LatticeFold

Memory-Efficient Proving at Scale

Binyi Chen

Joint work with Dan Boneh

Stanford University

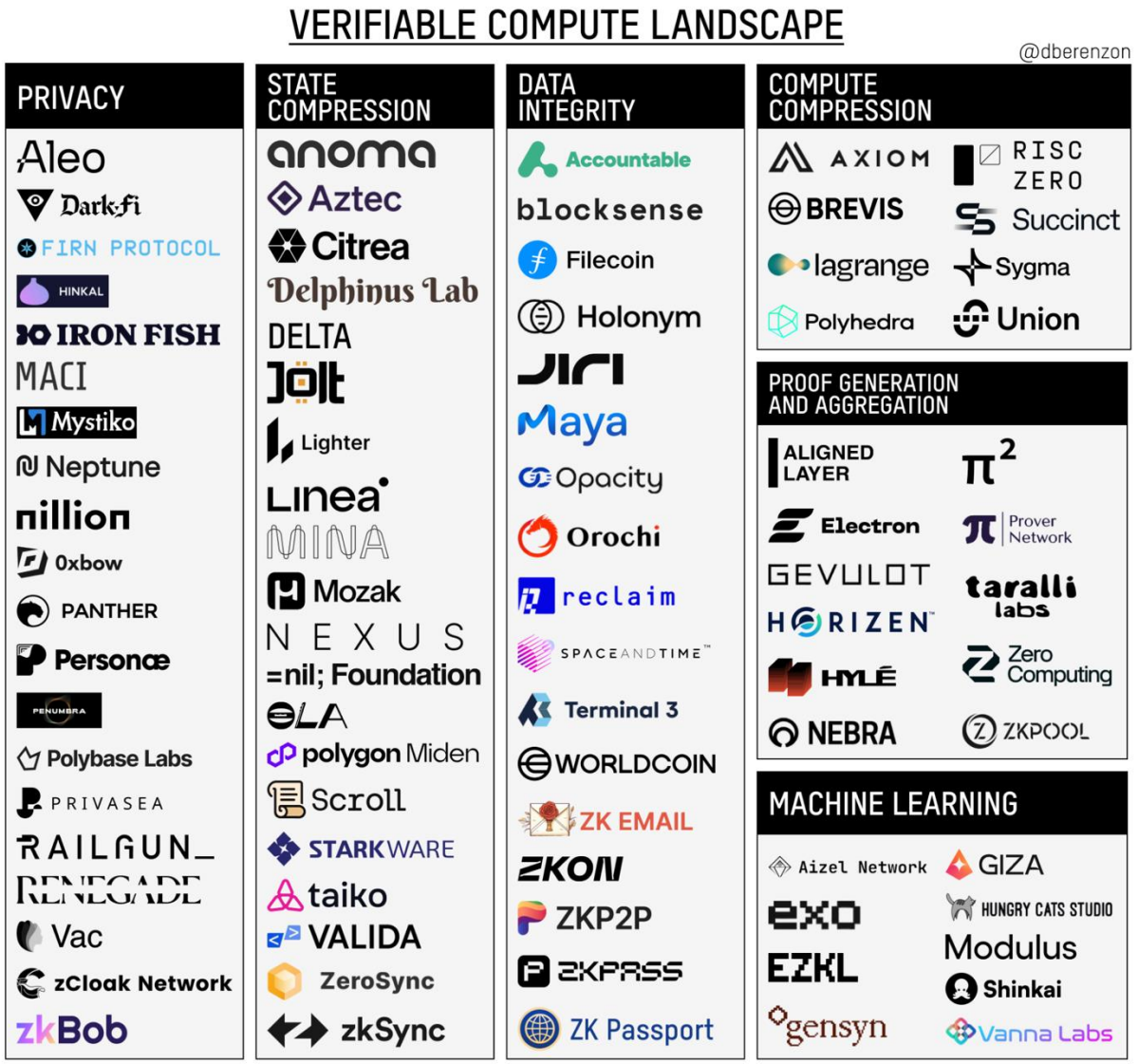
ZK-SNARKs: Advanced Applications

Applications

- zkVM/zkML
- Image/Video Provenance
- ZK Wallet/Passport
- Data Availability
- Decentralized Storage
- .....

Common Theme

Large-scale computation



ZK-SNARKs: Advanced Applications

Applications

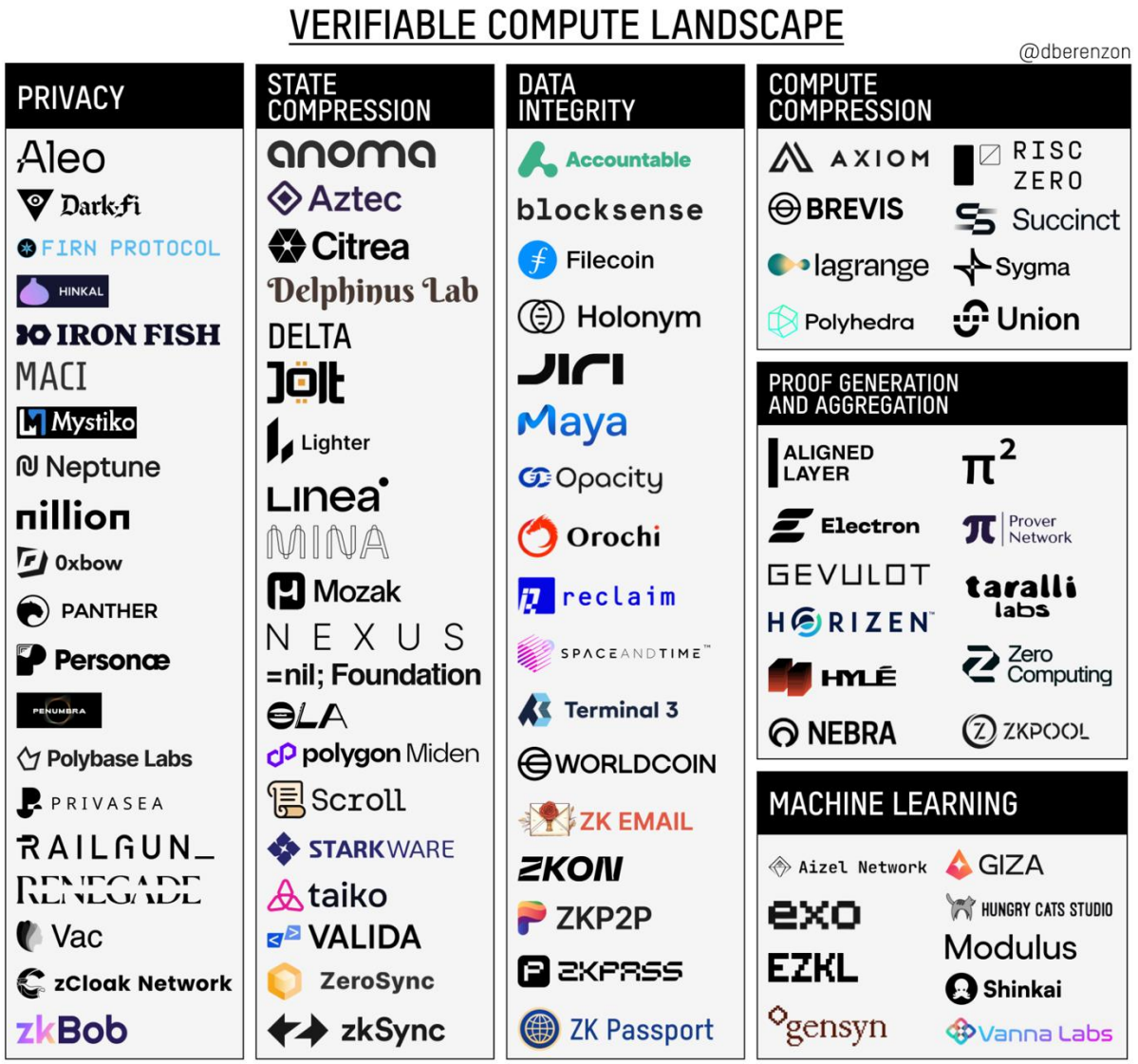
- zkVM/zkML
- Image/Video Provenance
- ZK Wallet/Passport
- Data Availability
- Decentralized Storage
- .....

Common Theme

Large-scale computation

Design Requirements

Scalable prover + post-quantum security

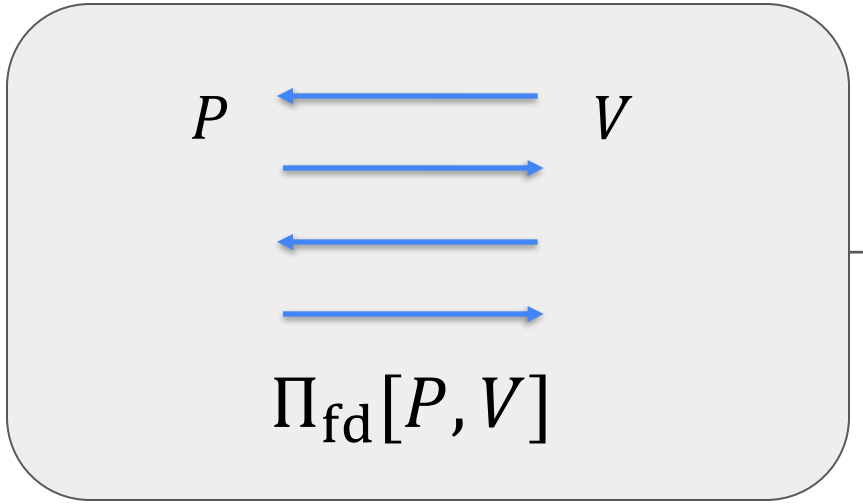


Folding Schemes [BGH'19, BCLMS'20, KST'21]

$|R_{NP}| \approx |R_{acc}|$

$(x_1, w_1) \in ? R_{NP}$

$(x_2, w_2) \in ? R_{acc}$



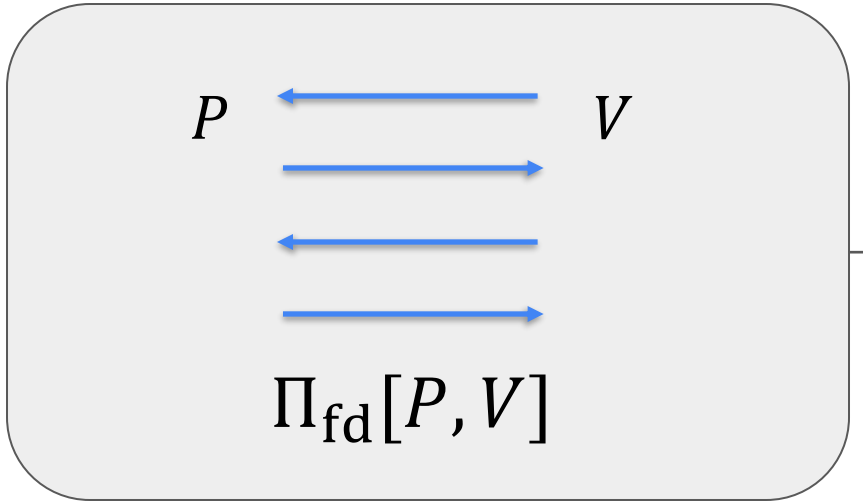
$(x, w) \in ? R_{acc}$

Folding Schemes [BGH'19, BCLMS'20, KST'21]

$|R_{NP}| \approx |R_{acc}|$

$(x_1, w_1) \in ? R_{NP}$

$(x_2, w_2) \in ? R_{acc}$



$(x, w) \in ? R_{acc}$

Non-Interactive Version:
$FS^{H}(\Pi_{fd}[P, V])$

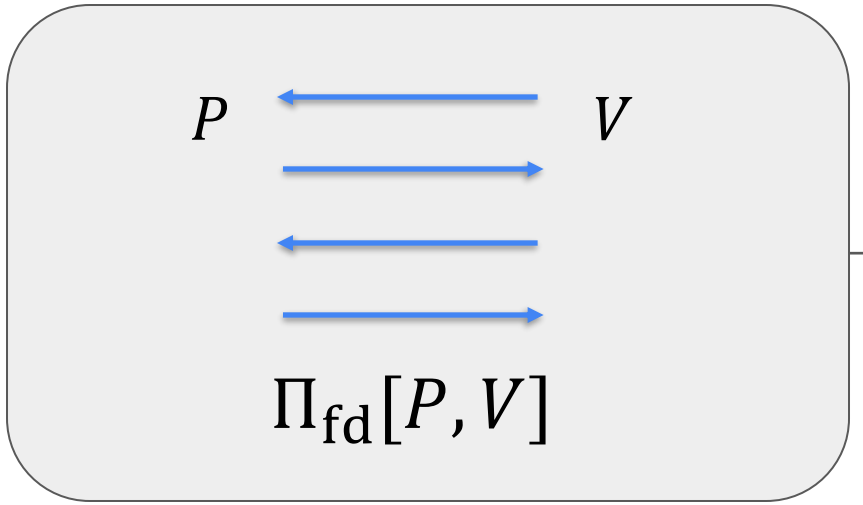
Folding Schemes [BGH'19, BCLMS'20, KST'21]

