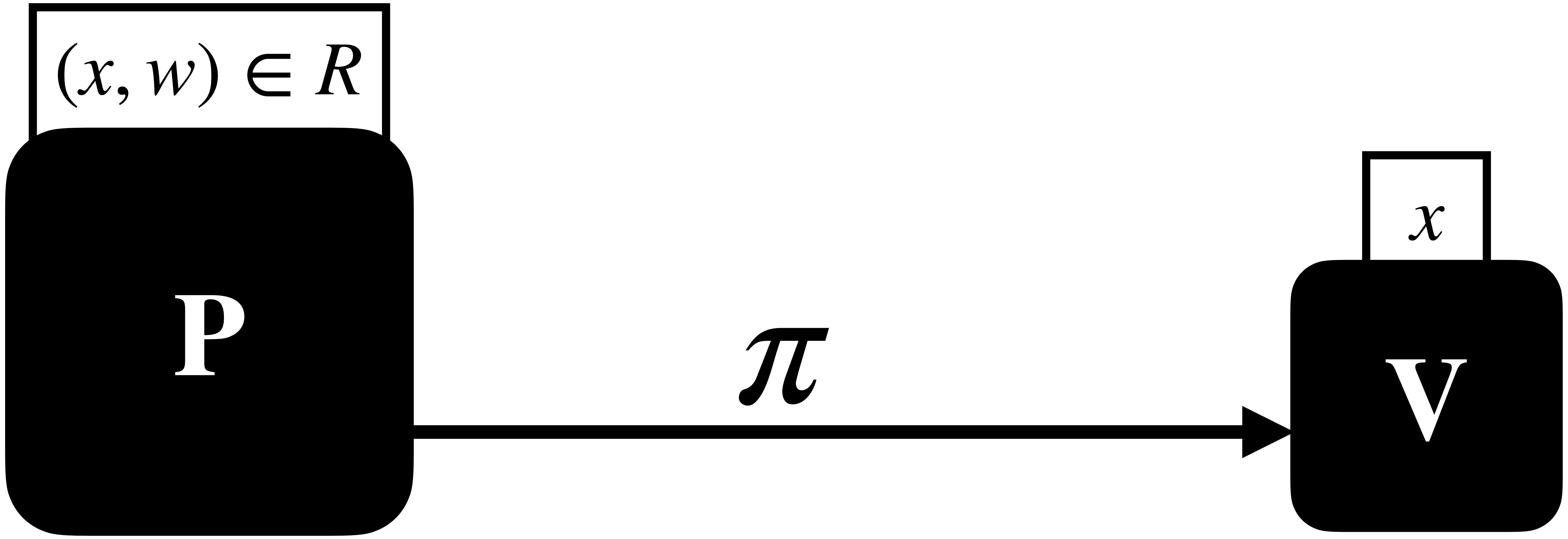
(Succinct Non-Interactive ARguments of Knowledge)

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)

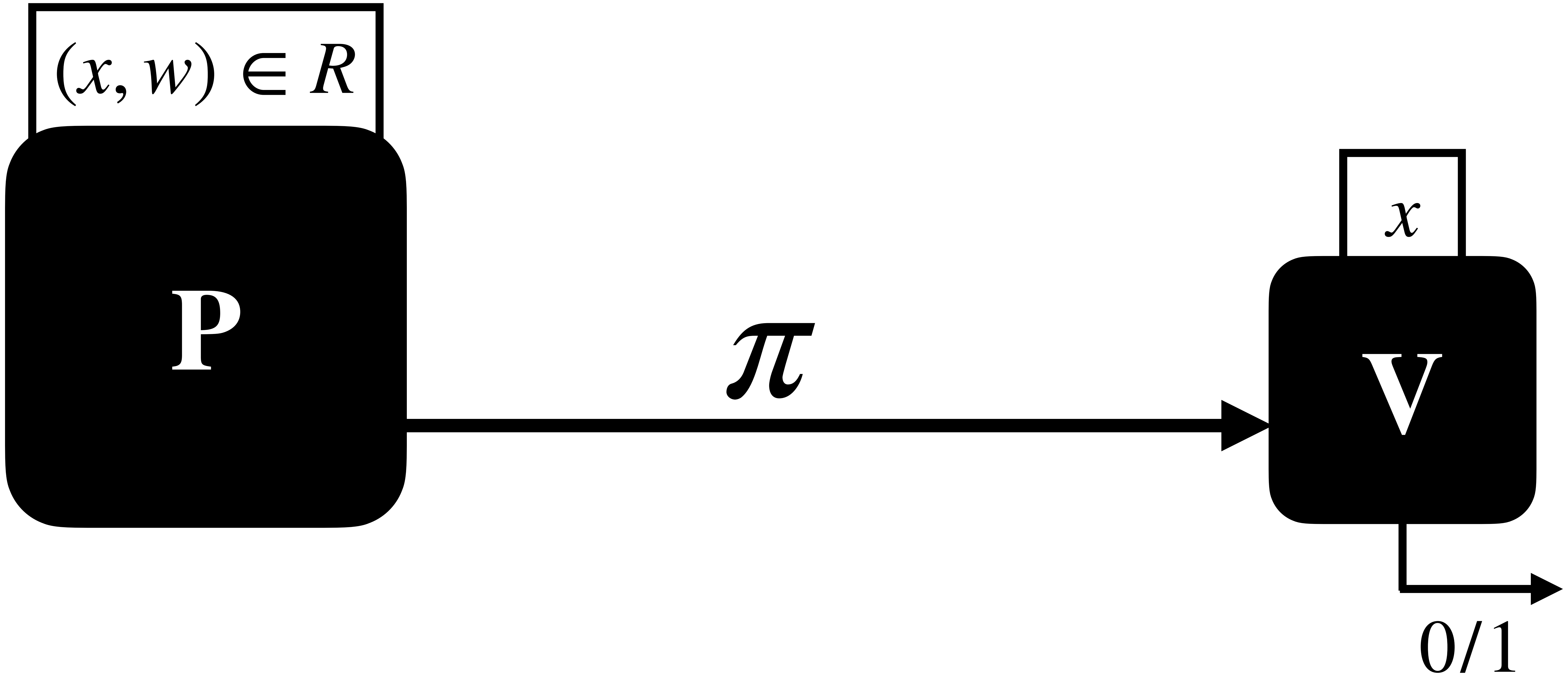
SNARKs

(Succinct Non-Interactive ARguments of Knowledge)



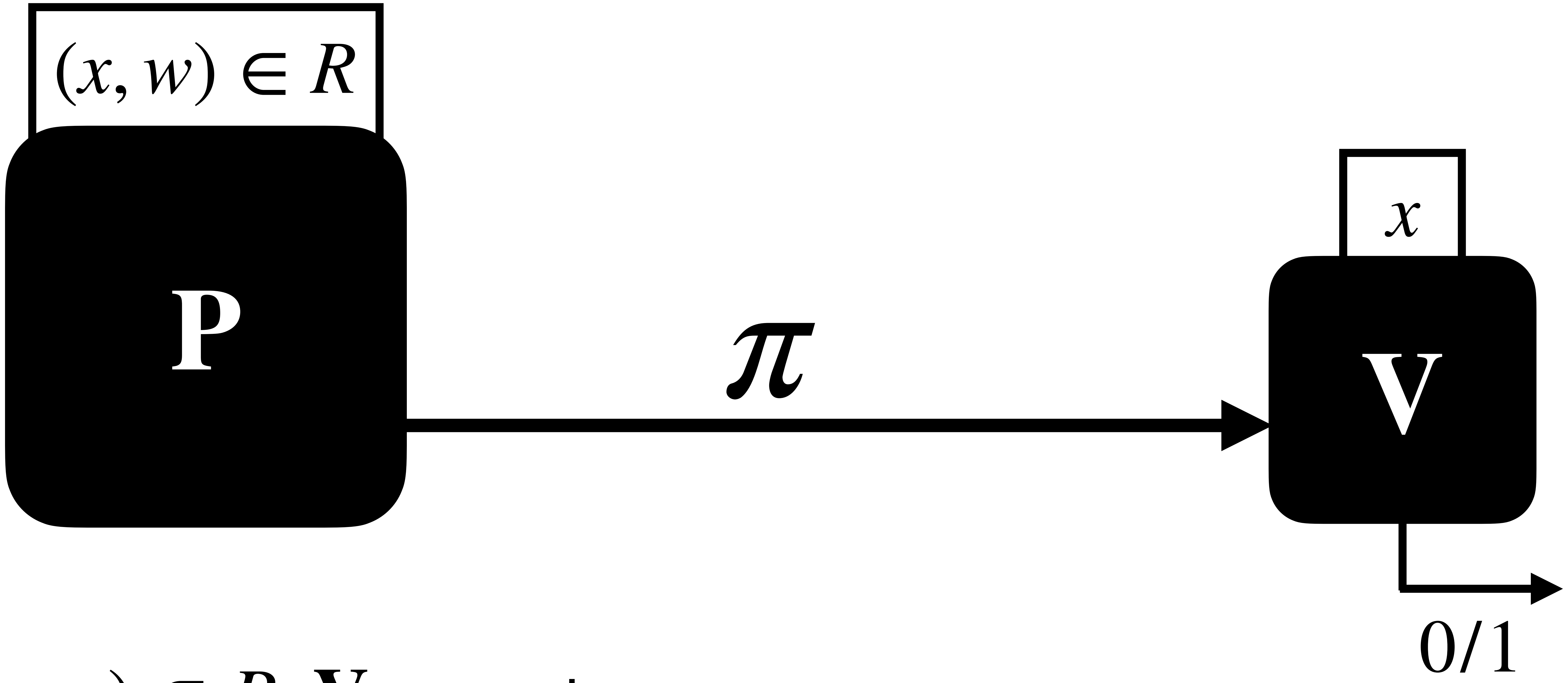
SNARKs

(Succinct Non-Interactive ARguments of Knowledge)



SNARKs

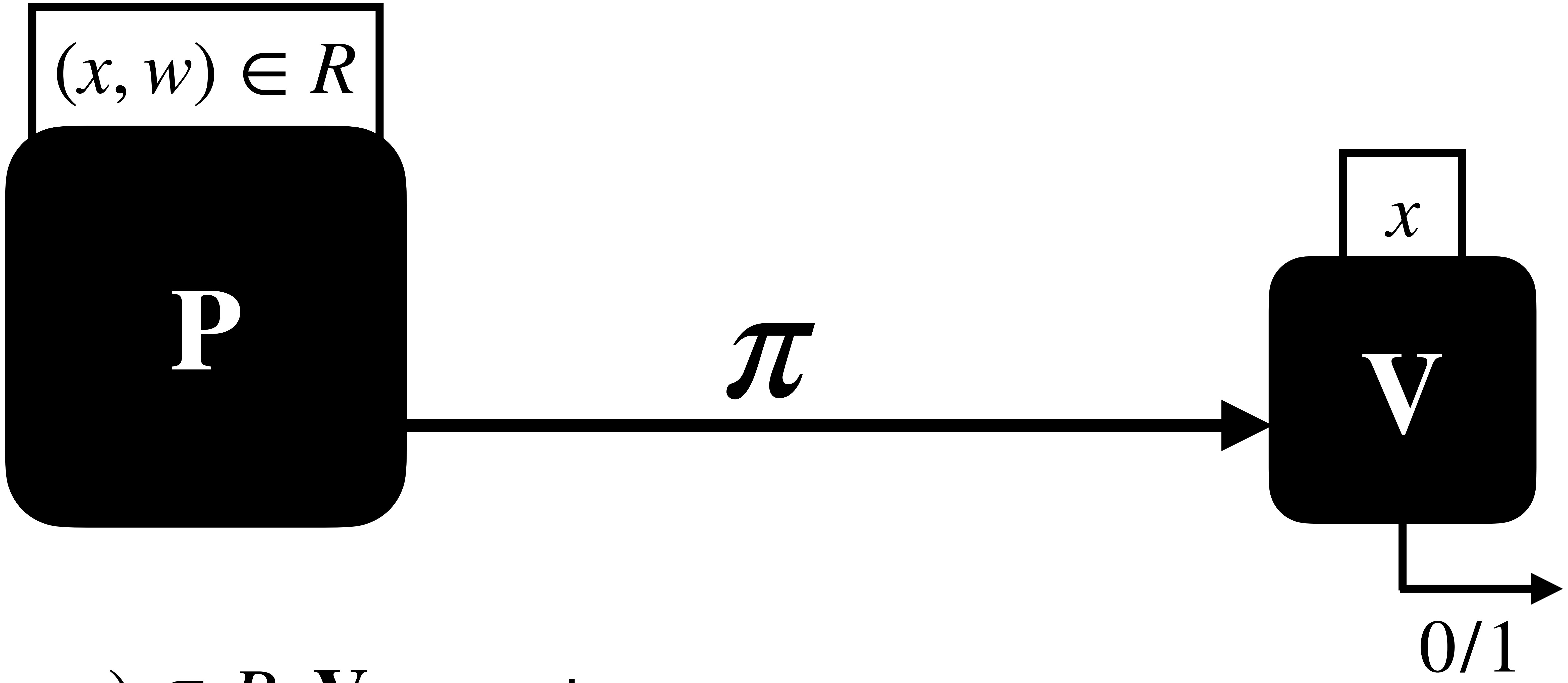
(Succinct Non-Interactive ARguments of Knowledge)



Complete: if $(x, w) \in R$, $\mathbf{V}$ accepts.

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)

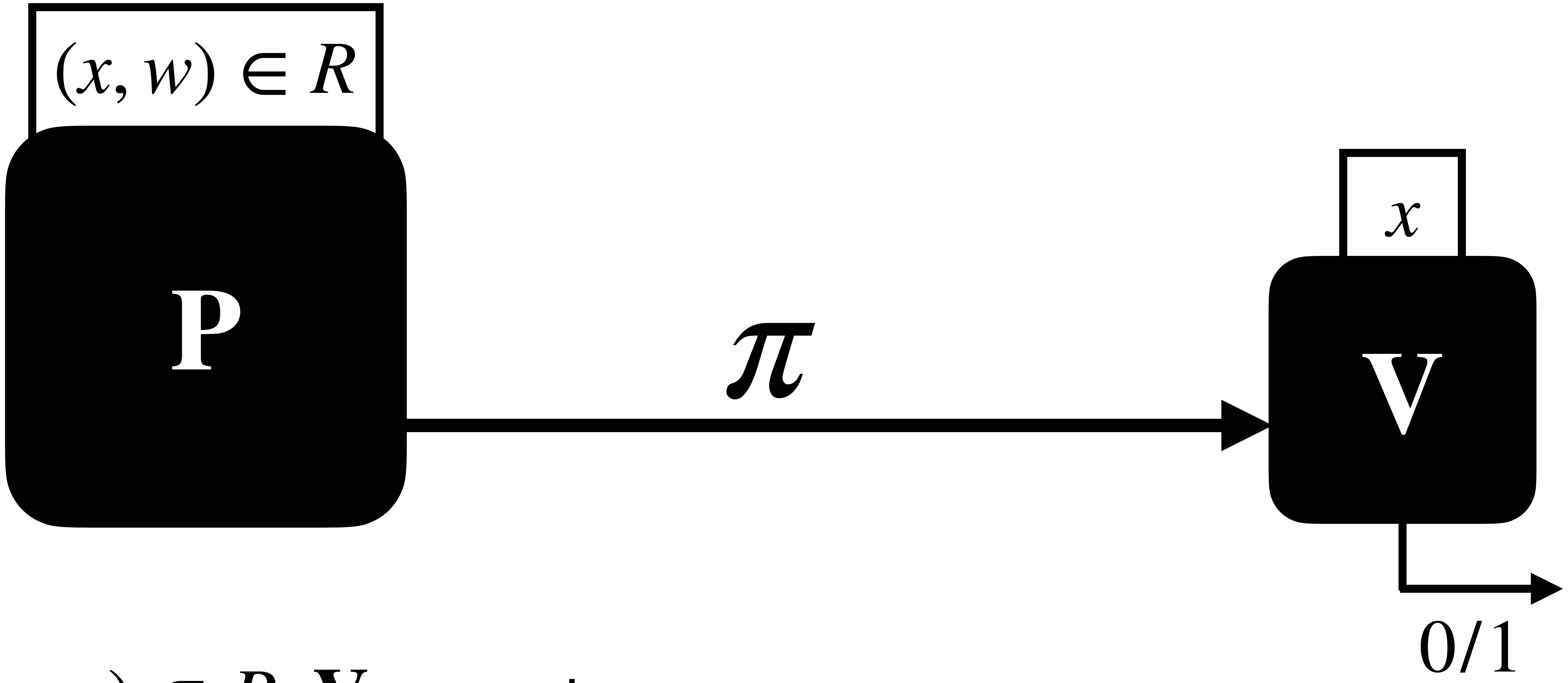


Complete: if $(x, w) \in R$, $\mathbf{V}$ accepts.

Non-interactive: $\mathbf{P}$ sends a single message.

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)



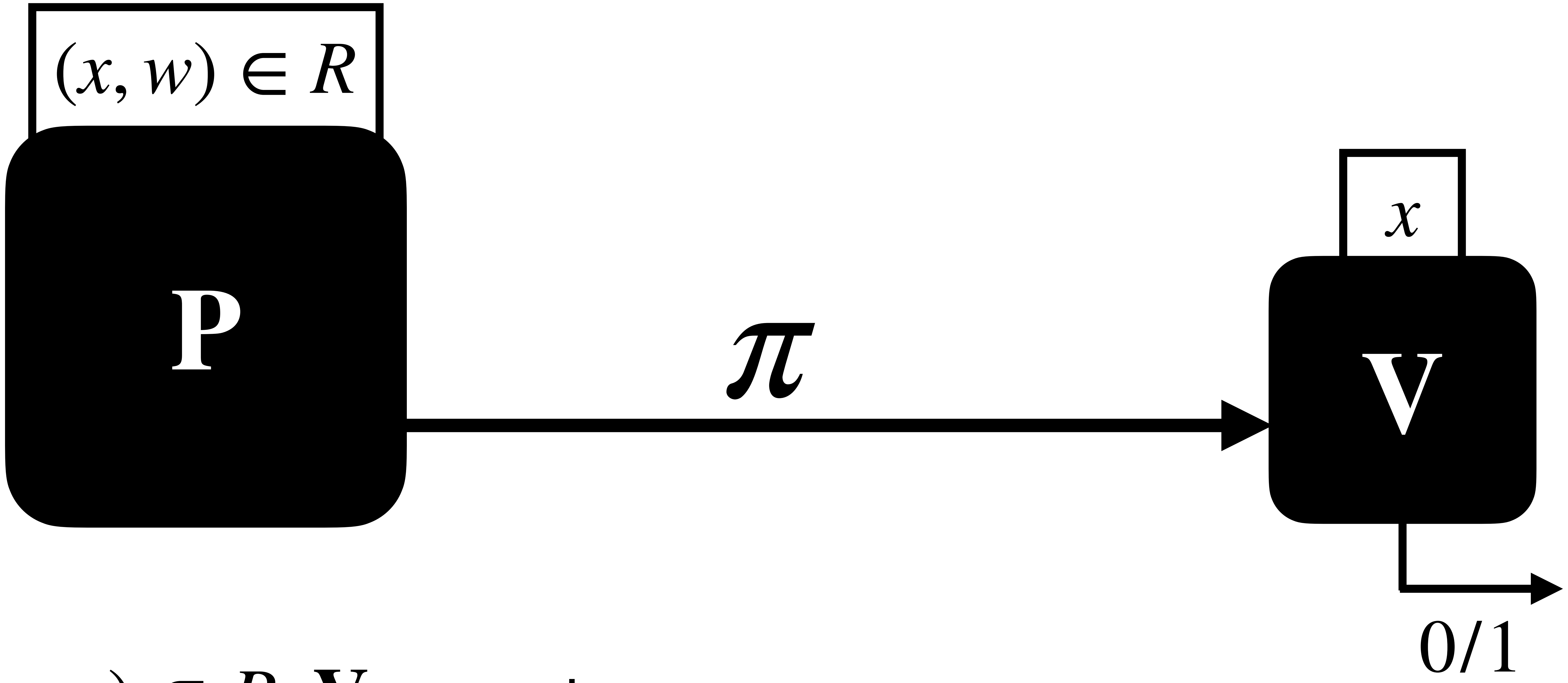
Complete: if $(x, w) \in R$, $\mathbf{V}$ accepts.

Non-interactive: $\mathbf{P}$ sends a single message.

Succinct: $|\pi| \ll |w|$ and verifier is fast.

SNARKs

(Succinct Non-Interactive ARguments of Knowledge)



Complete: if $(x, w) \in R$, $\mathbf{V}$ accepts.

Non-interactive: $\mathbf{P}$ sends a single message.

Succinct: $|\pi| \ll |w|$ and verifier is fast.

Knowledge Sound: if $\mathbf{V}(x, \pi) = 1$, can extract w such that $(x, w) \in R$