Complete + Knowledge sound

$|R_{NP}| \approx |R_{acc}|$

$(x_1, w_1) \in_? R_{NP}$

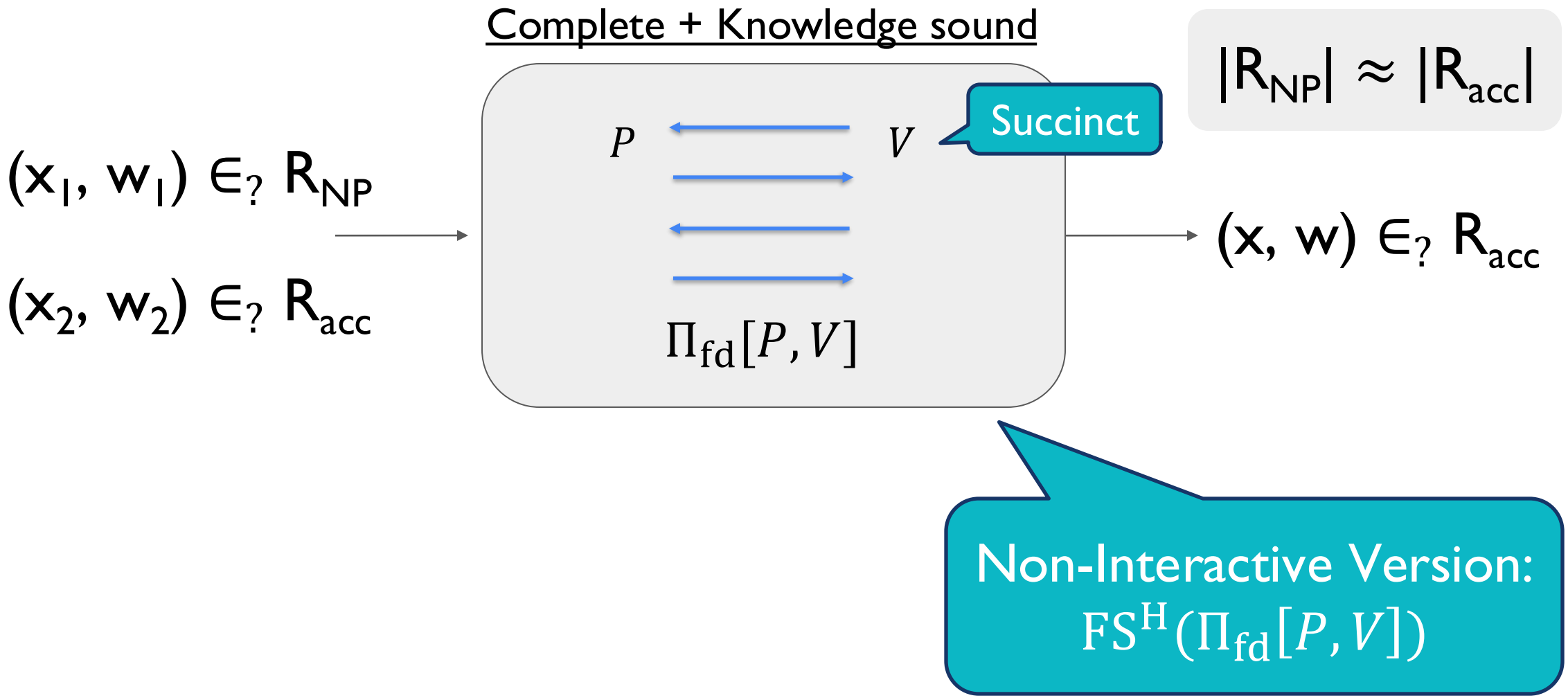
$(x_2, w_2) \in_? R_{acc}$



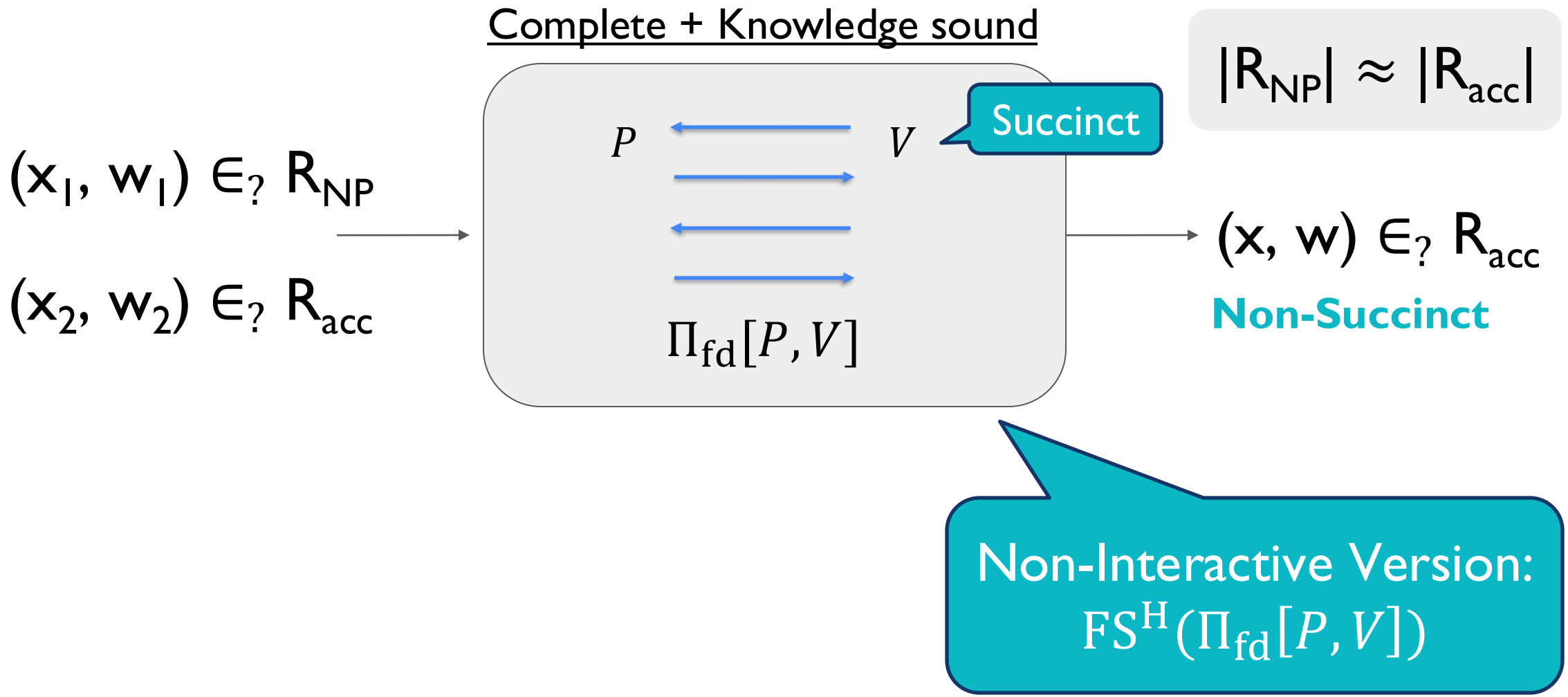
$(x, w) \in_? R_{acc}$

Non-Interactive Version:
$FS^{H}(\Pi_{fd}[P, V])$

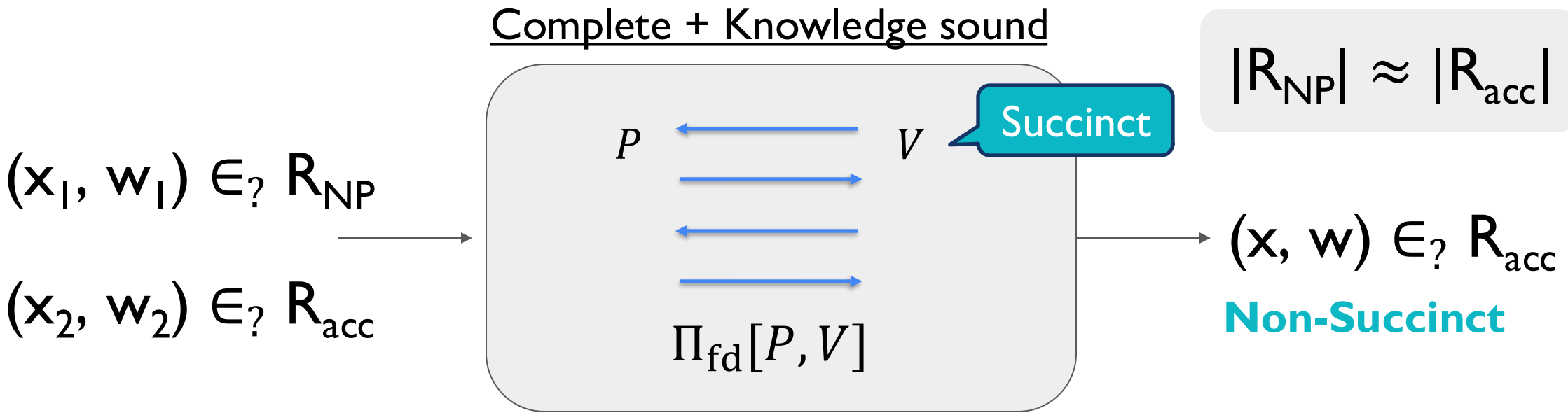
Folding Schemes [BGH'19, BCLMS'20, KST'21]



Folding Schemes [BGH'19, BCLMS'20, KST'21]



Folding Schemes [BGH'19, BCLMS'20, KST'21]

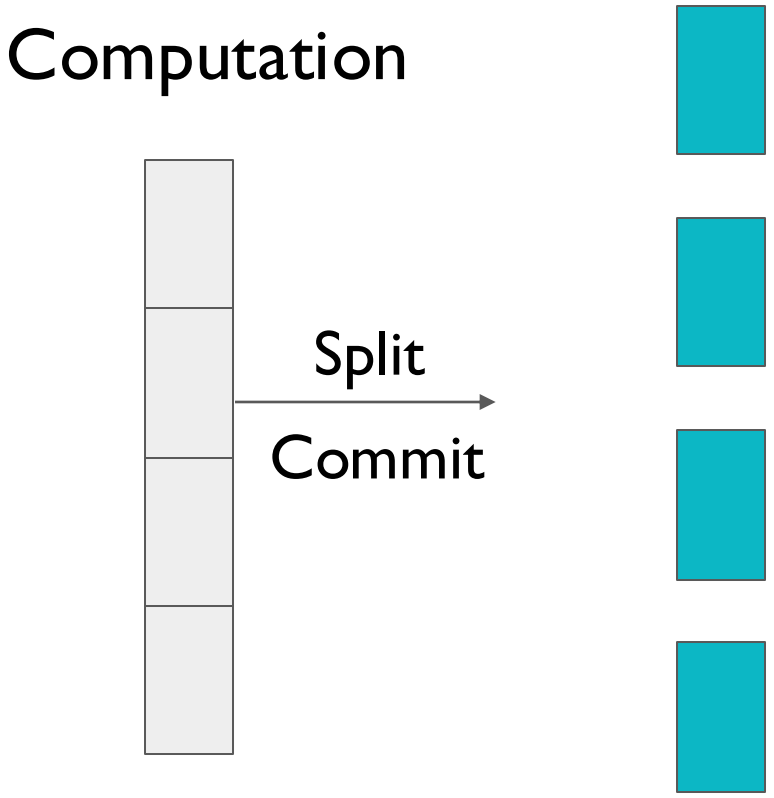


Advantages:

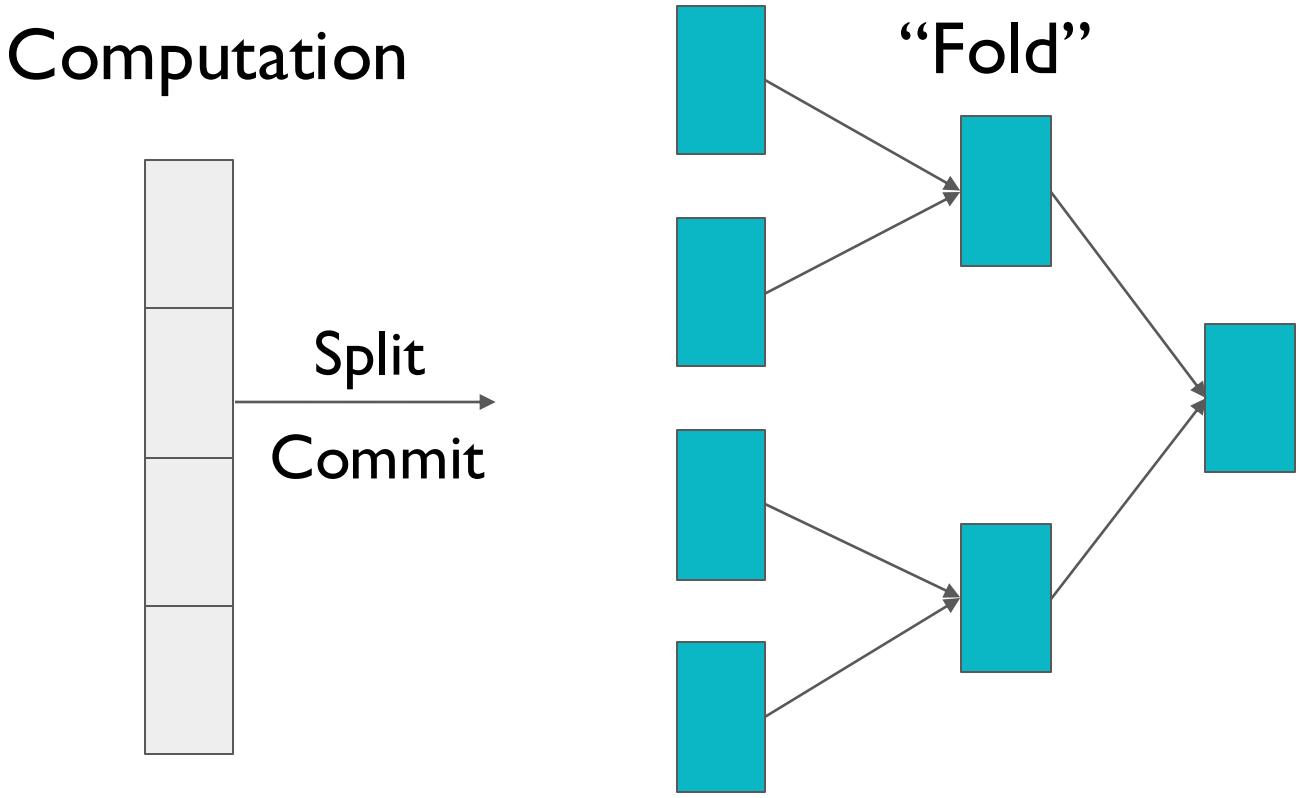
- Much faster than SNARKs
- Boost SNARK efficiency

Non-Interactive Version:
$FS^H(\Pi_{fd}[P, V])$

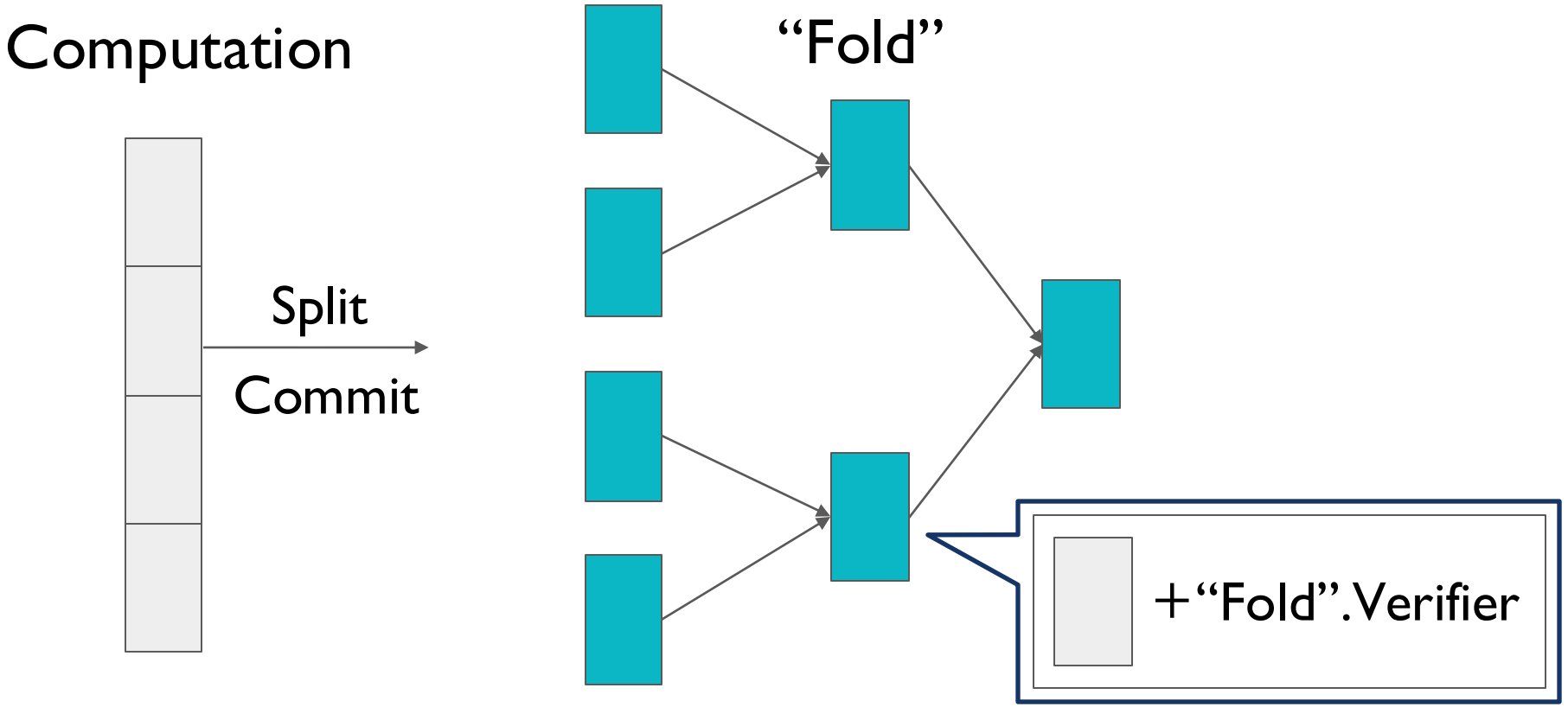
Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]



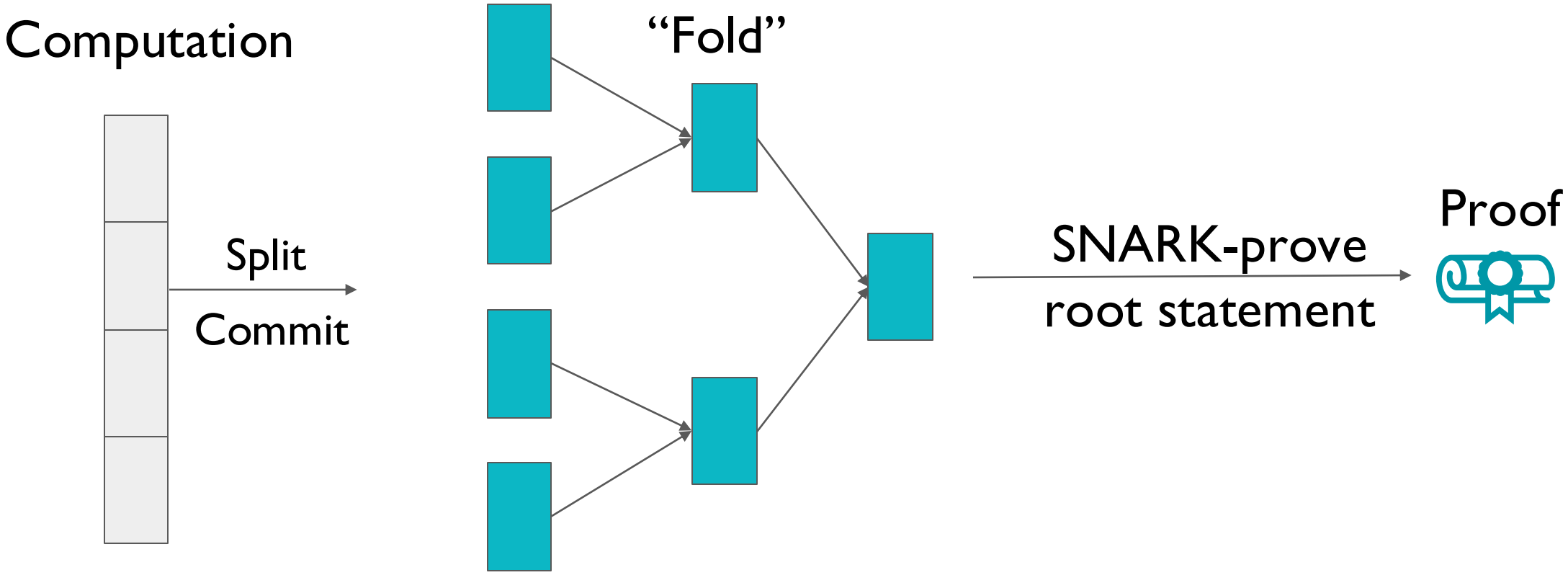
Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]



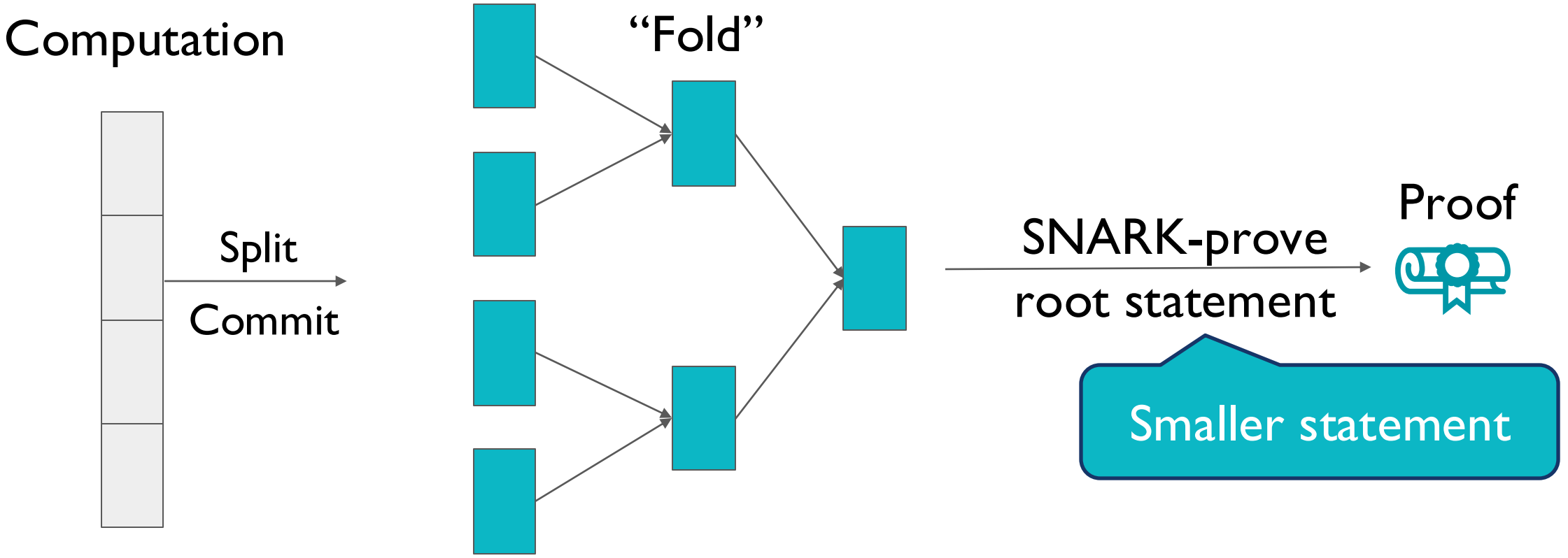
Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]



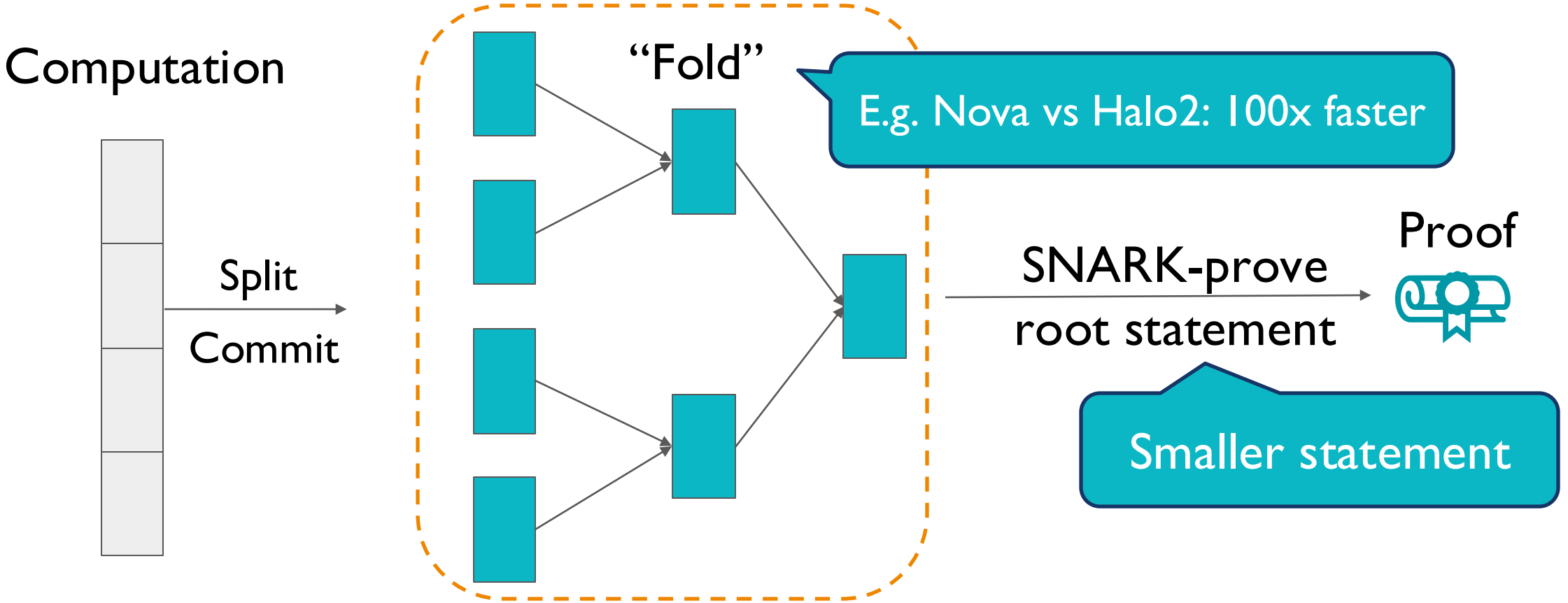
Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]



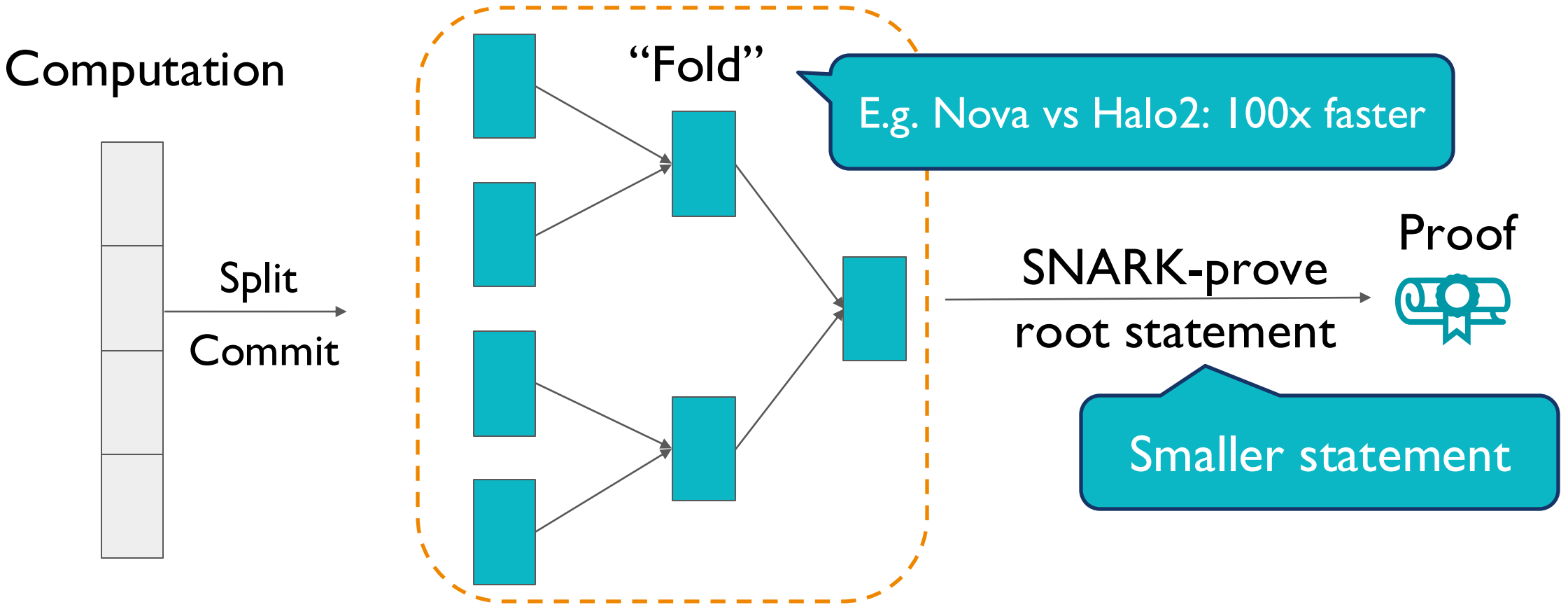
Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]



Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]



Folding-based SNARKs [BGH'19, COS'20, NDCTB'24…]

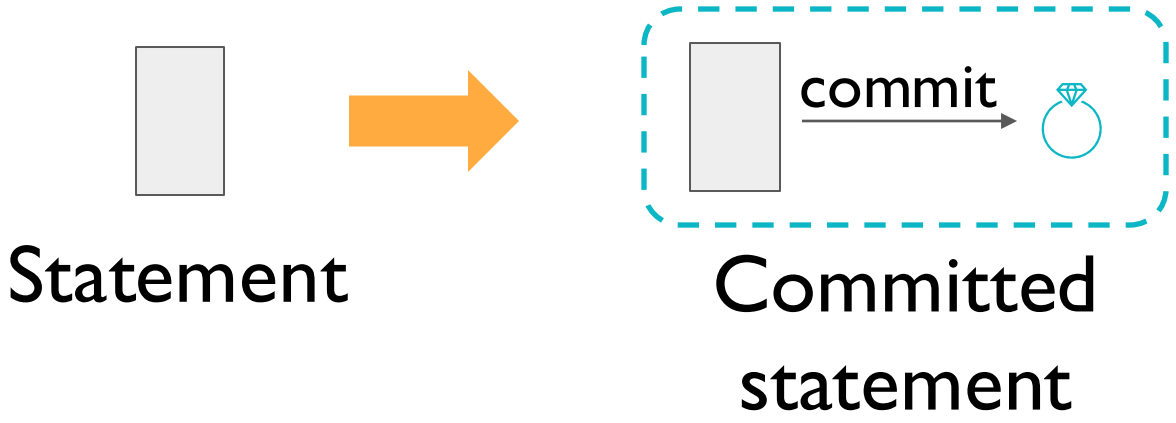


Qs: Which commitment should we use?