Constructing SNARKs

The modular way™

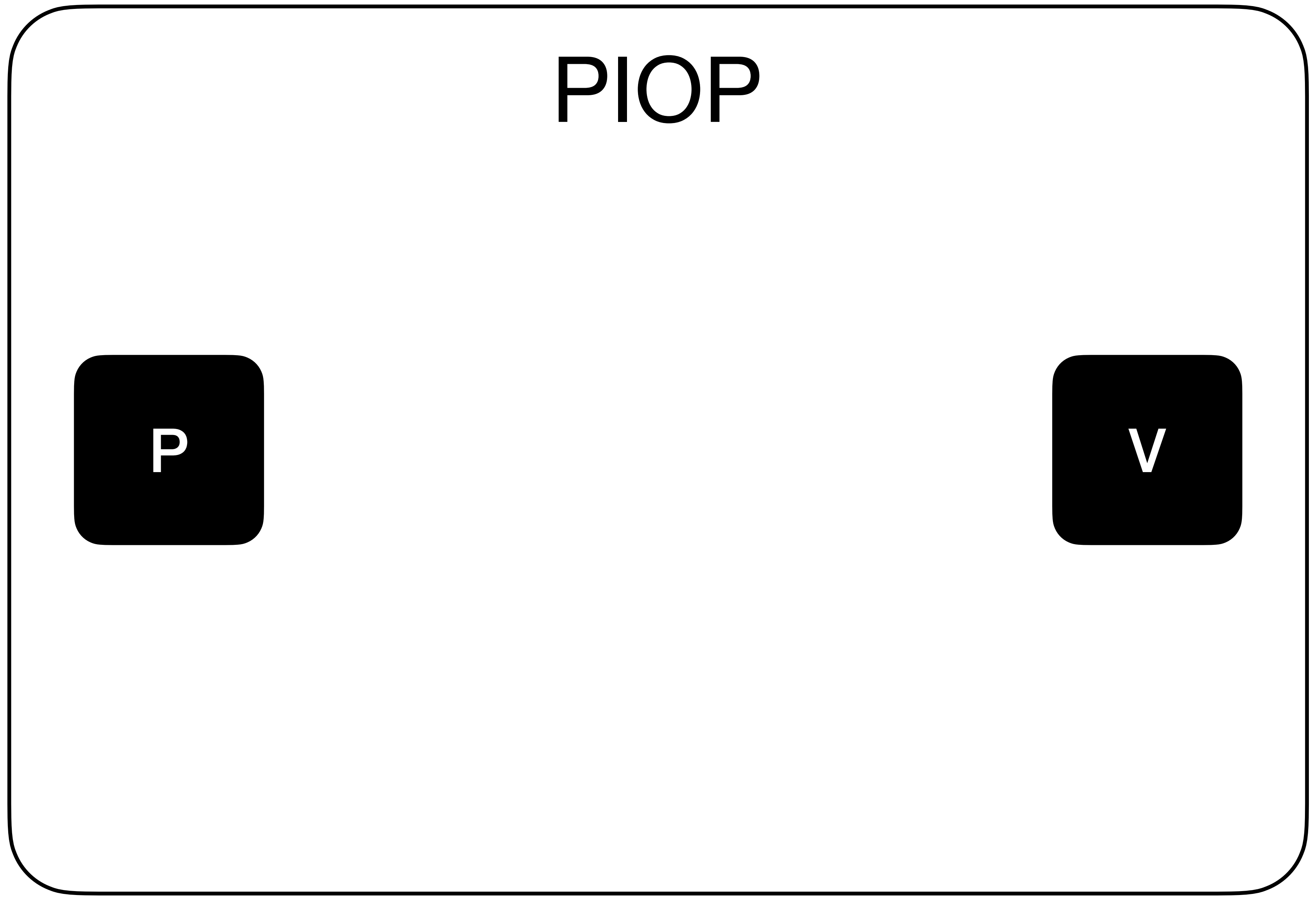
Constructing SNARKs

The modular way™

PIOP

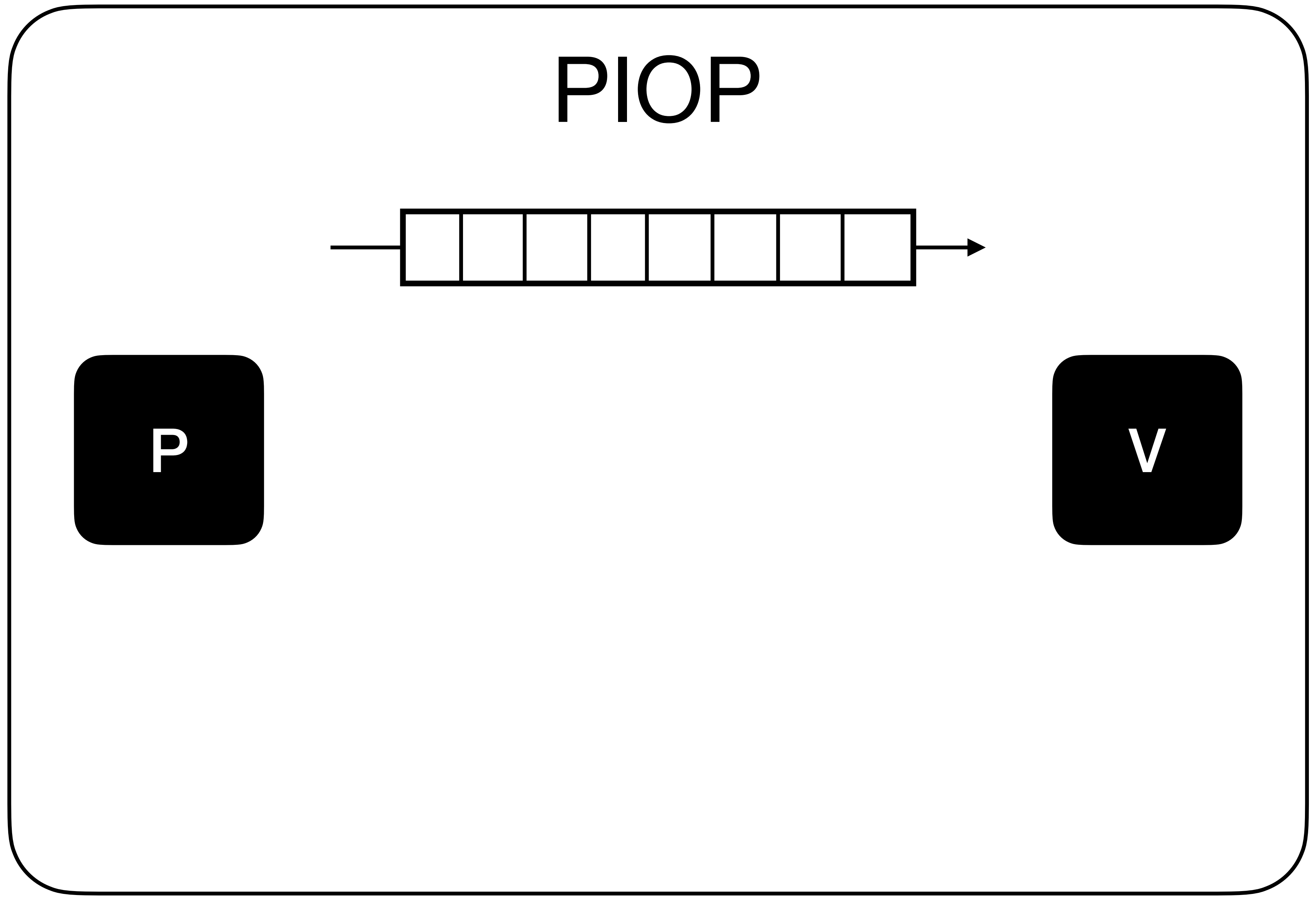
Constructing SNARKs

The modular way™



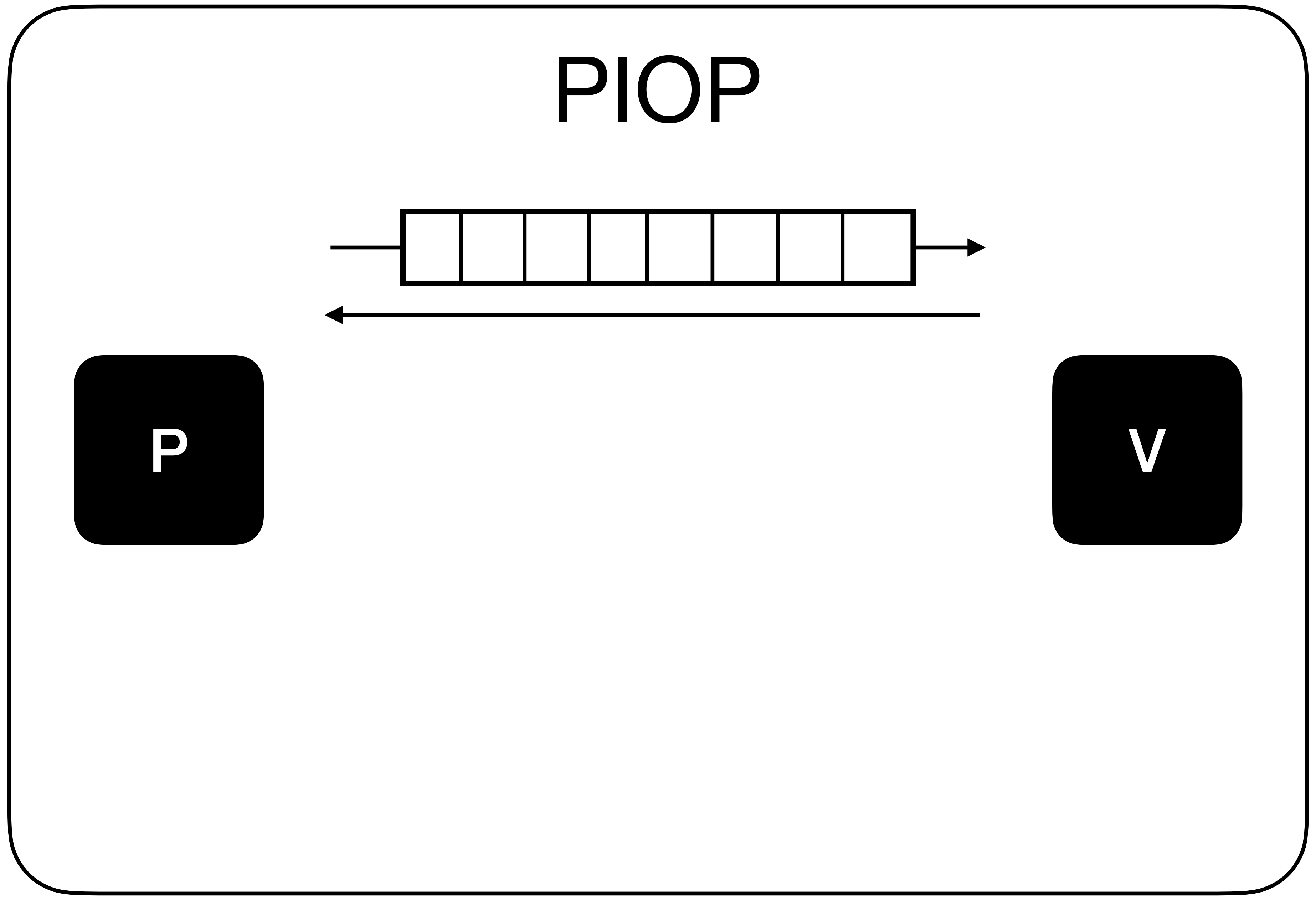
Constructing SNARKs

The modular way™



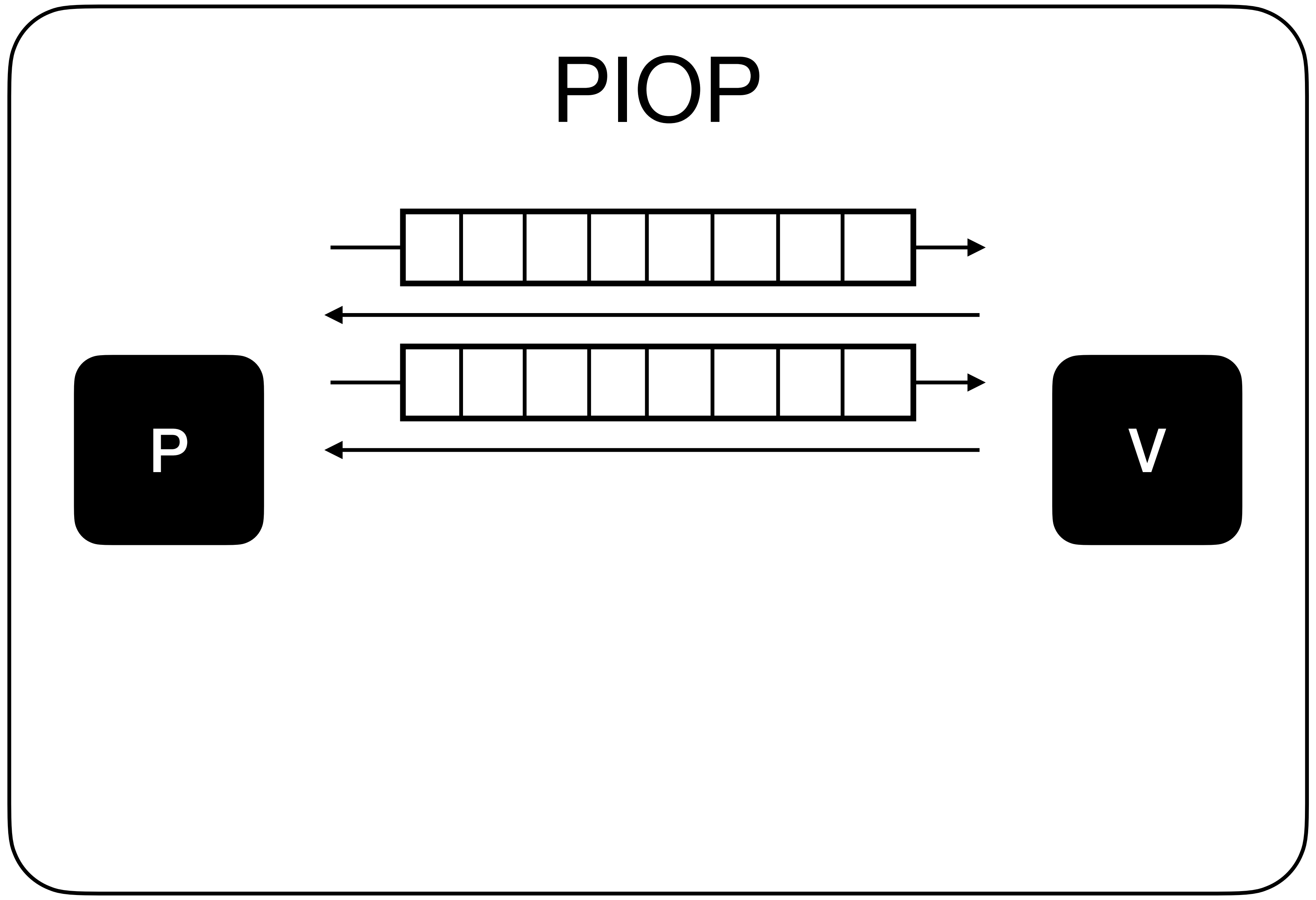
Constructing SNARKs

The modular way™



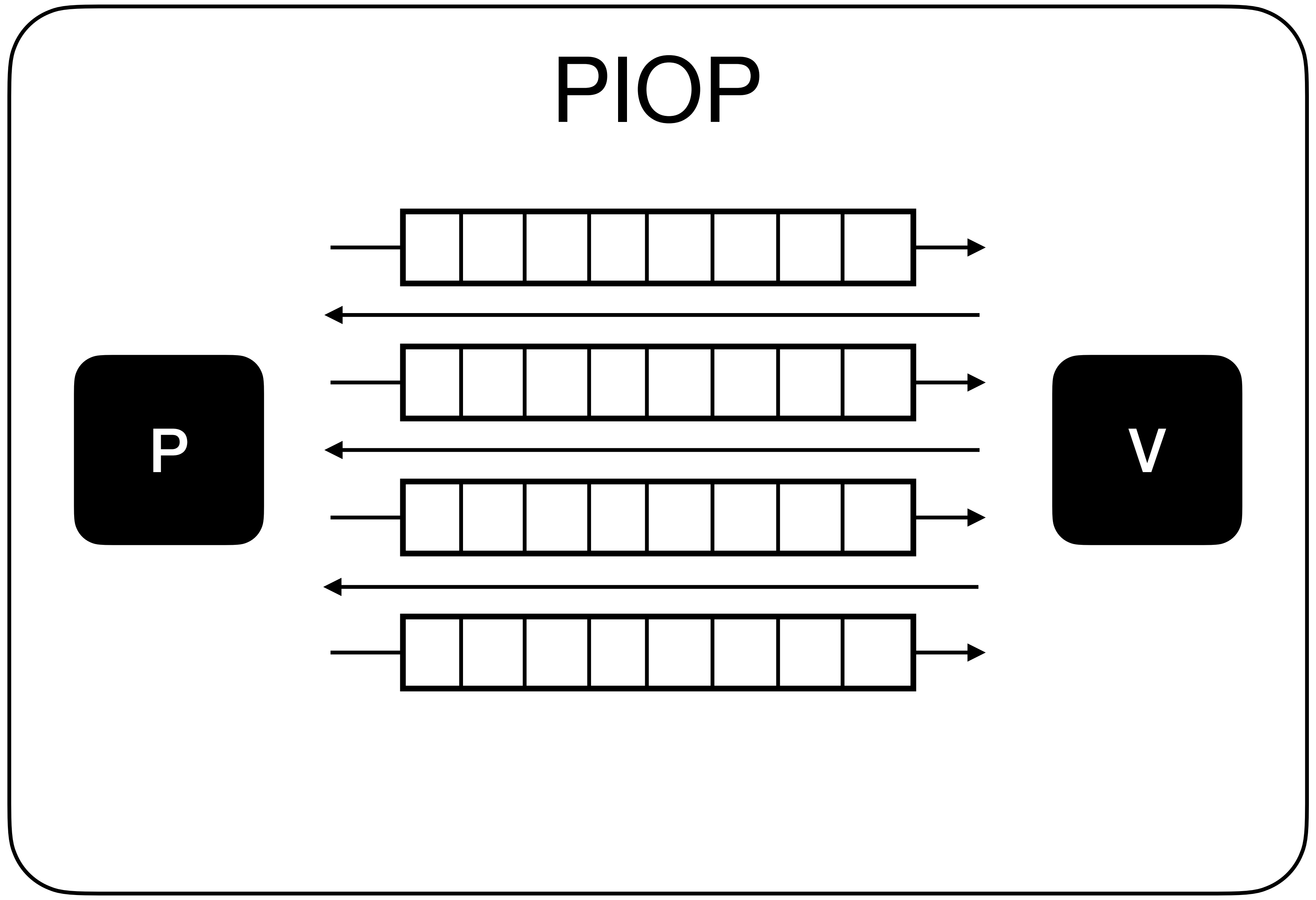
Constructing SNARKs

The modular way™



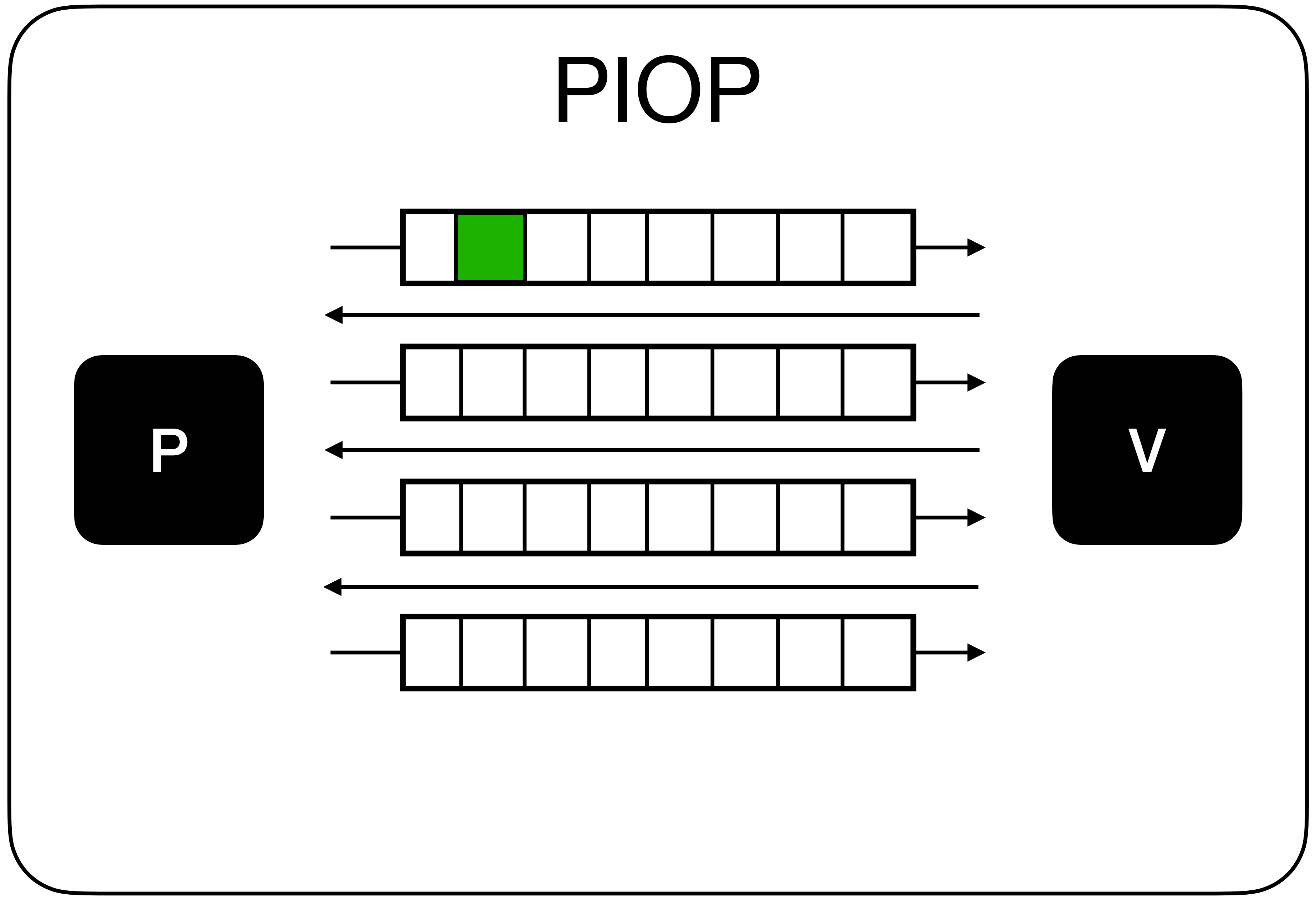
Constructing SNARKs

The modular way™



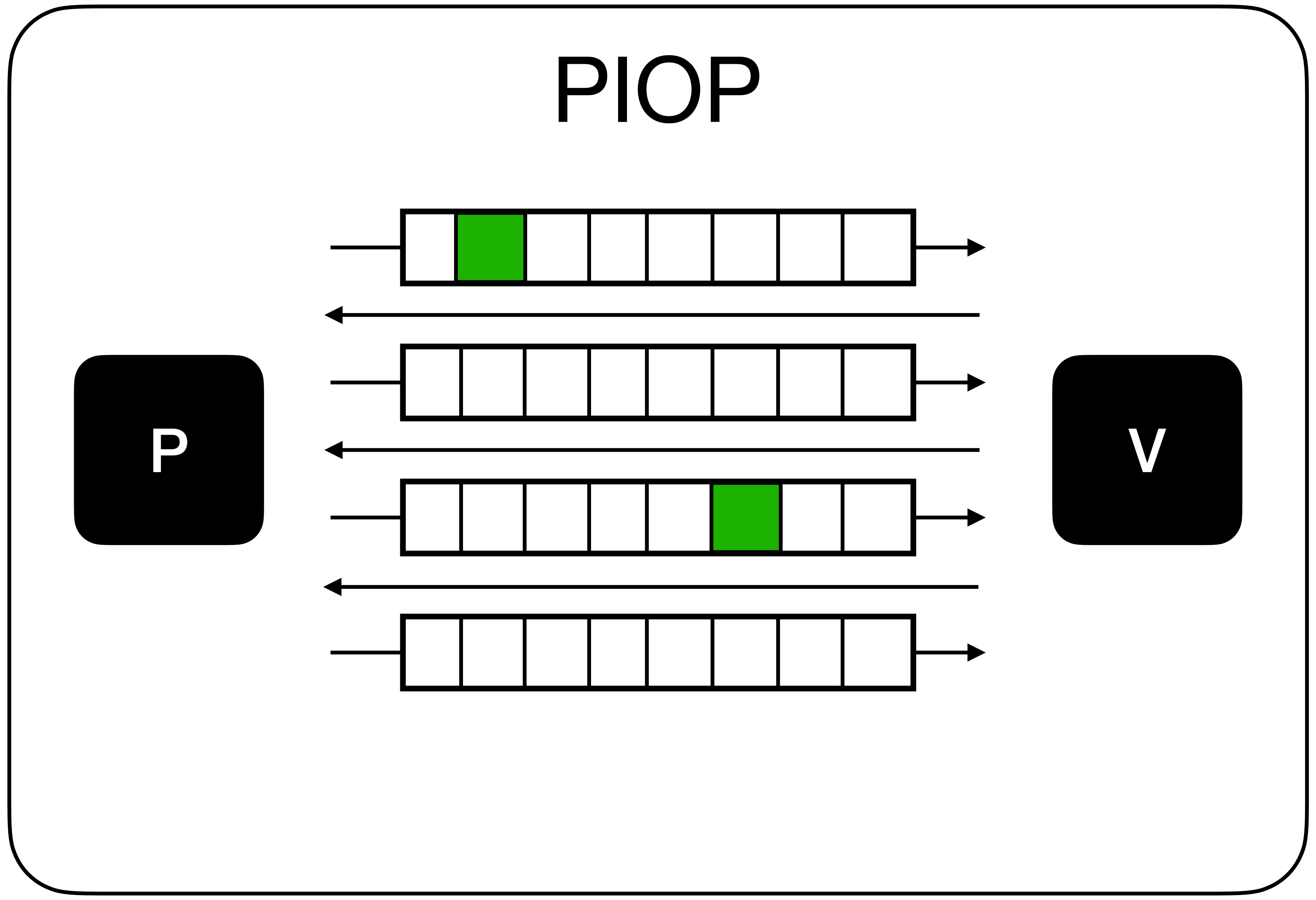
Constructing SNARKs

The modular way™



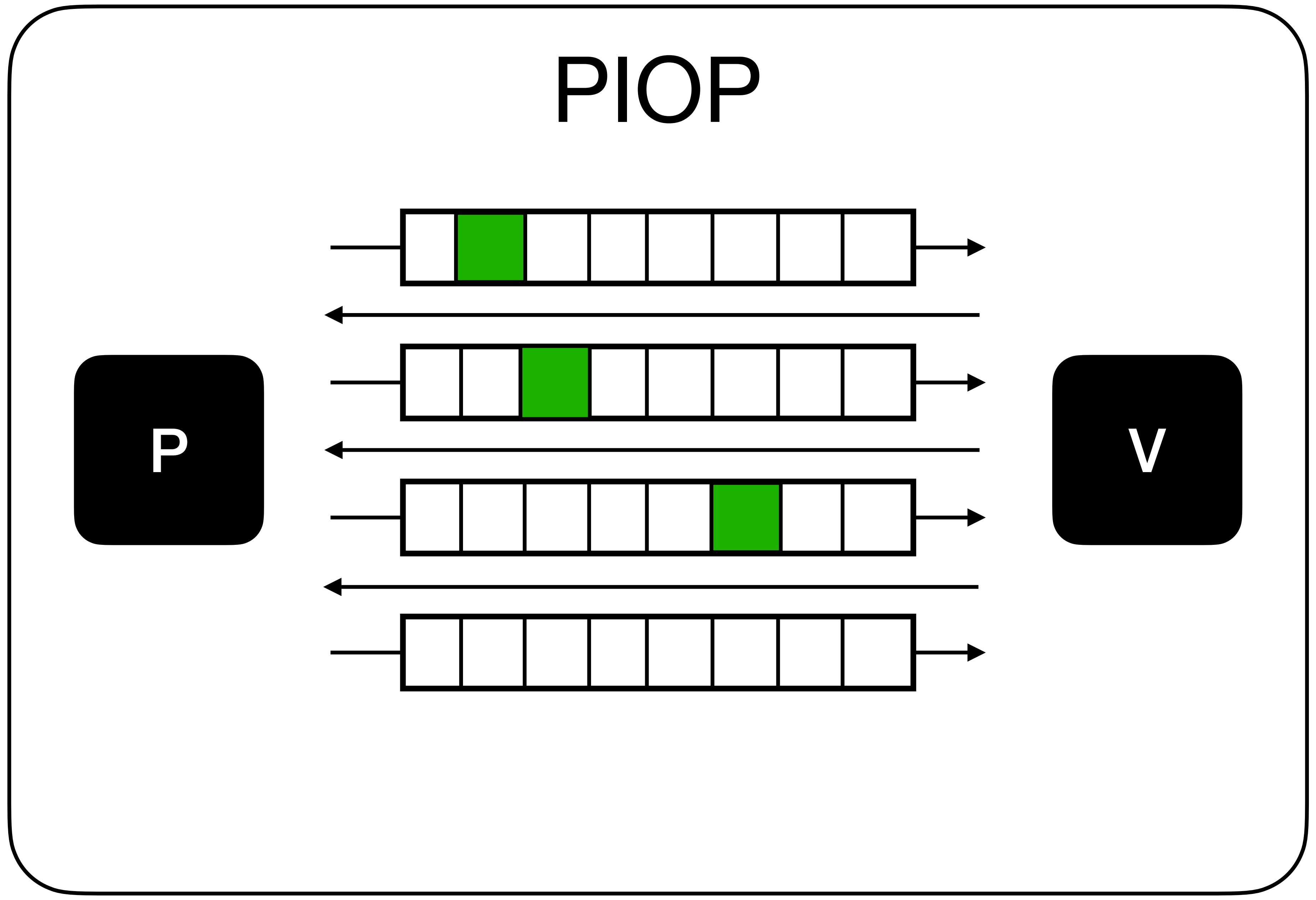
Constructing SNARKs

The modular way™



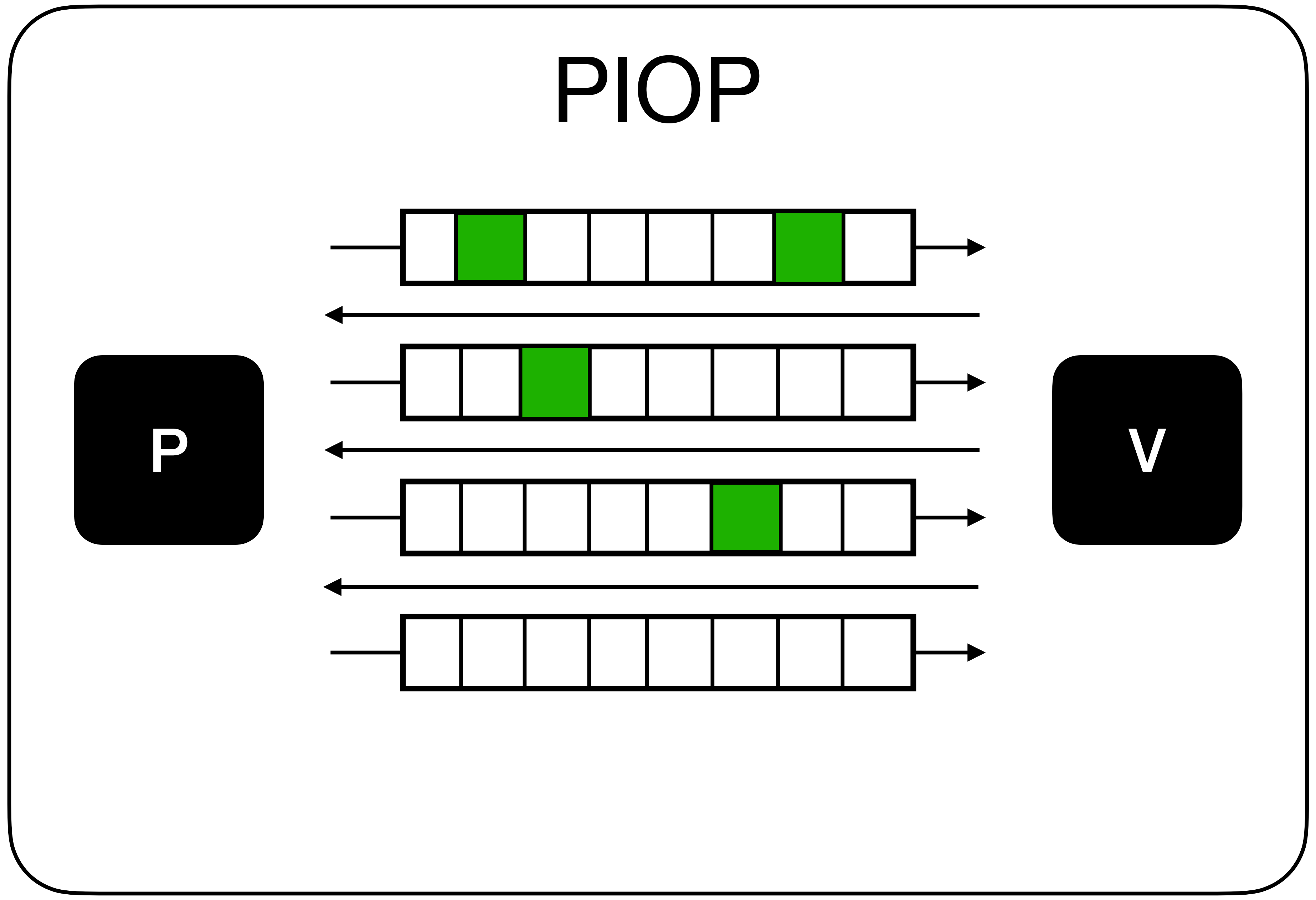
Constructing SNARKs

The modular way™



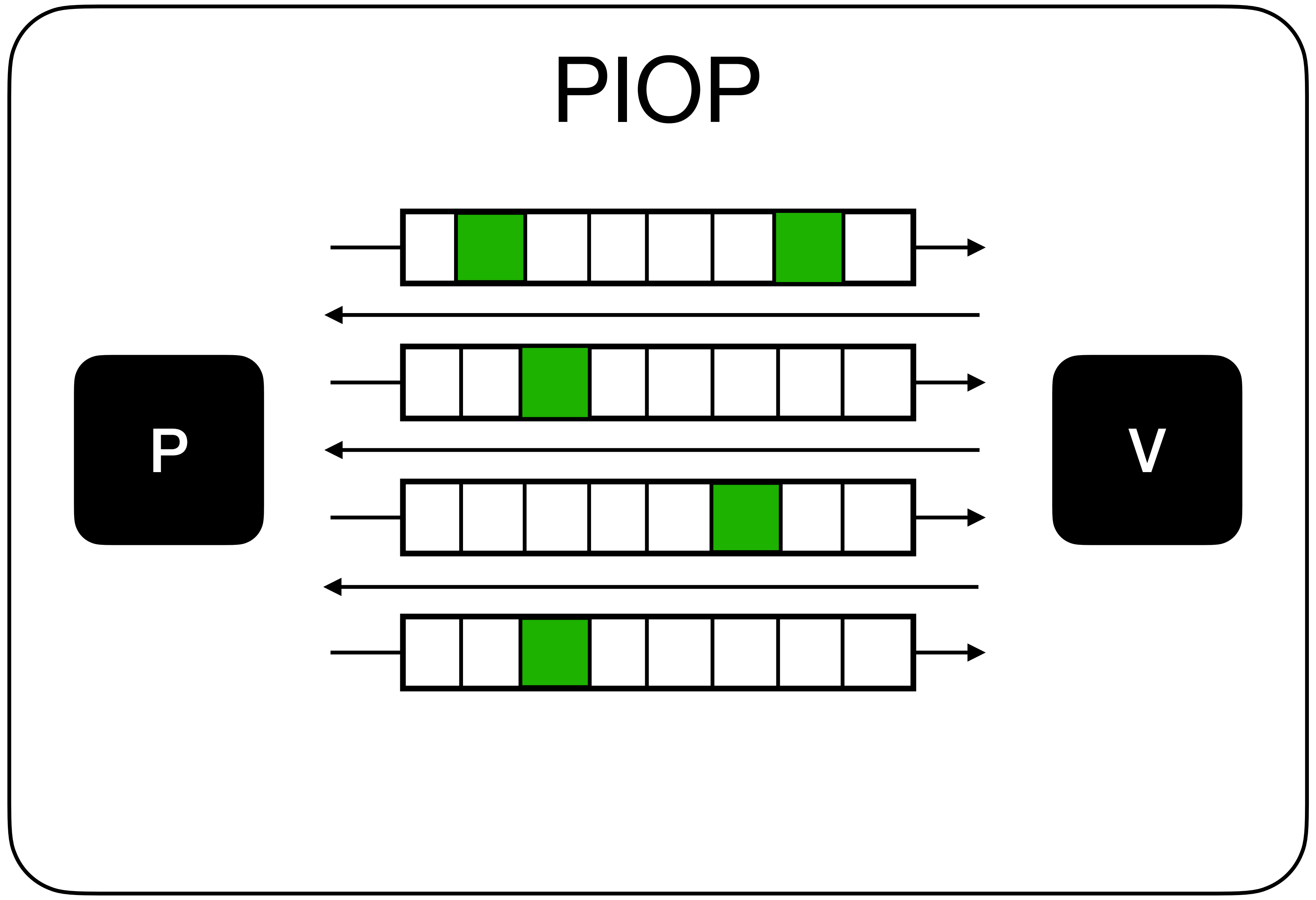
Constructing SNARKs

The modular way™



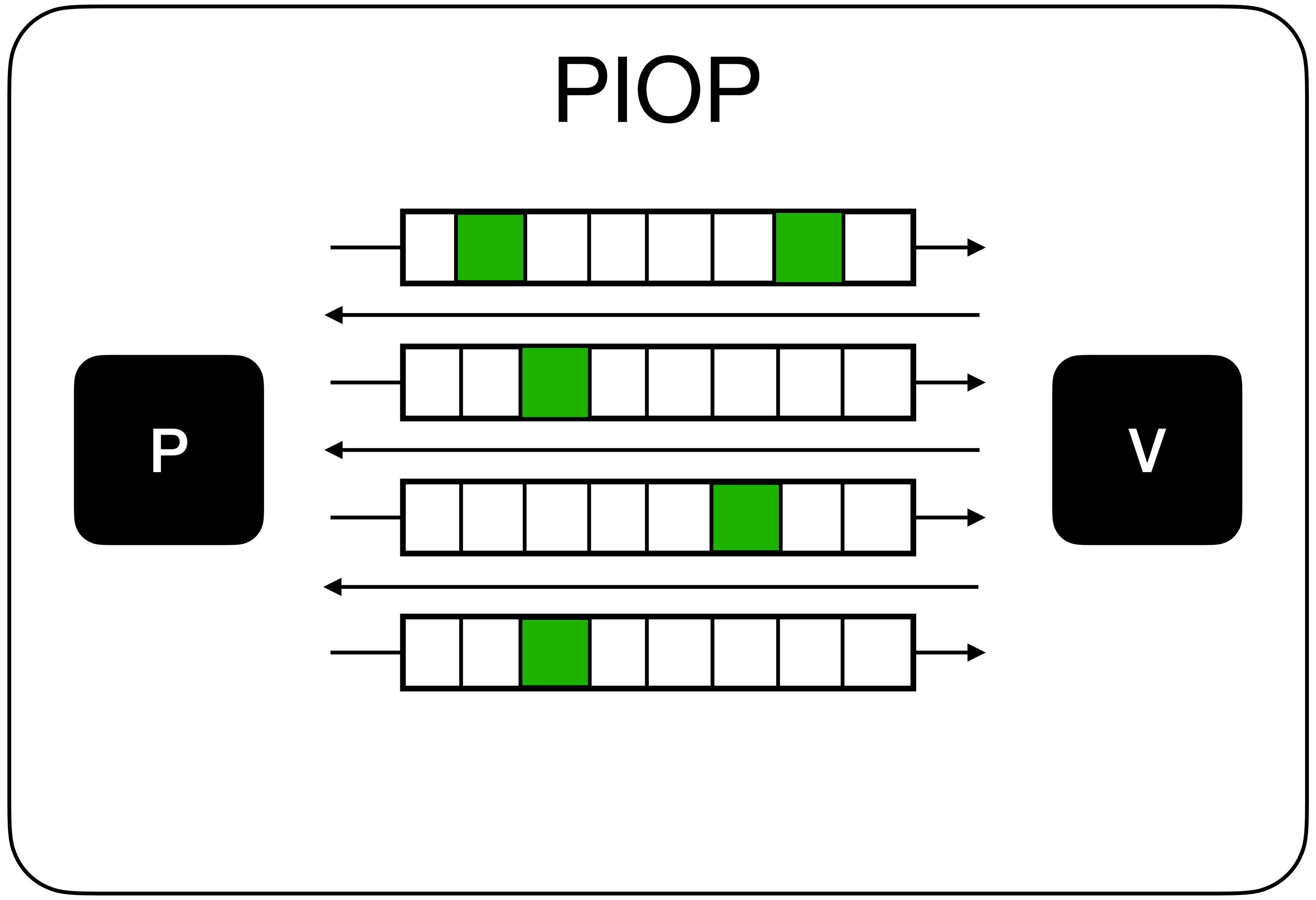
Constructing SNARKs

The modular way™



Constructing SNARKs

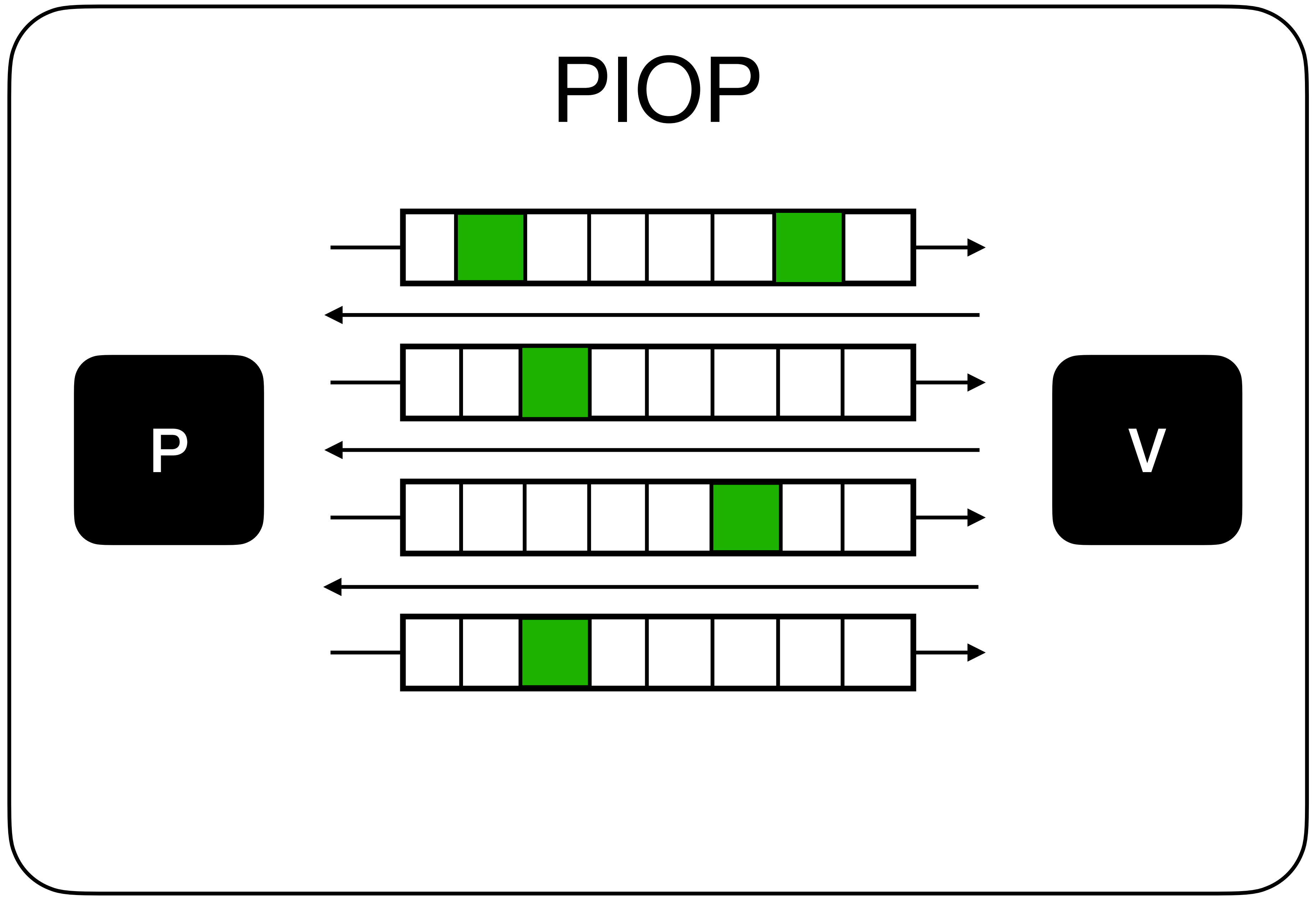
The modular way™



- Oracles are polynomials

Constructing SNARKs

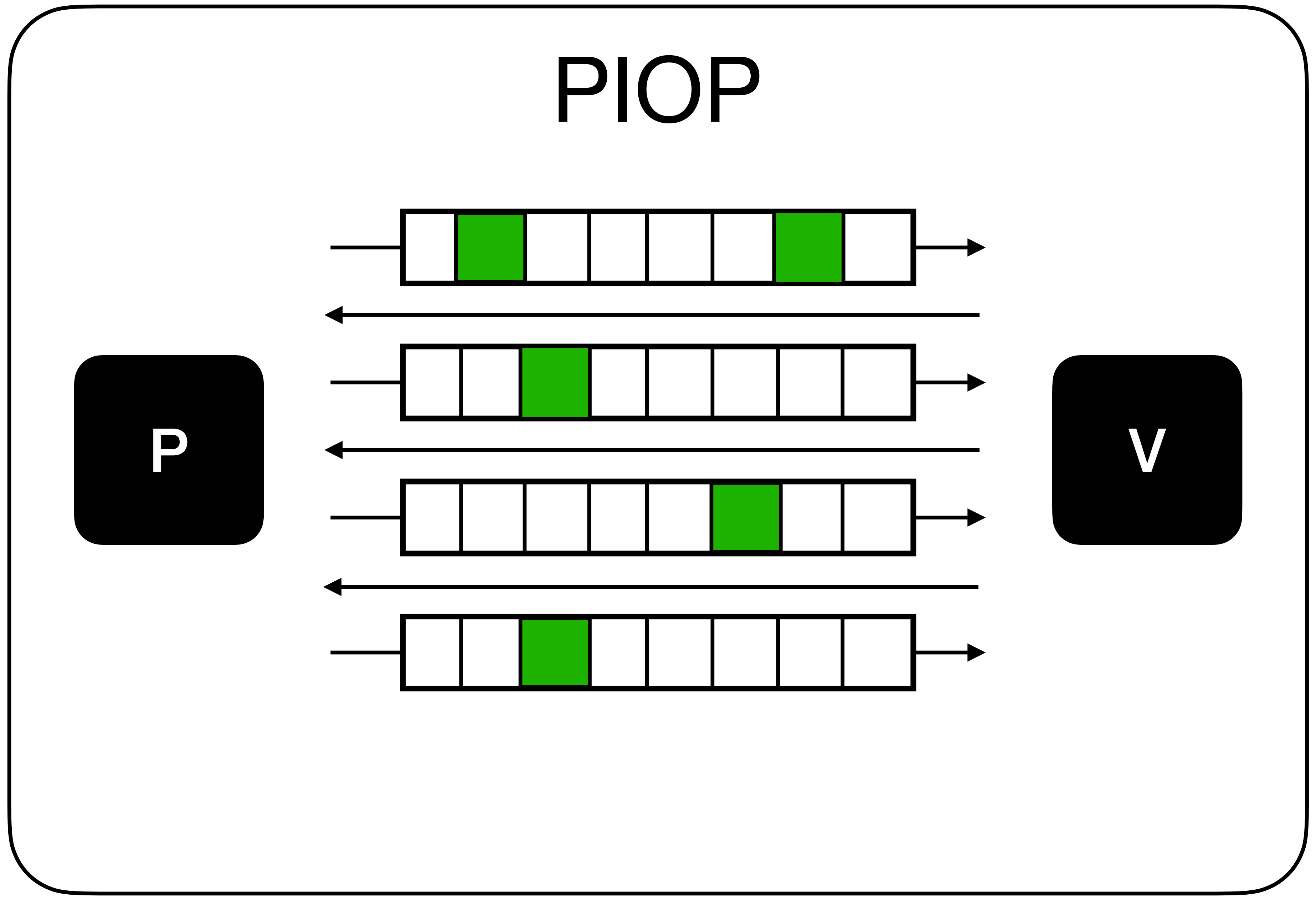
The modular way™



- Oracles are polynomials
- Security is information-theoretical

Constructing SNARKs

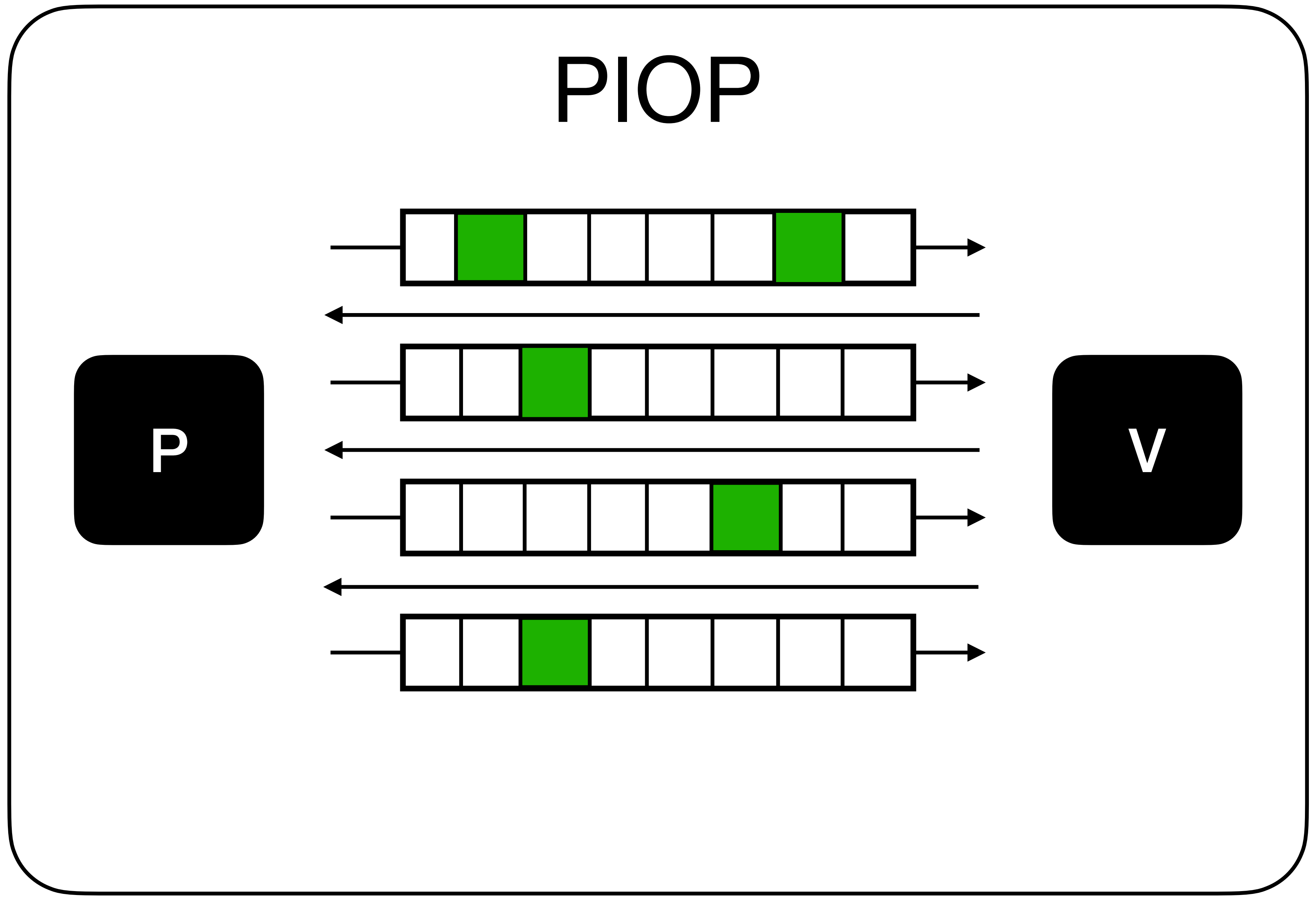
The modular way™



- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)

Constructing SNARKs

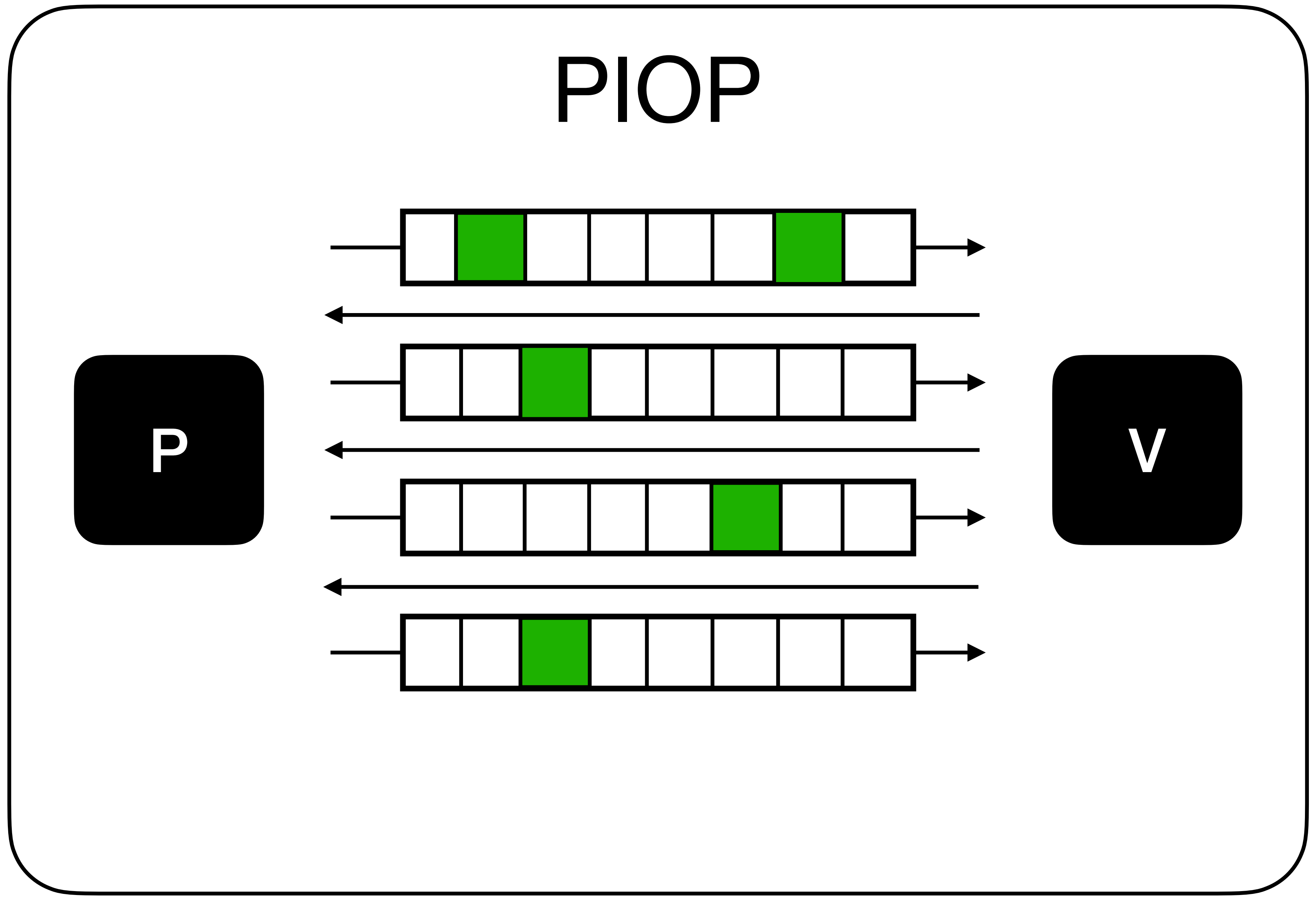
The modular way™

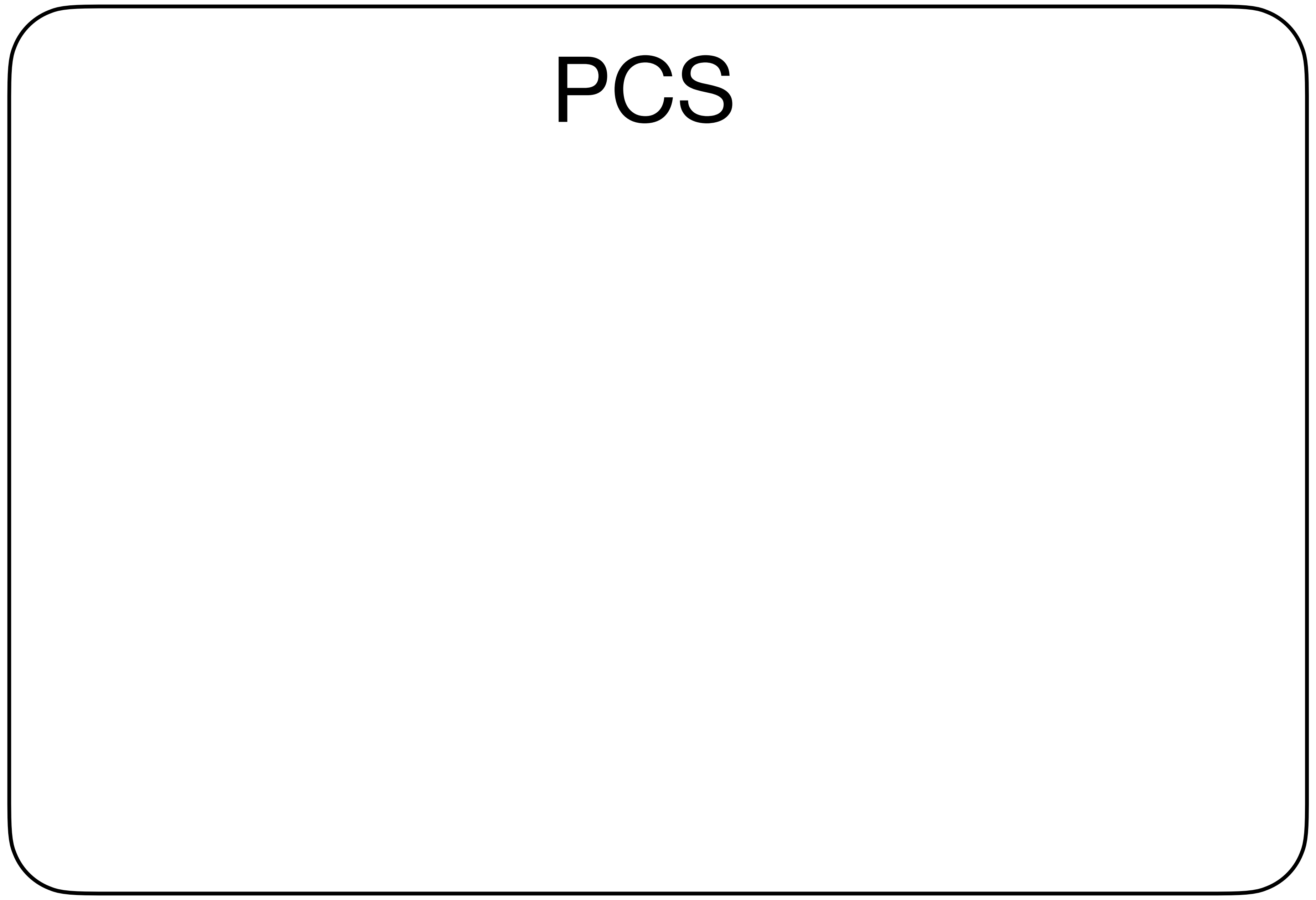


- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

Constructing SNARKs

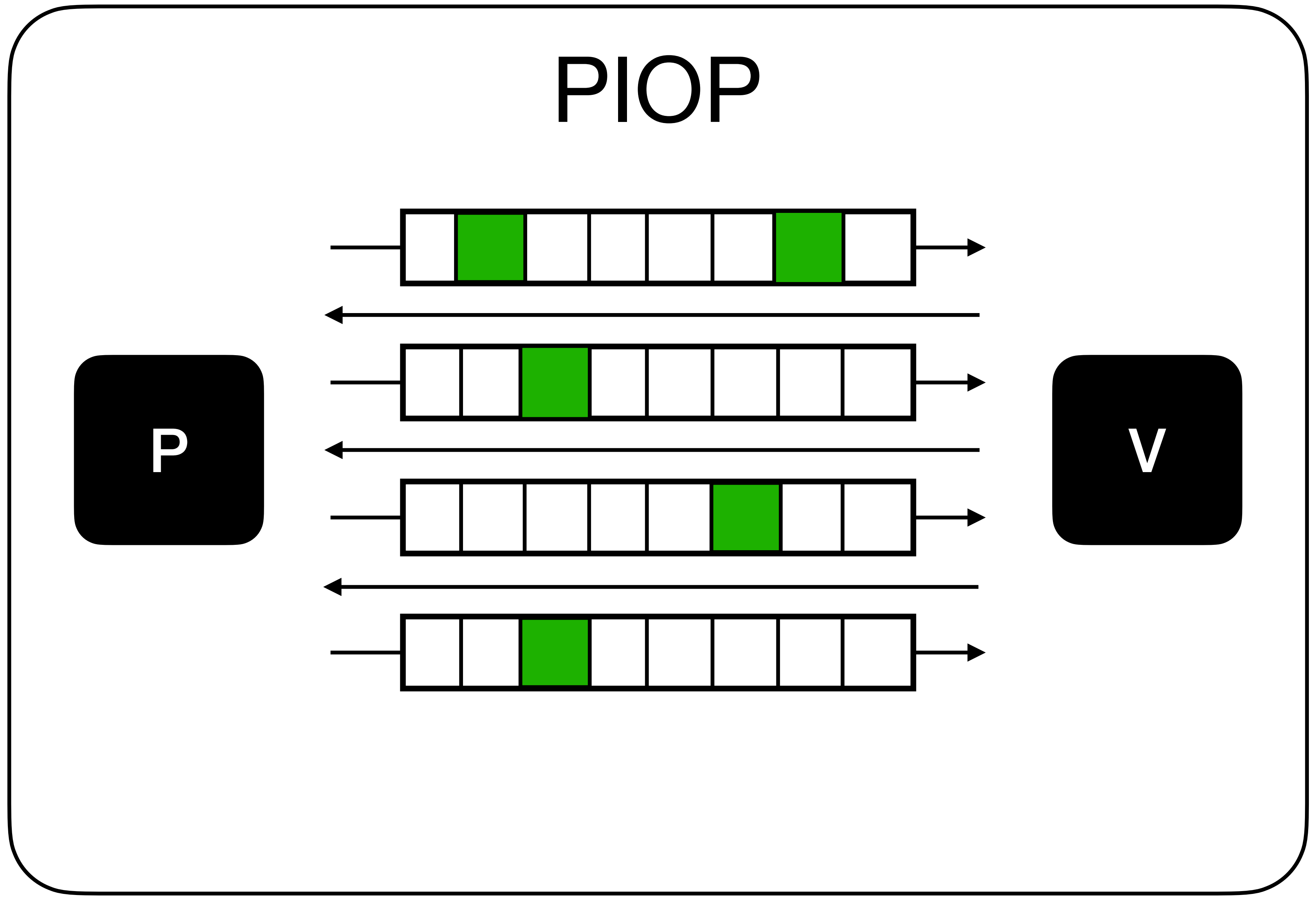
The modular way™

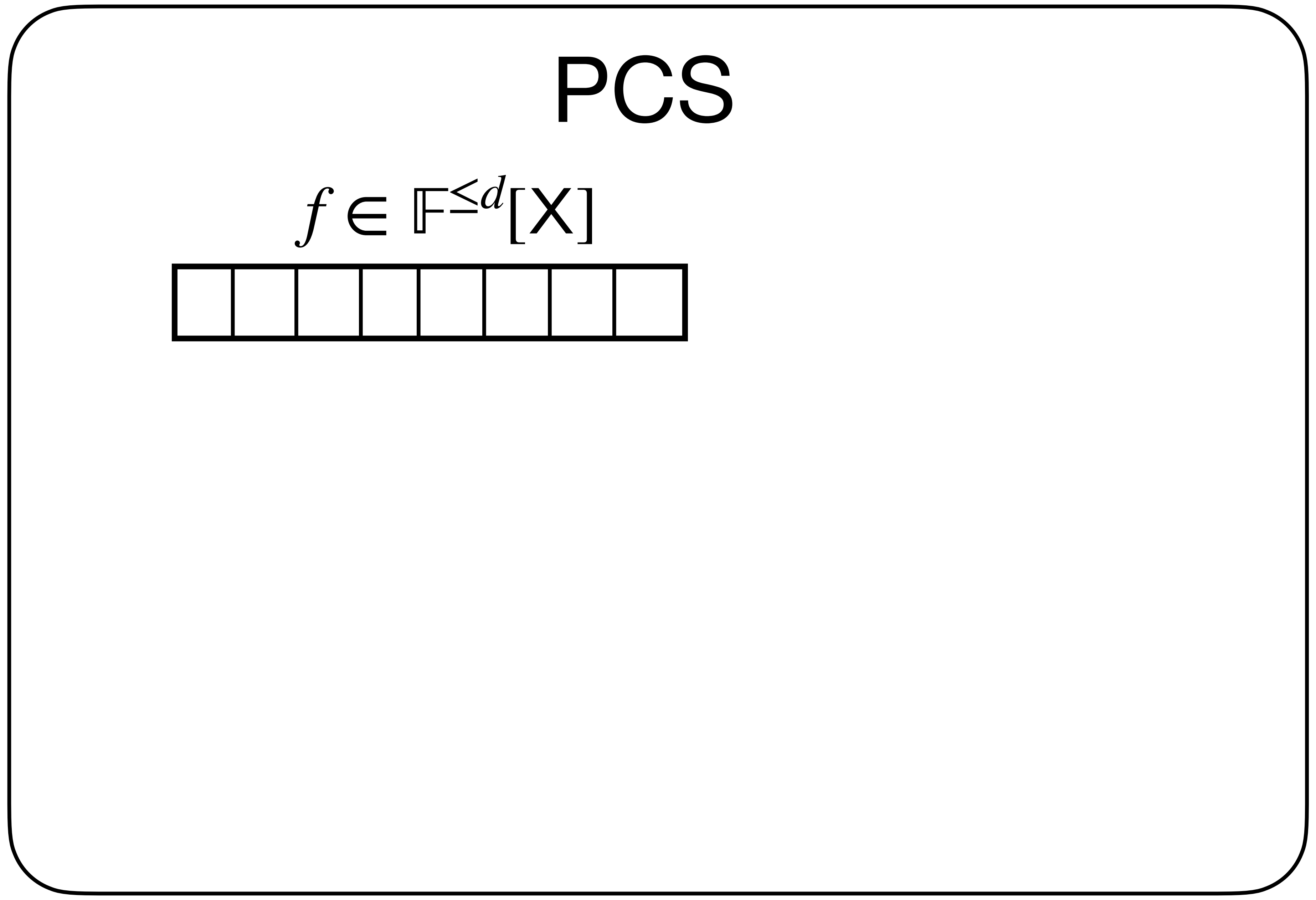




- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

Constructing SNARKs
The modular way™

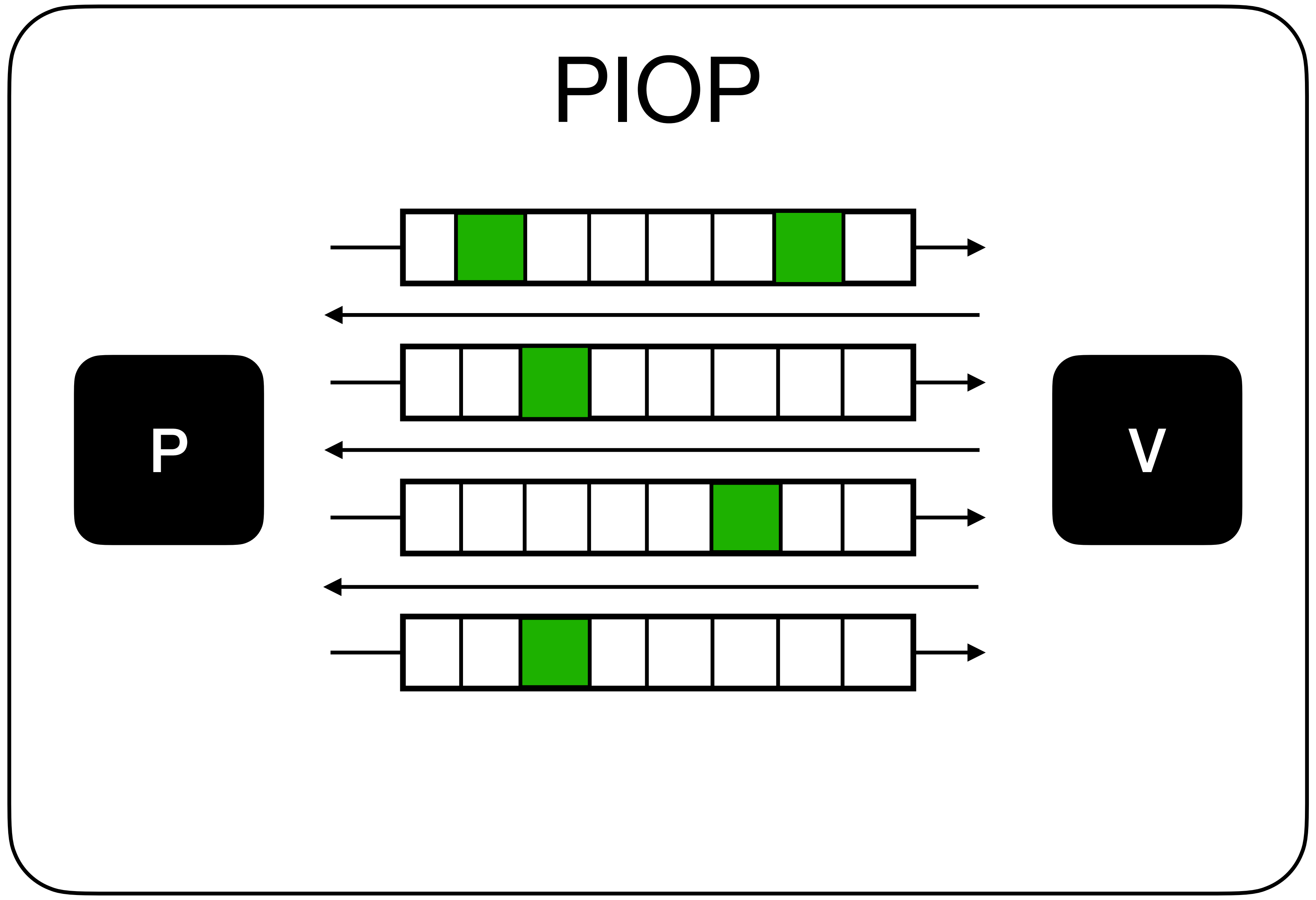


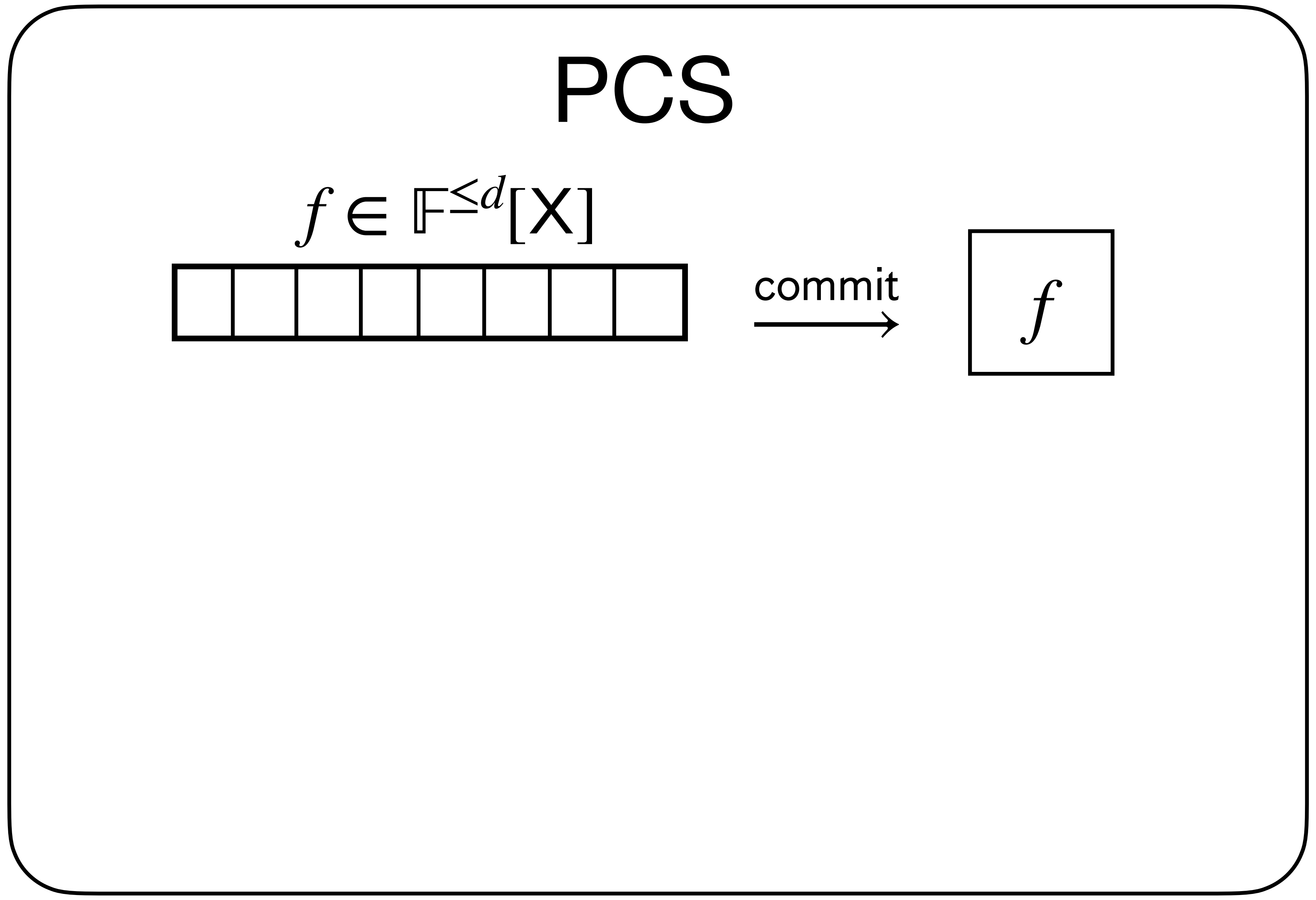


- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

Constructing SNARKs

The modular way™

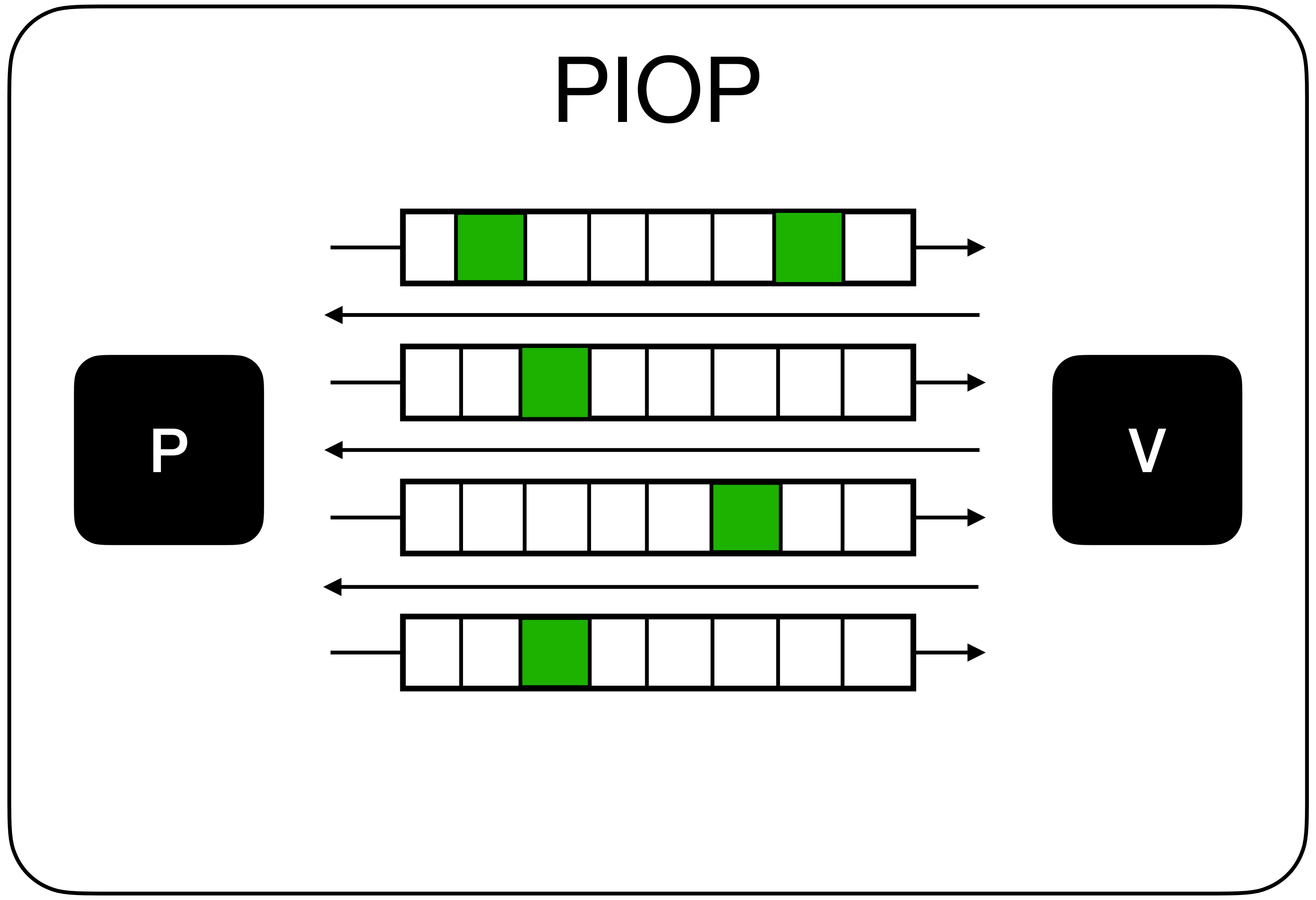


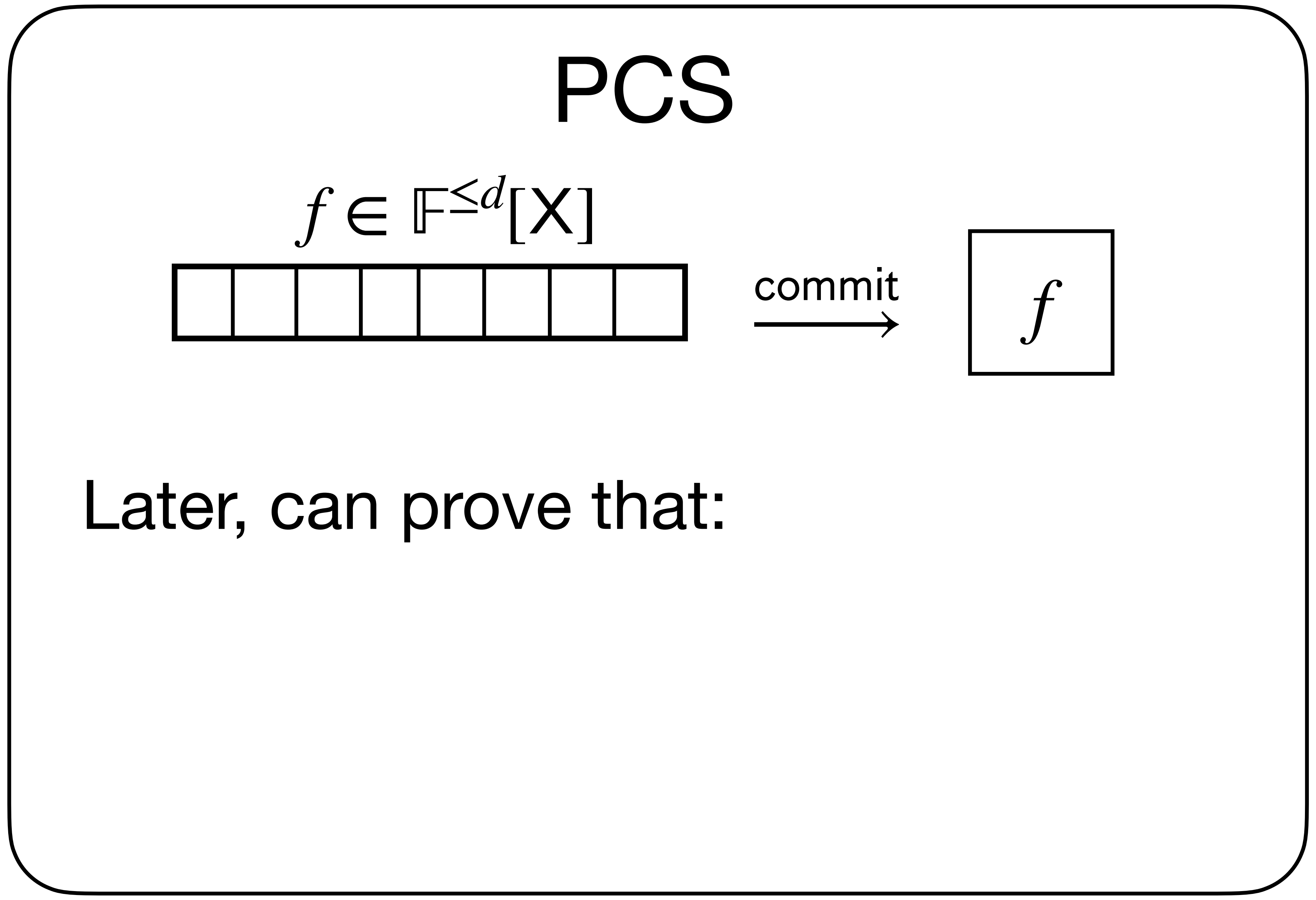


- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

Constructing SNARKs

The modular way™



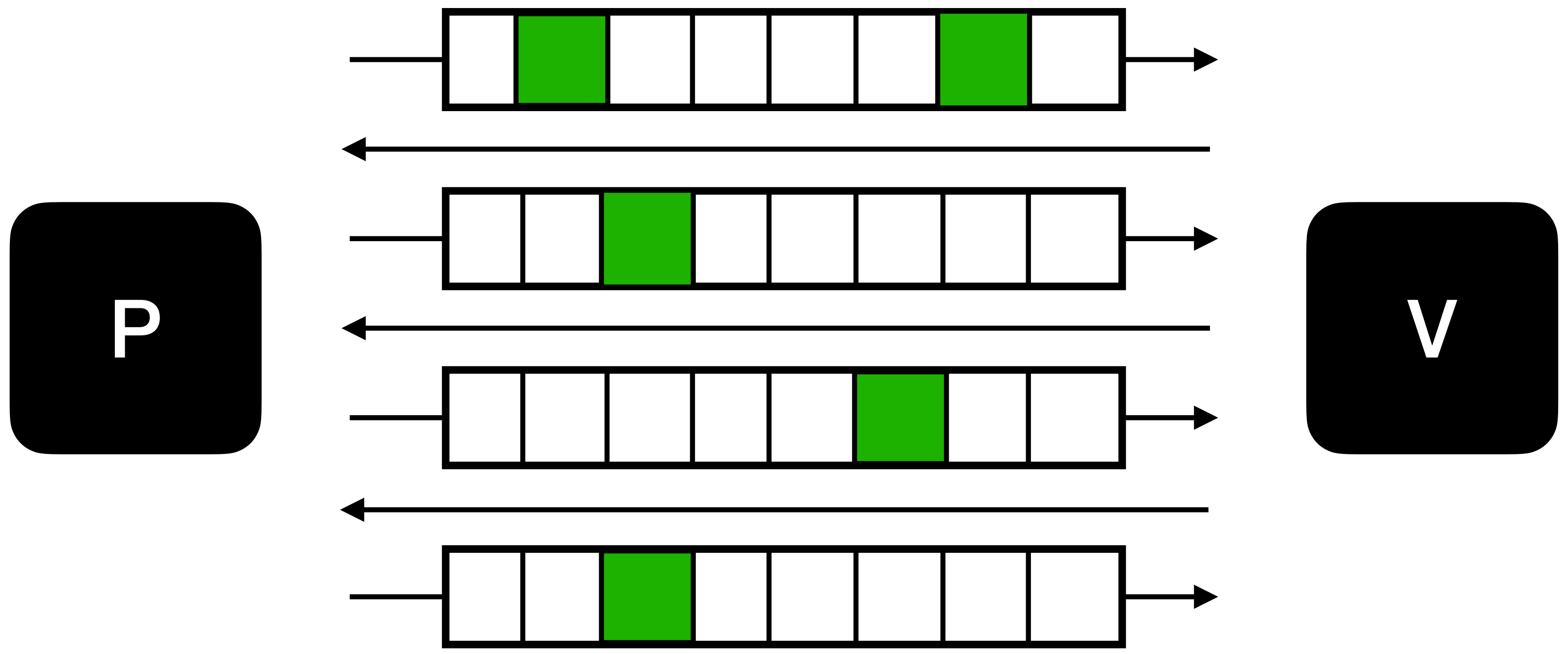


- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

Constructing SNARKs

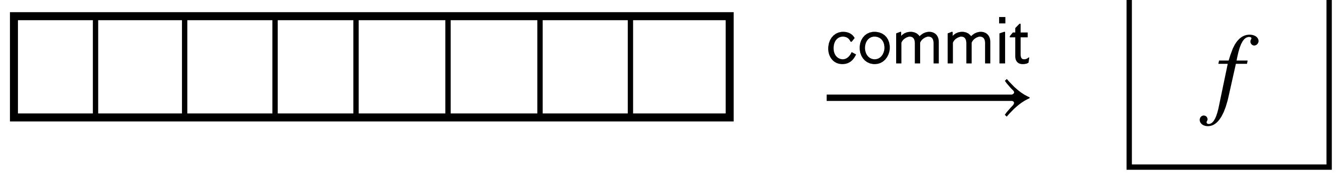
The modular way™

PIOP



PCS

$f \in \mathbb{F}^{\leq d}[X]$



Later, can prove that:

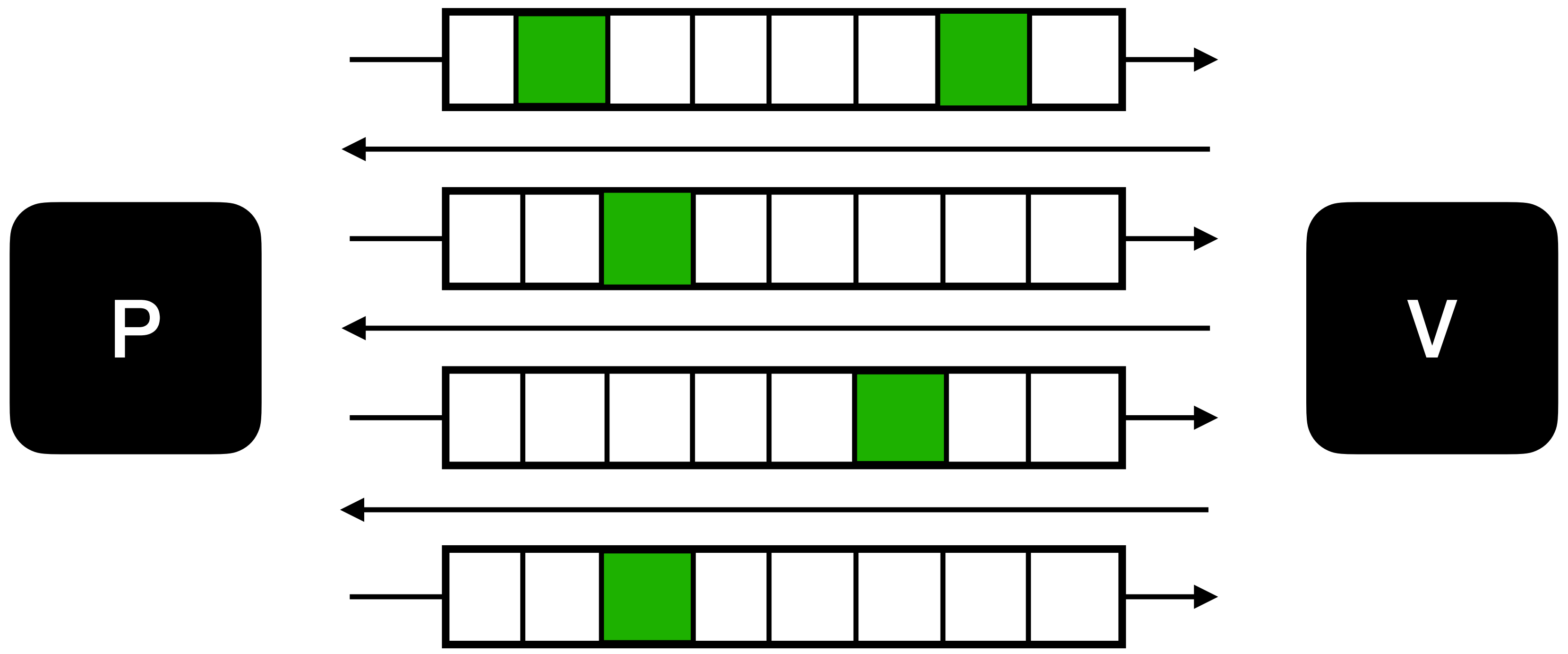
$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

Constructing SNARKs

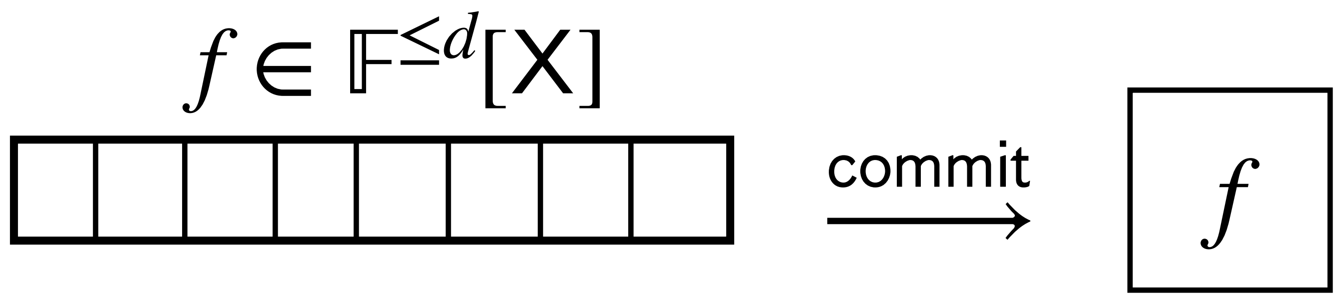
The modular way™

PIOP



- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

PCS



Later, can prove that:

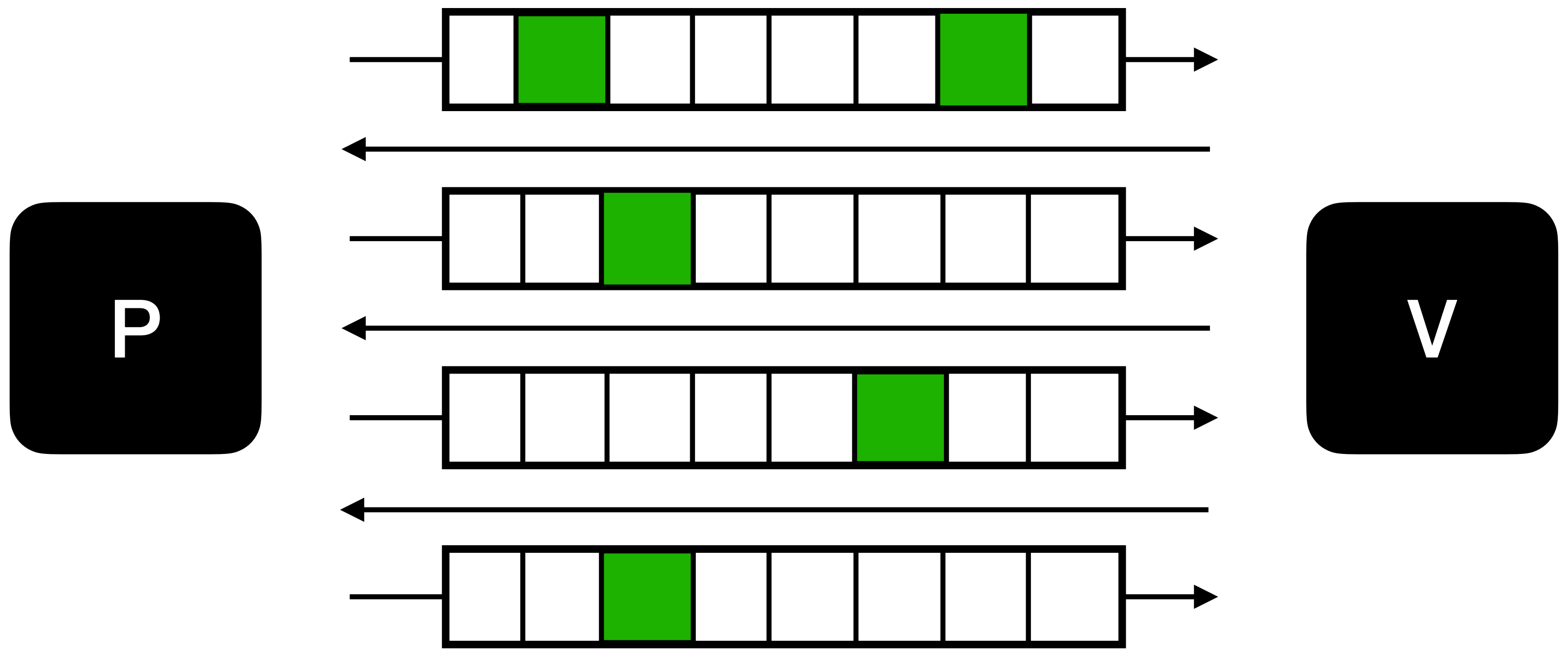
$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Cryptography goes here!

Constructing SNARKs

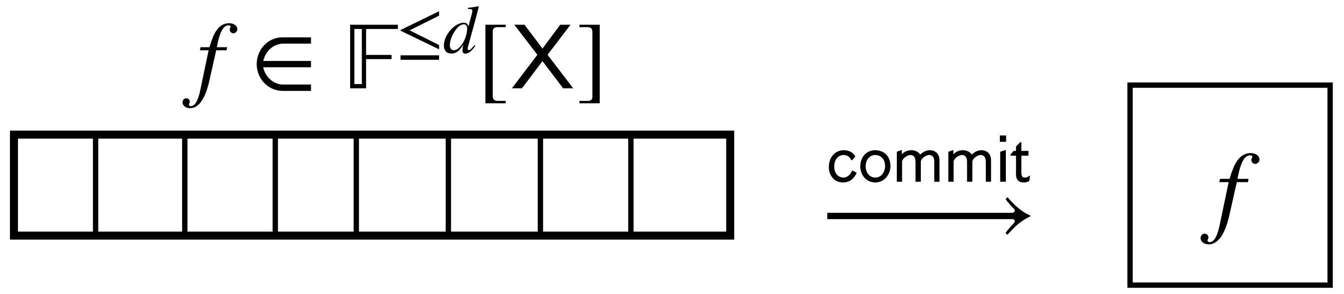
The modular way™

PIOP



- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

PCS



Later, can prove that:

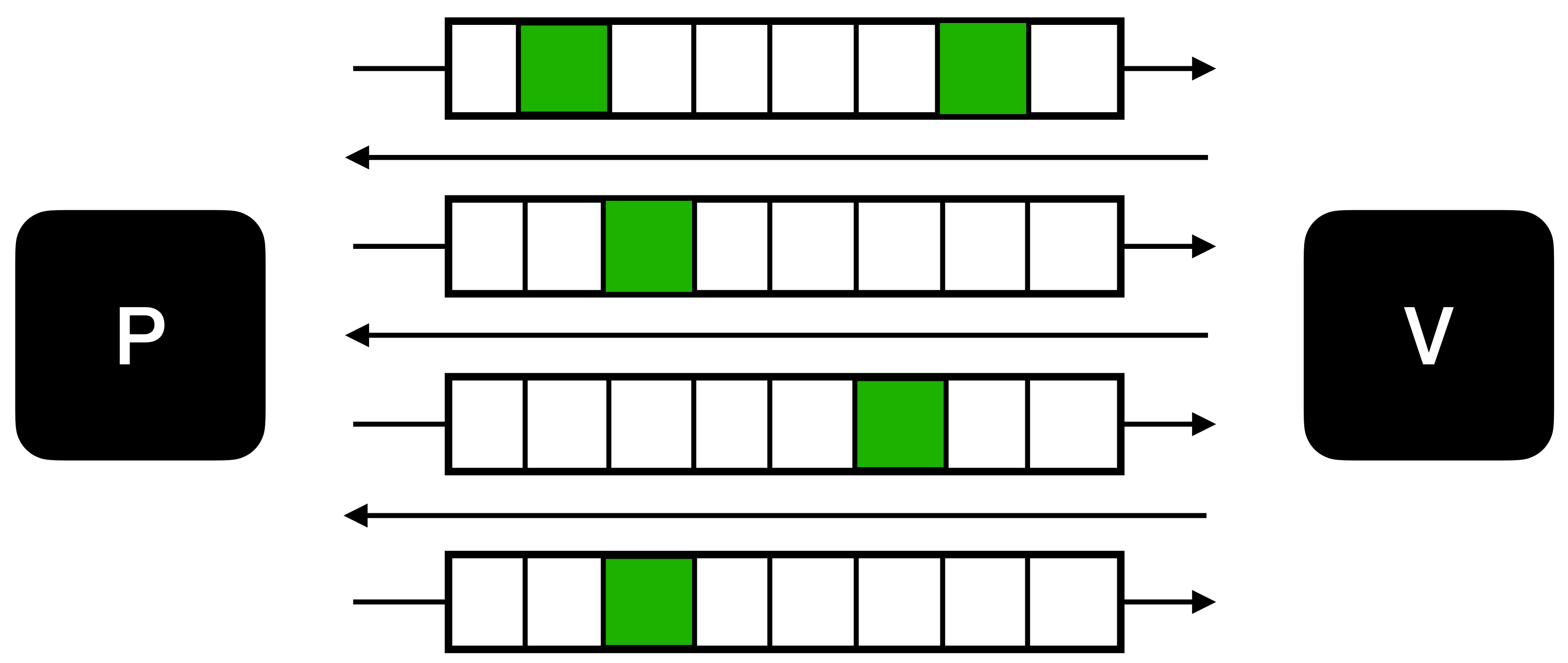
$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Cryptography goes here!
- Computational security

Constructing SNARKs

The modular way™

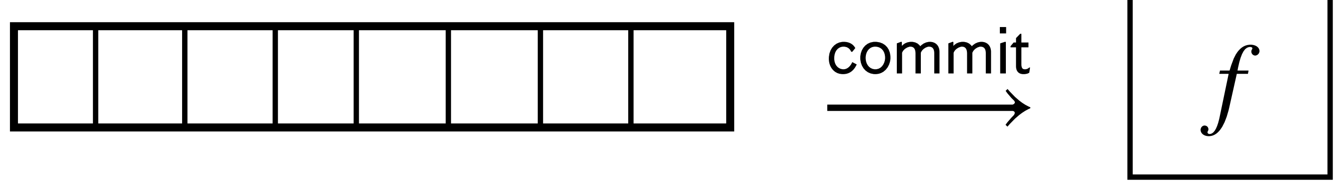
PIOP



- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

PCS

$f \in \mathbb{F}^{\leq d}[X]$



Later, can prove that:

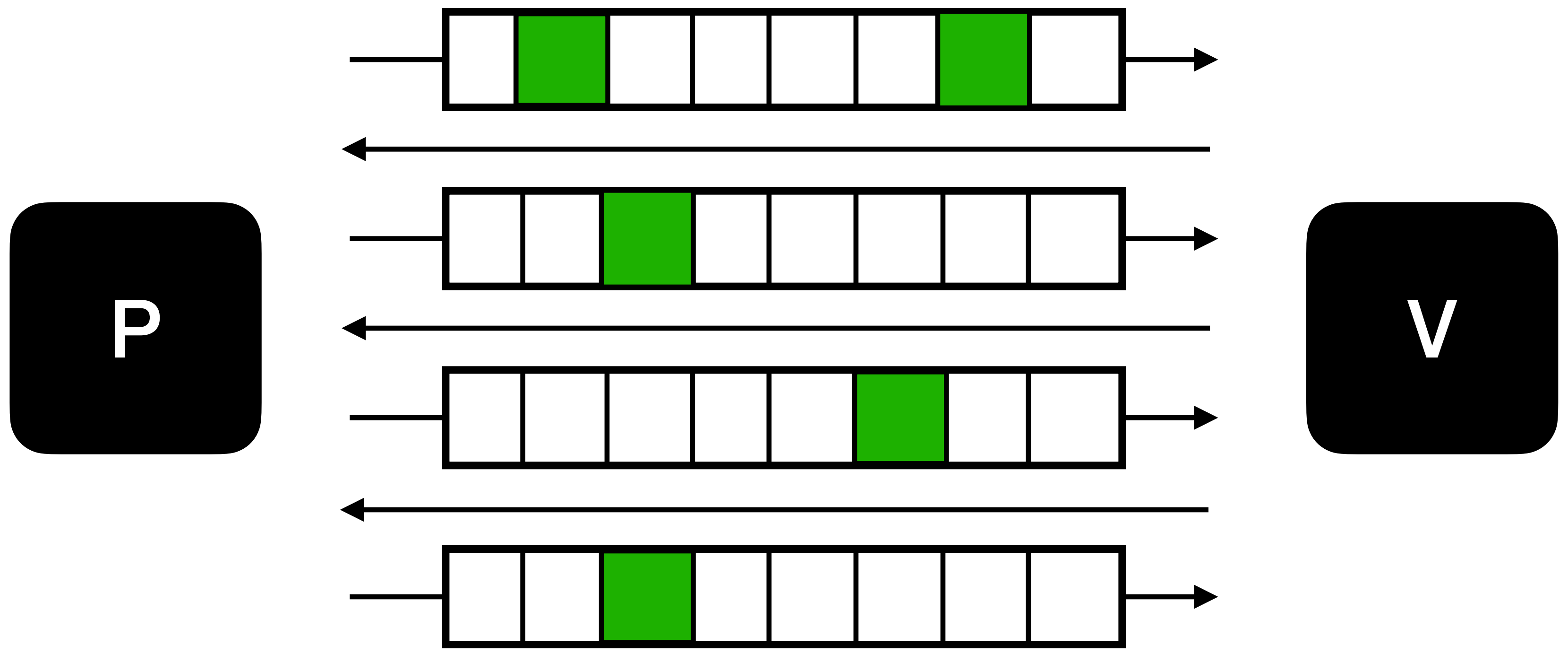
$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Cryptography goes here!
- Computational security
- We can achieve succinctness

Constructing SNARKs

The modular way™

PIOP

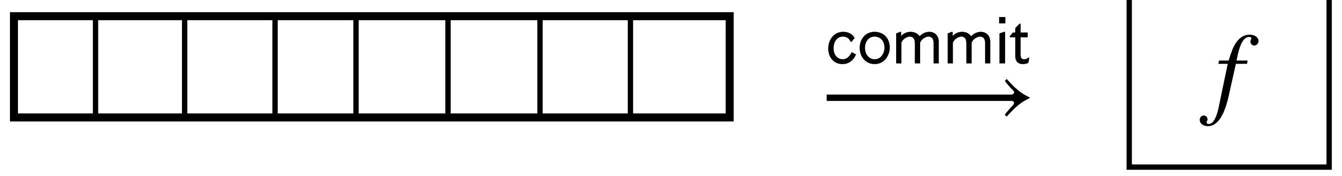


+

FS

PCS

$f \in \mathbb{F}^{\leq d}[X]$



Later, can prove that:

$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

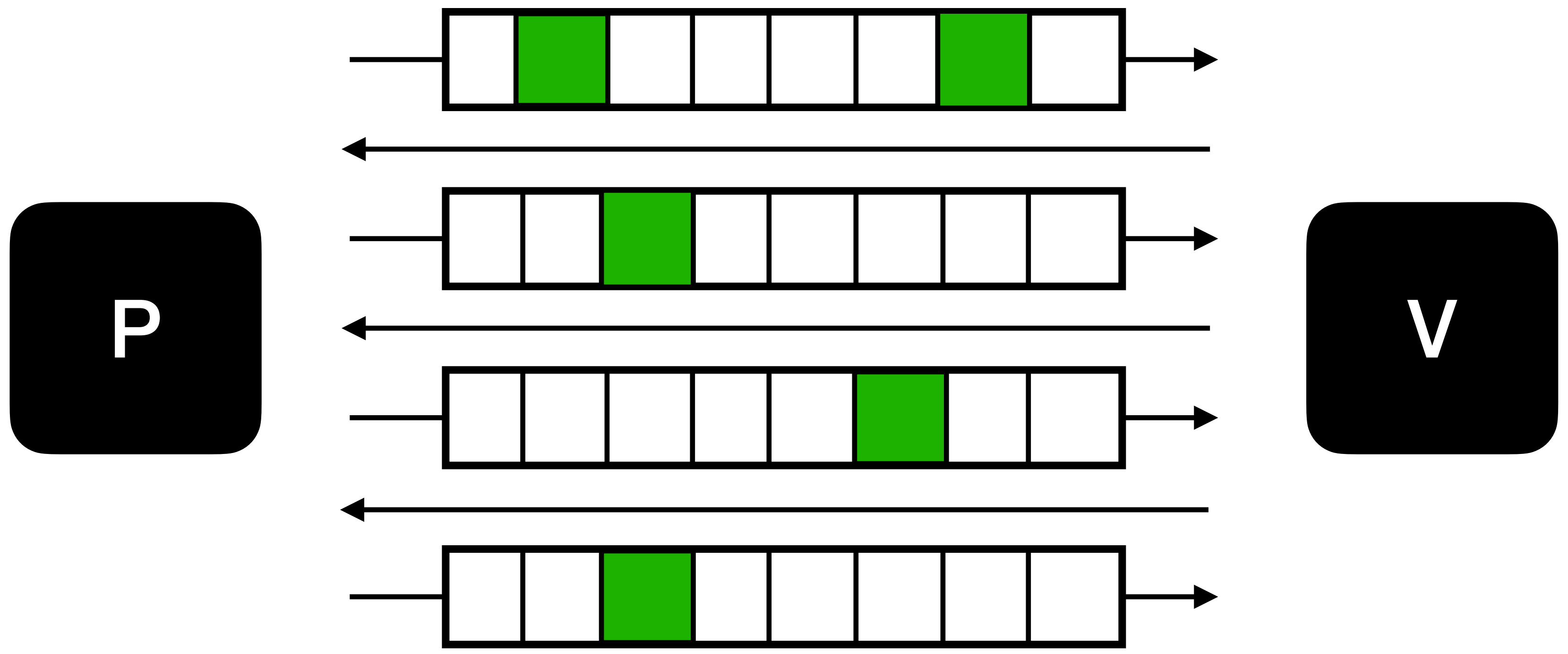
- Cryptography goes here!
- Computational security
- We can achieve succinctness

Constructing SNARKs

The modular way™

We focus on this!