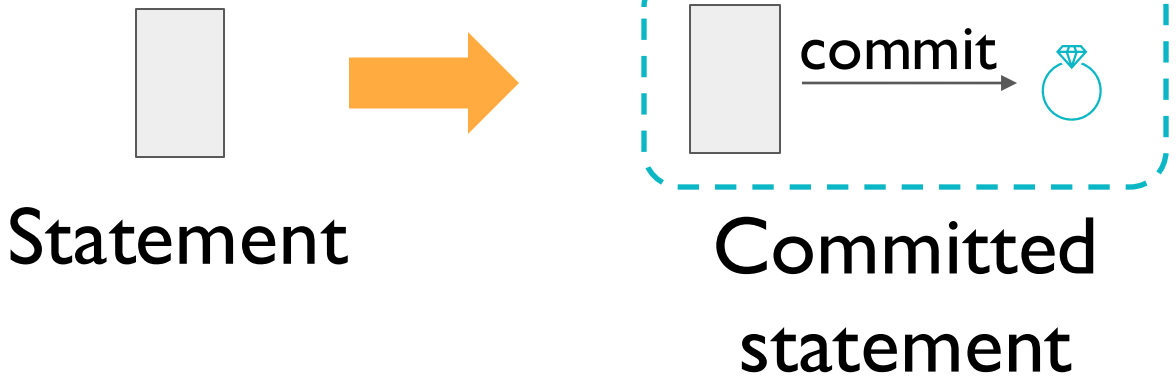
The Choice of Commitments

Statement

The Choice of Commitments

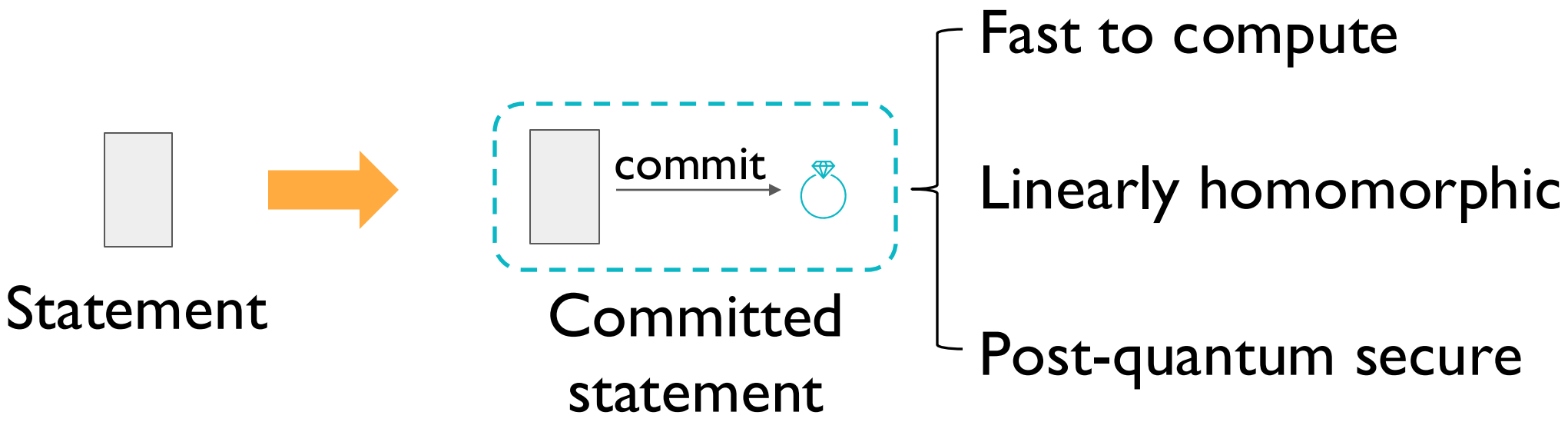


The Choice of Commitments



- Fast to compute
- Linearly homomorphic
- Post-quantum secure

The Choice of Commitments

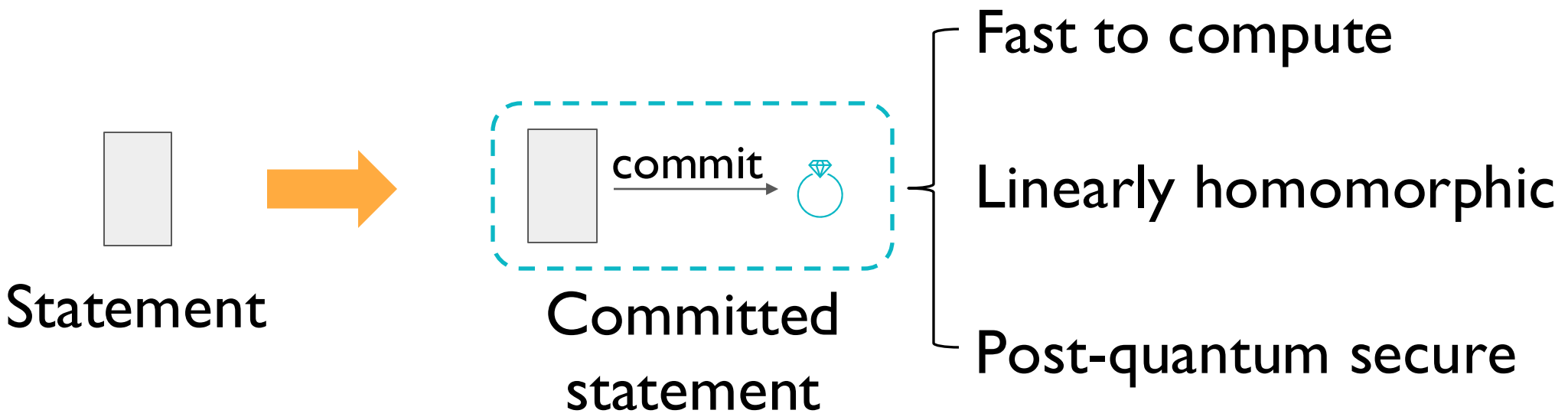


Before: Pedersen commitments [KST'21, BCLMS'21, BC'23, KS'23…]

Pros:

- Linearly homomorphic

The Choice of Commitments



Before: Pedersen commitments [KST'21, BCLMS'21, BC'23, KS'23…]

Pros:

- Linearly homomorphic

Cons:

- Pre-quantum
- Require expensive cycle-curves

Q: A folding scheme from a faster and PQ-secure commitment?

Our Contribution

Main Result

- The first plausibly PQ-secure folding scheme from Ajtai hashes
- Commitment space ≈ Witness space ⇒ Native verifier circuit

Our Contribution

Main Result

- The first plausibly PQ-secure folding scheme from Ajtai hashes
- Commitment space ≈ Witness space ⇒ Native verifier circuit

Follow-up Works:

- LF+ [BC'25], Symphony [C'25]
- Neo [NS'25]
- RoK-and-Roll [KLNO'25]
- Lova [FKNP'24]

Our Contribution

Main Result

- The first plausibly PQ-secure folding scheme from Ajtai hashes
- Commitment space ≈ Witness space ⇒ Native verifier circuit

Follow-up Works:

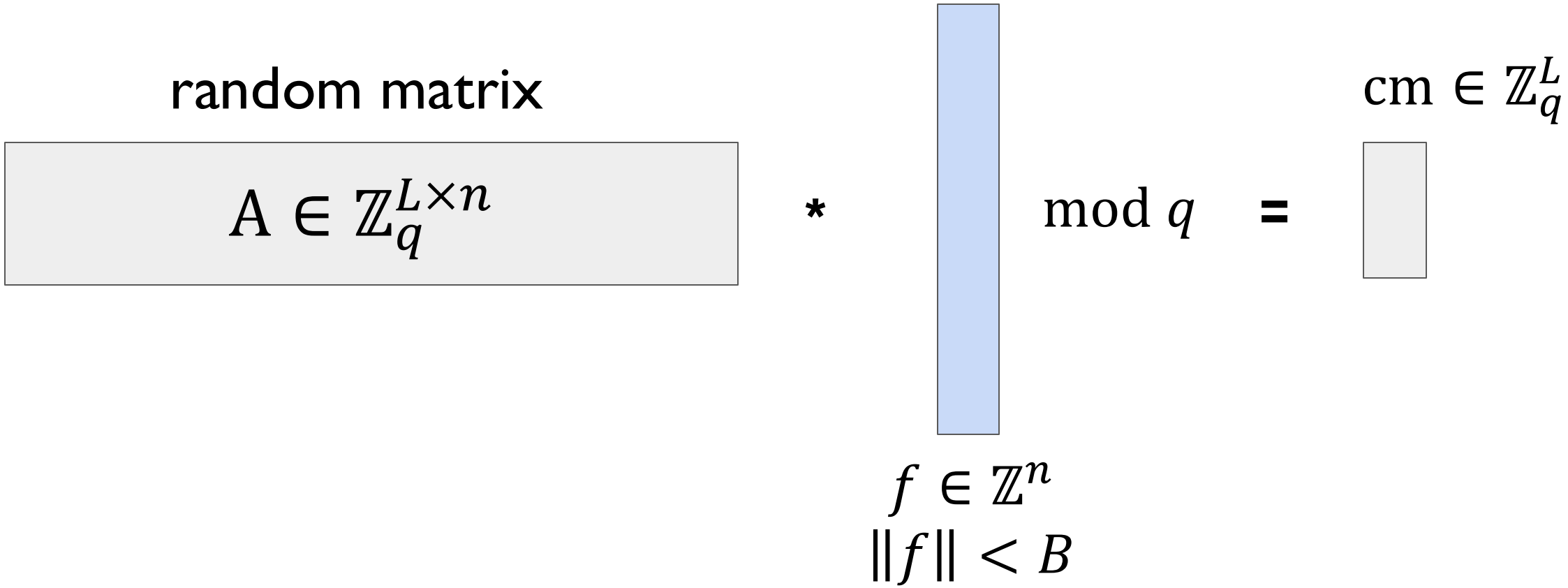
- LF+ [BC'25], Symphony [C'25]
- Neo [NS'25]
- RoK-and-Roll [KLNO'25]
- Lova [FKNP'24]

Hash-based Schemes:

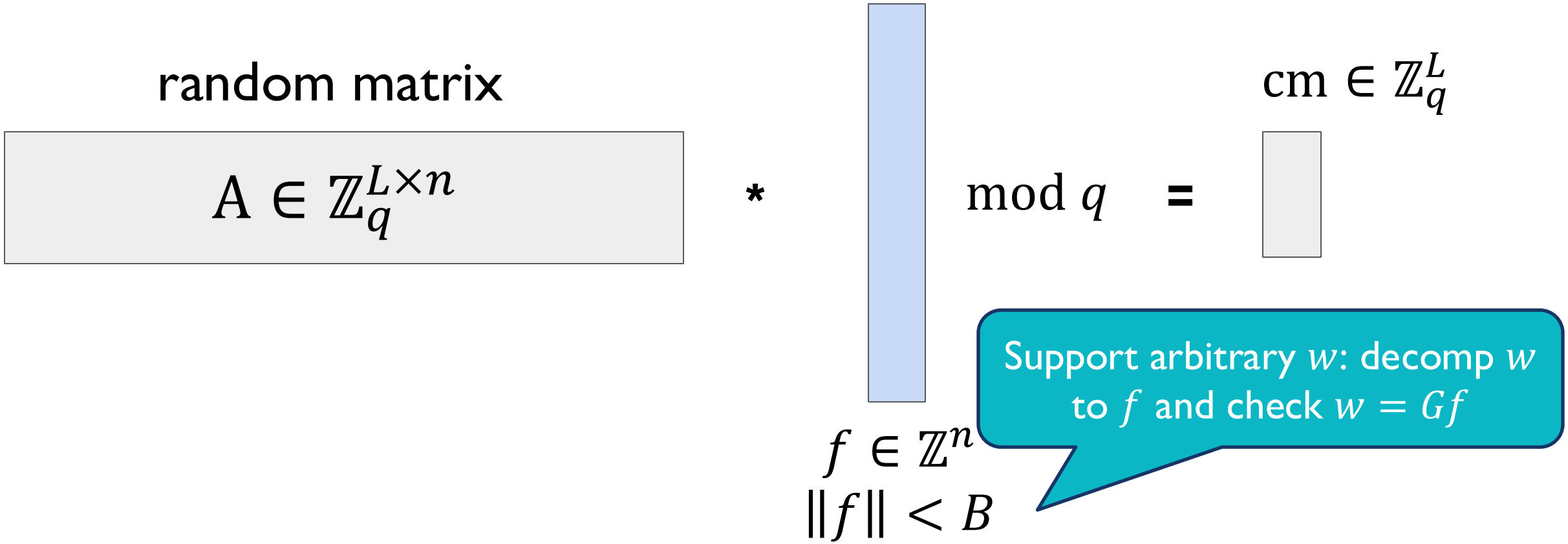
[BMNW'25a, BMNW'25b, BCFW25, BMMS'25…]

- Based on Merkle-commitments
- PQ-secure and fast committing
- More expensive verifier circuit

Lattice-based Commitments [Ajtai'96]



Lattice-based Commitments [Ajtai'96]



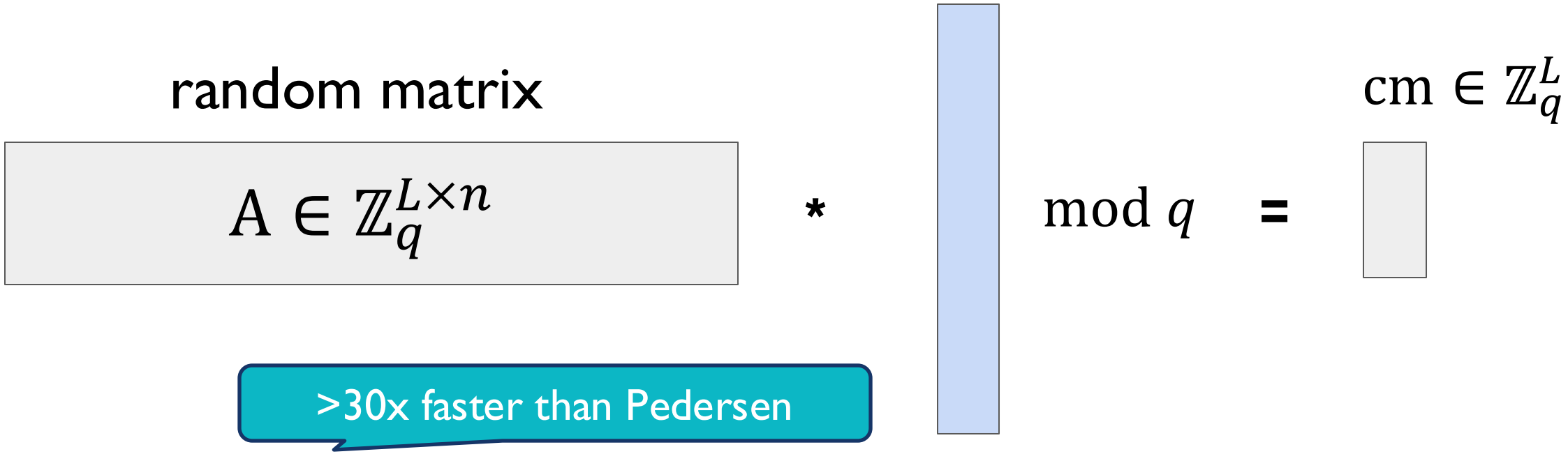
Lattice-based Commitments [Ajtai'96]

"Semi"-linearly homomorphic

random matrix

$$A \in \mathbb{Z}_q^{L \times n} \quad * \quad \begin{matrix} f \in \mathbb{Z}^n \\ \|f\| < B \end{matrix} \quad \bmod q \quad = \quad \mathrm{cm} \in \mathbb{Z}_q^L$$

Lattice-based Commitments [Ajtai'96]



>30x faster than Pedersen

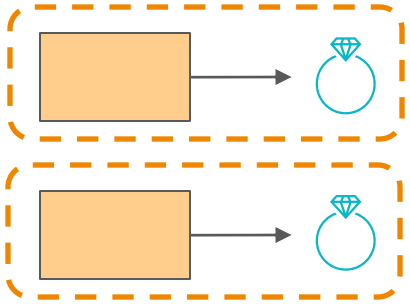
SWIFFT [LMPR'08]

A faster variant over R_q!