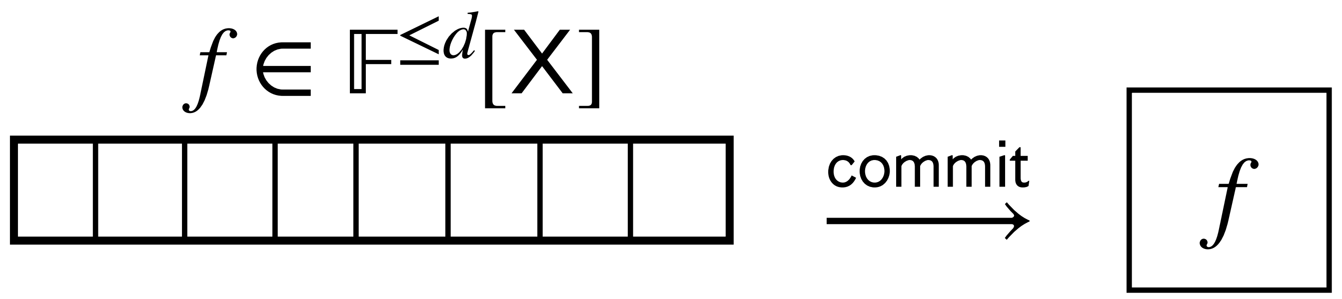
PIOP



+

FS

PCS



Later, can prove that:

$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

- Cryptography goes here!
- Computational security
- We can achieve succinctness

Zoo of Polynomial Commitments

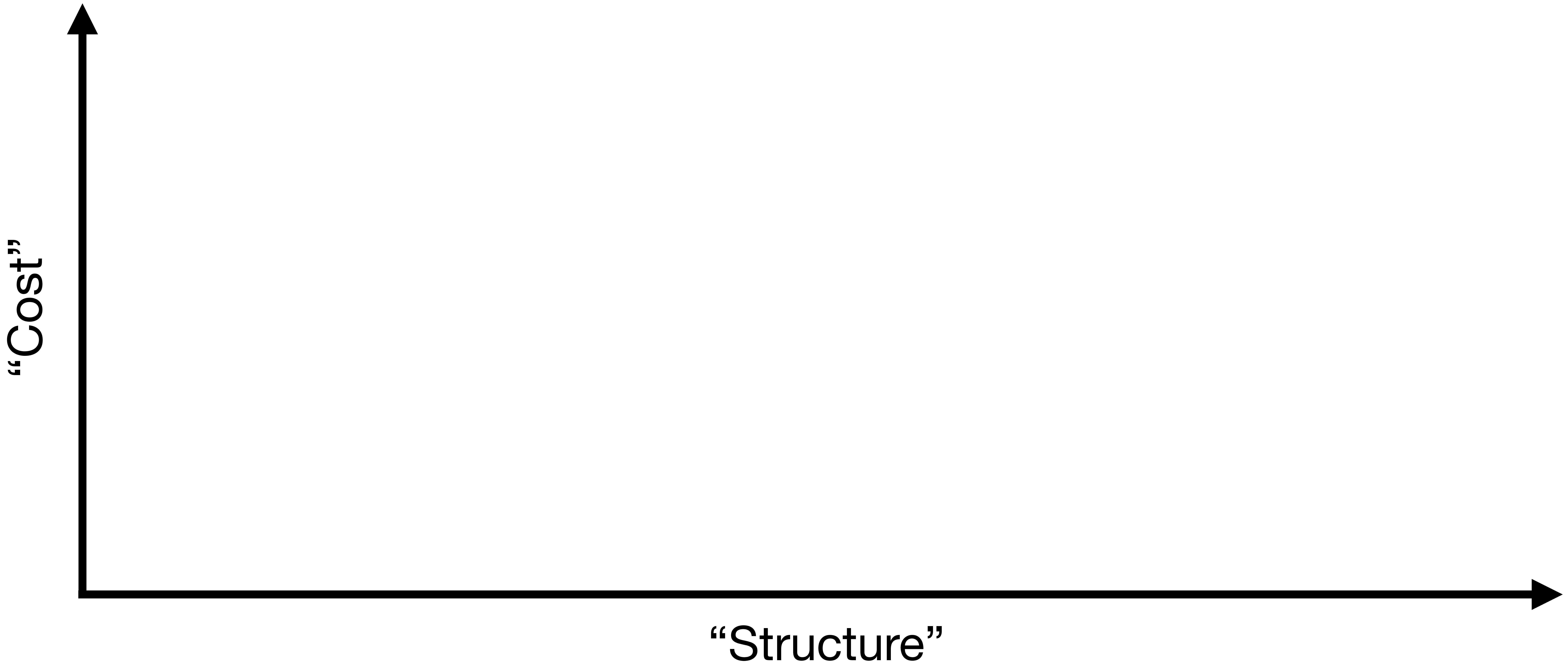
A very incomplete list…

Zoo of Polynomial Commitments

A very incomplete list…

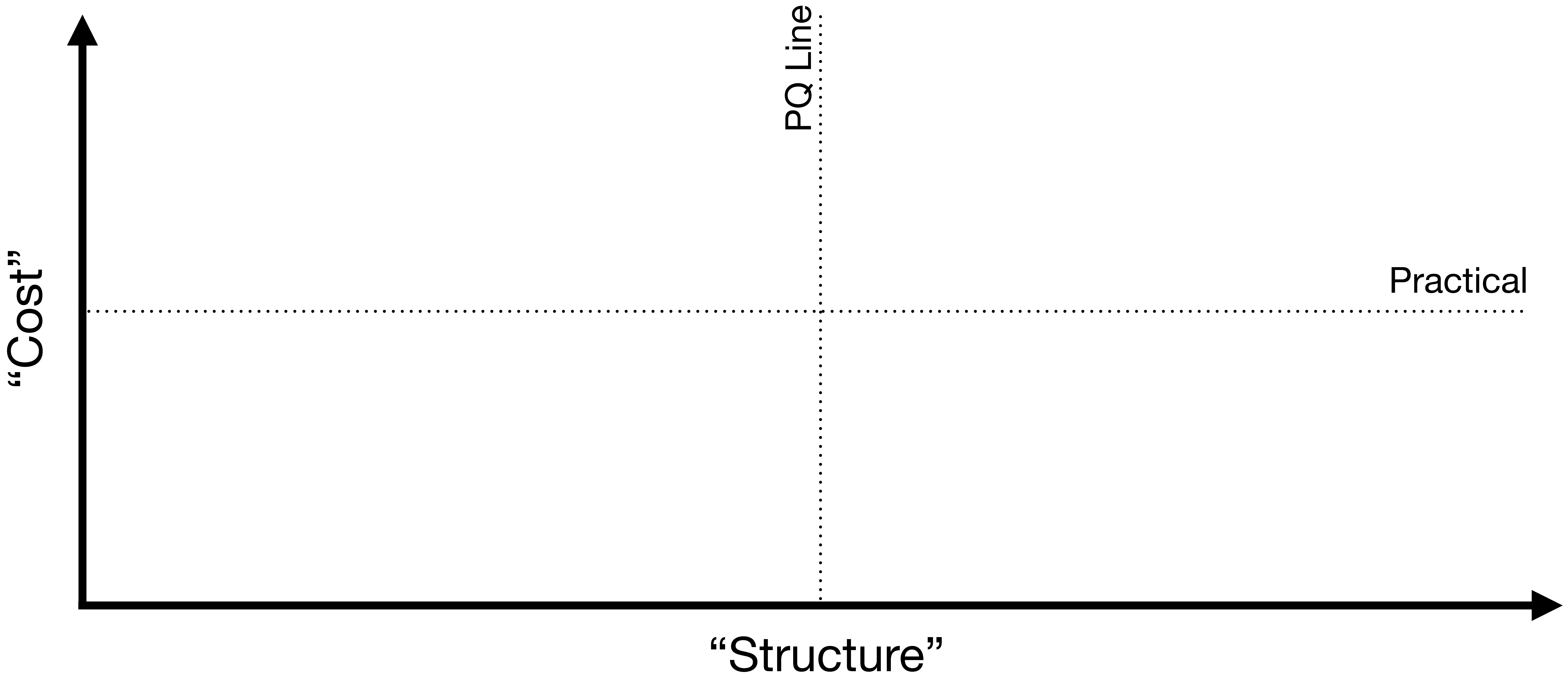
Zoo of Polynomial Commitments

A very incomplete list…



Zoo of Polynomial Commitments

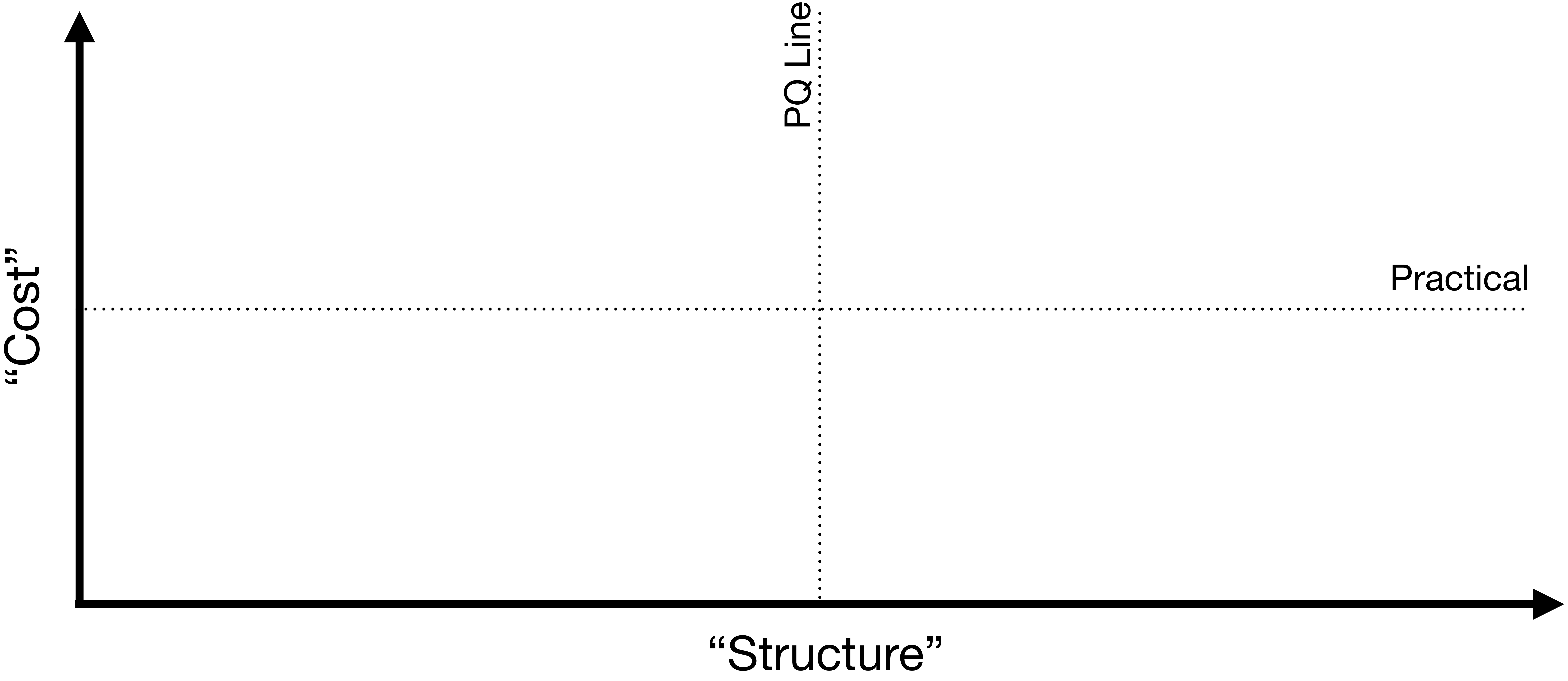
A very incomplete list…



Zoo of Polynomial Commitments

A very incomplete list…

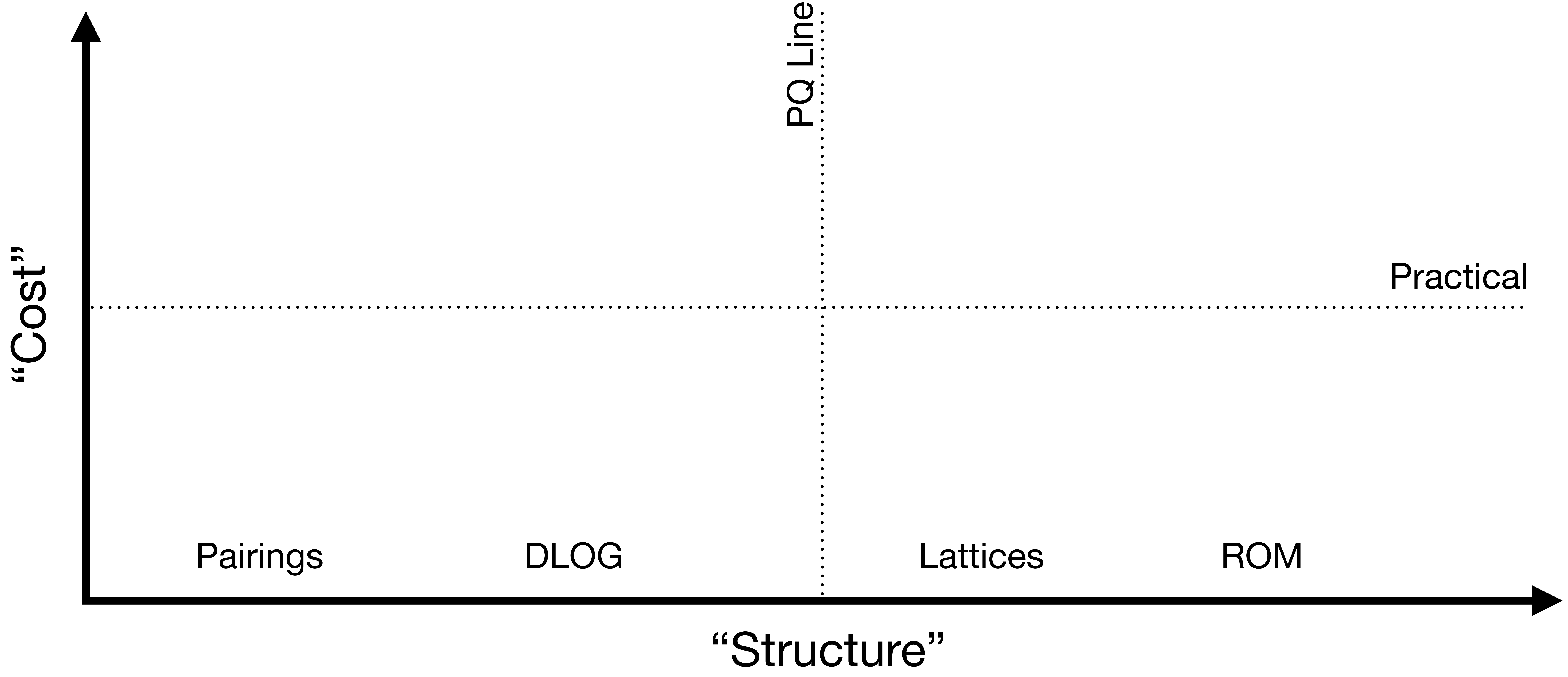
<u>Underlined:</u> succinct verification
*: interactive (no FS)
(T): trusted setup



Zoo of Polynomial Commitments

A very incomplete list…

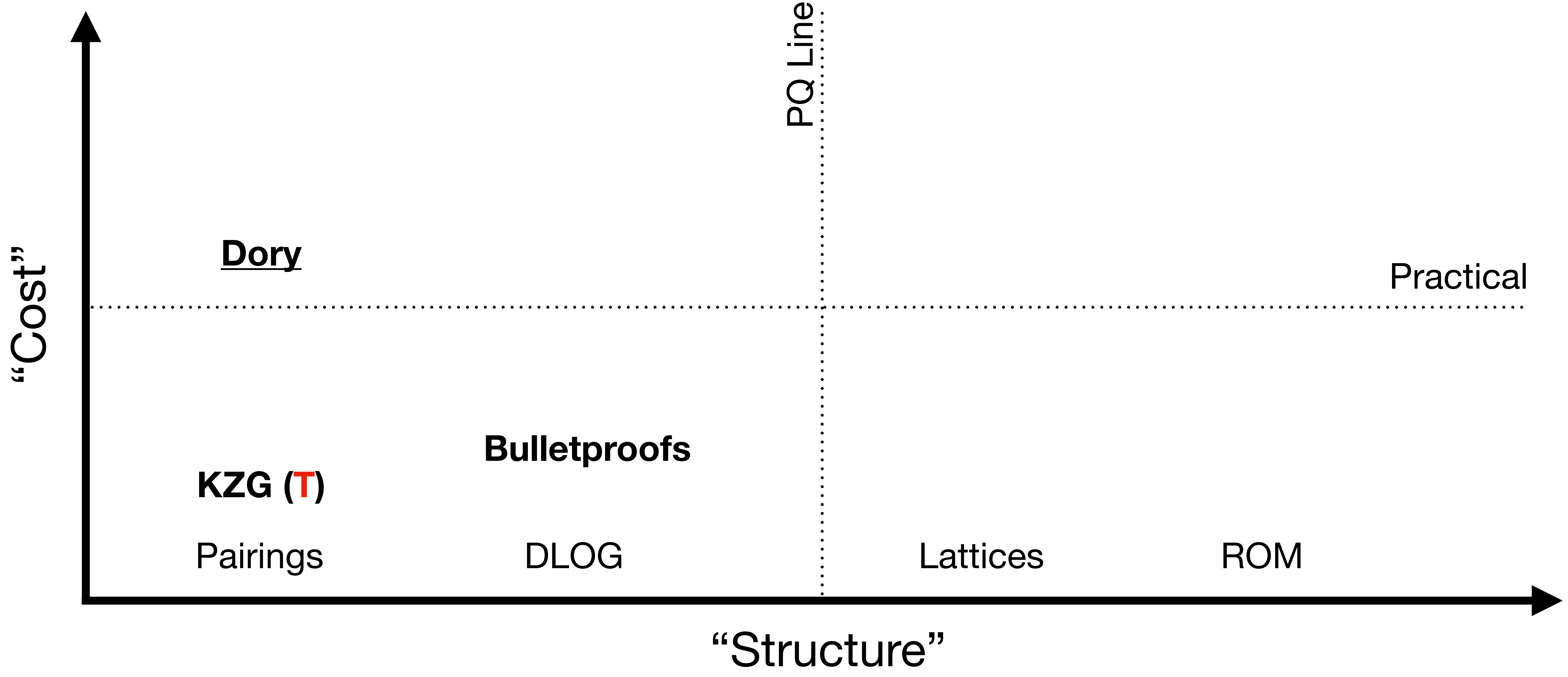
Underlined: succinct verification
*: interactive (no FS)
(T): trusted setup



Zoo of Polynomial Commitments

A very incomplete list…

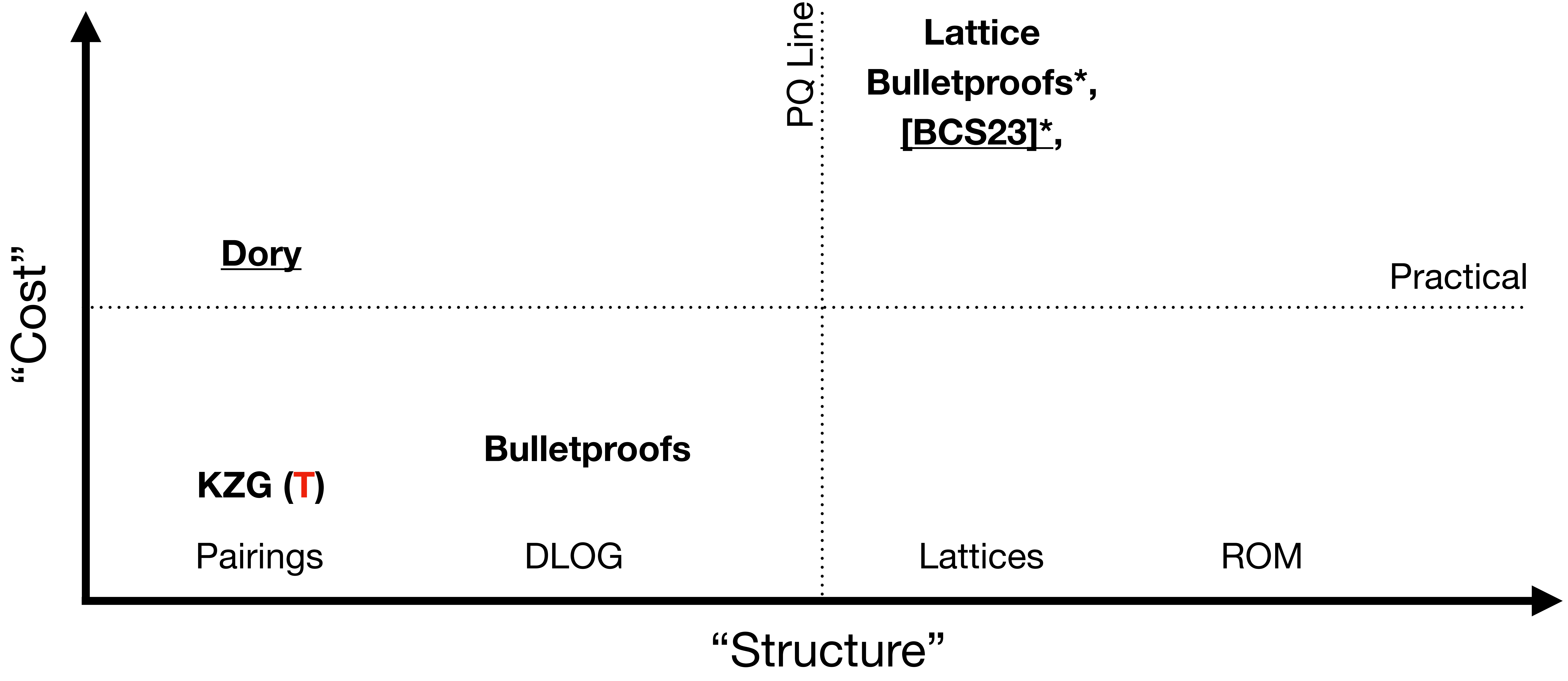
Underlined: succinct verification
*: interactive (no FS)
(T): trusted setup



Zoo of Polynomial Commitments

A very incomplete list...

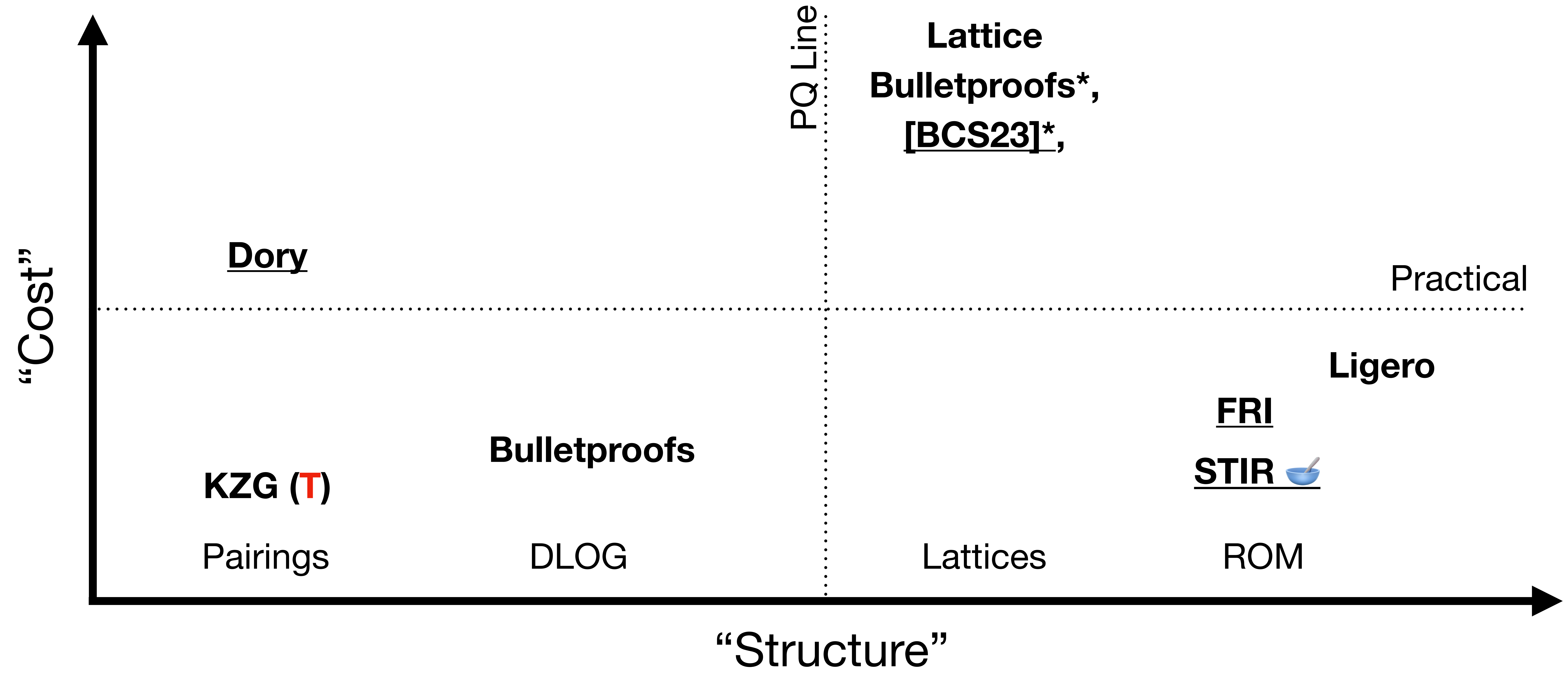
<u>Underlined:</u> succinct verification
*: interactive (no FS)
(T): trusted setup



Zoo of Polynomial Commitments

A very incomplete list…

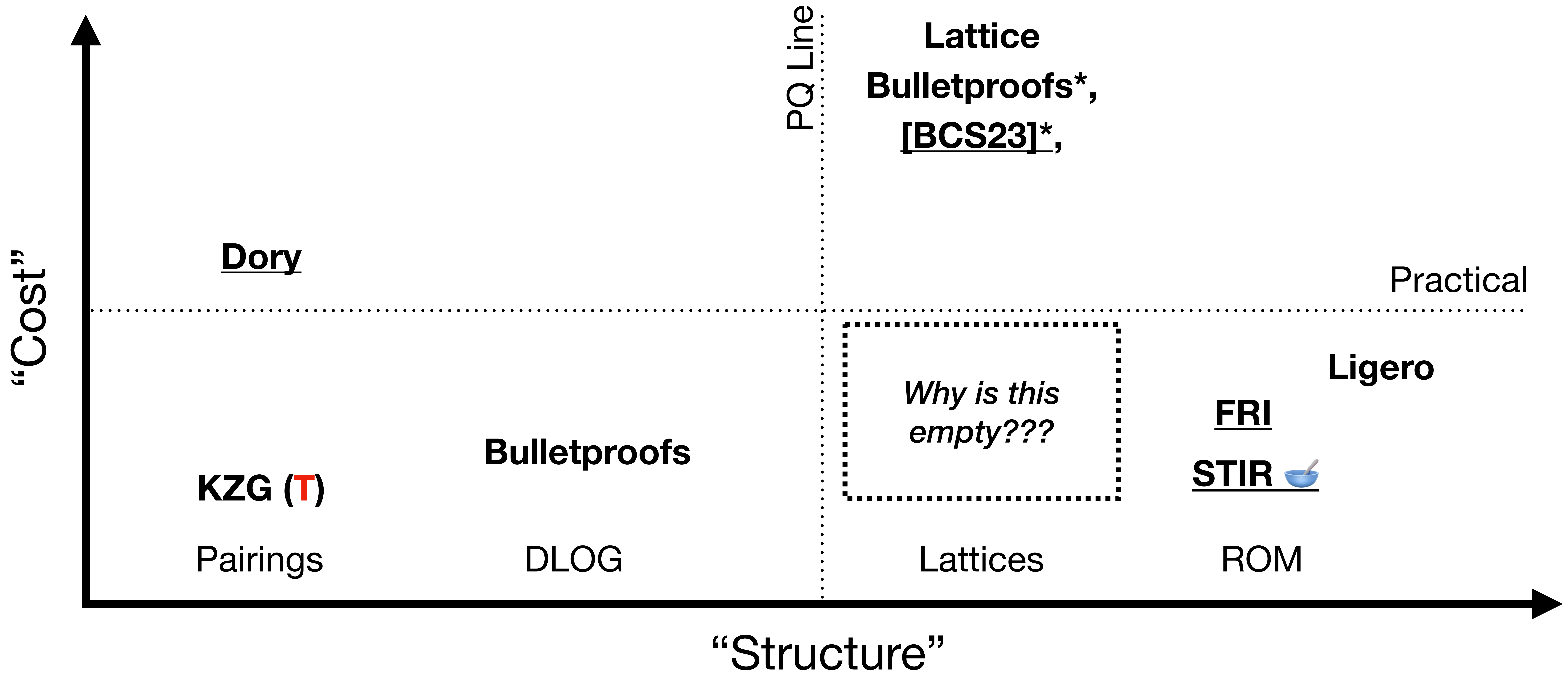
Underlined: succinct verification
*: interactive (no FS)
(T): trusted setup



Zoo of Polynomial Commitments

A very incomplete list…

<u>Underlined:</u> succinct verification
*: interactive (no FS)
(T): trusted setup



Our Results

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

Martin R. Albrecht
martin.albrecht@{kcl.ac.uk,sandboxaq.com}
King's College London and SandboxAQ

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Oleksandra Lapiha
sasha.lapiha.2021@live.rhul.ac.uk
Royal Holloway, University of London

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

Martin R. Albrecht
martin.albrecht@{kcl.ac.uk,sandboxaq.com}
King's College London and SandboxAQ

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Oleksandra Lapiha
sasha.lapiha.2021@live.rhul.ac.uk
Royal Holloway, University of London

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

We construct a non-interactive lattice-based polynomial commitment with:

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

Martin R. Albrecht
martin.albrecht@{kcl.ac.uk,sandboxaq.com}
King's College London and SandboxAQ

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Oleksandra Lapiha
sasha.lapiha.2021@live.rhul.ac.uk
Royal Holloway, University of London

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

We construct a non-interactive lattice-based polynomial commitment with:

1. **Succinct** proofs

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

Martin R. Albrecht
martin.albrecht@{kcl.ac.uk,sandboxaq.com}
King's College London and SandboxAQ

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Oleksandra Lapiha
sasha.lapiha.2021@live.rhul.ac.uk
Royal Holloway, University of London

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

We construct a non-interactive lattice-based polynomial commitment with:

1. **Succinct** proofs
2. **Succinct** verification time

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

Martin R. Albrecht
martin.albrecht@{kcl.ac.uk,sandboxaq.com}
King's College London and SandboxAQ

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Oleksandra Lapiha
sasha.lapiha.2021@live.rhul.ac.uk
Royal Holloway, University of London

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

We construct a non-interactive lattice-based polynomial commitment with:

1. **Succinct** proofs
2. **Succinct** verification time
3. Binding under **(M)SIS**

Techniques

Lattice-Based SNARKs

How to get around [GW11]?

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs

Rinocchio: SNARKs for Ring Arithmetic

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs

Amortized Efficient zk-SNARK from Linear-Only RLWE Encodings

Rinocchio: SNARKs for Ring Arithmetic

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs

Amortized Efficient zk-SNARK from Linear-Only RLWE Encodings

Rinocchio: SNARKs for Ring Arithmetic

Private Re-Randomization for Module LWE and Applications to Quasi-Optimal ZK-SNARKs

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs

Amortized Efficient zk-SNARK from Linear-Only RLWE Encodings

Rinocchio: SNARKs for Ring Arithmetic

Private Re-Randomization for Module LWE and Applications to Quasi-Optimal ZK-SNARKs

Lattice-Based zk-SNARKs from Square Span Programs

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs

Amortized Efficient zk-SNARK from Linear-Only RLWE Encodings

Rinocchio: SNARKs for Ring Arithmetic

Private Re-Randomization for Module LWE and Applications to Quasi-Optimal ZK-SNARKs

Lattice-Based zk-SNARKs from Square Span Programs

Shorter and Faster Post-Quantum Designated-Verifier zkSNARKs from Lattices*

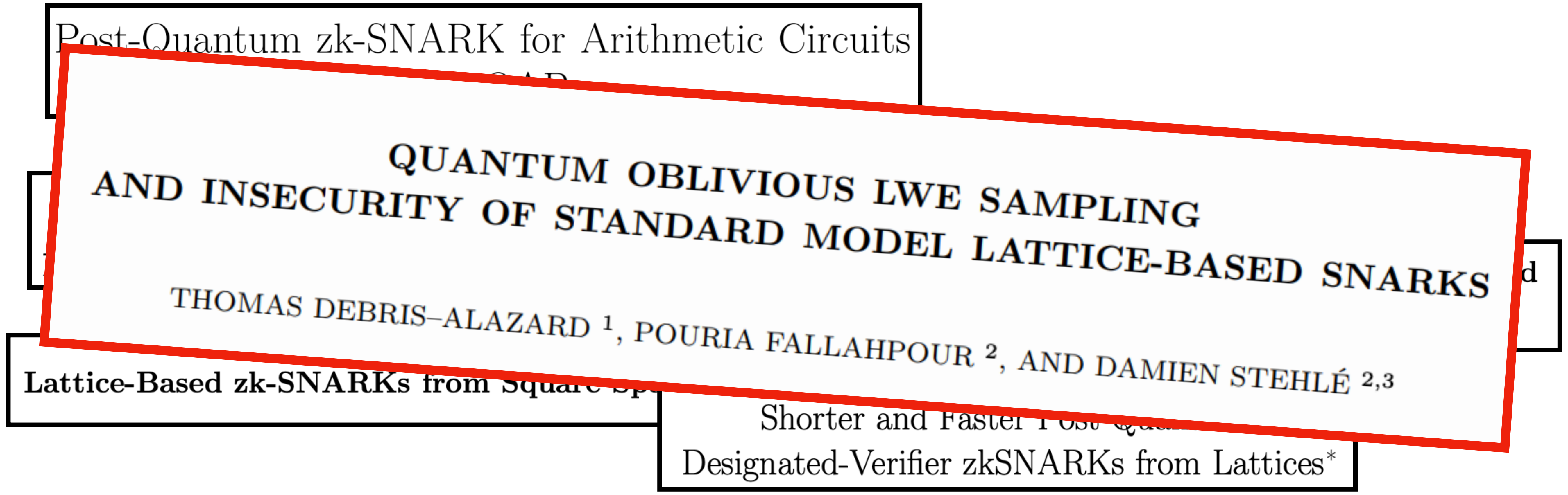
Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling



Lattice-Based SNARKs

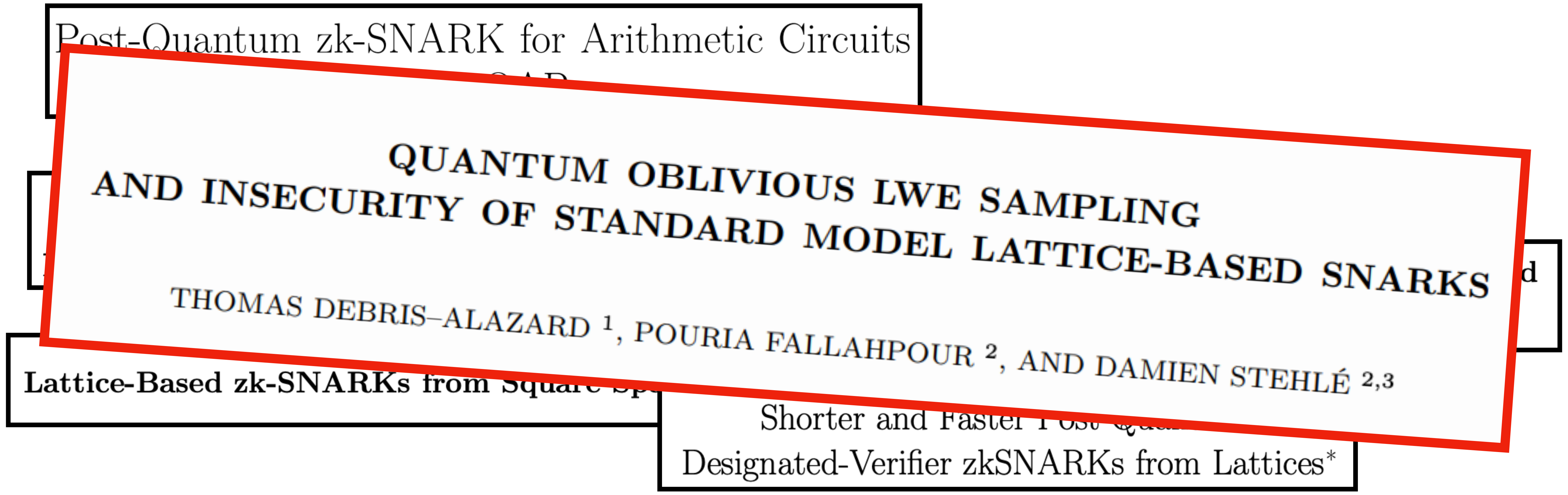
How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Knowledge k-RI-SIS



Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits

QUANTUM OBLIVIOUS LWE SAMPLING
AND INSECURITY OF STANDARD MODEL LATTICE-BASED SNARKS

THOMAS DEBRIS–ALAZARD [1], POURIA FALLAHPOUR [2], AND DAMIEN STEHLÉ [2,3]

Lattice-Based zk-SNARKs from Square

Shorter and Faster

Designated-Verifier zkSNARKs from Lattices*

Knowledge k-RI-SIS

Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable
(Full Version)

Lattice-Based SNARKs

How to get around [GW11]?

[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Post-Quantum zk-SNARK for Arithmetic Circuits

QUANTUM OBLIVIOUS LWE SAMPLING
AND INSECURITY OF STANDARD MODEL LATTICE-BASED SNARKS

THOMAS DEBRIS–ALAZARD [1], POURIA FALLAHPOUR [2], AND DAMIEN STEHLÉ [2,3]

Lattice-Based zk-SNARKs from Square Sp

Shorter and Faster Post-Qua

Designated-Verifier zkSNARKs from Lattices*

Knowledge k-RI-SIS

Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable
(Full Version)

Lattice-based Succinct Arguments from Vanishing Polynomials
(Full Version)

Lattice-Based SNARKs

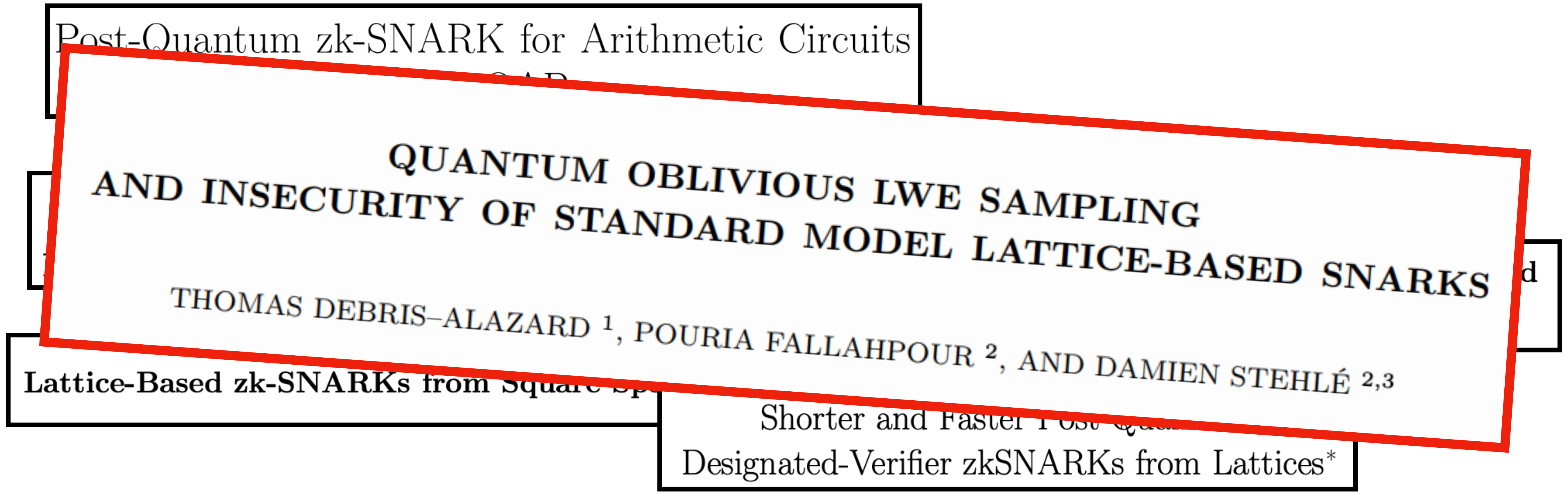
How to get around [GW11]?

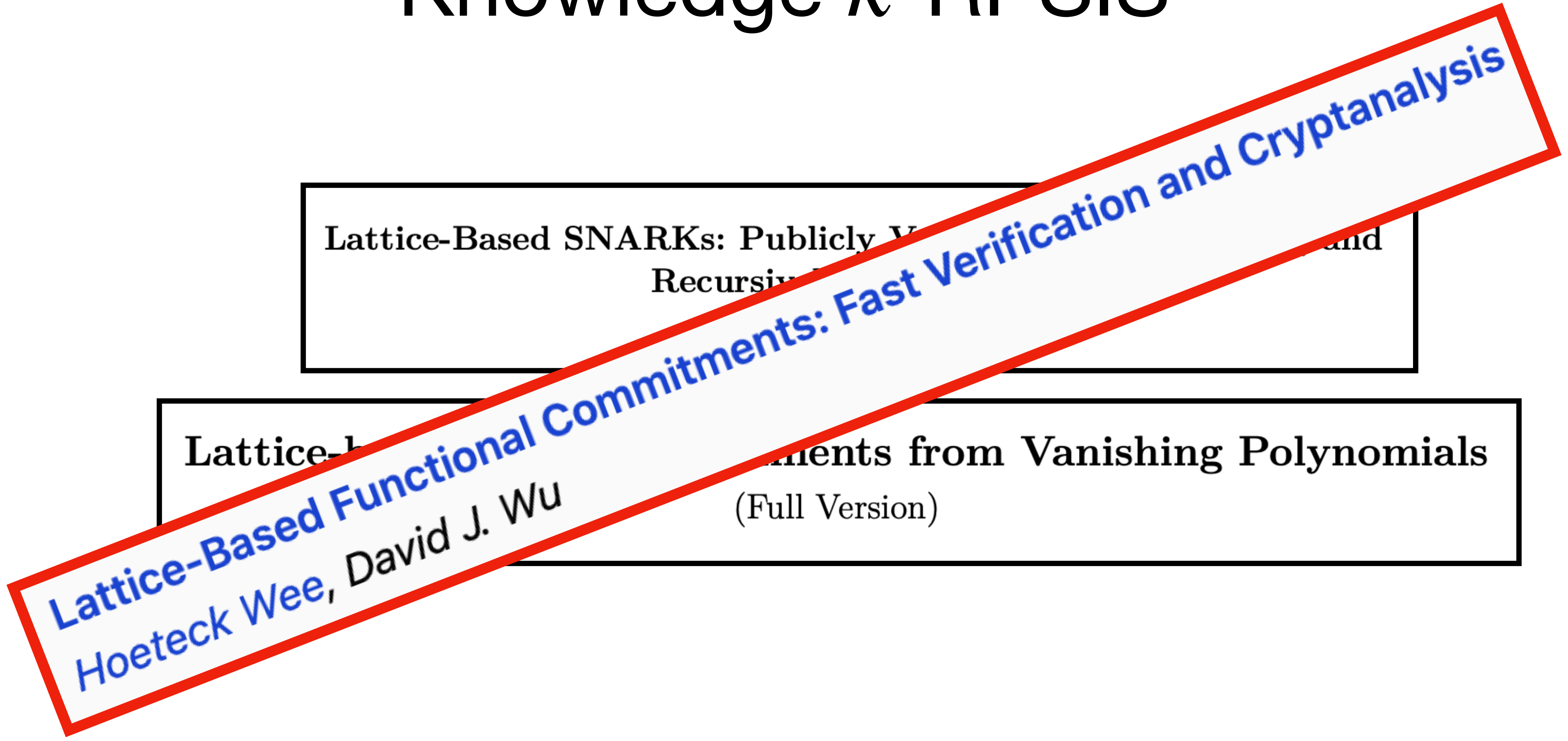
[GW11] - You cannot get **SN**ARG from falsifiable assumptions.

Knowledge Assumptions

Oblivious LWE Sampling

Knowledge k-RI-SIS





Lattice Assumptions ❤️ ROM

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in “lattice-land”: hard to define and easy-ish to break

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in “lattice-land”: hard to define and easy-ish to break
- ROM takes care of **extraction** and **non-interactivity**.

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in "lattice-land": hard to define and easy-ish to break
- ROM takes care of **extraction** and **non-interactivity**.

Special Sound
Interactive Protocol

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in “lattice-land”: hard to define and easy-ish to break
- ROM takes care of **extraction** and **non-interactivity**.

Special Sound
Interactive Protocol

+

Fiat-Shamir
Transform

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in “lattice-land”: hard to define and easy-ish to break
- ROM takes care of **extraction** and **non-interactivity**.

Special Sound Interactive Protocol + Fiat-Shamir Transform = Knowledge Sound PCS

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in “lattice-land”: hard to define and easy-ish to break
- ROM takes care of **extraction** and **non-interactivity**.

Special Sound Interactive Protocol + Fiat-Shamir Transform = Knowledge Sound PCS

- Use lattices to get succinctness in the interactive protocol.

Lattice Assumptions ❤️ ROM

- Knowledge assumptions in “lattice-land”: hard to define and easy-ish to break
- ROM takes care of **extraction** and **non-interactivity**.

Special Sound Interactive Protocol + Fiat-Shamir Transform = Knowledge Sound PCS

- Use lattices to get succinctness in the interactive protocol.
- **Open Question**: ROM alone is sufficient for efficient PCS (e.g. STIR), can we gain by using lattices?

Building succinct PCS

Building succinct PCS

Commitment Scheme

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathscr{R}_q^d$

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathscr{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

- Need commitment $|\mathbf{t}| \ll d$

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathscr{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathscr{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol

- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol

$\mathbf{t}, u, v$

V

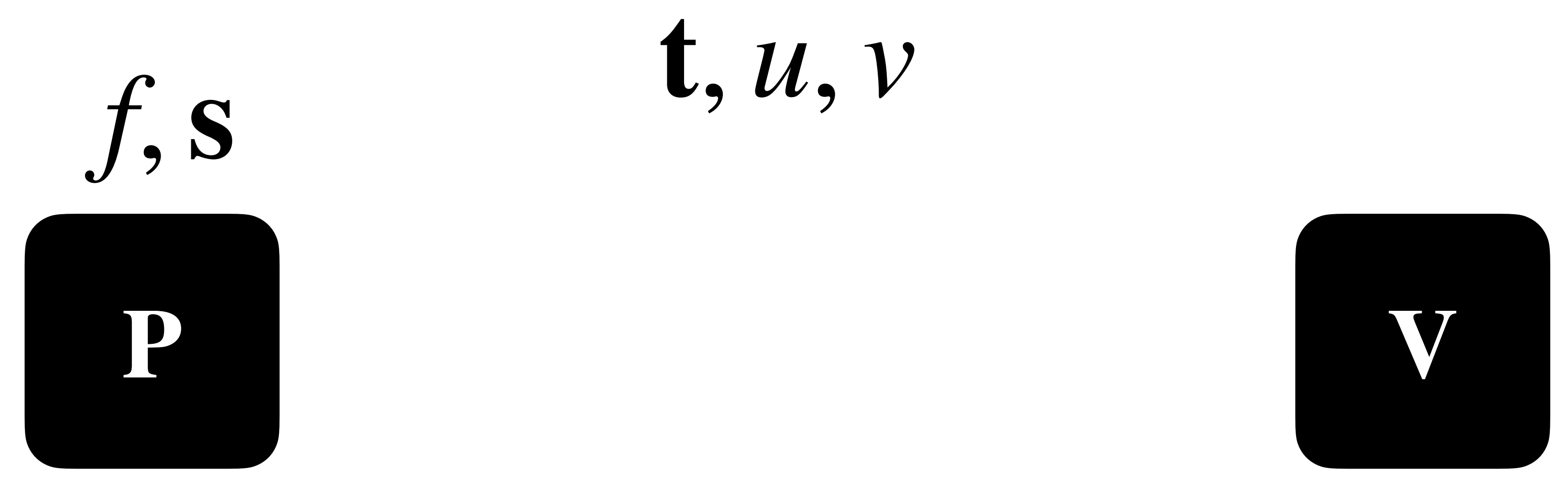
- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol



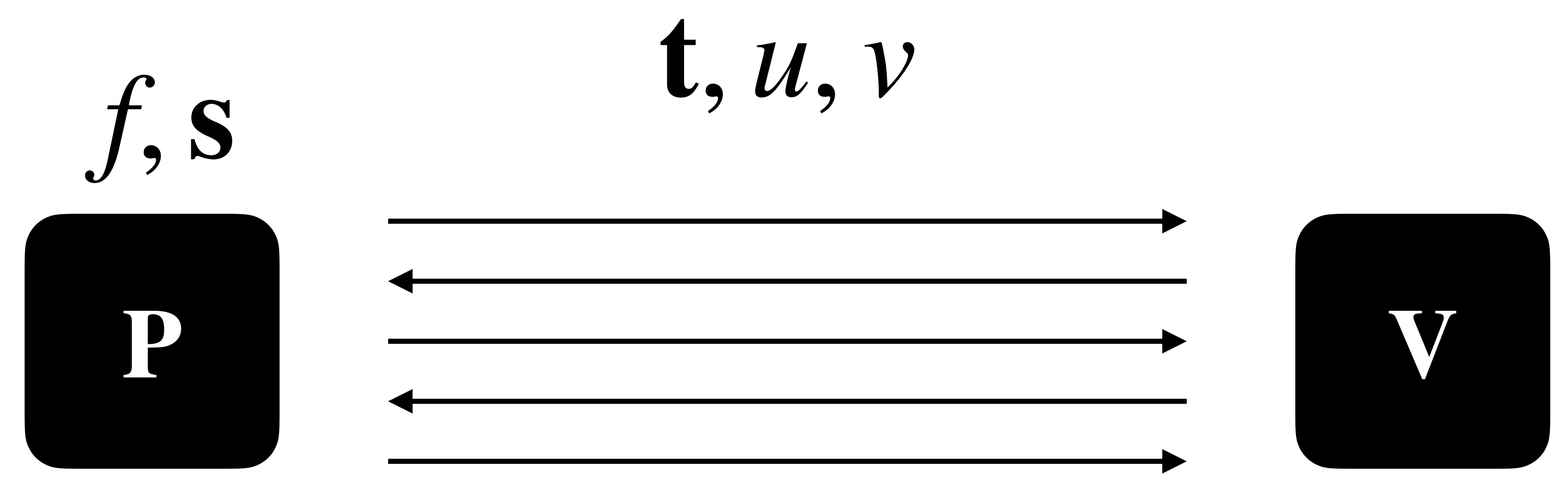
- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol



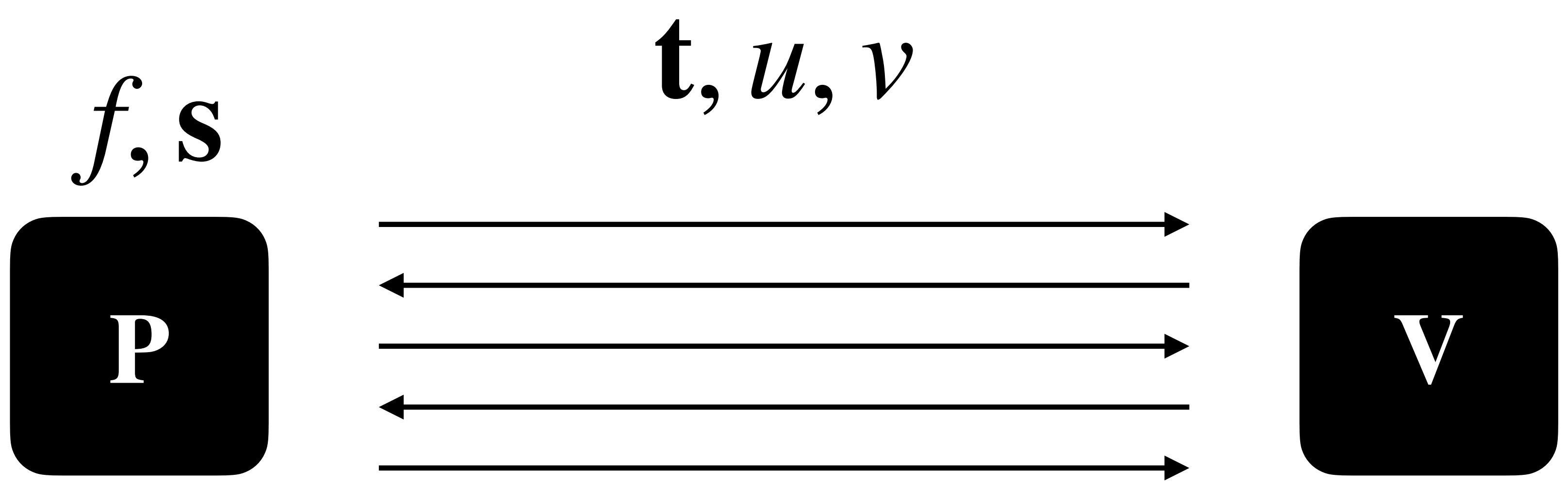
- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol



"I know f such that $f(u) = v$ and an opening $\mathbf{s}$ for $\mathbf{f} :=$ coeff(f) to $\mathbf{t}$"

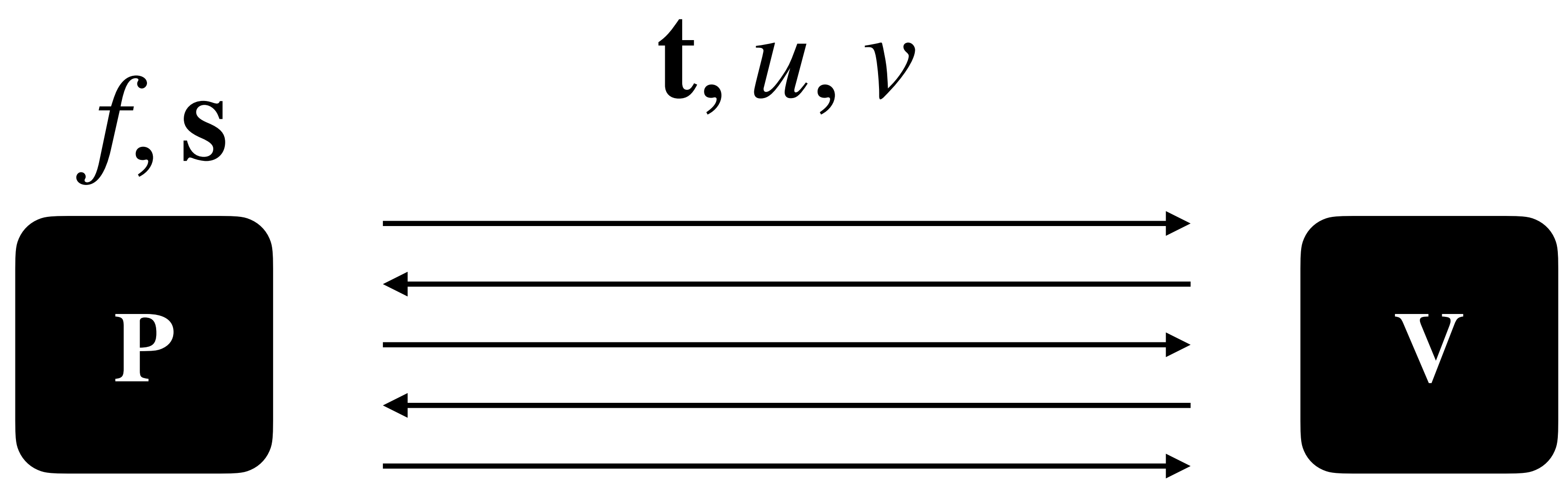
- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol



"I know f such that $f(u) = v$ and an opening $\mathbf{s}$ for $\mathbf{f} :=$ coeff(f) to $\mathbf{t}$"

- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

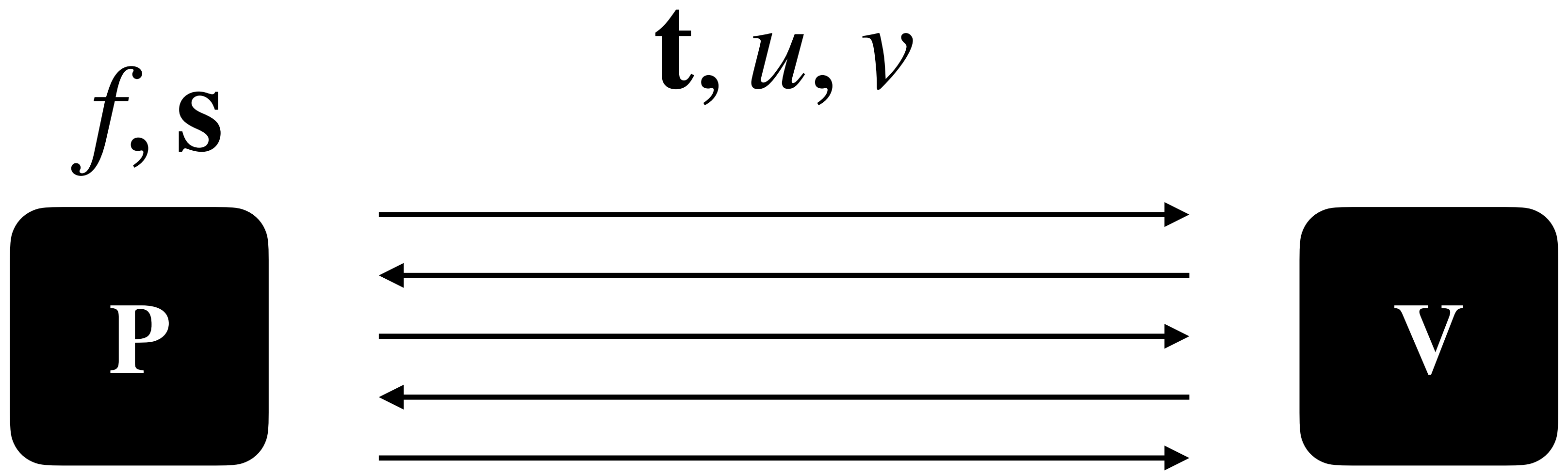
- Need $\mathbf{V}$'s running time to be $\ll d$

Building succinct PCS

Commitment Scheme

- Commit to a vector $\mathbf{f} \in \mathcal{R}_q^d$
- Commitment $\mathbf{t}$, opening $\mathbf{s}$
- Binding under lattice assumption

Evaluation Protocol



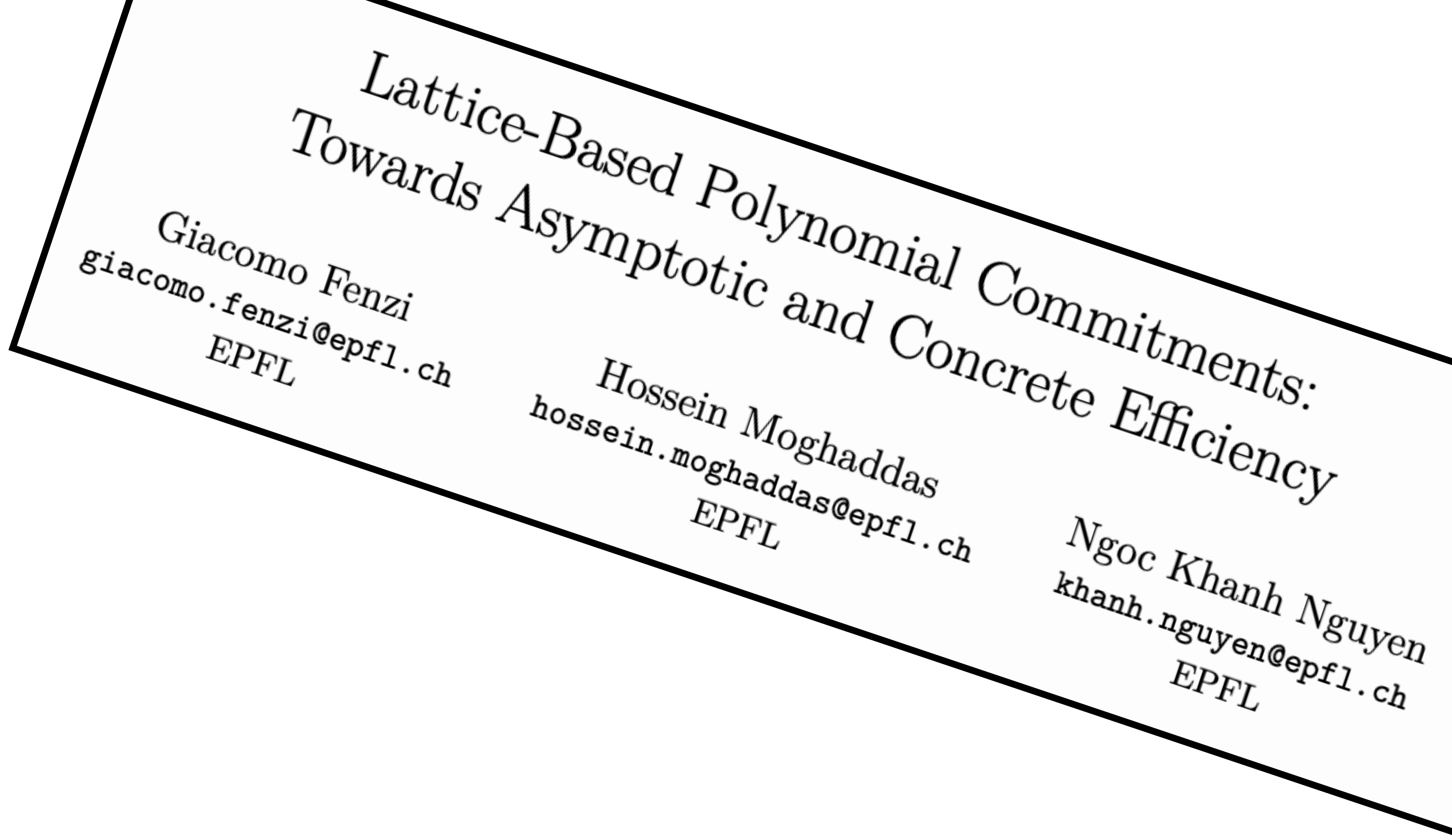
"I know f such that $f(u) = v$ and an opening $\mathbf{s}$ for $\mathbf{f} := \text{coeff}(f)$ to $\mathbf{t}$"

- Need commitment $|\mathbf{t}| \ll d$
- Must be binding for $\mathbf{f}$ of arbitrary norm

- Need $\mathbf{V}$'s running time to be $\ll d$
- Need communication complexity $\ll d$

PRISIS Commitments I

A starting point [FMN23]



Lattice-Based Polynomial Commitments:
Towards Asymptotic and Concrete Efficiency

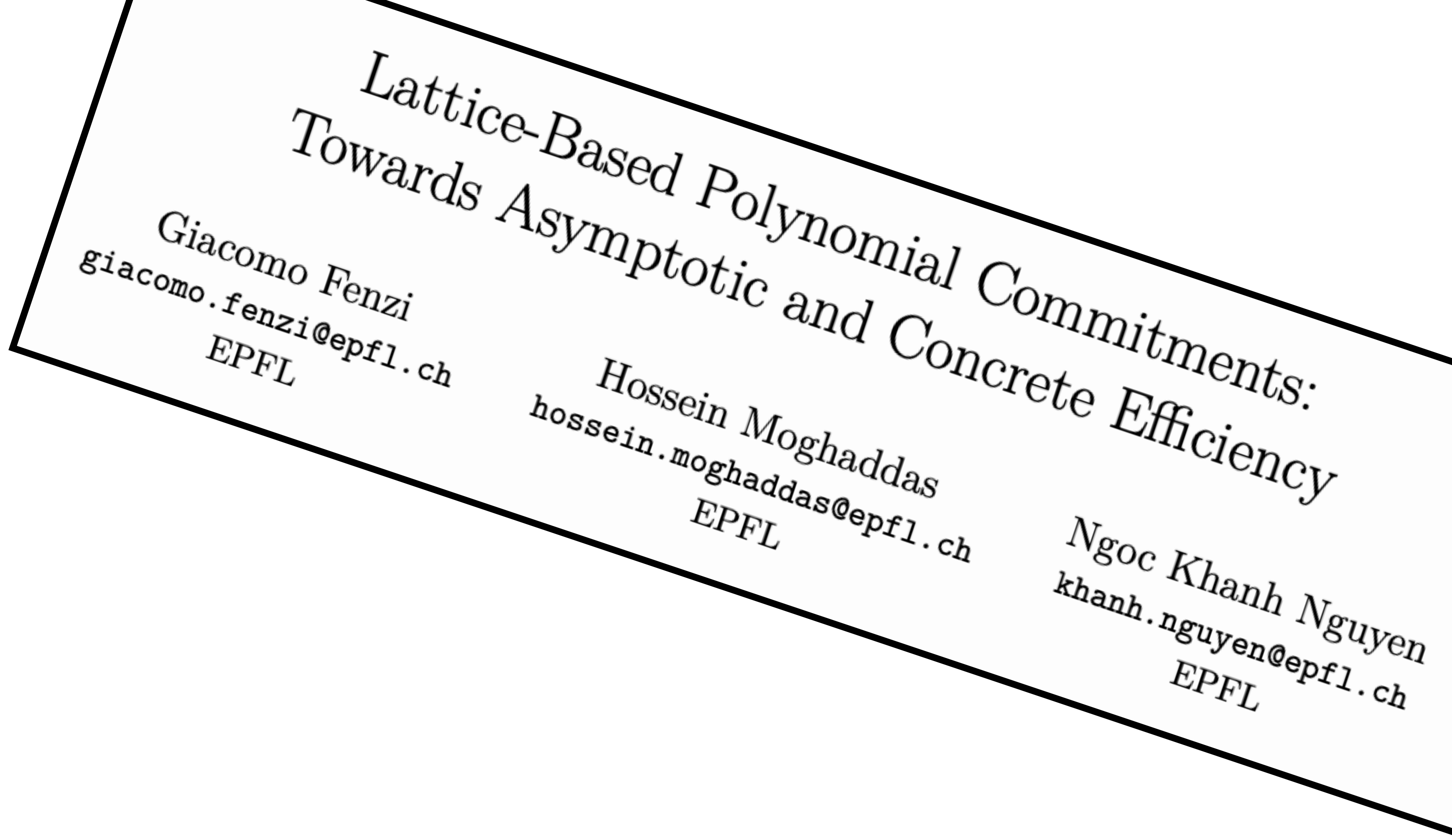
Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Hossein Moghaddas
hossein.moghaddas@epfl.ch
EPFL

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

PRISIS Commitments I

A starting point [FMN23]


Lattice-Based Polynomial Commitments:
Towards Asymptotic and Concrete Efficiency

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

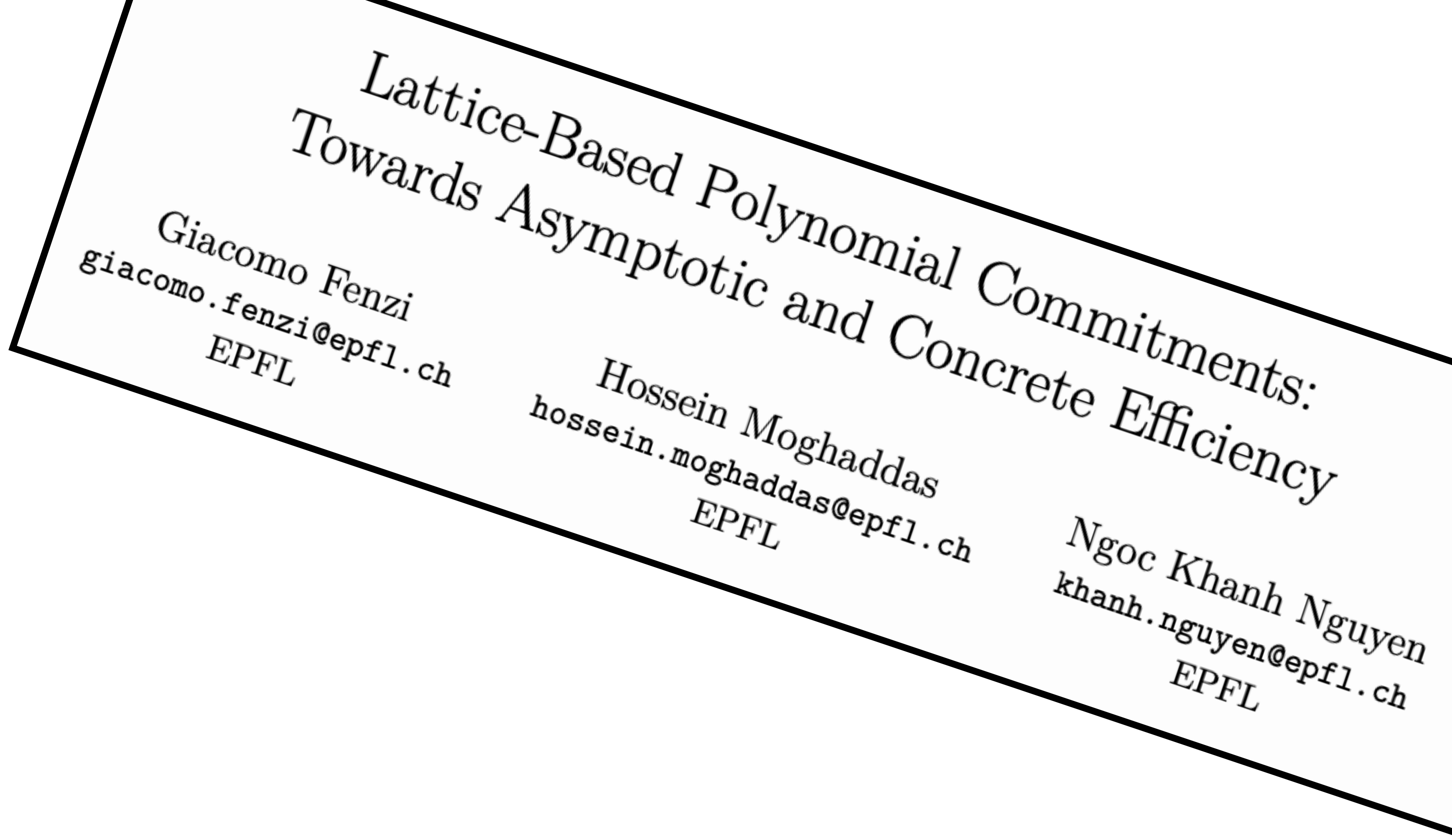
Hossein Moghaddas
hossein.moghaddas@epfl.ch
EPFL

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

Given $\mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \dots & & -\mathbf{G} \\ & \ddots & & \\ & \dots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$ and **trapdoor** $\mathbf{T}$ for $\mathbf{B}$

PRISIS Commitments I

A starting point [FMN23]


Lattice-Based Polynomial Commitments:
Towards Asymptotic and Concrete Efficiency

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Hossein Moghaddas
hossein.moghaddas@epfl.ch
EPFL

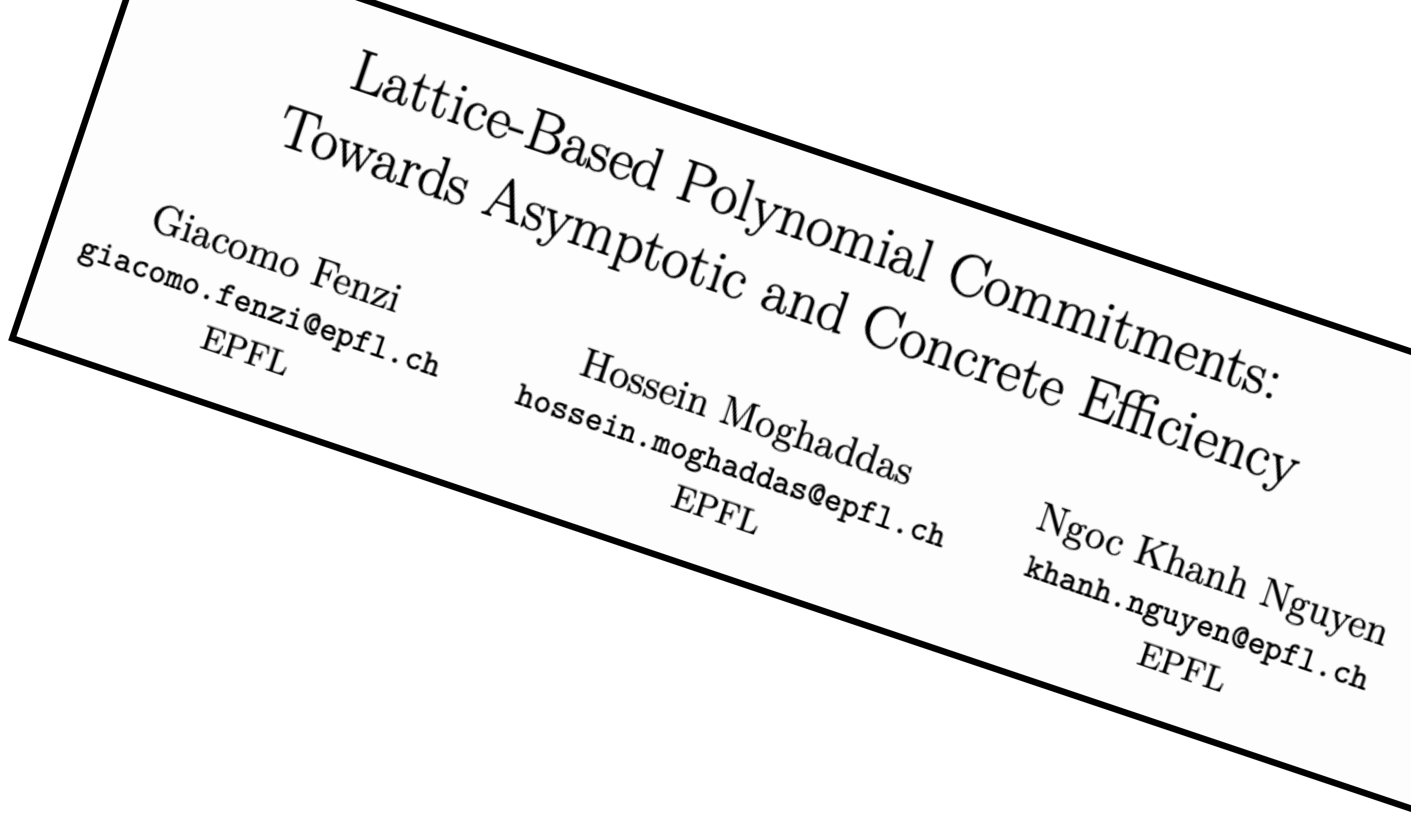
Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

Given $\mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \dots & & -\mathbf{G} \\ & \ddots & & \\ & \dots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$ and **trapdoor** $\mathbf{T}$ for $\mathbf{B}$

Use $\mathbf{T}$ to sample short $\mathbf{s}_0, \dots, \mathbf{s}_{\ell-1}, \hat{\mathbf{t}}$ such that:

PRISIS Commitments I

A starting point [FMN23]


Lattice-Based Polynomial Commitments:
Towards Asymptotic and Concrete Efficiency

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Hossein Moghaddas
hossein.moghaddas@epfl.ch
EPFL

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

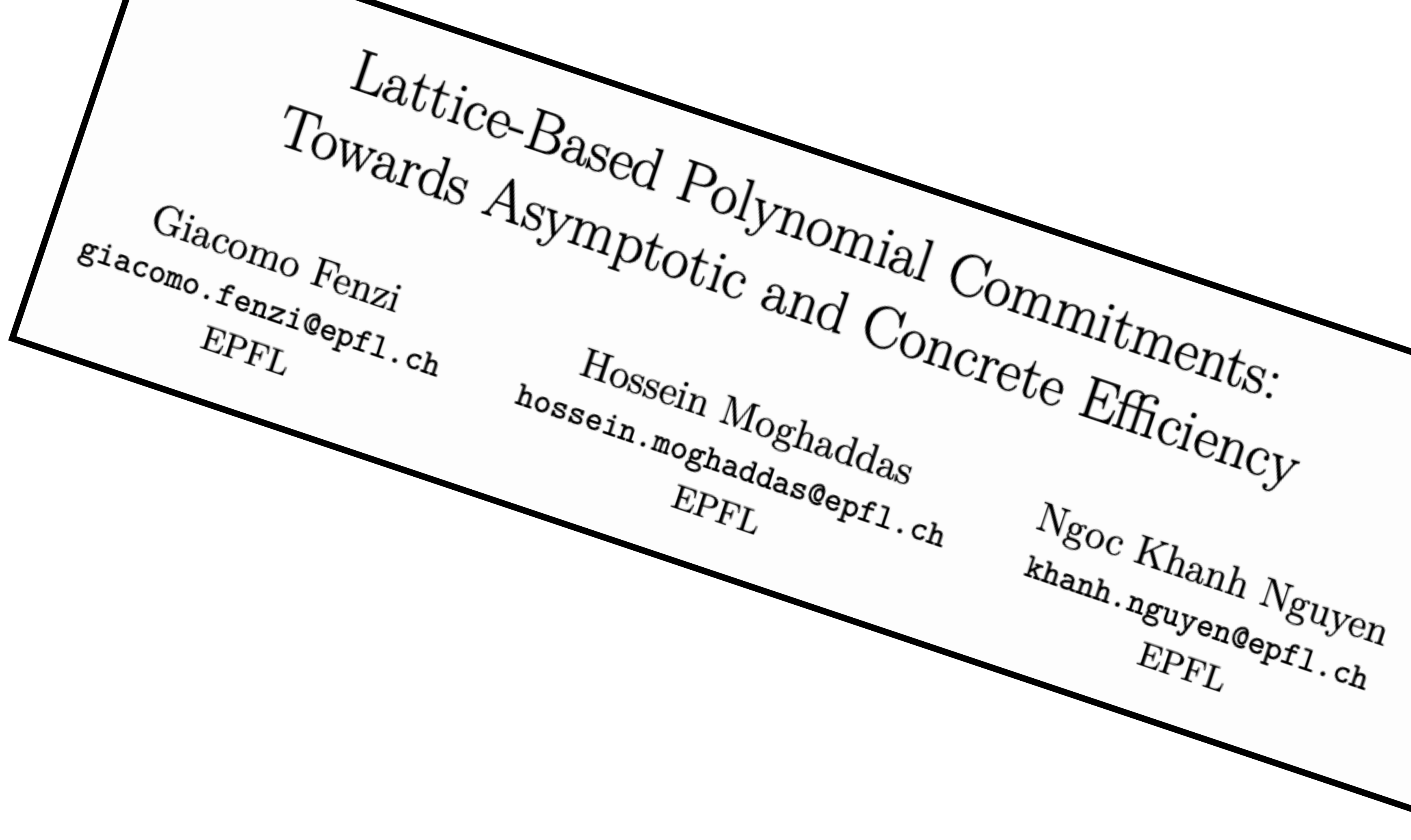
Given $\mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \dots & & -\mathbf{G} \\ & \ddots & & \\ & \dots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$ and **trapdoor** $\mathbf{T}$ for $\mathbf{B}$

Use $\mathbf{T}$ to sample short $\mathbf{s}_0, \dots, \mathbf{s}_{\ell-1}, \hat{\mathbf{t}}$ such that:

$$\mathbf{B} \underbrace{\begin{bmatrix} \mathbf{s}_0 \\ \vdots \\ \mathbf{s}_{\ell-1} \\ \hat{\mathbf{t}} \end{bmatrix}}_{\text{preimage}} = \underbrace{\begin{bmatrix} -f_0 w^0 \mathbf{e}_1 \\ \vdots \\ -f_{\ell-1} w^{\ell-1} \mathbf{e}_1 \end{bmatrix}}_{\text{target}}$$

PRISIS Commitments I

A starting point [FMN23]


Lattice-Based Polynomial Commitments:
Towards Asymptotic and Concrete Efficiency

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Hossein Moghaddas
hossein.moghaddas@epfl.ch
EPFL

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

Given $\mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \dots & & -\mathbf{G} \\ & \ddots & & \\ & \dots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$ and **trapdoor** $\mathbf{T}$ for $\mathbf{B}$

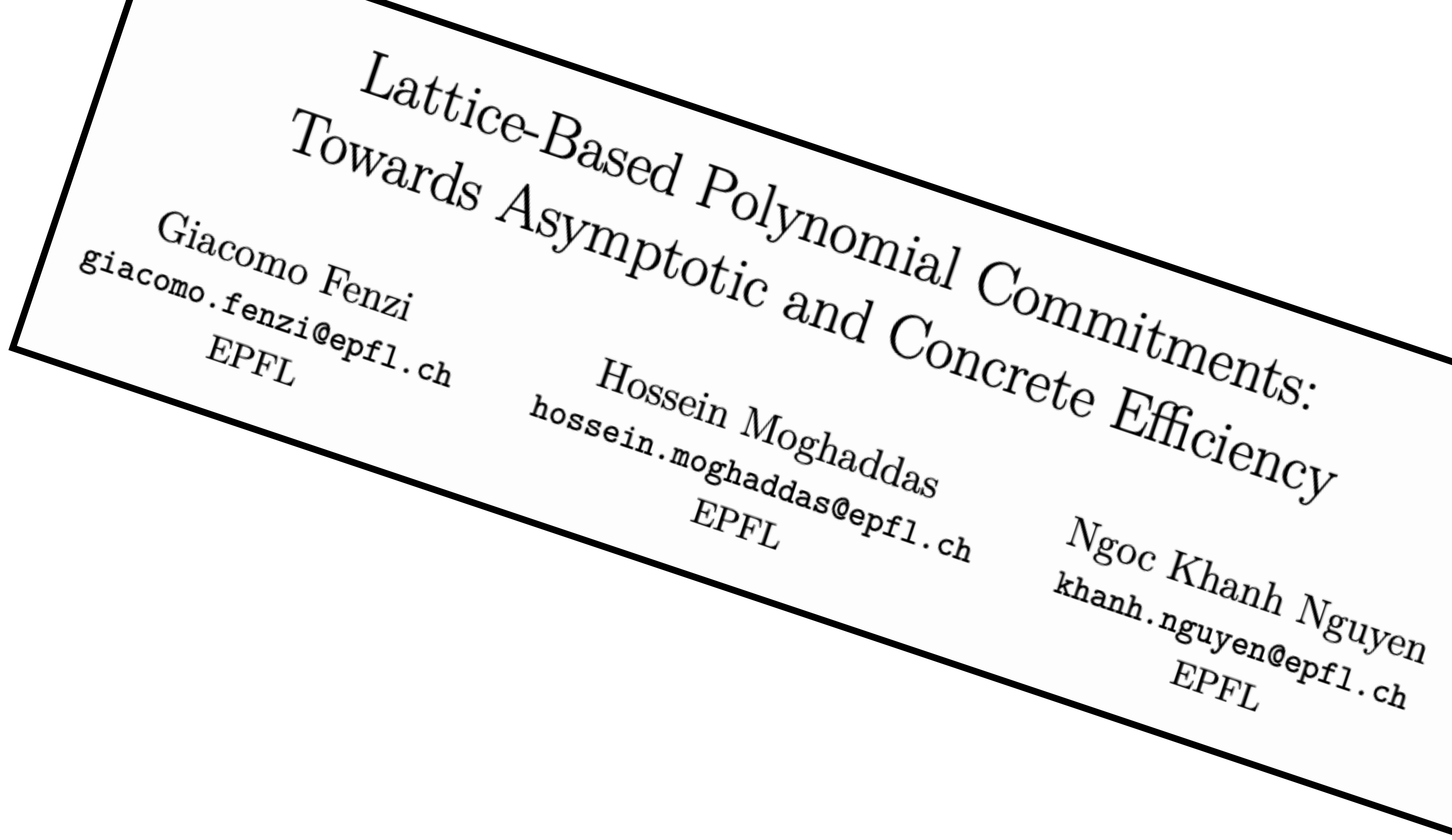
Use $\mathbf{T}$ to sample short $\mathbf{s}_0, \dots, \mathbf{s}_{\ell-1}, \hat{\mathbf{t}}$ such that:

$$\mathbf{B} \underbrace{\begin{bmatrix} \mathbf{s}_0 \\ \vdots \\ \mathbf{s}_{\ell-1} \\ \hat{\mathbf{t}} \end{bmatrix}}_{\text{preimage}} = \underbrace{\begin{bmatrix} -f_0 w^0 \mathbf{e}_1 \\ \vdots \\ -f_{\ell-1} w^{\ell-1} \mathbf{e}_1 \end{bmatrix}}_{\text{target}}$$

The commitment is $\mathbf{t} := \mathbf{G}\hat{\mathbf{t}}$ and the openings are $(\mathbf{s}_i)_i$.

PRISIS Commitments I

A starting point [FMN23]



Given $\mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \dots & & -\mathbf{G} \\ & \ddots & & \\ & \dots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$ and **trapdoor** $\mathbf{T}$ for $\mathbf{B}$

Use $\mathbf{T}$ to sample short $\mathbf{s}_0, \dots, \mathbf{s}_{\ell-1}, \hat{\mathbf{t}}$ such that:

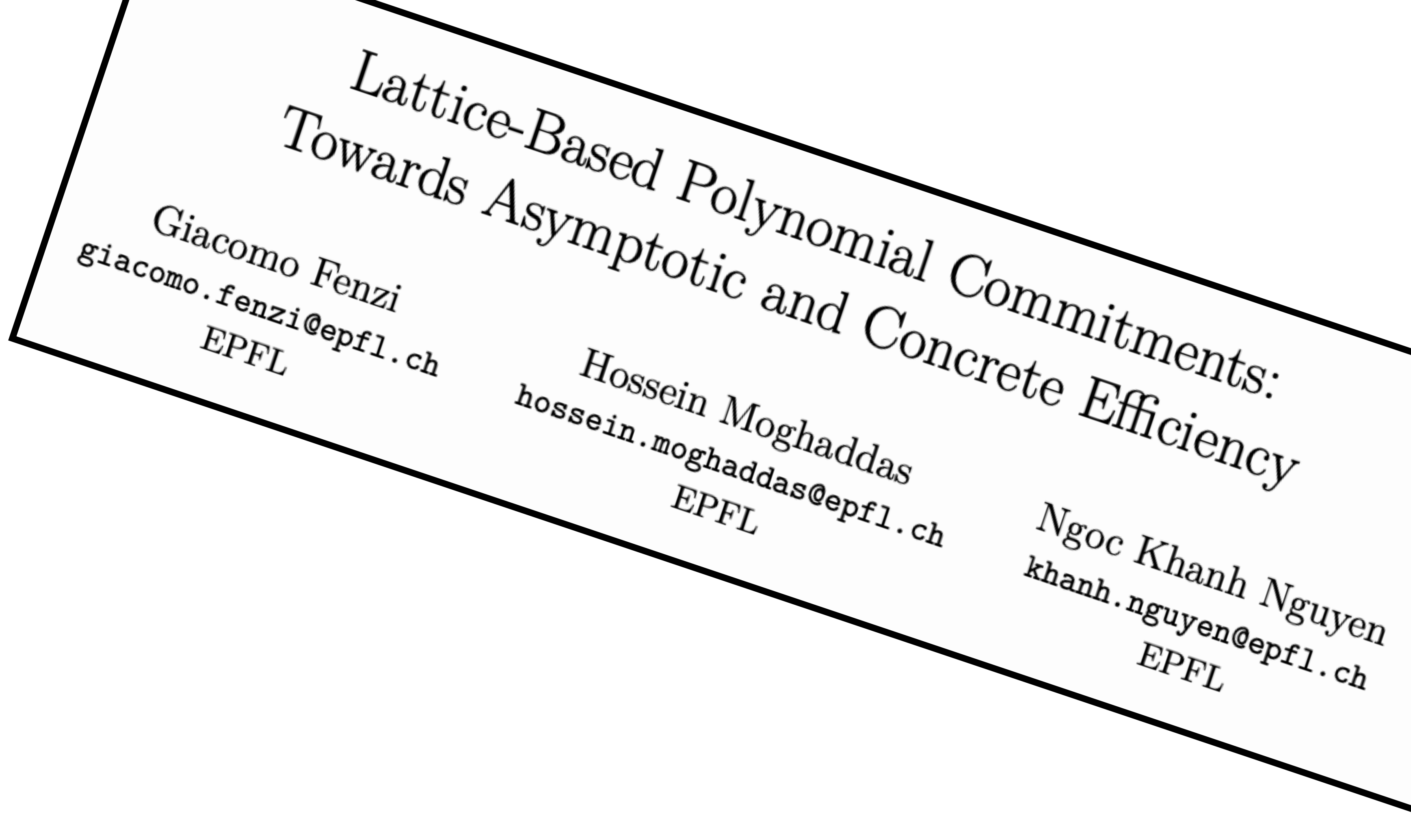
$$\mathbf{B} \underbrace{\begin{bmatrix} \mathbf{s}_0 \\ \vdots \\ \mathbf{s}_{\ell-1} \\ \hat{\mathbf{t}} \end{bmatrix}}_{\text{preimage}} = \underbrace{\begin{bmatrix} -f_0 w^0 \mathbf{e}_1 \\ \vdots \\ -f_{\ell-1} w^{\ell-1} \mathbf{e}_1 \end{bmatrix}}_{\text{target}}$$

The commitment is $\mathbf{t} := \mathbf{G}\hat{\mathbf{t}}$ and the openings are $(\mathbf{s}_i)_i$.

To open check that

PRISIS Commitments I

A starting point [FMN23]


Lattice-Based Polynomial Commitments:
Towards Asymptotic and Concrete Efficiency

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Hossein Moghaddas
hossein.moghaddas@epfl.ch
EPFL

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

Given $\mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \dots & & -\mathbf{G} \\ & \ddots & & \\ & \dots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$ and **trapdoor** $\mathbf{T}$ for $\mathbf{B}$

Use $\mathbf{T}$ to sample short $\mathbf{s}_0, \dots, \mathbf{s}_{\ell-1}, \hat{\mathbf{t}}$ such that:

$$\mathbf{B} \underbrace{\begin{bmatrix} \mathbf{s}_0 \\ \vdots \\ \mathbf{s}_{\ell-1} \\ \hat{\mathbf{t}} \end{bmatrix}}_{\text{preimage}} = \underbrace{\begin{bmatrix} -f_0 w^0 \mathbf{e}_1 \\ \vdots \\ -f_{\ell-1} w^{\ell-1} \mathbf{e}_1 \end{bmatrix}}_{\text{target}}$$

The commitment is $\mathbf{t} := \mathbf{G}\hat{\mathbf{t}}$ and the openings are $(\mathbf{s}_i)_i$.

To open check that

$\mathbf{A}\mathbf{s}_i + f_i\mathbf{e}_1 = w^{-i}\mathbf{t}$ and $\mathbf{s}_i$ short

PRISIS Commitments II

Pros ✅ and Cons ❌

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Algebraic structure enables **efficient evaluation protocol**.

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Algebraic structure enables **efficient evaluation protocol**.

- Binding under **non-standard** PRISIS assumption.

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Algebraic structure enables **efficient evaluation protocol**.

- Binding under **non-standard** PRISIS assumption.
- Time to commit is **quadratic**.

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Algebraic structure enables **efficient evaluation protocol**.

- Binding under **non-standard** PRISIS assumption.
- Time to commit is **quadratic**.
- Common reference string is **quadratic**.