Q: Plug to existing folding templates?

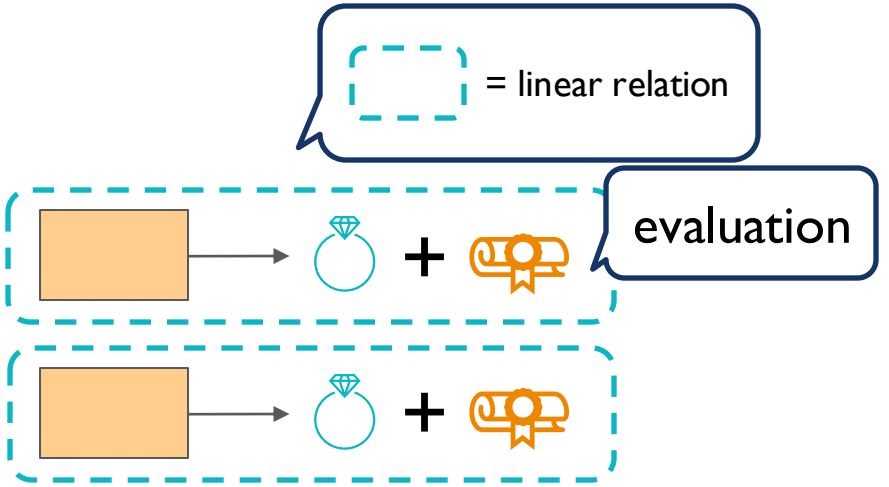
A Typical Folding Framework [KS'23]

Step 1: Commit



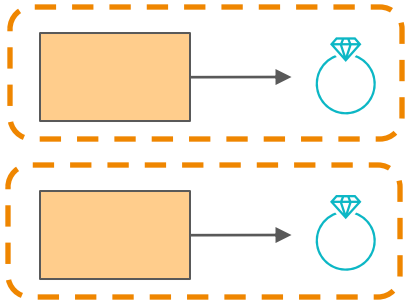
Witnesses + Comm

Step 2: Linearization



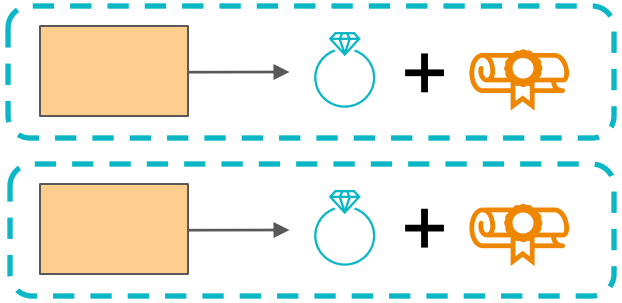
A Typical Folding Framework [KS'23]

Step 1: Commit

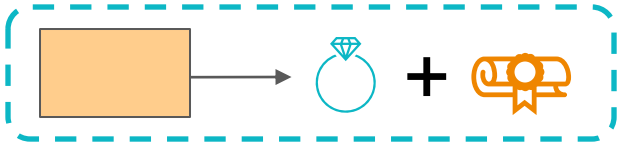


Witnesses + Comm

Step 2: Linearization

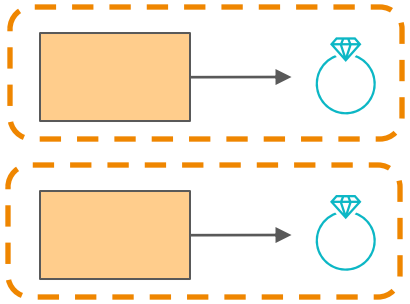


Step 3: Fold



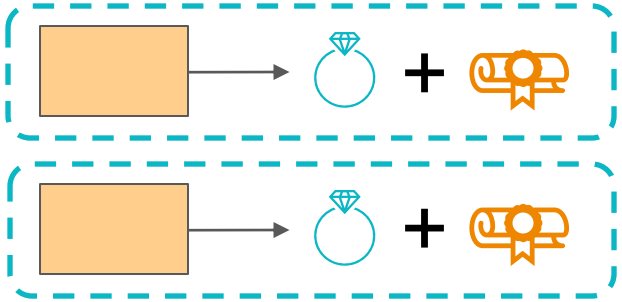
A Typical Folding Framework [KS'23]

Step 1: Commit



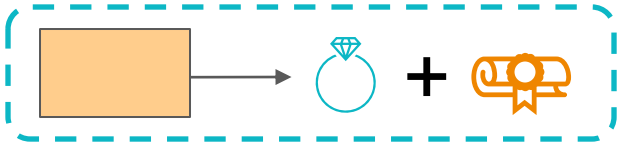
Witnesses + Comm

Step 2: Linearization



rand. lincomb

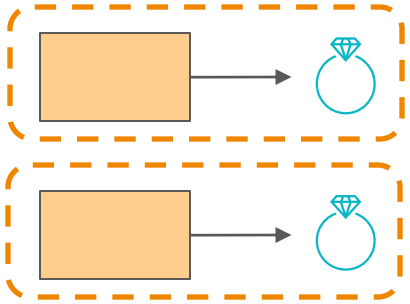
Step 3: Fold



Challenges in using lattice commitments

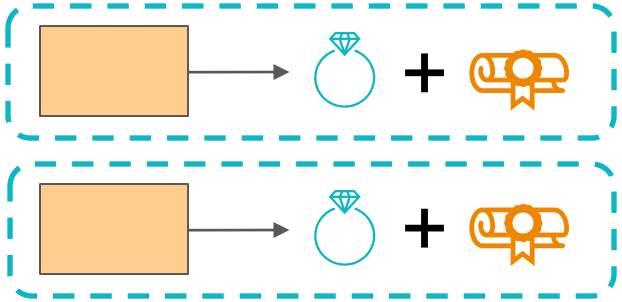
A Typical Folding Framework [KS'23]

Step 1: Commit



Witnesses + Comm

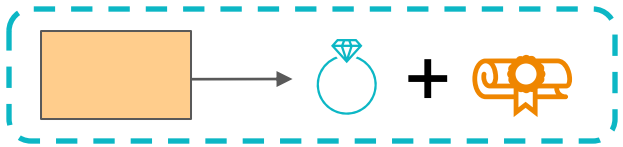
Step 2: Linearization



rand. lincomb

Step 3: Fold

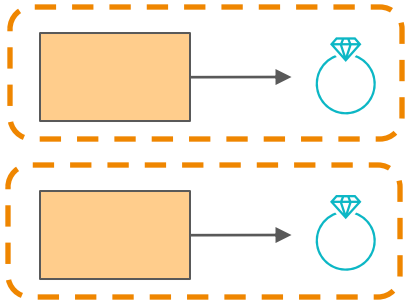
Challenge 1: Control norm blowup



Challenges in using lattice commitments

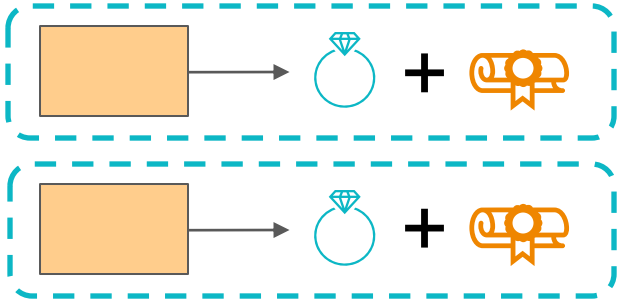
A Typical Folding Framework [KS'23]