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Algebraic structure enables **efficient evaluation protocol**.

- Binding under **non-standard** PRISIS assumption.
- Time to commit is **quadratic**.
- Common reference string is **quadratic**.
- **Trusted** setup

PRISIS Commitments II

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Algebraic structure enables **efficient evaluation protocol**.

- Binding under **non-standard** PRISIS assumption.
- Time to commit is **quadratic**.
- Common reference string is **quadratic**.
- **Trusted** setup

Can we do better?

Small-Dimension PRISIS

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS)**.** *Let* $n > 0, m \geq n$ *and denote* $t = (n+1)\tilde{q}$. *Let* $q = \omega(N)$. *Take* $\epsilon \in (0, 1/3)$ *and* $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ *such that* $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ *is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS). *Let $n > 0, m \geq n$ and denote $t = (n+1)\tilde{q}$. Let $q = \omega(N)$. Take $\epsilon \in (0, 1/3)$ and $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ such that $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS). *Let $n > 0, m \geq n$ and denote $t = (n+1)\tilde{q}$. Let $q = \omega(N)$. Take $\epsilon \in (0, 1/3)$ and $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ such that $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

h-instance BASIS Game

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS)**.** *Let* $n > 0, m \geq n$ *and denote* $t = (n+1)\tilde{q}$*. Let* $q = \omega(N)$*. Take* $\epsilon \in (0, 1/3)$ *and* $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ *such that* $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ *is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

h-instance BASIS Game

$$\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star \leftarrow \mathcal{R}_q^{m \times n}$$

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS)**.** *Let $n > 0, m \geq n$ and denote $t = (n+1)\tilde{q}$. Let $q = \omega(N)$. Take $\epsilon \in (0, 1/3)$ and $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ such that $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

h-instance BASIS Game

$\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star \leftarrow \mathscr{R}_q^{m \times n}$

$\mathsf{aux}_i \leftarrow \mathsf{Samp}(\mathbf{A}_i^\star)$ for $i \in [h]$

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS)**.** *Let* $n > 0, m \geq n$ *and denote* $t = (n+1)\tilde{q}$*. Let* $q = \omega(N)$*. Take* $\epsilon \in (0, 1/3)$ *and* $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ *such that* $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ *is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

h-instance BASIS Game

$\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star \leftarrow \mathcal{R}_q^{m \times n}$

$\mathsf{aux}_i \leftarrow \mathsf{Samp}(\mathbf{A}_i^\star)$ for $i \in [h]$

return $((\mathbf{A}_i^\star, \mathsf{aux}_i)_i)$ to $\mathcal{A}$

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS). *Let $n > 0, m \geq n$ and denote $t = (n+1)\tilde{q}$. Let $q = \omega(N)$. Take $\epsilon \in (0, 1/3)$ and $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ such that $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

h-instance BASIS Game

$\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star \leftarrow \mathscr{R}_q^{m \times n}$

$\mathsf{aux}_i \leftarrow \mathsf{Samp}(\mathbf{A}_i^\star)$ for $i \in [h]$

return $((\mathbf{A}_i^\star, \mathsf{aux}_i)_i)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $[\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star] \cdot \mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Small-Dimension PRISIS

[FMN23]: $\ell = 2$ reduces to MSIS

Lemma 3.6 (PRISIS $\implies$ MSIS)**.** *Let* $n > 0, m \geq n$ *and denote* $t = (n+1)\tilde{q}$. *Let* $q = \omega(N)$. *Take* $\epsilon \in (0, 1/3)$ *and* $\mathfrak{s} \geq \max(\sqrt{N \ln(8Nq)} \cdot q^{1/2+\epsilon}, \omega(N^{3/2} \ln^{3/2} N))$ *such that* $2^{10N} q^{-\lfloor \epsilon N \rfloor}$ *is negligible. Let*

$$\sigma \geq \delta \sqrt{tN \cdot (N^2 \mathfrak{s}^2 m + 2t)} \cdot \omega(\sqrt{N \log nN}).$$

Then, $\mathsf{PRISIS}_{n,m,N,q,2,\sigma,\beta}$ *is hard under the* $\mathsf{MSIS}_{n,m,N,q,\beta}$ *assumption.*

Multi-Instance BASIS

h-instance BASIS Game

$\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star \leftarrow \mathcal{R}_q^{m \times n}$

$\mathsf{aux}_i \leftarrow \mathsf{Samp}(\mathbf{A}_i^\star)$ for $i \in [h]$

return $((\mathbf{A}_i^\star, \mathsf{aux}_i)_i)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $[\mathbf{A}_1^\star, \ldots, \mathbf{A}_h^\star] \cdot \mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

For $\ell = O(1)$, if PRISIS_ℓ is hard so is h-PRISIS_ℓ!

Merkle-PRISIS I

Example with $d = 8$

Merkle-PRISIS I

Example with $d = 8$

f_{000}

Merkle-PRISIS I

Example with $d = 8$

f_{000} f_{001}

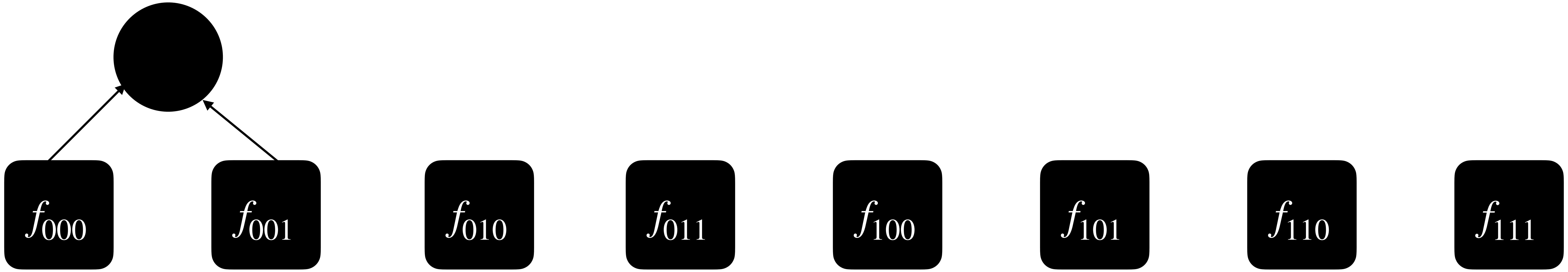
Merkle-PRISIS I

Example with $d = 8$

f_{000} f_{001} f_{010} f_{011} f_{100} f_{101} f_{110} f_{111}

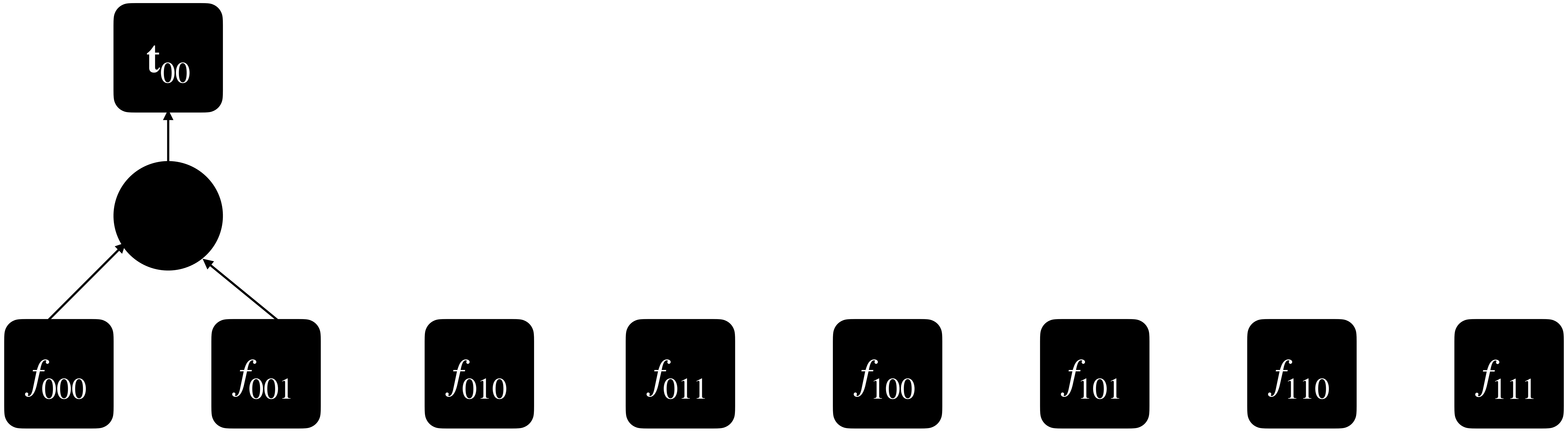
Merkle-PRISIS I

Example with $d = 8$



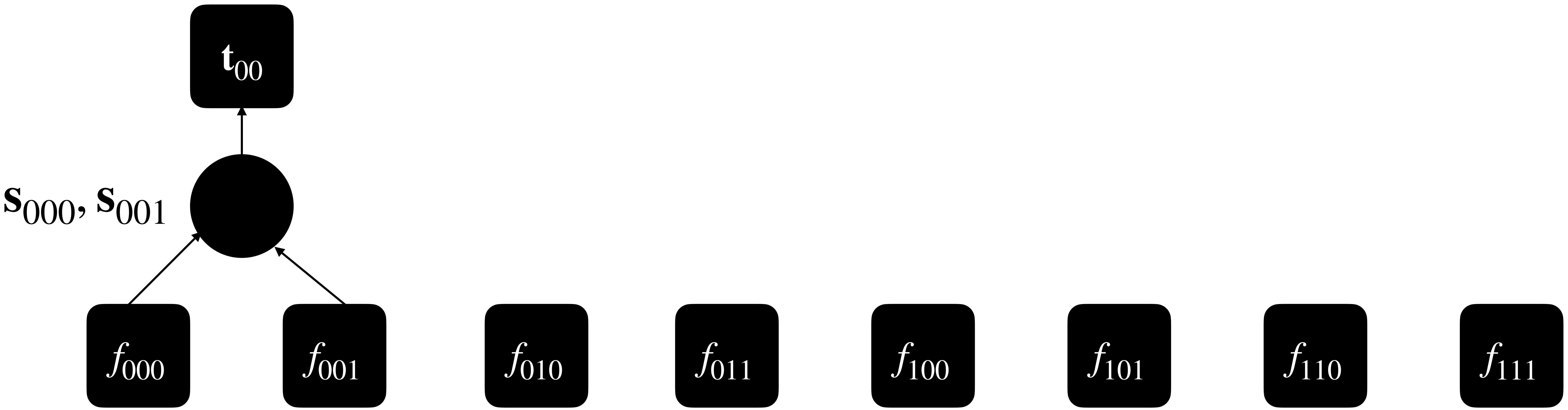
Merkle-PRISIS I

Example with $d = 8$



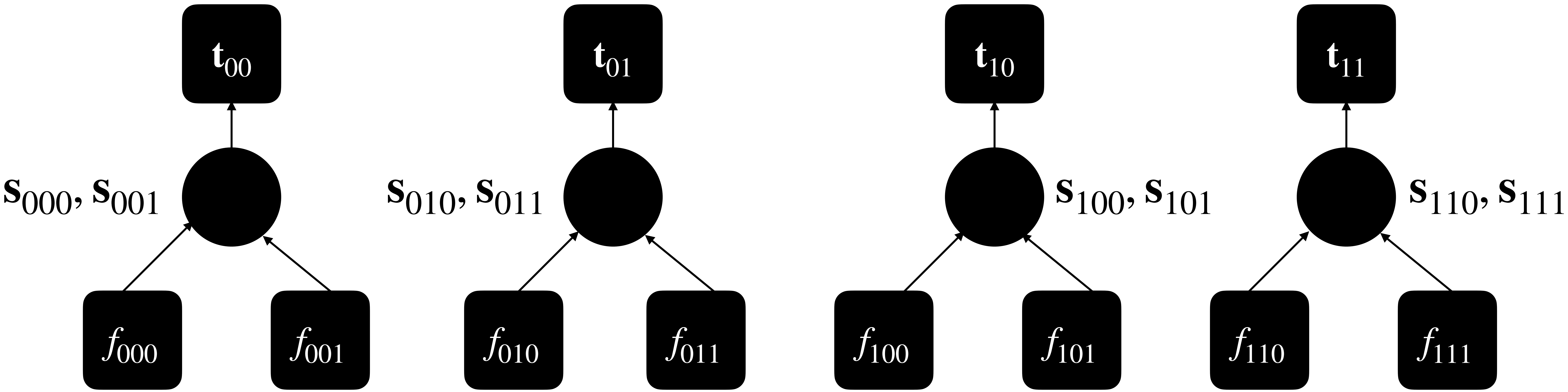
Merkle-PRISIS I

Example with $d = 8$



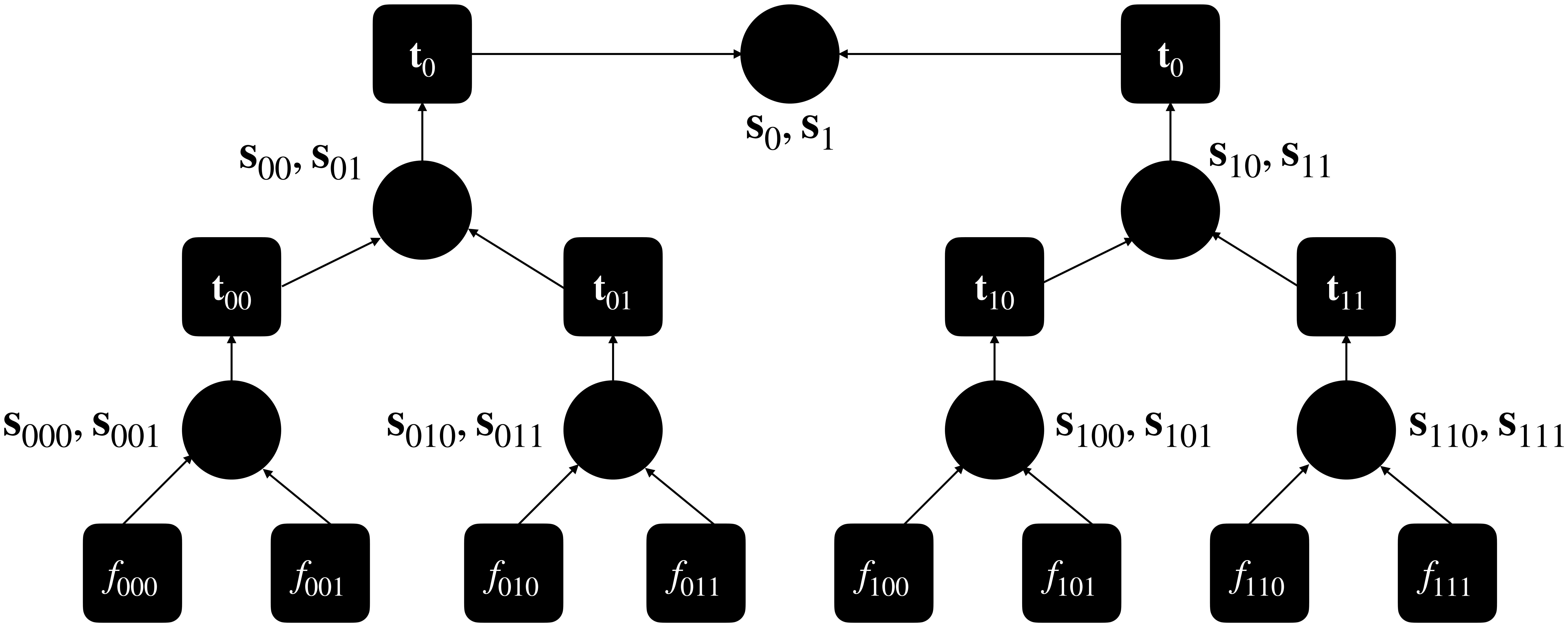
Merkle-PRISIS I

Example with $d = 8$



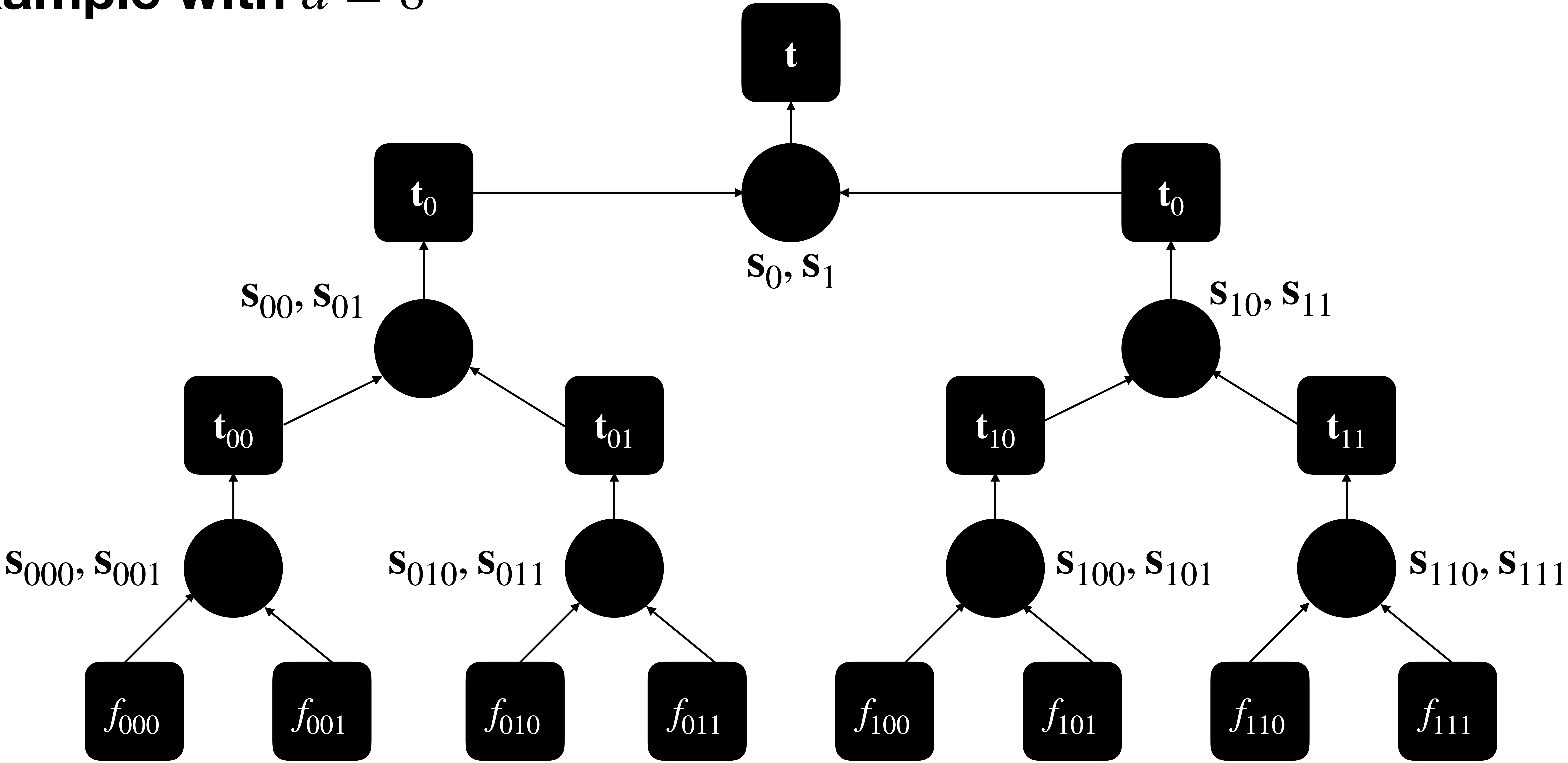
Merkle-PRISIS I

Example with $d = 8$



Merkle-PRISIS I

Example with $d = 8$



Merkle-PRISIS II

How to check an opening

Merkle-PRISIS II

How to check an opening

- Each layer has its own $\mathsf{crs}_j := (\mathbf{A}_j, w_j, \mathbf{T}_j)$ for $j \in [h := \log d]$

Merkle-PRISIS II

How to check an opening

- Each layer has its own $\mathsf{crs}_j := (\mathbf{A}_j, w_j, \mathbf{T}_j)$ for $j \in [h := \log d]$
- Check that all local openings are correct. I.e. check that, for $\mathbf{b} \in \{0,1\}^h$:

Merkle-PRISIS II

How to check an opening

- Each layer has its own $\mathsf{crs}_j := (\mathbf{A}_j, w_j, \mathbf{T}_j)$ for $j \in [h := \log d]$
- Check that all local openings are correct. I.e. check that, for $\mathbf{b} \in \{0,1\}^h$:

$$\sum_{j \in [h]} w_j^{b_j} \mathbf{A}_j \mathbf{s}_{\mathbf{b}:j} + f_{\mathbf{b}} \cdot \mathbf{e} = \mathbf{t}$$

Merkle-PRISIS II

How to check an opening

- Each layer has its own $\mathsf{crs}_j := (\mathbf{A}_j, w_j, \mathbf{T}_j)$ for $j \in [h := \log d]$
- Check that all local openings are correct. I.e. check that, for $\mathbf{b} \in \{0,1\}^h$:

$$\sum_{j \in [h]} w_j^{b_j} \mathbf{A}_j \mathbf{s}_{\mathbf{b}:j} + f_{\mathbf{b}} \cdot \mathbf{e} = \mathbf{t}$$

- And, of course, that all the openings $\mathbf{s}_{\mathbf{b}}$ are short for $\mathbf{b} \in \{0,1\}^{\leq h}$

Merkle-PRISIS II

How to check an opening

- Each layer has its own $\mathsf{crs}_j := (\mathbf{A}_j, w_j, \mathbf{T}_j)$ for $j \in [h := \log d]$
- Check that all local openings are correct. I.e. check that, for $\mathbf{b} \in \{0,1\}^h$:

$$\sum_{j \in [h]} w_j^{b_j} \mathbf{A}_j \mathbf{s}_{\mathbf{b}:j} + f_{\mathbf{b}} \cdot \mathbf{e} = \mathbf{t}$$

- And, of course, that all the openings $\mathbf{s}_{\mathbf{b}}$ are short for $\mathbf{b} \in \{0,1\}^{\leq h}$
- **Binding**: subtract two verification equation:

Merkle-PRISIS II

How to check an opening

- Each layer has its own $\mathsf{crs}_j := (\mathbf{A}_j, w_j, \mathbf{T}_j)$ for $j \in [h := \log d]$
- Check that all local openings are correct. I.e. check that, for $\mathbf{b} \in \{0,1\}^h$:

$$\sum_{j \in [h]} w_j^{b_j} \mathbf{A}_j \mathbf{s}_{\mathbf{b}:j} + f_{\mathbf{b}} \cdot \mathbf{e} = \mathbf{t}$$

- And, of course, that all the openings $\mathbf{s}_{\mathbf{b}}$ are short for $\mathbf{b} \in \{0,1\}^{\leq h}$
- **Binding**: subtract two verification equation:

  reduces to h-PRISIS$_\ell$ i.e. **MSIS**!

Merkle-PRISIS III

Pros ✅ and Cons ❌

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Time to commit is **quasi-linear**.

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Time to commit is **quasi-linear**.
- Common reference string is **logarithmic**.

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Time to commit is **quasi-linear**.
- Common reference string is **logarithmic**.
- Binding under **standard** SIS assumption.

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Time to commit is **quasi-linear**.
- Common reference string is **logarithmic**.
- Binding under **standard** SIS assumption.

- **Trusted** setup

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Time to commit is **quasi-linear**.
- Common reference string is **logarithmic**.
- Binding under **standard** SIS assumption.

- **Trusted** setup

Polynomial Commitments from Lattices: Post-Quantum Security, Fast Verification and Transparent Setup

Valerio Cini[1], Giulio Malavolta[2], Ngoc Khanh Nguyen[3], and Hoeteck Wee[1]

[1] NTT Research, Sunnyvale, CA, USA
[2] Bocconi University, Milan, Italy
[3] King's College London, London, UK

Merkle-PRISIS III

Pros ✅ and Cons ❌

- Commitment is **succinct**.
- Supports committing to messages of **arbitrary** size.
- Time to commit is **quasi-linear**.
- Common reference string is **logarithmic**.
- Binding under **standard** SIS assumption.

- **Trusted** setup

Polynomial Commitments from Lattices: Post-Quantum Security, Fast Verification and Transparent Setup

Valerio Cini[1], Giulio Malavolta[2], Ngoc Khanh Nguyen[3], and Hoeteck Wee[1]

[1] NTT Research, Sunnyvale, CA, USA
[2] Bocconi University, Milan, Italy
[3] King's College London, London, UK

Can we do an efficient evaluation protocol?

Evaluation Protocol

FRI Inspired folding + CWSS

Basic Σ-Protocol

Prover		**Verifier**
$f(\mathsf{X}) = f_0(\mathsf{X}^2) + \mathsf{X} f_1(\mathsf{X}^2)$		
$z_i \coloneqq f_i(u^2)$ for $i \in \mathbb{Z}_2$	$\xrightarrow{z_0, z_1, \mathbf{s}_0, \mathbf{s}_1}$	Check: $z_0 + u z_1 =_? z$; Check: $\mathbf{s}_0, \mathbf{s}_1$ short
$g(\mathsf{X}) \coloneqq \alpha_0 f_0(\mathsf{X}) + \alpha_1 f_1(\mathsf{X})$	$\xleftarrow{\alpha_0, \alpha_1}$	$\alpha_0, \alpha_1 \leftarrow \{ X^i : i \in \mathbb{Z} \}$
$\mathbf{z_b} \coloneqq \alpha_0 \mathbf{s}_{\mathbf{b},0} + \alpha_1 \mathbf{s}_{\mathbf{b},1}$ for $\mathbf{b} \in \mathbb{Z}_2^{\leq h-1}$	$\xrightarrow{g, (\mathbf{z_b})_\mathbf{b}}$	$\mathsf{crs}' \coloneqq (\mathbf{A}_{1+t}, w_{1+t}, \mathbf{T}_{1+t})_{t \in [h-1]}$
		$\mathbf{t}' \coloneqq \alpha_0 \cdot (\mathbf{t} - w_1^0 \mathbf{A}_1 \mathbf{s}_0) + \alpha_1 \cdot (\mathbf{t} - w_1^1 \mathbf{A}_1 \mathbf{s}_1)$
		$u' \coloneqq u^2; z' \coloneqq \alpha_0 \cdot z_0 + \alpha_1 \cdot z_1$
		Check: $g(u') = z'$
		Check: $\mathsf{Open}(\mathsf{crs}', \mathbf{t}', g, (\mathbf{z_b})_\mathbf{b}) = 1$

Are we done?

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.
- Commitment is **succinct**, verifier also **succinct**.

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.
- Commitment is **succinct**, verifier also **succinct**.
- **Problem** 🤔: Knowledge soundness error is $1/\text{poly}(\lambda)$.

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.
- Commitment is **succinct**, verifier also **succinct**.
- **Problem** 🤔: Knowledge soundness error is $1/\mathsf{poly}(\lambda)$.
- Can be made negligible by parallel repetition, but then **no Fiat-Shamir**!

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.
- Commitment is **succinct**, verifier also **succinct**.
- **Problem** 🤔: Knowledge soundness error is $1/\mathsf{poly}(\lambda)$.
- Can be made negligible by parallel repetition, but then **no Fiat-Shamir**!
- Change the challenge space?

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.
- Commitment is **succinct**, verifier also **succinct**.
- **Problem** 🤔: Knowledge soundness error is $1/\text{poly}(\lambda)$.
- Can be made negligible by parallel repetition, but then **no Fiat-Shamir**!
- Change the challenge space?
    - Non-subtractive challenge space => Blowup in extraction, cannot do more than $\log \log d$ recursions => only **quasi-polylogarithmic** sizes.

Are we done?

- Apply protocol recursively $\log d$ times and send final opening $O(1)$.
- Knowledge soundness follows from **coordinate-wise special soundness**.
- Commitment is **succinct**, verifier also **succinct**.
- **Problem** 🤔: Knowledge soundness error is $1/\text{poly}(\lambda)$.
- Can be made negligible by parallel repetition, but then **no Fiat-Shamir**!
- Change the challenge space?
  - Non-subtractive challenge space => Blowup in extraction, cannot do more than $\log \log d$ recursions => only **quasi-polylogarithmic** sizes.
  - Subtractive challenge space => Challenge space of size at most $\text{poly}(\lambda)$ [AL21]

Claim bundling I

Let’s prove something harder!

Claim bundling I

Let's prove something harder!

- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$

Claim bundling I

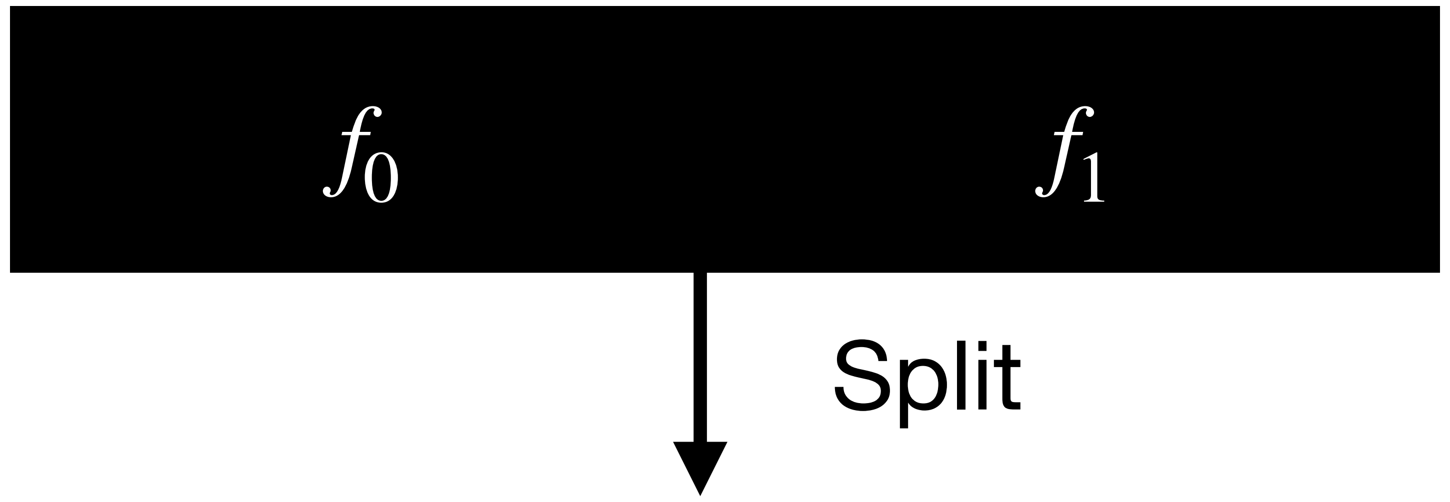
Let's prove something harder!

- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.

Claim bundling I

Let's prove something harder!

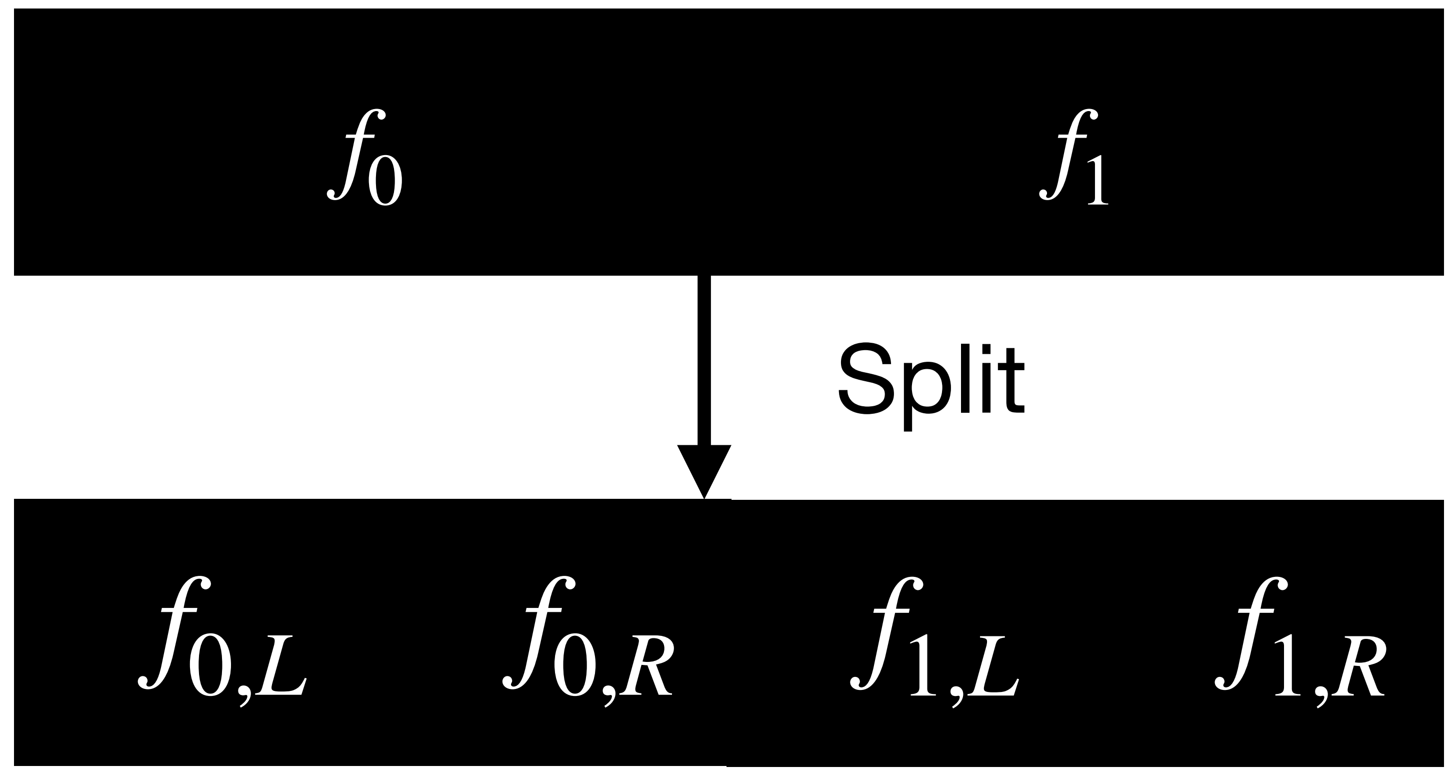
- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.



Claim bundling I

Let's prove something harder!

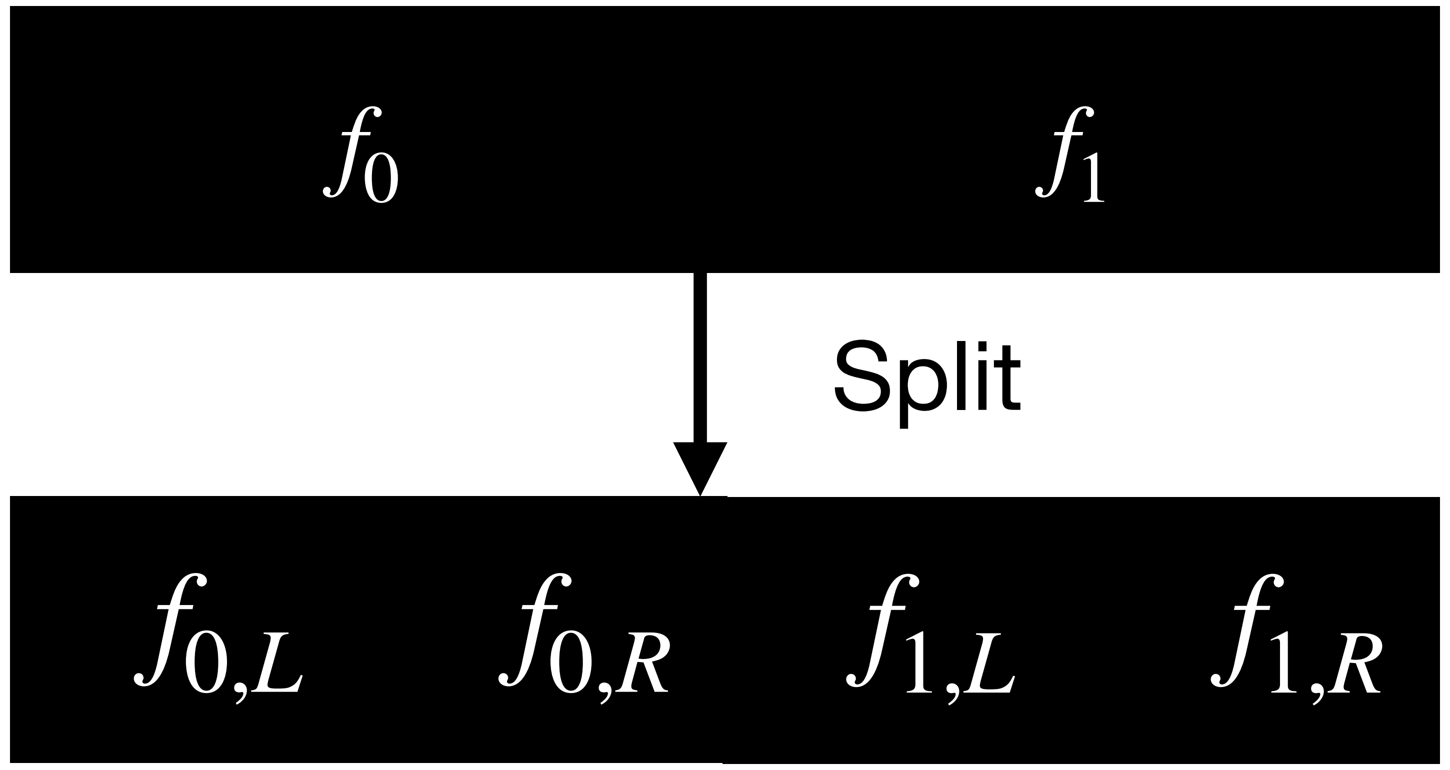
- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.



Claim bundling I

Let's prove something harder!

- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.



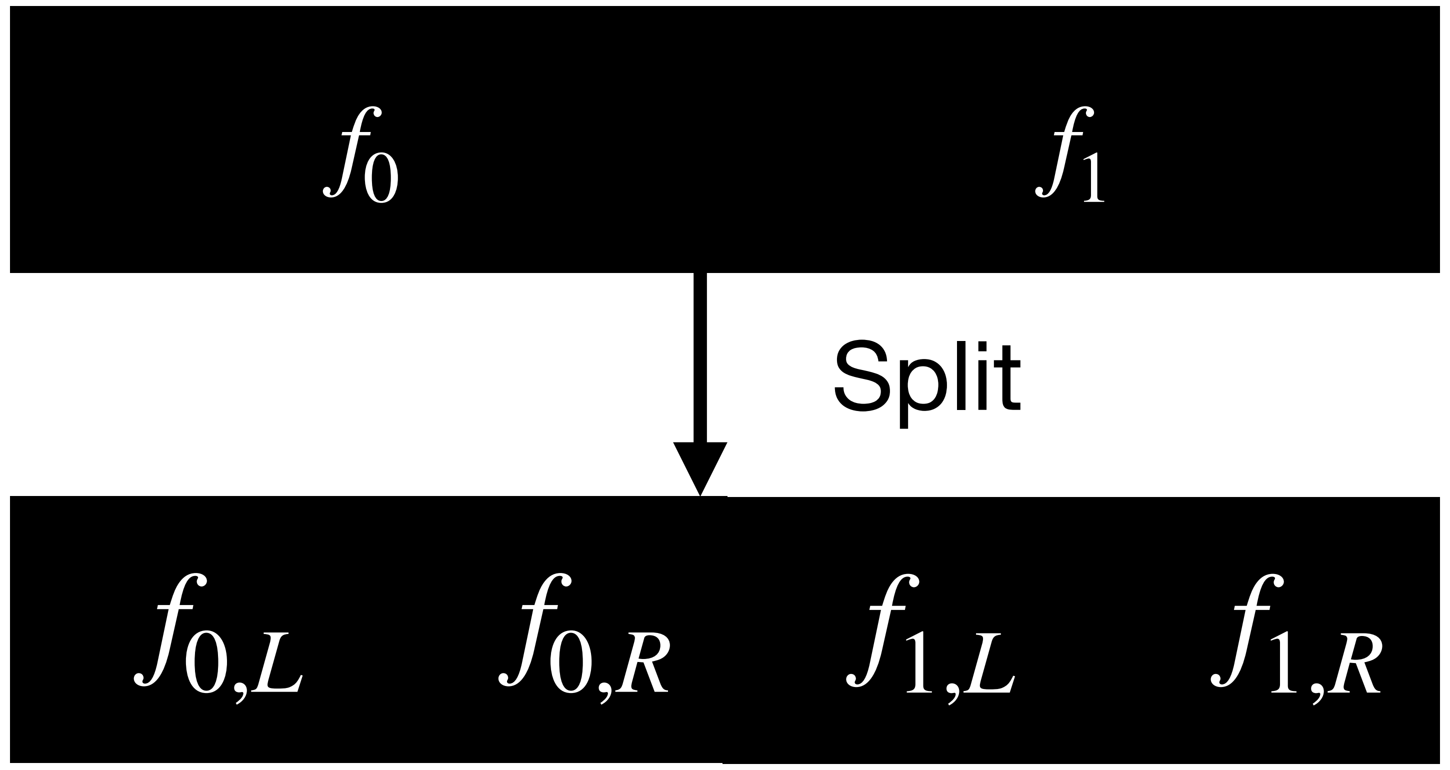
Randomness is now:

$$\begin{bmatrix} \alpha_{0,L,0}, \alpha_{0,R,0}, \alpha_{1,L,0}, \alpha_{1,R,0} \\ \alpha_{0,L,1}, \alpha_{0,R,1}, \alpha_{1,L,1}, \alpha_{1,R,1} \end{bmatrix} \in (\mathscr{C}^r)^{2r}$$

Claim bundling I

Let's prove something harder!

- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.



Randomness is now:

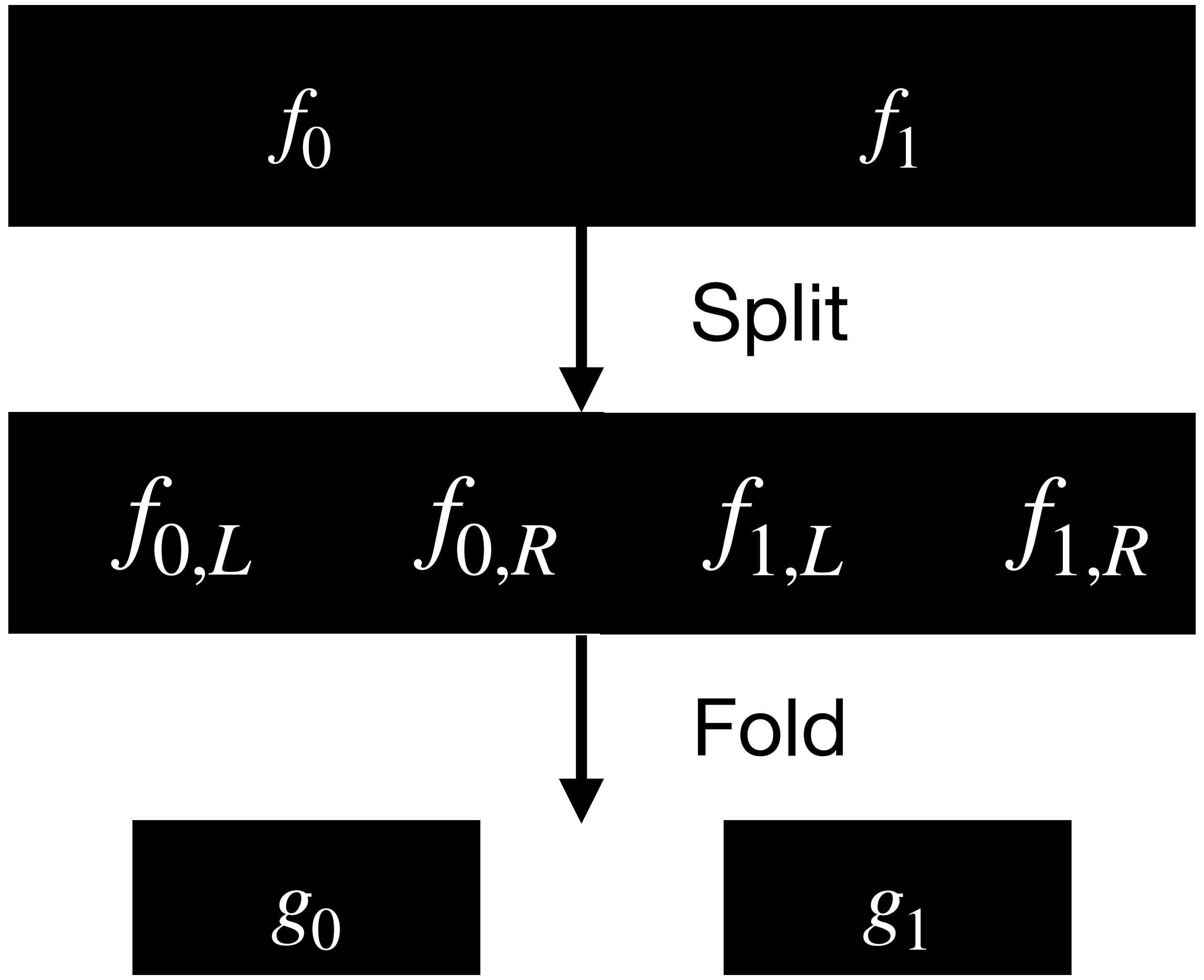
$$\begin{bmatrix} \alpha_{0,L,0}, \alpha_{0,R,0}, \alpha_{1,L,0}, \alpha_{1,R,0} \\ \alpha_{0,L,1}, \alpha_{0,R,1}, \alpha_{1,L,1}, \alpha_{1,R,1} \end{bmatrix} \in (\mathscr{C}^r)^{2r}$$

$\alpha_{\iota,i,\kappa}$ folds $f_{\iota,i}$ into g_κ

Claim bundling I

Let's prove something harder!

- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.



Randomness is now:

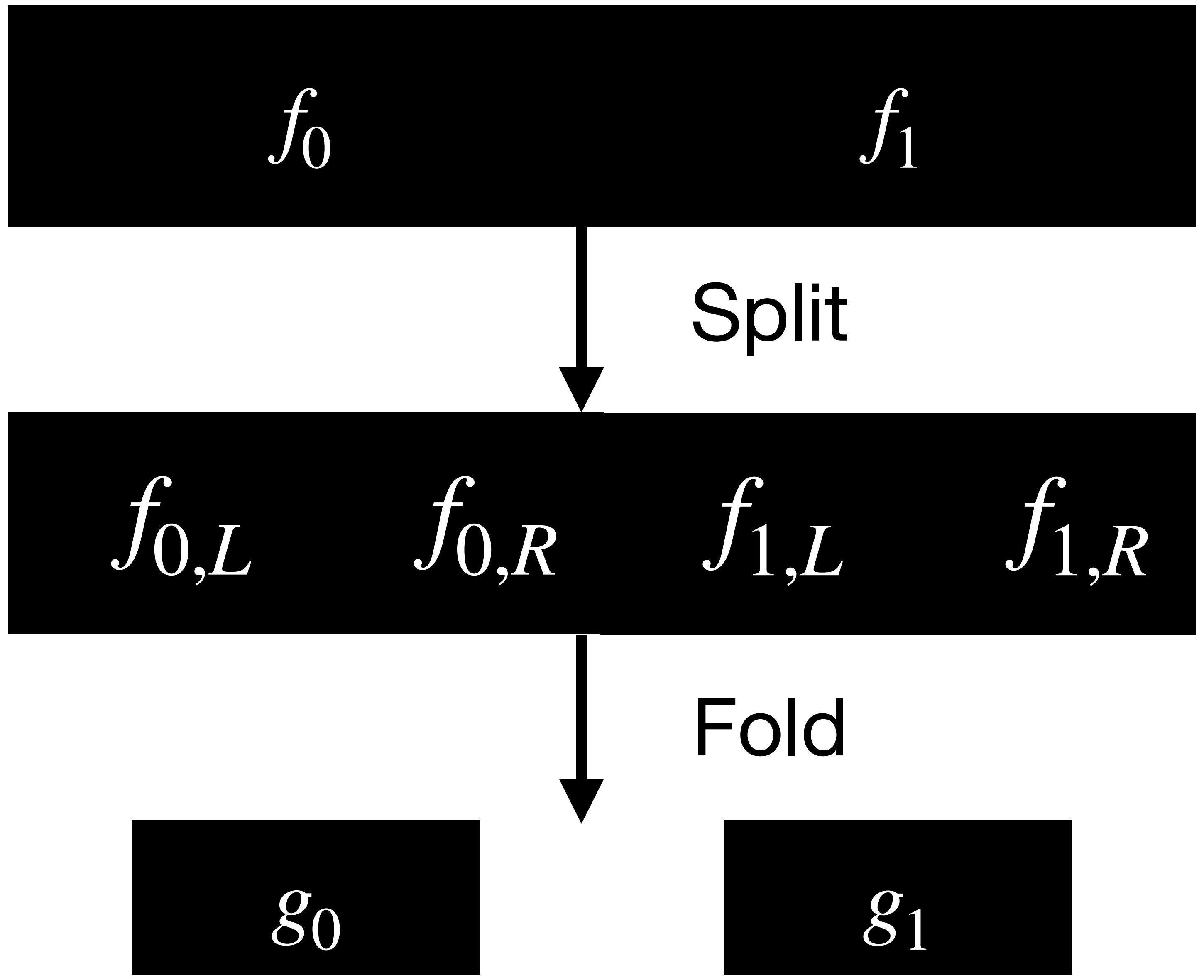
$$\begin{bmatrix} \alpha_{0,L,0}, \alpha_{0,R,0}, \alpha_{1,L,0}, \alpha_{1,R,0} \\ \alpha_{0,L,1}, \alpha_{0,R,1}, \alpha_{1,L,1}, \alpha_{1,R,1} \end{bmatrix} \in (\mathscr{C}^r)^{2r}$$

$\alpha_{\iota,i,\kappa}$ folds $f_{\iota,i}$ into g_κ

Claim bundling I

Let's prove something harder!

- Instead of proving $f(u) = v$, show that, for $\iota \in [r], f_\iota(u) = v_\iota$
- As in [FMN23], our protocol can be easily extended to deal with this.



Randomness is now:

$$\begin{bmatrix} \alpha_{0,L,0}, \alpha_{0,R,0}, \alpha_{1,L,0}, \alpha_{1,R,0} \\ \alpha_{0,L,1}, \alpha_{0,R,1}, \alpha_{1,L,1}, \alpha_{1,R,1} \end{bmatrix} \in (\mathscr{C}^r)^{2r}$$

$\alpha_{\iota,i,\kappa}$ folds $f_{\iota,i}$ into g_κ

Folded polynomial:

$$g_0 := \alpha_{0,L,0} f_{0,L} + \alpha_{0,R,0} f_{0,R} + \alpha_{1,L,0} f_{1,L} + \alpha_{1,R,0} f_{1,R}$$

$$g_1 := \alpha_{0,L,1} f_{0,L} + \alpha_{0,R,1} f_{0,R} + \alpha_{1,L,1} f_{1,L} + \alpha_{1,R,1} f_{1,R}$$

Claim bundling II

What did we gain?

Claim bundling II

What did we gain?

- Now, protocol is $2r$ coordinate-wise special sound with challenge space of size roughly poly$(\lambda)^r$

Claim bundling II

What did we gain?

- Now, protocol is $2r$ coordinate-wise special sound with challenge space of size roughly $\text{poly}(\lambda)^r$
- Setting r to be $\text{polylog}(\lambda)$, we achieve **negligible knowledge error**!

Claim bundling II

What did we gain?

- Now, protocol is $2r$ coordinate-wise special sound with challenge space of size roughly $\mathsf{poly}(\lambda)^r$
- Setting r to be $\mathsf{polylog}(\lambda)$, we achieve **negligible knowledge error**!
- Our protocol can now be made **non-interactive** using FS.

Claim bundling II

What did we gain?

- Now, protocol is $2r$ coordinate-wise special sound with challenge space of size roughly $\text{poly}(\lambda)^r$
- Setting r to be $\text{polylog}(\lambda)$, we achieve **negligible knowledge error**!
- Our protocol can now be made **non-interactive** using FS.
- To prove a single claim $f(u) = v$, simply set $f_1, \ldots, f_r = f$ and $v_1, \ldots, v_r = v$.

Conclusion

👋 - SLAP

A non-interactive lattice-based polynomial commitment with succinct proofs and verification time, from standard lattice assumptions.

There is more!

What we did not talk about

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness
- Honest-verifier zero knowledge for our PCS

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness
- Honest-verifier zero knowledge for our PCS
- Transforming PCS for $\mathscr{R}_q$ in those for $\mathbb{Z}_q$ (efficient packing)

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness
- Honest-verifier zero knowledge for our PCS
- Transforming PCS for $\mathscr{R}_q$ in those for $\mathbb{Z}_q$ (efficient packing)
- Twin-k-M-ISIS is no easier than $2k$-M-ISIS

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness
- Honest-verifier zero knowledge for our PCS
- Transforming PCS for $\mathscr{R}_q$ in those for $\mathbb{Z}_q$ (efficient packing)
- Twin-k-M-ISIS is no easier than $2k$-M-ISIS
- Setting concrete parameters

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness
- Honest-verifier zero knowledge for our PCS
- Transforming PCS for $\mathcal{R}_q$ in those for $\mathbb{Z}_q$ (efficient packing)
- Twin-k-M-ISIS is no easier than $2k$-M-ISIS
- Setting concrete parameters
- Reductions… all the reductions

There is more!

What we did not talk about

- Succinct evaluation protocol for Merkle-PRISIS
- Folding more at each step
- Coordinate-wise special soundness
- Honest-verifier zero knowledge for our PCS
- Transforming PCS for $\mathcal{R}_q$ in those for $\mathbb{Z}_q$ (efficient packing)
- Twin-k-M-ISIS is no easier than $2k$-M-ISIS
- Setting concrete parameters
- Reductions… all the reductions

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

Martin R. Albrecht
martin.albrecht@{kcl.ac.uk,sandboxaq.com}
King's College London and SandboxAQ

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Oleksandra Lapiha
sasha.lapiha.2021@live.rhul.ac.uk
Royal Holloway, University of London

Ngoc Khanh Nguyen
khanh.nguyen@epfl.ch
EPFL

ia.cr/2023/1469

Details here!

SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions

September 2023 · Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, Ngoc Khanh Nguyen · EUROCRYPT 2024 - ePrint: 2023/1469

This blog-post is a short introduction to our new work: "SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions". This is joint work with Martin Albrecht, Oleksandra Lapiha and Ngoc Khanh Nguyen, and the full version is available on ePrint . Here are also some slides that might be helpful.

gfenzi.io/papers/slap

Open Questions 🔬

Open Questions 🔬

- Can we get succinct lattice-based polynomial commitments under **100KB**?

Open Questions 🔬

- Can we get succinct lattice-based polynomial commitments under **100KB**?
- Can we get $\mathrm{negl}(\lambda)$ knowledge error in one-shot (no claim bundling)?

Open Questions 🔬

- Can we get succinct lattice-based polynomial commitments under **100KB**?
- Can we get $\mathrm{negl}(\lambda)$ knowledge error in one-shot (no claim bundling)?
- Is PRISIS_ℓ with $\ell > 2$ still secure?

Open Questions

- Can we get succinct lattice-based polynomial commitments under **100KB**?
- Can we get $\mathrm{negl}(\lambda)$ knowledge error in one-shot (no claim bundling)?
- Is PRISIS_{ℓ} with $\ell > 2$ still secure?

Thank you!

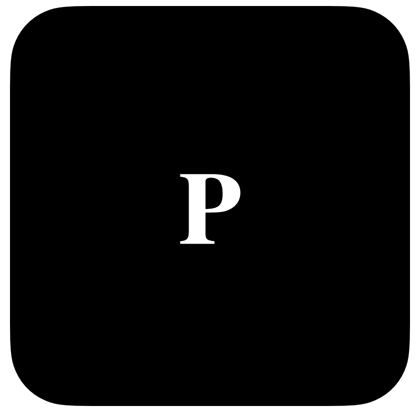
Extra slides

Evaluation Protocol I

Strategy

Evaluation Protocol I

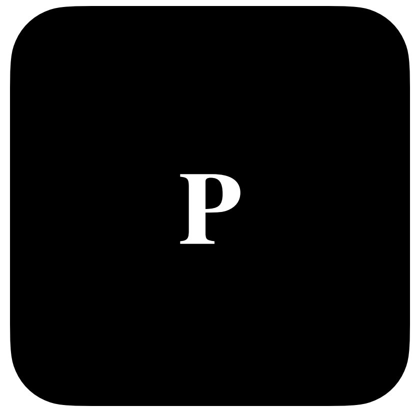
Strategy



Evaluation Protocol I

Strategy

Prover knows:

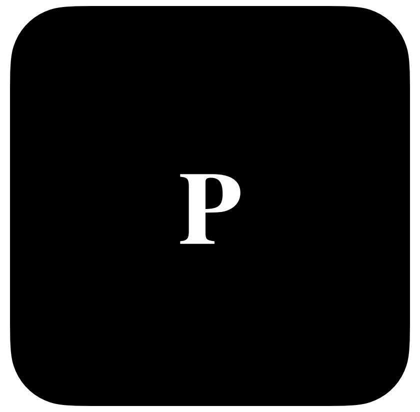


Evaluation Protocol I

Strategy

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$



Evaluation Protocol I

Strategy

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

P

V

Evaluation Protocol I

Strategy

Verifier knows:

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

P

V

Evaluation Protocol I

Strategy

Verifier knows:

- Common reference string crs

Prover knows:

- Polynomial $f \in \mathscr{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

P

V

Evaluation Protocol I

Strategy

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

P

V

Evaluation Protocol I

Strategy

Prover knows:

- Polynomial $f \in \mathscr{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and Open(crs, $\mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$

P

V

Evaluation Protocol I

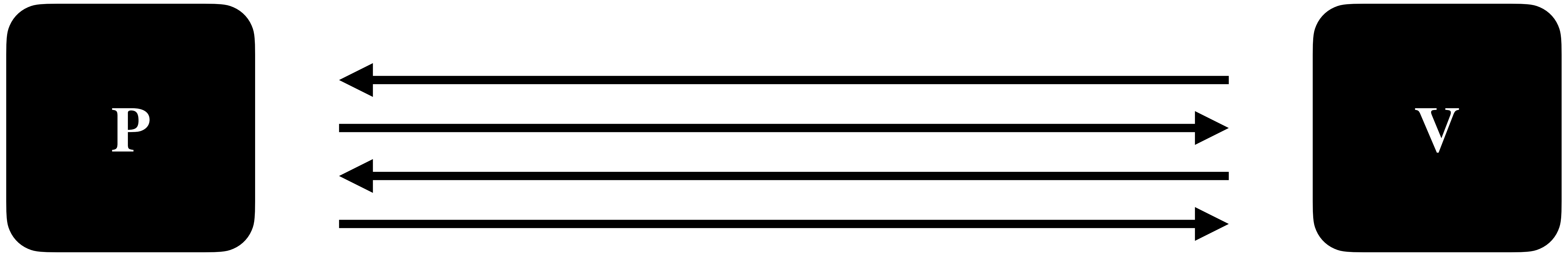
Strategy

Prover knows:

- Polynomial $f \in \mathscr{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and $\mathsf{Open}(\mathsf{crs}, \mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$



Evaluation Protocol I

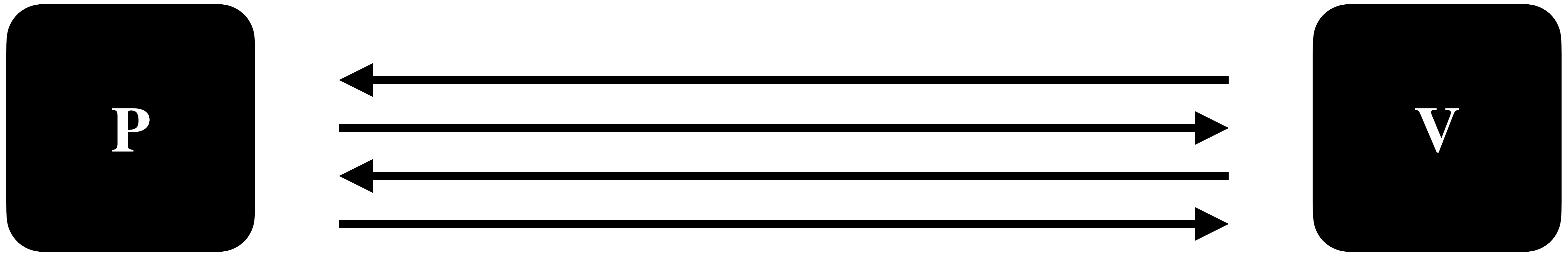
Strategy

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and Open(crs, $\mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$



Prover now knows:

Evaluation Protocol I

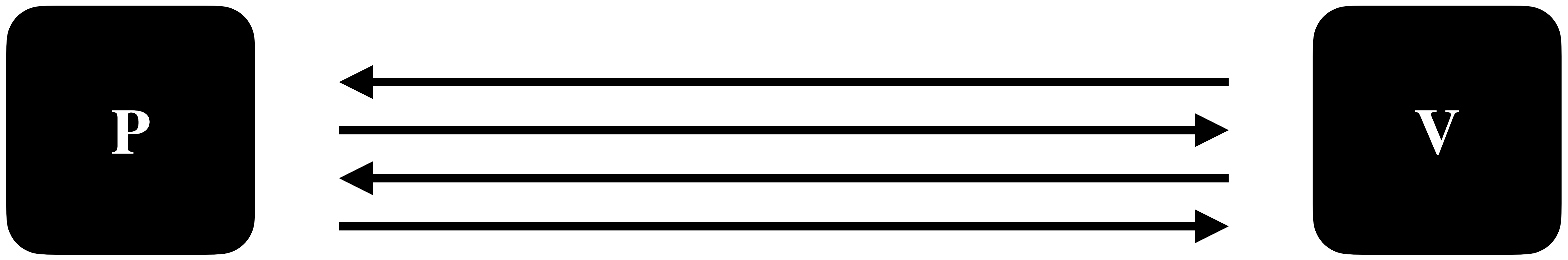
Strategy

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and $\mathsf{Open}(\mathsf{crs}, \mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$



Prover now knows:

- Polynomial $g \in \mathcal{R}_q^{<d/2}[X]$ and openings $(\mathbf{z_b})_\mathbf{b}$

Evaluation Protocol I

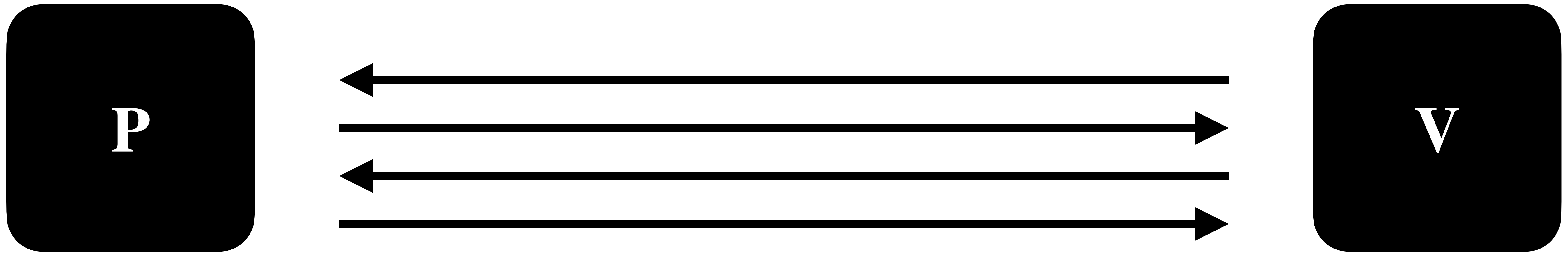
Strategy

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and $\mathsf{Open}(\mathsf{crs}, \mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$

Prover knows:

- Polynomial $f \in \mathscr{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$



Verifier now knows:

Prover now knows:

- Polynomial $g \in \mathscr{R}_q^{<d/2}[X]$ and openings $(\mathbf{z_b})_\mathbf{b}$

Evaluation Protocol I

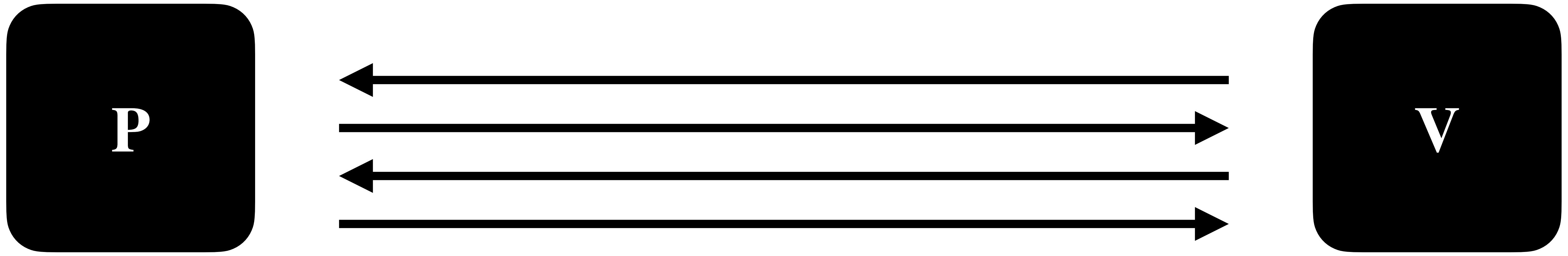
Strategy

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and $\mathsf{Open}(\mathsf{crs}, \mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$



Verifier now knows:

- Common reference string crs'

Prover now knows:

- Polynomial $g \in \mathcal{R}_q^{<d/2}[X]$ and openings $(\mathbf{z_b})_\mathbf{b}$

Evaluation Protocol I

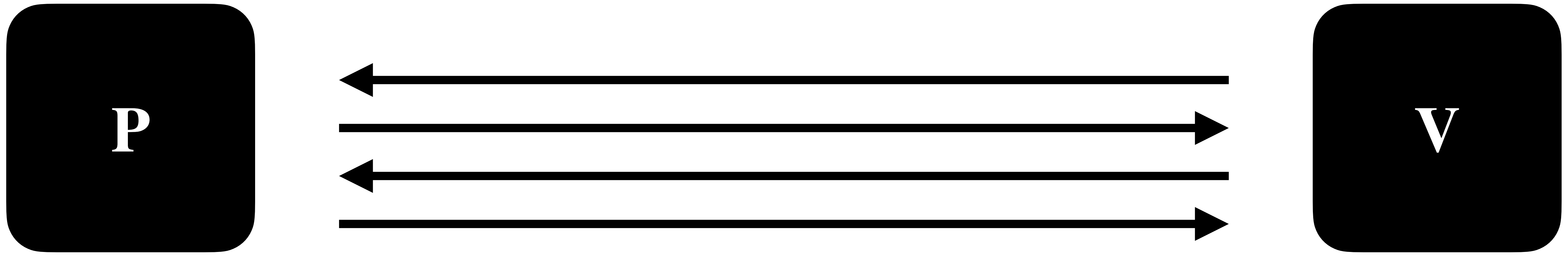
Strategy

Prover knows:

- Polynomial $f \in \mathscr{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and $\mathsf{Open}(\mathsf{crs}, \mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$



Prover now knows:

- Polynomial $g \in \mathscr{R}_q^{<d/2}[X]$ and openings $(\mathbf{z_b})_\mathbf{b}$

Verifier now knows:

- Common reference string crs'
- Commitment $\mathbf{t}'$

Evaluation Protocol I

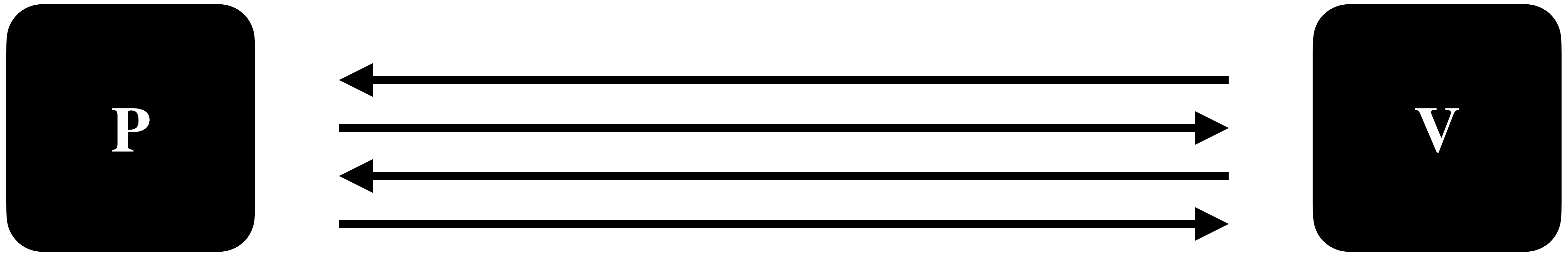
Strategy

Prover knows:

- Polynomial $f \in \mathcal{R}_q^{<d}[X]$ and openings $(\mathbf{s_b})_\mathbf{b}$

Verifier knows:

- Common reference string crs
- Commitment $\mathbf{t}$
- Claim: $f(u) = v$ and Open(crs, $\mathbf{t}, f, (\mathbf{s_b})_\mathbf{b}) = 1$



Prover now knows:

- Polynomial $g \in \mathcal{R}_q^{<d/2}[X]$ and openings $(\mathbf{z_b})_\mathbf{b}$

Verifier now knows:

- Common reference string crs′
- Commitment $\mathbf{t}'$
- New claim: $g(u') = v'$ and Open(crs′, $\mathbf{t}', g, (\mathbf{z_b})_\mathbf{b}) = 1$

Evaluation Protocol II

Split and fold (Evaluations)

Evaluation Protocol II

Split and fold (Evaluations)

$$f \in \mathscr{R}_q^{<d}[X]$$

Evaluation Protocol II

Split and fold (Evaluations)

$$f \in \mathscr{R}_q^{<d}[X]$$

Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

Evaluation Protocol II

Split and fold (Evaluations)

Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

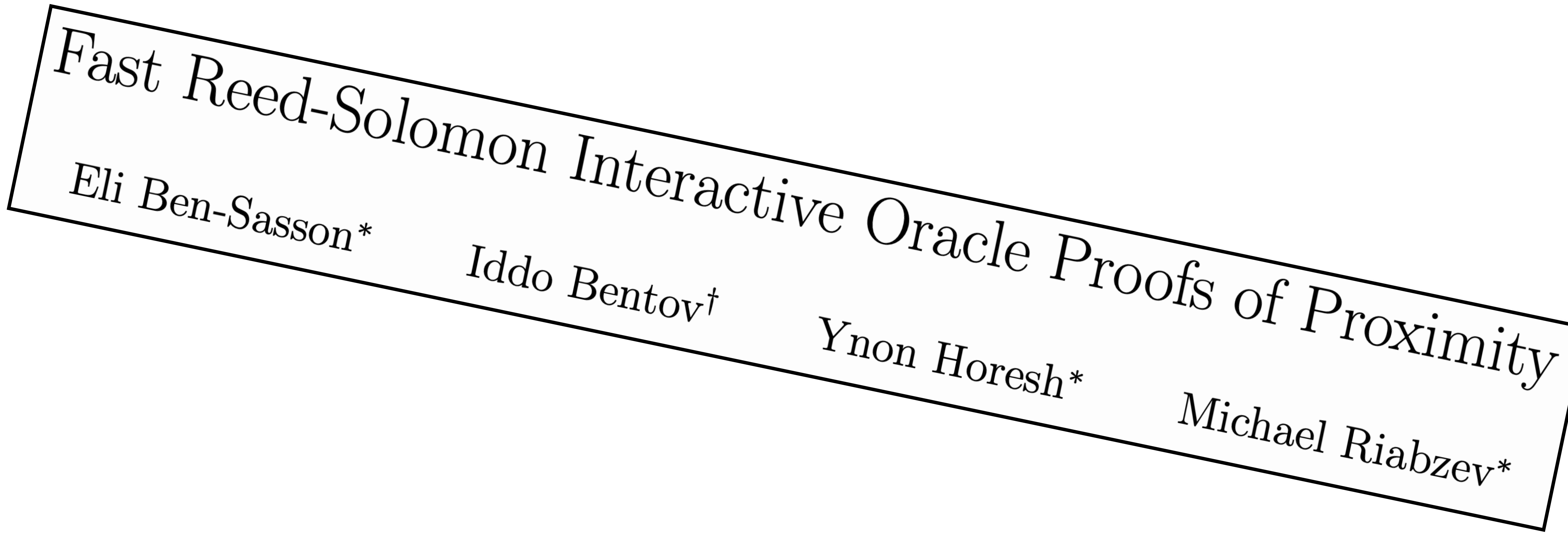
$f \in \mathscr{R}_q^{<d}[X]$

$\downarrow$ Split

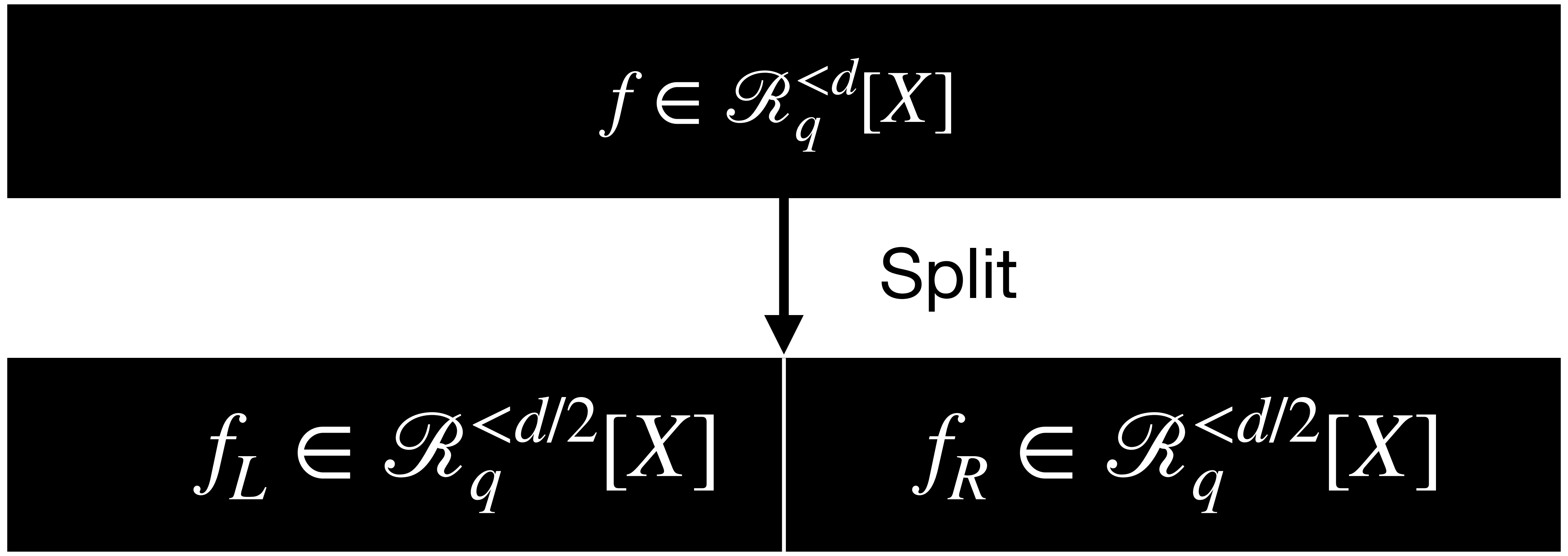
$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

Evaluation Protocol II

Split and fold (Evaluations)


Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*



$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

Evaluation Protocol II

Split and fold (Evaluations)

Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

$f \in \mathscr{R}_q^{<d}[X]$

Split

$f_L \in \mathscr{R}_q^{<d/2}[X]$ $f_R \in \mathscr{R}_q^{<d/2}[X]$

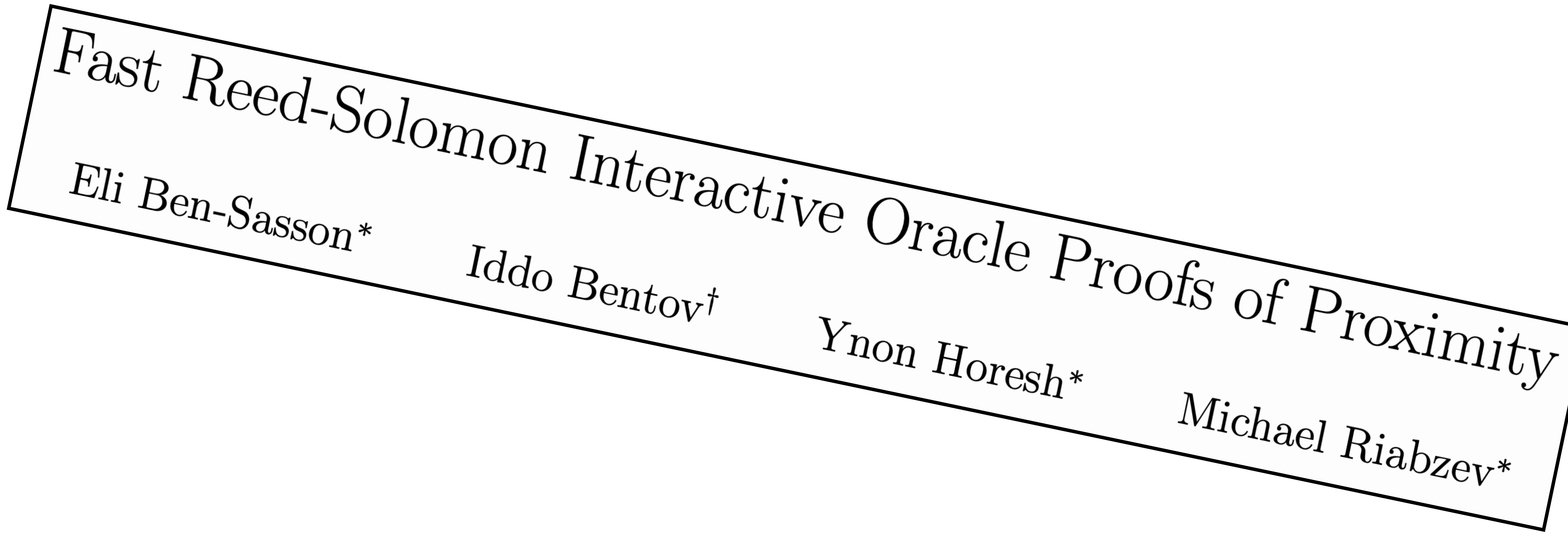
$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

α_0, α_1

V

Evaluation Protocol II

Split and fold (Evaluations)


Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

$f \in \mathcal{R}_q^{<d}[X]$

Split

$f_L \in \mathcal{R}_q^{<d/2}[X]$ | $f_R \in \mathcal{R}_q^{<d/2}[X]$

Fold

$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

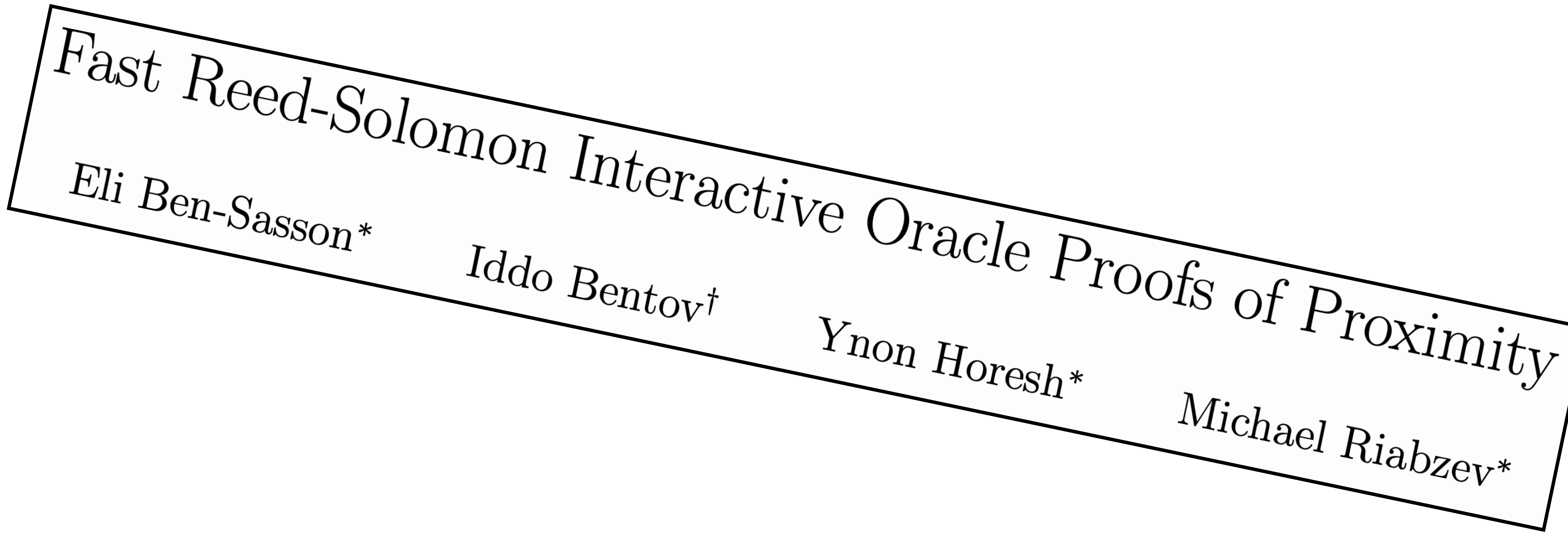
α_0, α_1

V

$$g(X) = \alpha_0 f_L(X) + \alpha_1 f_R(X)$$

Evaluation Protocol II

Split and fold (Evaluations)


Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

$f \in \mathcal{R}_q^{<d}[X]$

Split

$f_L \in \mathcal{R}_q^{<d/2}[X]$ $f_R \in \mathcal{R}_q^{<d/2}[X]$

Fold

$g \in \mathcal{R}_q^{<d/2}[X]$

$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

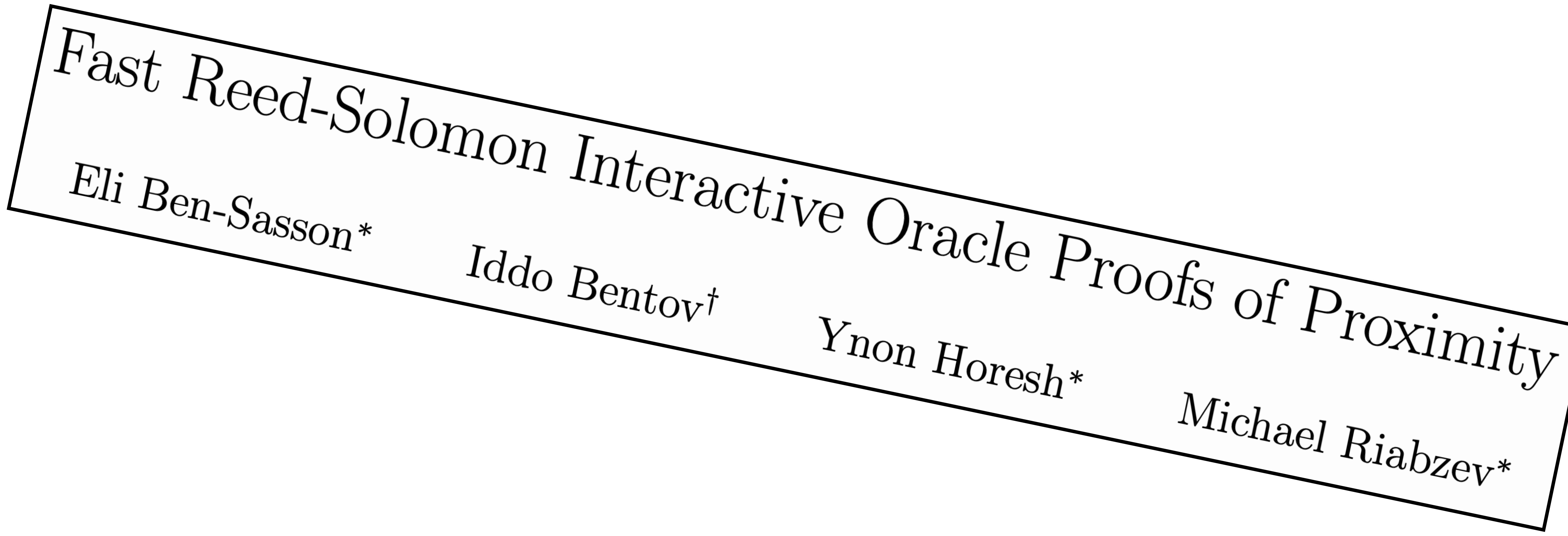
α_0, α_1

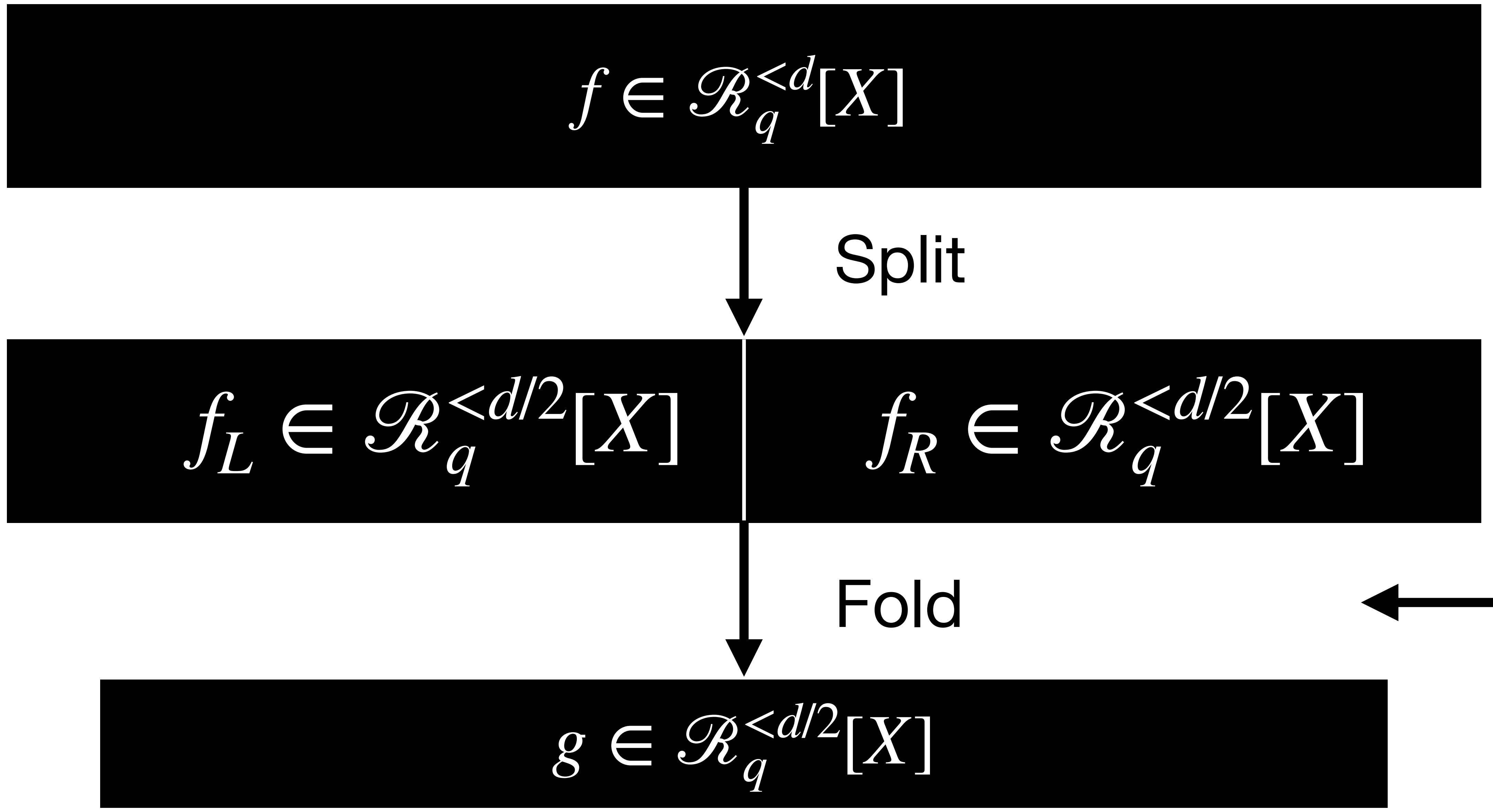
V

$$g(X) = \alpha_0 f_L(X) + \alpha_1 f_R(X)$$

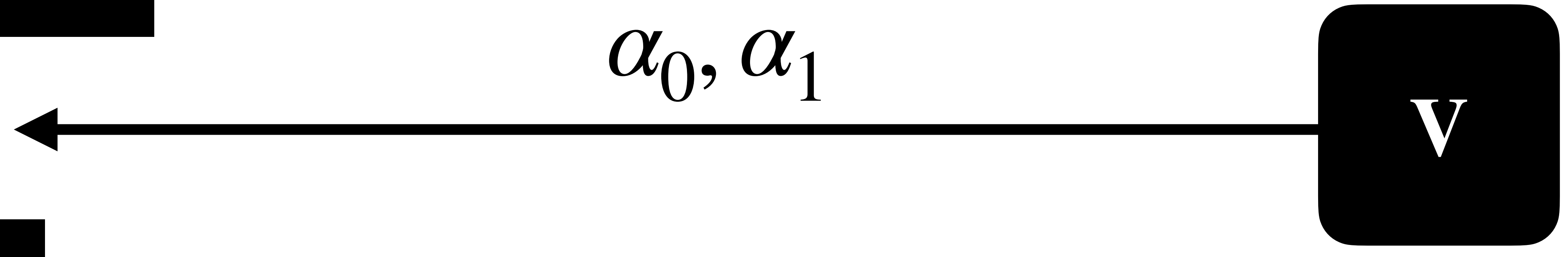
Evaluation Protocol II

Split and fold (Evaluations)





$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

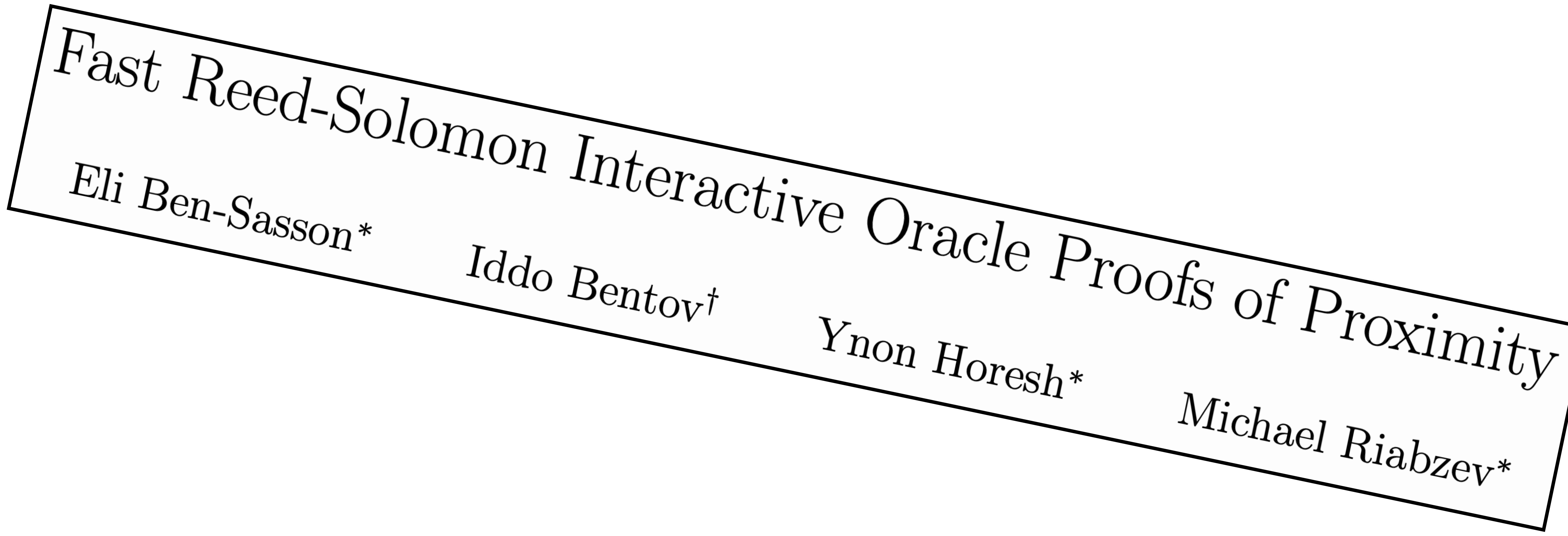


$$g(X) = \alpha_0 f_L(X) + \alpha_1 f_R(X)$$

Ask prover to send $z_0 = f_L(u^2), z_1 = f_R(u^2)$. Check $z_0 + uz_1 = z$

Evaluation Protocol II

Split and fold (Evaluations)


Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

$f \in \mathcal{R}_q^{<d}[X]$

Split

$$f(X) = f_L(X^2) + X \cdot f_R(X^2)$$

$f_L \in \mathcal{R}_q^{<d/2}[X]$ | $f_R \in \mathcal{R}_q^{<d/2}[X]$

Fold

α_0, α_1

V

$g \in \mathcal{R}_q^{<d/2}[X]$

$$g(X) = \alpha_0 f_L(X) + \alpha_1 f_R(X)$$

Ask prover to send $z_0 = f_L(u^2), z_1 = f_R(u^2)$. Check $z_0 + uz_1 = z$

If $f(u) = v$, then $g(u^2) = \alpha_0 z_0 + \alpha_1 z_1$.

Evaluation Protocol III

Split and fold (Openings)

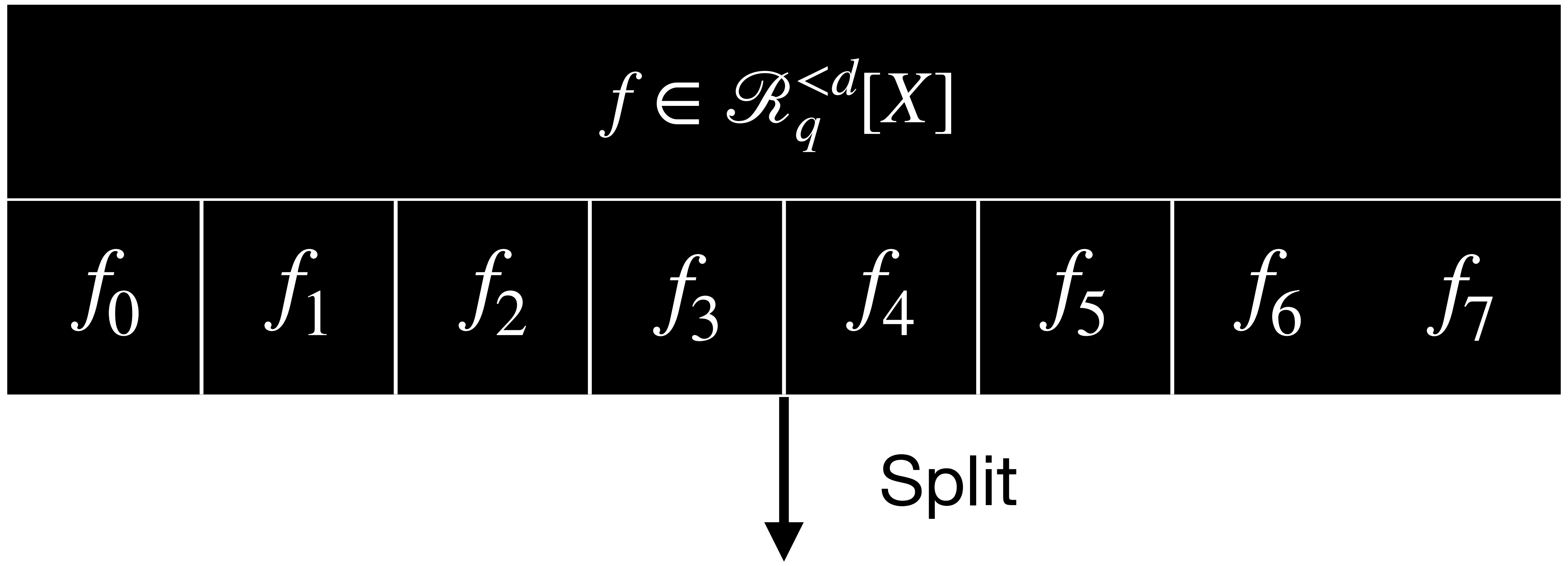
Evaluation Protocol III

Split and fold (Openings)

$f \in \mathscr{R}_q^{<d}[X]$							
f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7

Evaluation Protocol III

Split and fold (Openings)



Evaluation Protocol III

Split and fold (Openings)

$f \in \mathcal{R}_q^{<d}[X]$							
f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7

$\downarrow$ Split

$f_L \in \mathcal{R}_q^{<d/2}[X]$				$f_R \in \mathcal{R}_q^{<d/2}[X]$			
f_0	f_2	f_4	f_6	f_1	f_3	f_5	f_7

Evaluation Protocol III

Split and fold (Openings)

$f \in \mathcal{R}_q^{<d}[X]$							
f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7

Split

$f_L \in \mathcal{R}_q^{<d/2}[X]$				$f_R \in \mathcal{R}_q^{<d/2}[X]$			
f_0	f_2	f_4	f_6	f_1	f_3	f_5	f_7

Fold

Evaluation Protocol III

Split and fold (Openings)

$f \in \mathscr{R}_q^{<d}[X]$

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7

$\downarrow$ Split

$f_L \in \mathscr{R}_q^{<d/2}[X]$				$f_R \in \mathscr{R}_q^{<d/2}[X]$			
f_0	f_2	f_4	f_6	f_1	f_3	f_5	f_7

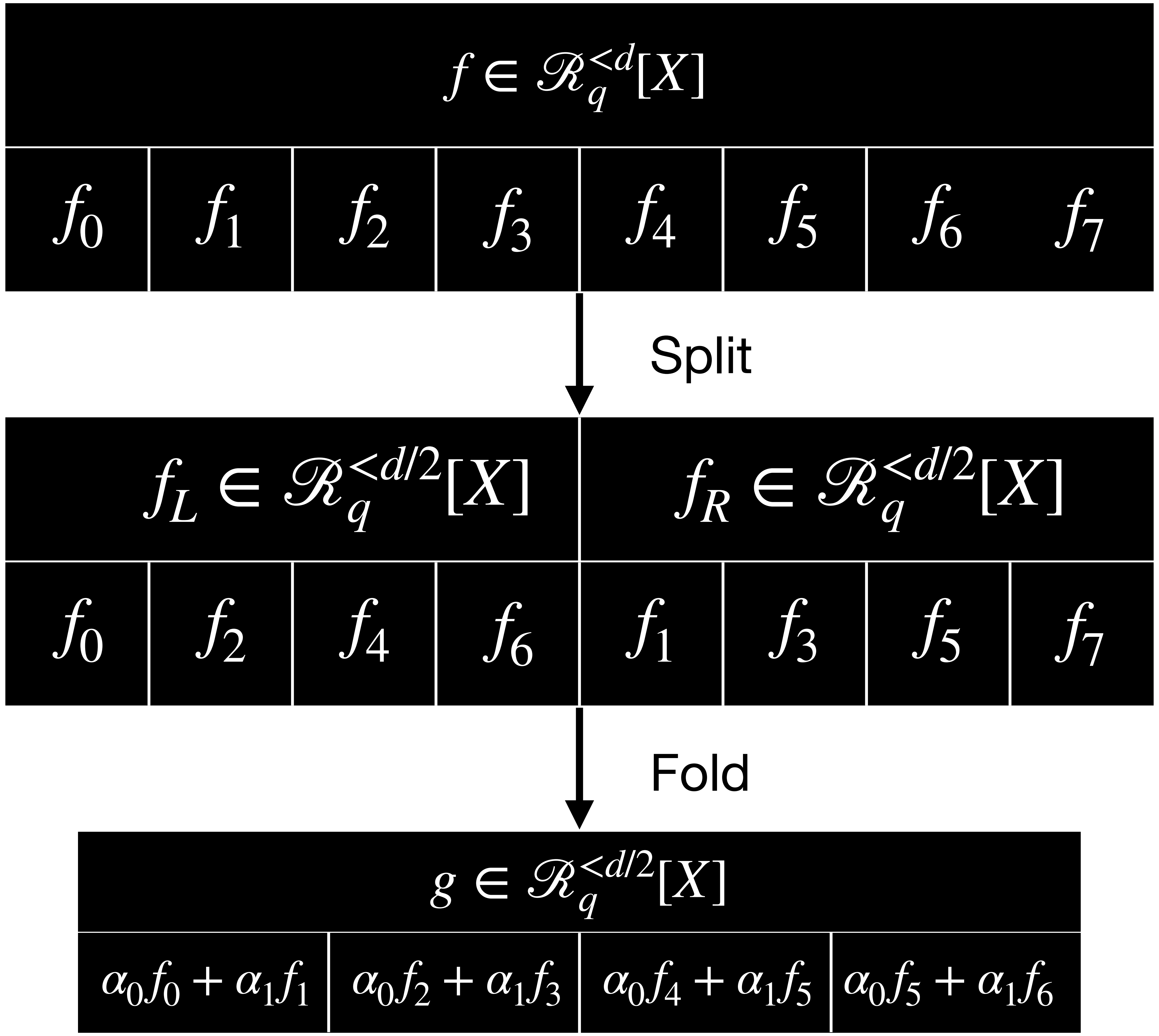
$\downarrow$ Fold

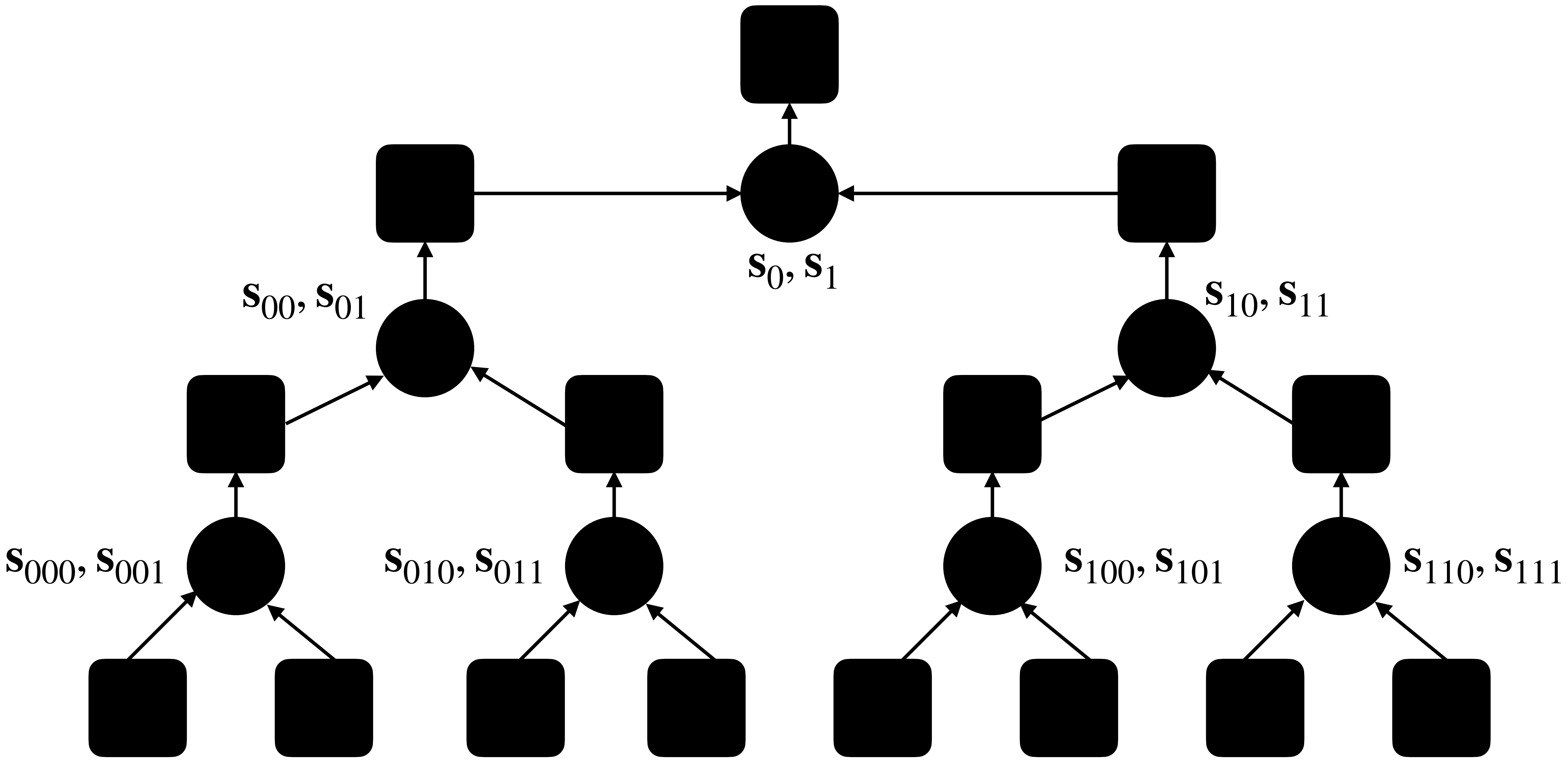
$g \in \mathscr{R}_q^{<d/2}[X]$

$\alpha_0 f_0 + \alpha_1 f_1$	$\alpha_0 f_2 + \alpha_1 f_3$	$\alpha_0 f_4 + \alpha_1 f_5$	$\alpha_0 f_5 + \alpha_1 f_6$

Evaluation Protocol III

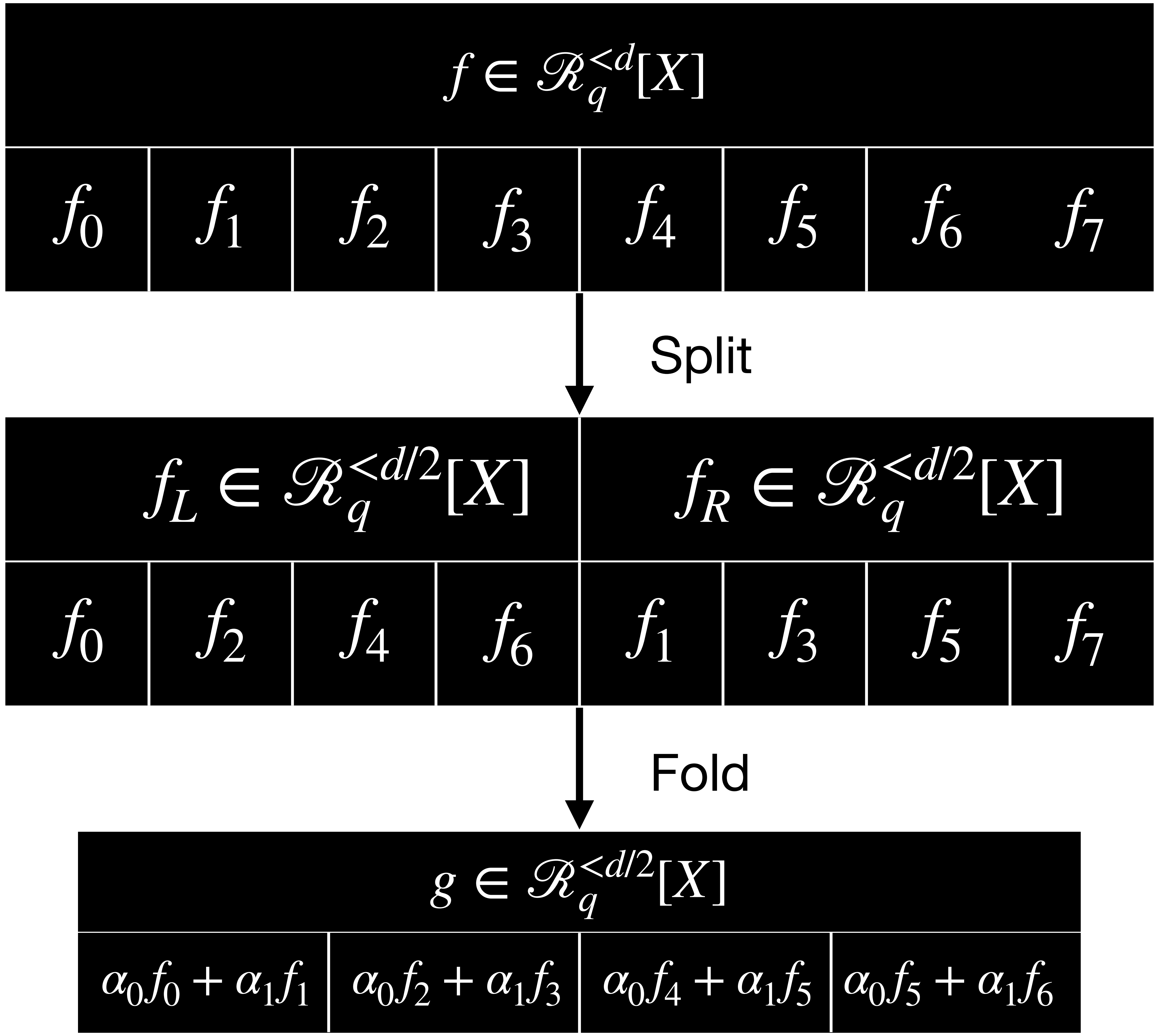
Split and fold (Openings)

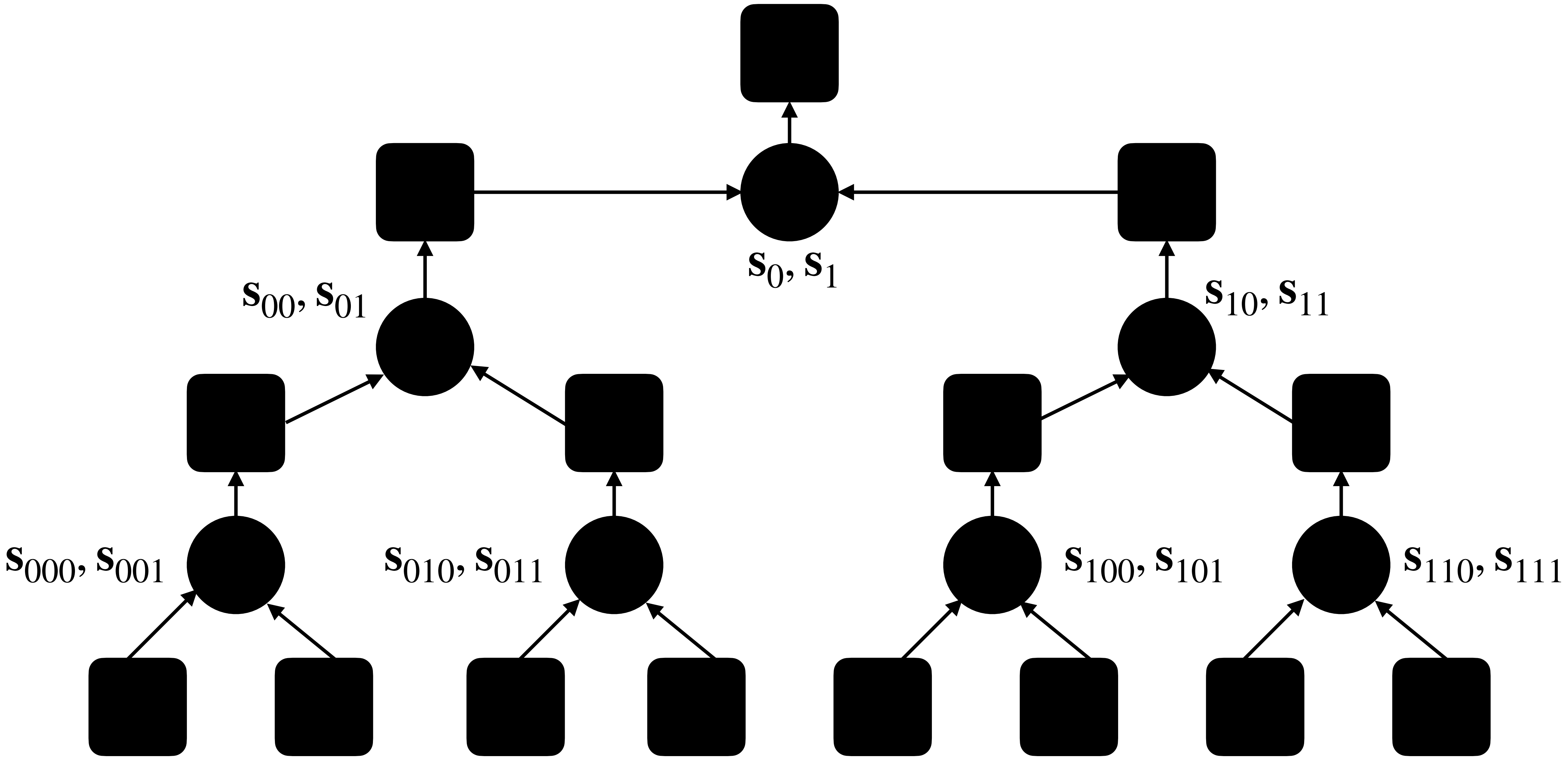


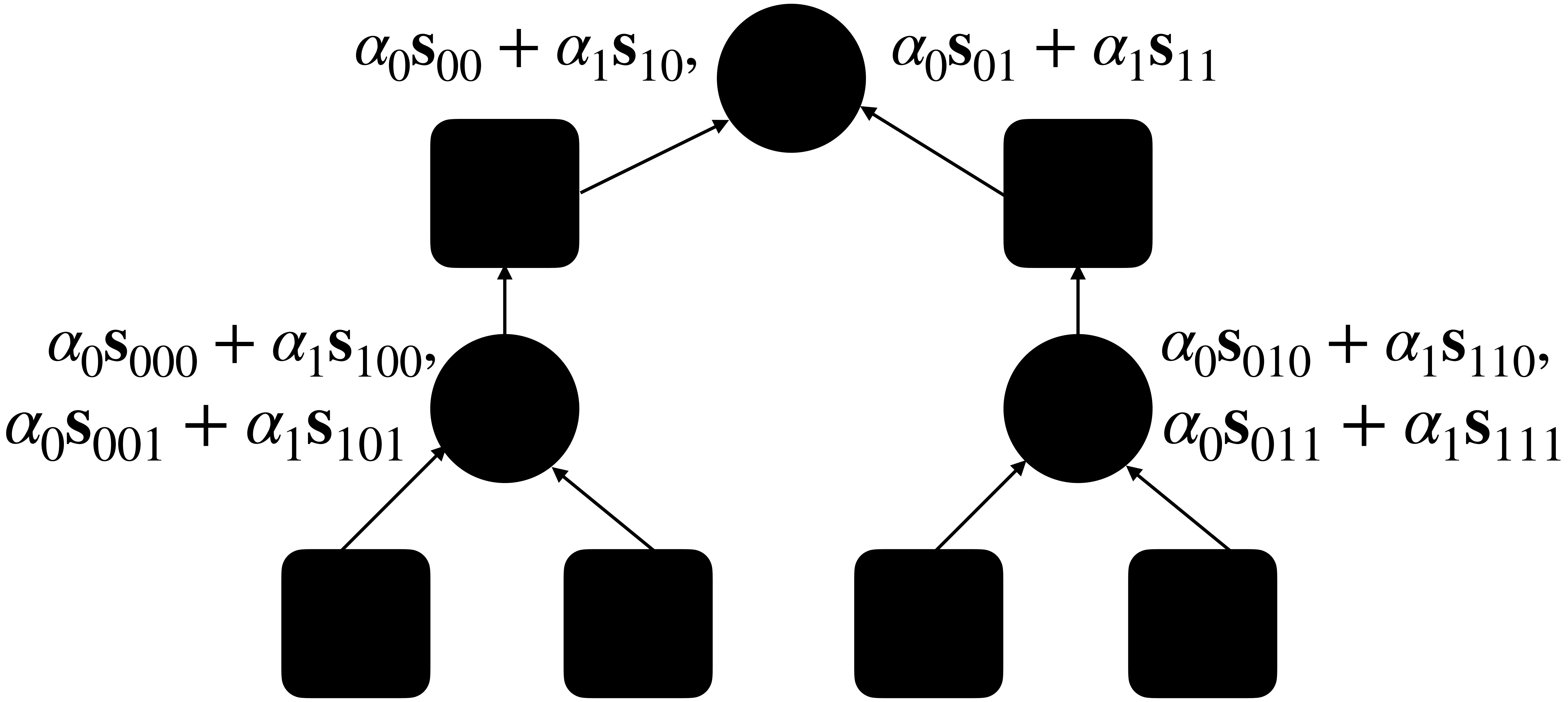


Evaluation Protocol III

Split and fold (Openings)







Evaluation Protocol IV

Split and fold (Commitment)

Evaluation Protocol IV

Split and fold (Commitment)

- We have shown how to compute new evaluations and openings

Evaluation Protocol IV

Split and fold (Commitment)

- We have shown how to compute new evaluations and openings
- If α_i are short, the new openings also are.

Evaluation Protocol IV

Split and fold (Commitment)

- We have shown how to compute new evaluations and openings
- If α_i are short, the new openings also are.
- How does the verifier compute new commitment? With some magic:

Evaluation Protocol IV

Split and fold (Commitment)

- We have shown how to compute new evaluations and openings
- If α_i are short, the new openings also are.
- How does the verifier compute new commitment? With some magic:

$$\sum_{j \in [h-1]} w_{1+j}^{b_{1+j}} \mathbf{A}_{1+j} \mathbf{s}_{\mathbf{b}:1+j} + g_{\mathbf{b}} \mathbf{e} = \alpha_0 \cdot (\mathbf{t} - w_1^0 \mathbf{A}_1 \mathbf{s}_0) + \alpha_1 \cdot (\mathbf{t} - w_1^1 \mathbf{A}_1 \mathbf{s}_1)$$

Evaluation Protocol IV

Split and fold (Commitment)

- We have shown how to compute new evaluations and openings
- If α_i are short, the new openings also are.
- How does the verifier compute new commitment? With some magic:

$$\sum_{j \in [h-1]} w_{1+j}^{b_{1+j}} \mathbf{A}_{1+j} \mathbf{s}_{\mathbf{b}:1+j} + g_{\mathbf{b}} \mathbf{e} = \alpha_0 \cdot (\mathbf{t} - w_1^0 \mathbf{A}_1 \mathbf{s}_0) + \alpha_1 \cdot (\mathbf{t} - w_1^1 \mathbf{A}_1 \mathbf{s}_1)$$

- Prover reveals $\mathbf{s}_0, \mathbf{s}_1$. Verifier sets RHS as new updated commitment.

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$$\mathbf{A}^{\star} \leftarrow \mathcal{R}_q^{m \times n}$$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m\times n}$$

$$\mathsf{aux} \leftarrow \mathsf{Samp}(\mathbf{A}^\star)$$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m \times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star$, aux$)$ to $\mathscr{A}$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathscr{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star,$ aux$)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star \mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

BASIS-👨‍👩‍👧‍👧 [WW23]

Samp$_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

BASIS Game

$\mathbf{A}^\star \leftarrow \mathscr{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star,$ aux$)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \le \beta$

BASIS-👨‍👩‍👧‍👧 [WW23]

$\mathsf{Samp}_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

$\mathsf{Samp}_{\mathsf{BASIS},\ell}(\mathbf{A}^\star)$

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star, \text{aux})$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m \times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star, \mathsf{aux})$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star \mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Samp$_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

Samp$_{\mathsf{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

BASIS-👨‍👩‍👧‍👧 [WW23]

Samp$_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star,$ aux$)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Samp$_{\mathsf{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

$$\mathbf{A}_1 := \begin{bmatrix}\mathbf{a}^\top \\ \mathbf{A}^\star\end{bmatrix}, \mathbf{B} := \begin{bmatrix}\mathbf{A}_1 & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & \mathbf{A}_d & -\mathbf{G}\end{bmatrix}$$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m \times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star$, aux$)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star \mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Samp$_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

Samp$_{\mathsf{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

$$\mathbf{A_1} := \begin{bmatrix} \mathbf{a}^\top \\ \mathbf{A}^\star \end{bmatrix}, \mathbf{B} := \begin{bmatrix} \mathbf{A}_1 & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & \mathbf{A}_d & -\mathbf{G} \end{bmatrix}$$

return $(\mathbf{a}, (\mathbf{A}_i)_i, \mathbf{B}^{-1}(\mathbf{G}))$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathscr{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star,$ aux$)$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Samp$_{\text{SIS}}(\mathbf{A}^\star)$

return $\perp$

Samp$_{\text{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

$$\mathbf{A}_1 := \begin{bmatrix} \mathbf{a}^\top \\ \mathbf{A}^\star \end{bmatrix}, \mathbf{B} := \begin{bmatrix} \mathbf{A}_1 & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & \mathbf{A}_d & -\mathbf{G} \end{bmatrix}$$

return $(\mathbf{a}, (\mathbf{A}_i)_i, \mathbf{B}^{-1}(\mathbf{G}))$

Samp$_{\text{PRISIS},\ell}(\mathbf{A}^\star)$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m\times n}$

$\mathsf{aux} \leftarrow \mathsf{Samp}(\mathbf{A}^\star)$

return $(\mathbf{A}^\star, \mathsf{aux})$ to $\mathcal{A}$

$\mathcal{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

$\mathsf{Samp}_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

$\mathsf{Samp}_{\mathsf{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

$$\mathbf{A_1} := \begin{bmatrix} \mathbf{a}^\top \\ \mathbf{A}^\star \end{bmatrix}, \mathbf{B} := \begin{bmatrix} \mathbf{A}_1 & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & \mathbf{A}_d & -\mathbf{G} \end{bmatrix}$$

return $(\mathbf{a}, (\mathbf{A}_i)_i, \mathbf{B}^{-1}(\mathbf{G}))$

$\mathsf{Samp}_{\mathsf{PRISIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, w$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star,$ aux$)$ to $\mathcal{A}$

$\mathcal{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Samp$_{\mathsf{SIS}}(\mathbf{A}^\star)$

return $\perp$

Samp$_{\mathsf{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

$$\mathbf{A}_1 := \begin{bmatrix} \mathbf{a}^\top \\ \mathbf{A}^\star \end{bmatrix}, \mathbf{B} := \begin{bmatrix} \mathbf{A}_1 & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & \mathbf{A}_d & -\mathbf{G} \end{bmatrix}$$

return $(\mathbf{a}, (\mathbf{A}_i)_i, \mathbf{B}^{-1}(\mathbf{G}))$

Samp$_{\mathsf{PRISIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, w$

$$\mathbf{A} := \begin{bmatrix} \mathbf{a}^\top \\ \mathbf{A}^\star \end{bmatrix}, \mathbf{B} := \begin{bmatrix} w^0\mathbf{A} & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & w^{\ell-1}\mathbf{A} & -\mathbf{G} \end{bmatrix}$$

BASIS-👨‍👩‍👧‍👧 [WW23]

BASIS Game

$\mathbf{A}^\star \leftarrow \mathcal{R}_q^{m\times n}$

aux $\leftarrow$ Samp$(\mathbf{A}^\star)$

return $(\mathbf{A}^\star, \text{aux})$ to $\mathscr{A}$

$\mathscr{A}$ wins if it finds $\mathbf{x}$:

- $\mathbf{A}^\star\mathbf{x} = 0$
- $0 < |\mathbf{x}| \leq \beta$

Samp$_{\text{SIS}}(\mathbf{A}^\star)$

return $\perp$

Samp$_{\text{BASIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, \mathbf{A}_2, \ldots \mathbf{A}_\ell$

$$\mathbf{A}_1 := \begin{bmatrix}\mathbf{a}^\top \\ \mathbf{A}^\star\end{bmatrix}, \mathbf{B} := \begin{bmatrix}\mathbf{A}_1 & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & \mathbf{A}_d & -\mathbf{G}\end{bmatrix}$$

return $(\mathbf{a}, (\mathbf{A}_i)_i, \mathbf{B}^{-1}(\mathbf{G}))$

Samp$_{\text{PRISIS},\ell}(\mathbf{A}^\star)$

Sample $\mathbf{a}, w$

$$\mathbf{A} := \begin{bmatrix}\mathbf{a}^\top \\ \mathbf{A}^\star\end{bmatrix}, \mathbf{B} := \begin{bmatrix}w^0\mathbf{A} & \ldots & & -\mathbf{G} \\ & \ddots & & \\ & \ldots & w^{\ell-1}\mathbf{A} & -\mathbf{G}\end{bmatrix}$$

return $(\mathbf{a}, w, \mathbf{B}^{-1}(\mathbf{G}))$

Recap:

What we talked about

Recap:

What we talked about

- PRISIS and Merkle-PRISIS commitments

Recap:

What we talked about

- PRISIS and Merkle-PRISIS commitments
- Multi-instance PRISIS assumptions

Recap:

What we talked about

- PRISIS and Merkle-PRISIS commitments
- Multi-instance PRISIS assumptions
- h-PRISIS$_2$ reduces to MSIS

Recap:

What we talked about

- PRISIS and Merkle-PRISIS commitments
- Multi-instance PRISIS assumptions
- h-PRISIS$_2$ reduces to MSIS
- Succinct evaluation protocol for Merkle-PRISIS

Recap:

What we talked about

- PRISIS and Merkle-PRISIS commitments
- Multi-instance PRISIS assumptions
- h-PRISIS$_2$ reduces to MSIS
- Succinct evaluation protocol for Merkle-PRISIS
- Boosting soundness via claim bundling

Trapdoors [MP12]

Trapdoors [MP12]

- Let $\mathbf{G}$ be a “gadget matrix”

Trapdoors [MP12]

- Let $\mathbf{G}$ be a “gadget matrix”
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{AR} = \mathbf{G}$, with $\mathbf{R}$ short.

Trapdoors [MP12]

"Not nice"

$\Lambda^{\perp}(\mathbf{A})$

- Let $\mathbf{G}$ be a "gadget matrix"
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{AR} = \mathbf{G}$, with $\mathbf{R}$ short.

Trapdoors [MP12]

“Not nice”

$\Lambda^{\perp}(\mathbf{A})$

- Let $\mathbf{G}$ be a “gadget matrix”
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{AR} = \mathbf{G}$, with $\mathbf{R}$ short.

$\Lambda^{\perp}(\mathbf{G})$

“Nice”

Trapdoors [MP12]

- Let $\mathbf{G}$ be a “gadget matrix”
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{AR} = \mathbf{G}$, with $\mathbf{R}$ short.

“Not nice”

$$\begin{tikzcd} \Lambda^{\perp}(\mathbf{A}) \arrow[d, bend left] \arrow[d, phantom, "\mathbf{R}"] \\ \Lambda^{\perp}(\mathbf{G}) \arrow[u, bend left] \end{tikzcd}$$

“Nice”

Trapdoors [MP12]

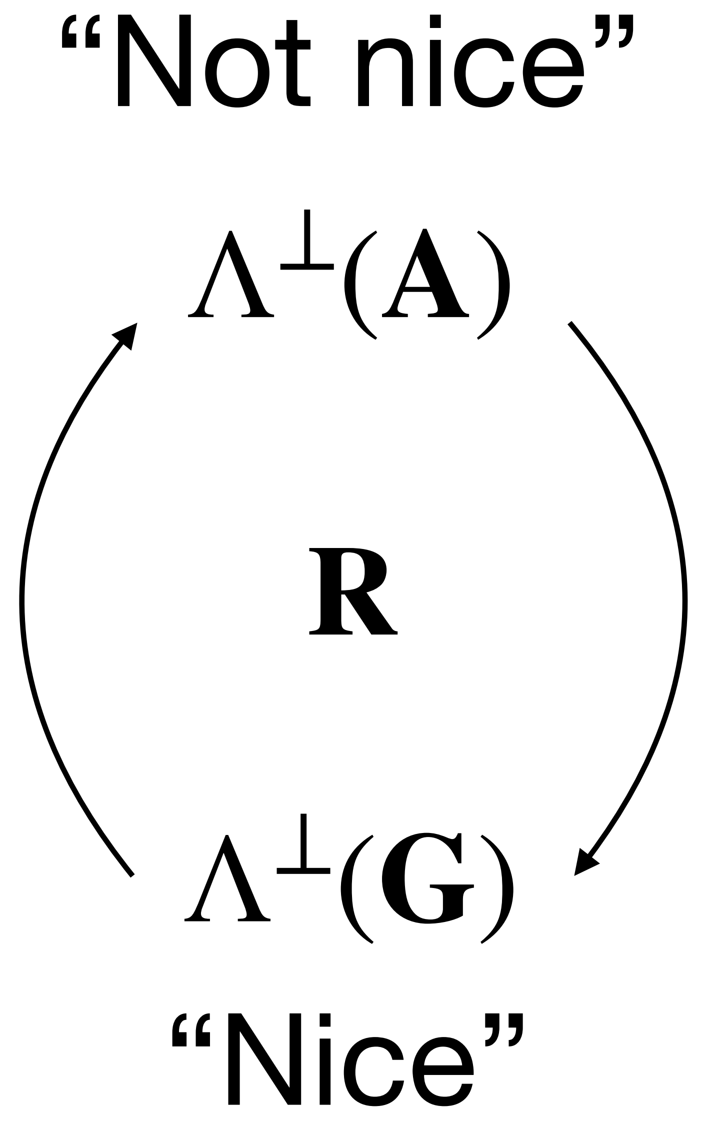
- Let $\mathbf{G}$ be a “gadget matrix”
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{A}\mathbf{R} = \mathbf{G}$, with $\mathbf{R}$ short.
- Given $\mathbf{A}, \mathbf{R}, \mathbf{v}$, can sample short $\mathbf{s}$ such that $\mathbf{A}\mathbf{s} = \mathbf{v}$.

“Not nice”

$$\begin{tikzcd} \Lambda^{\perp}(\mathbf{A}) \arrow[d, bend left] \arrow[d, phantom, "\mathbf{R}"] \\ \Lambda^{\perp}(\mathbf{G}) \arrow[u, bend left] \end{tikzcd}$$

“Nice”

Trapdoors [MP12]



- Let $\mathbf{G}$ be a "gadget matrix"
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{AR} = \mathbf{G}$, with $\mathbf{R}$ short.
- Given $\mathbf{A}, \mathbf{R}, \mathbf{v}$, can sample short $\mathbf{s}$ such that $\mathbf{As} = \mathbf{v}$.

Trapdoor Resampling [WW23]

Trapdoors [MP12]

- Let $\mathbf{G}$ be a “gadget matrix”
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{AR} = \mathbf{G}$, with $\mathbf{R}$ short.
- Given $\mathbf{A}, \mathbf{R}, \mathbf{v}$, can sample short $\mathbf{s}$ such that $\mathbf{As} = \mathbf{v}$.

“Not nice”

$$\begin{tikzcd} \Lambda^{\perp}(\mathbf{A}) \arrow[d, bend left] \arrow[d, phantom, "\mathbf{R}"] \\ \Lambda^{\perp}(\mathbf{G}) \arrow[u, bend left] \end{tikzcd}$$

“Nice”

Trapdoor Resampling [WW23]

- Given $(\mathbf{A}, \mathbf{R})$, can sample new trapdoor $\mathbf{T}$ for some matrix $\mathbf{B}$ “related” to $\mathbf{A}$

Trapdoors [MP12]

“Not nice”

$$\begin{tikzcd} \Lambda^{\perp}(\mathbf{A}) \arrow[d, bend left=60, ""{name=U}] \\ \Lambda^{\perp}(\mathbf{G}) \arrow[u, bend left=60, ""{name=D}] \arrow[from=U, to=D, phantom, "\mathbf{R}"] \end{tikzcd}$$

“Nice”

- Let $\mathbf{G}$ be a “gadget matrix”
- Can sample $(\mathbf{A}, \mathbf{R})$ such that $\mathbf{A}\mathbf{R} = \mathbf{G}$, with $\mathbf{R}$ short.
- Given $\mathbf{A}, \mathbf{R}, \mathbf{v}$, can sample short $\mathbf{s}$ such that $\mathbf{A}\mathbf{s} = \mathbf{v}$.

Trapdoor Resampling [WW23]

- Given $(\mathbf{A}, \mathbf{R})$, can sample new trapdoor $\mathbf{T}$ for some matrix $\mathbf{B}$ “related” to $\mathbf{A}$
- BASIS style assumption say:

“Given $\mathbf{A}, \mathbf{B}, \mathbf{T}$, hard to find short $\mathbf{x}$ for $\mathbf{A}\mathbf{x} = \mathbf{0}$”