Step 1: Commit



Witnesses + Comm

Step 2: Linearization

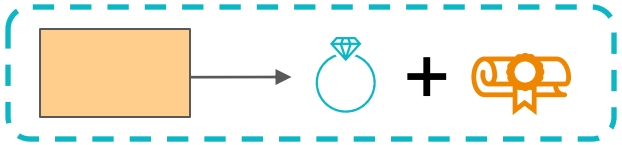


Challenge 2: Extract low-normed witnesses

rand. lincomb

Step 3: Fold

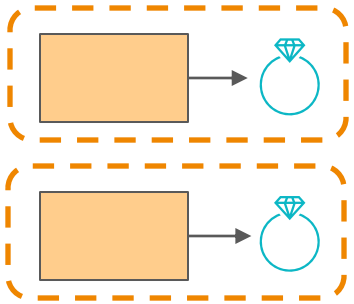
Challenge 1: Control norm blowup



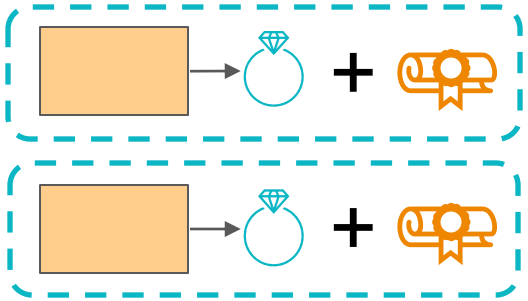
Challenges in using lattice commitments

A New Lattice-Folding Framework

Step 1: Commit

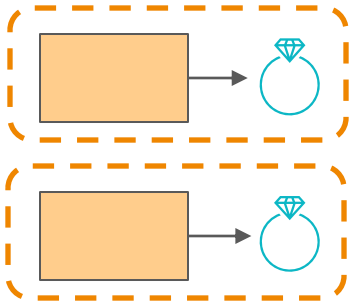


Step 2: Linearize

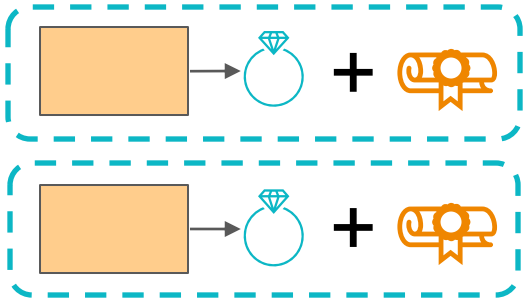


A New Lattice-Folding Framework

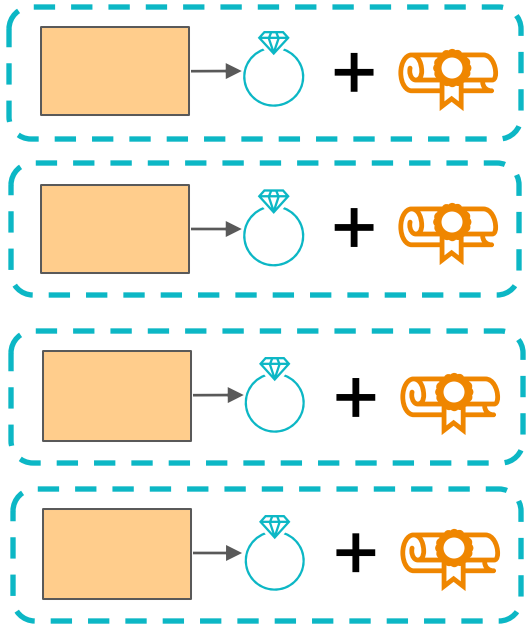
Step 1: Commit



Step 2: Linearize

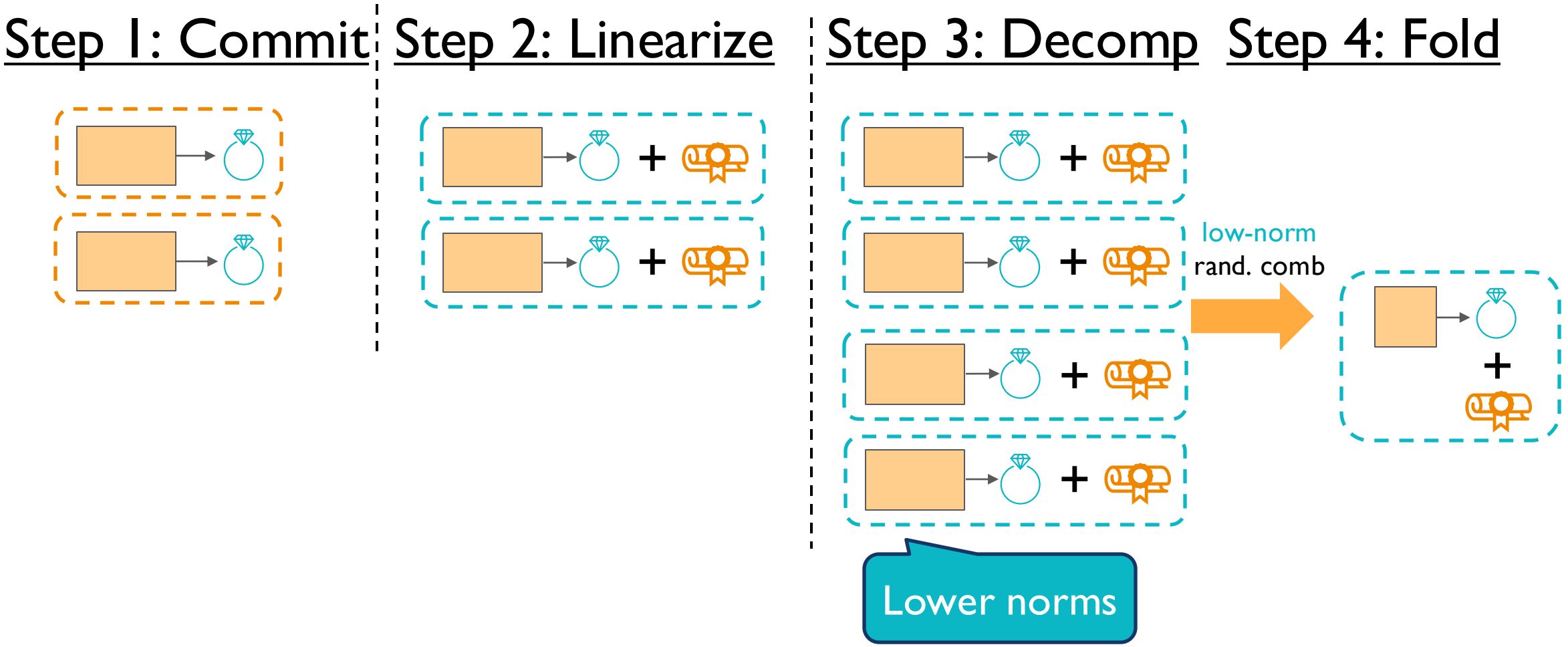


Step 3: Decomp

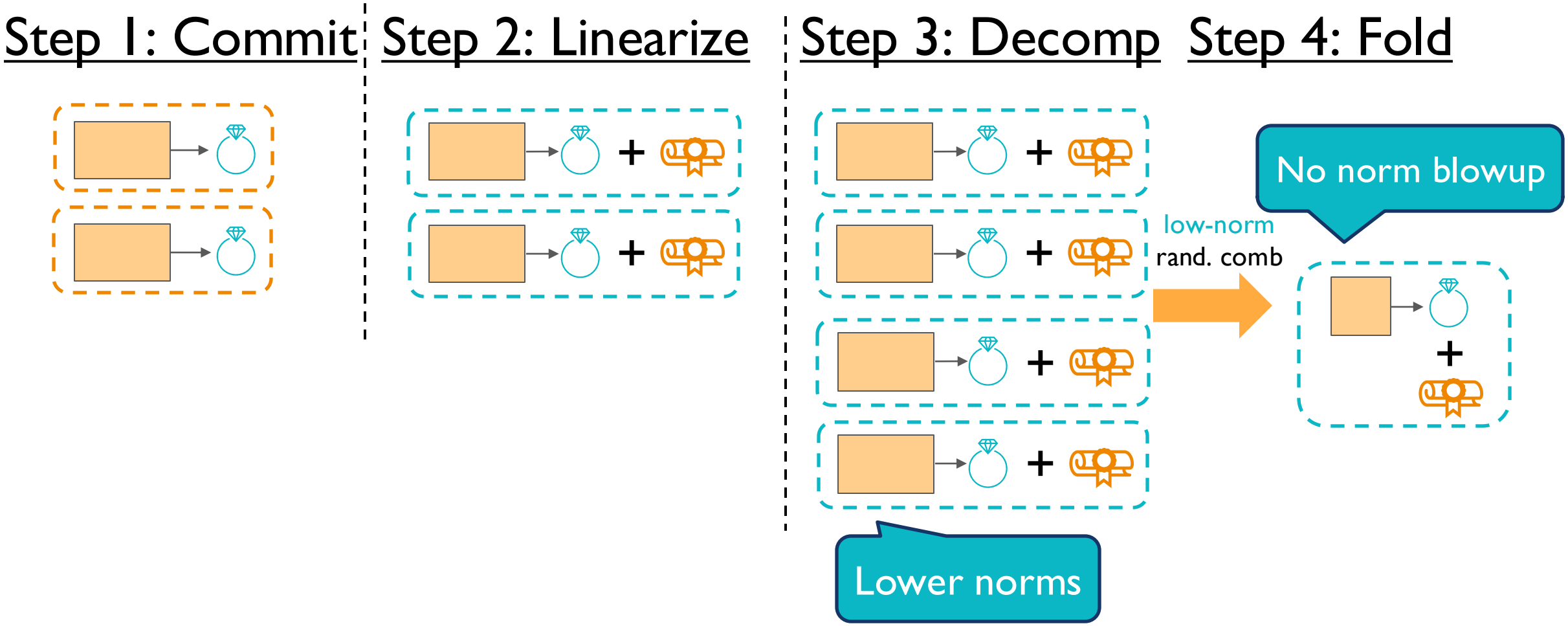


Lower norms

A New Lattice-Folding Framework

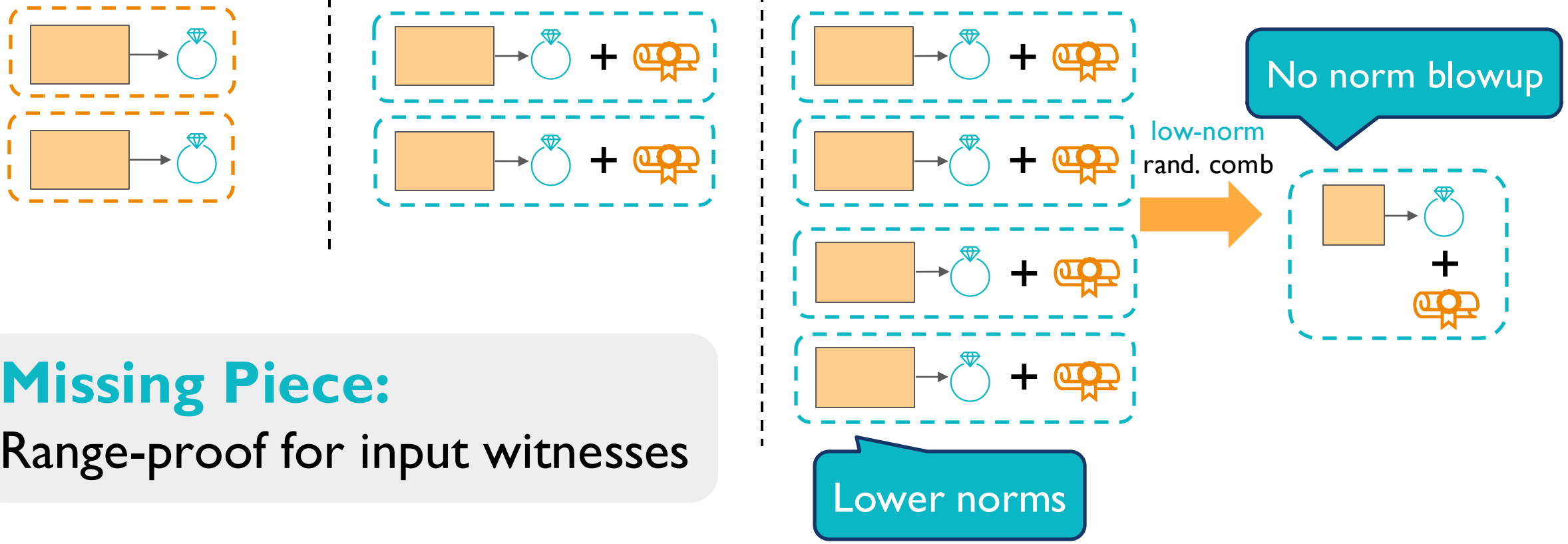


A New Lattice-Folding Framework



A New Lattice-Folding Framework

Step 1: Commit | Step 2: Linearize | Step 3: Decomp | Step 4: Fold

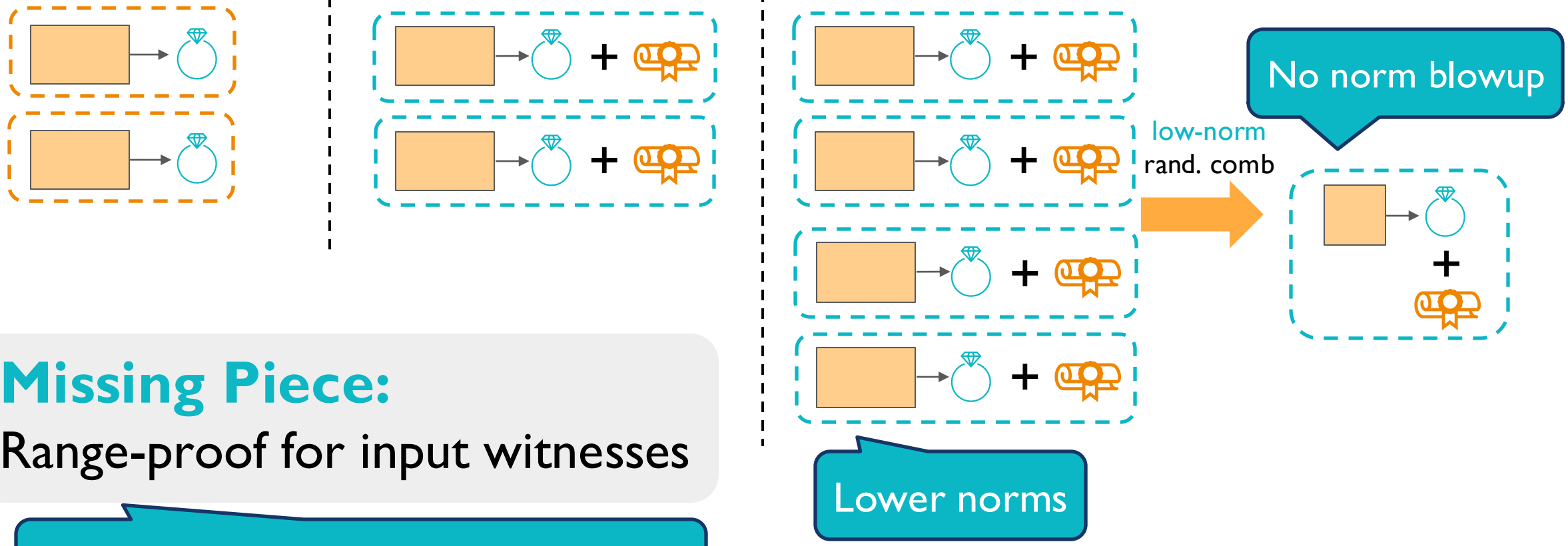


Missing Piece:

Range-proof for input witnesses

A New Lattice-Folding Framework

Step 1: Commit | Step 2: Linearize | Step 3: Decomp | Step 4: Fold



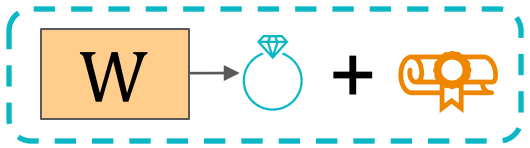
Missing Piece:

Range-proof for input witnesses

Challenge: Sublinear verification

Rephrasing Range Statements

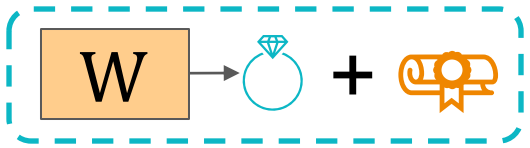
Range statement:



where each entry of W is in $[0, b)$

Rephrasing Range Statements

Range statement:



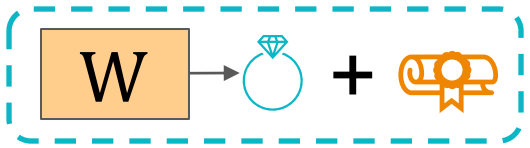
b is small

where each entry of W is in $[0, b)$

Rephrasing Range Statements

Range statement:

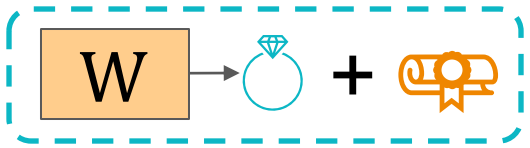
Generalize to $(-b, b)$



where each entry of W is in $[0, b)$

Rephrasing Range Statements

Range statement:



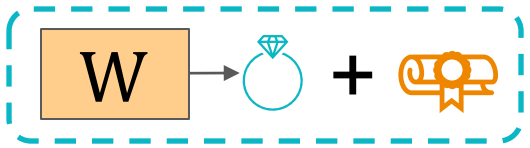
where each entry of W is in $[0, b)$

Observation:

$$\mathrm{W}_{i,j} \in [0, b) \cap \mathbb{Z} \iff F(i,j) := \prod_{t=[0,b)} \left(\mathrm{W}_{i,j} - t\right) = 0 \bmod q$$

Rephrasing Range Statements

Range statement:



where each entry of W is in $[0, b)$

Observation:

Degree-b equation

$$\mathrm{W}_{i,j} \in [0, b) \cap \mathbb{Z} \iff F(i,j) := \prod_{t=[0,b)} \left(\mathrm{W}_{i,j} - t\right) = 0 \bmod q$$

Range Proof from Sumcheck

Observation:

$$\forall (i,j) \in \mathcal{D}: \quad F(i,j) = 0$$

[Set'20, CBBZ'23]

Sumcheck [LFKN'92]

Random

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

Range Proof from Sumcheck

Observation:

$$\forall (i,j) \in \mathcal{D}: \quad F(i,j) = 0$$

[Set'20, CBBZ'23]

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

Sumcheck redcution:

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

Range Proof from Sumcheck

Observation:

$\forall (i,j) \in \mathcal{D}$:
$F(i,j) = 0$

[Set'20, CBBZ'23]

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

Sumcheck redcution:

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

[LFKN'92]

Linear evaluation

$$\widetilde{W}(\vec{\gamma}) = v$$

Range Proof from Sumcheck

Observation:

$\forall (i,j) \in \mathcal{D}$:
$F(i,j) = 0$

[Set'20, CBBZ'23]

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

Sumcheck redcution:

$$\sum_{(i,j)\in\mathcal{D}} F(i,j) \cdot eq\big((i,j),(r,r')\big) = 0$$

[LFKN'92]

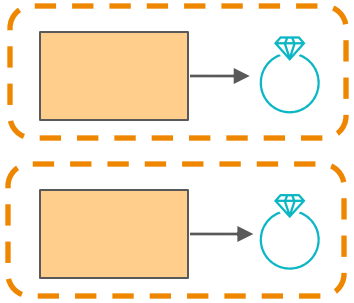
$\widetilde{W}(\vec{\gamma}) = v$

Linear evaluation

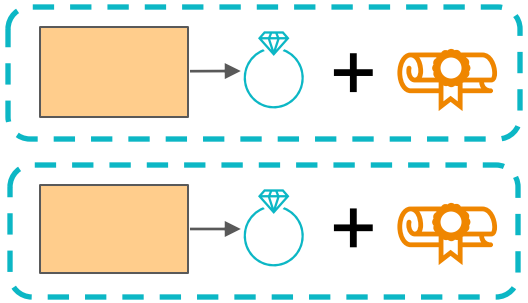
Folding-friendly

Final Scheme

Step 1: Commit

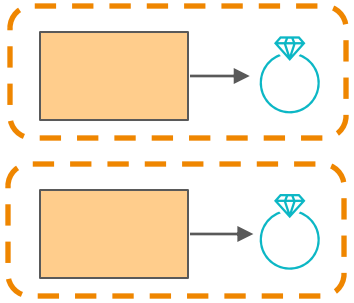


Step 2: Linearize

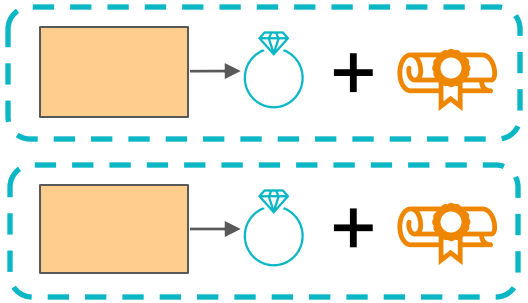


Final Scheme

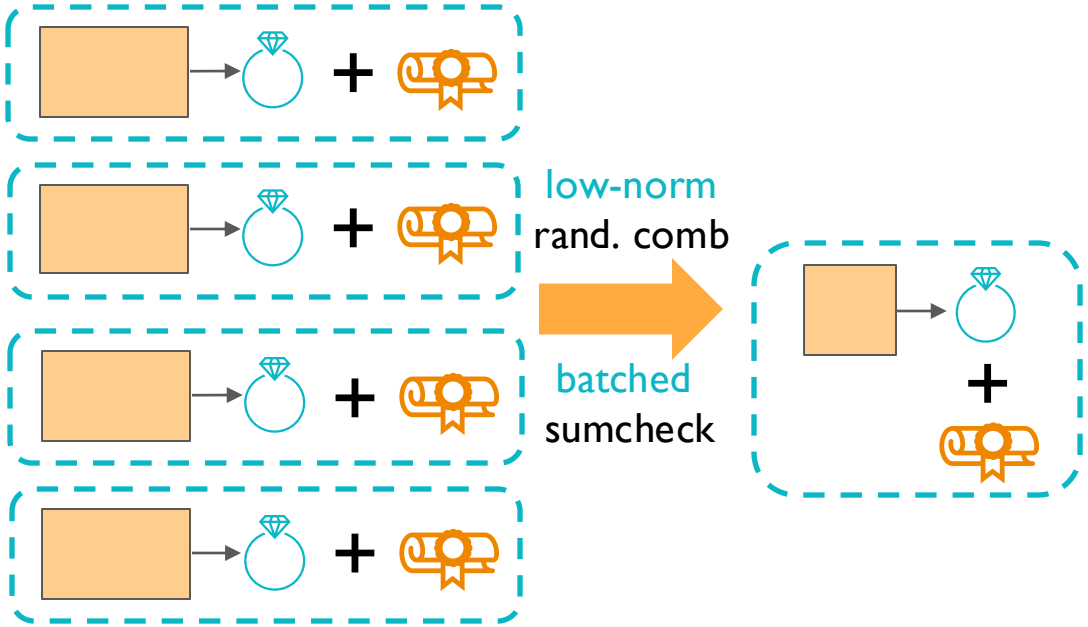
Step 1: Commit



Step 2: Linearize

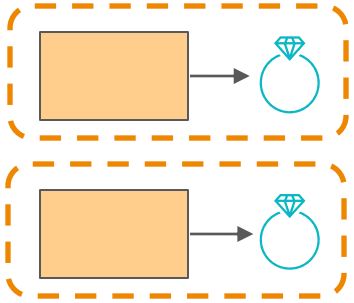


(Combined) Step 3:
Decomp + Range-chk + Fold

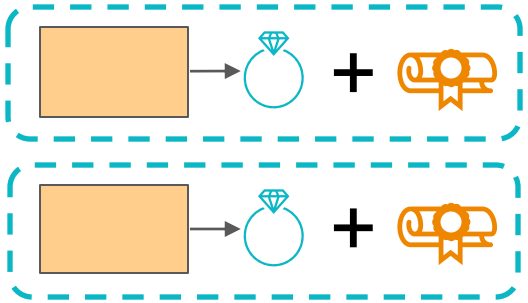


Final Scheme

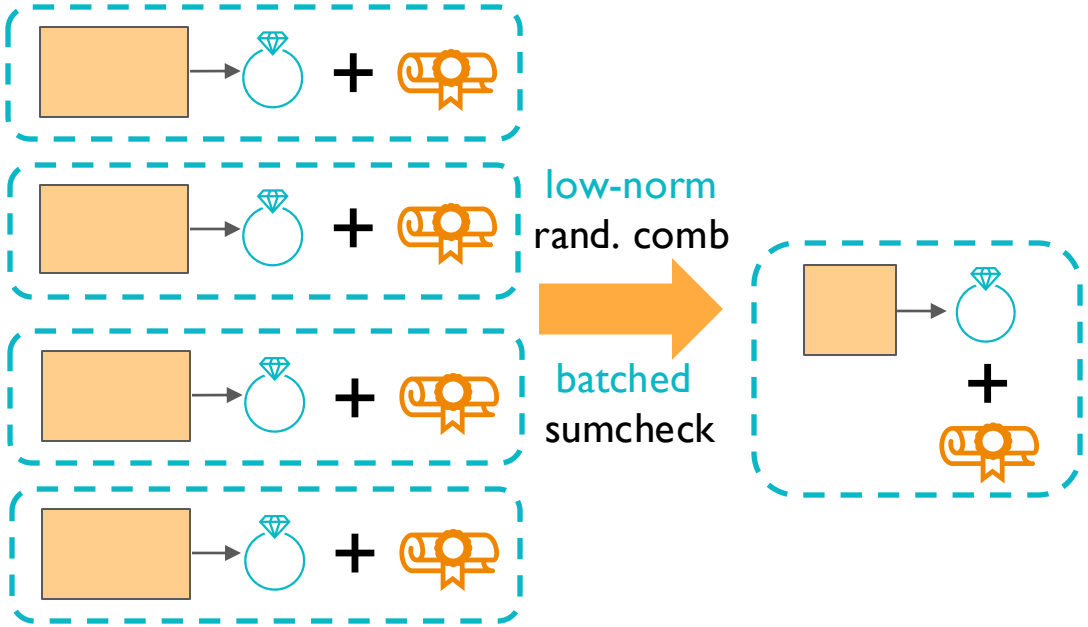
Step 1: Commit



Step 2: Linearize



(Combined) Step 3:
Decomp + Range-chk + Fold



Folding Verifier:

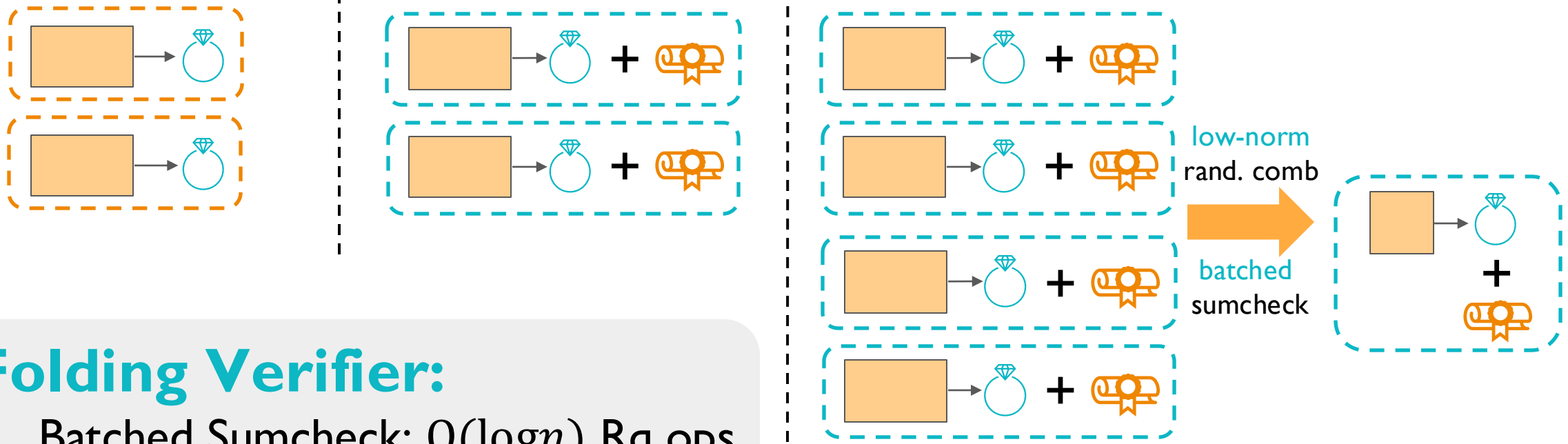
- Batched Sumcheck: $O(\log n)$ Rq ops
- Rand. comb: $O(\lambda)$ field ops

Final Scheme

Step 1: Commit

Step 2: Linearize

(Combined) Step 3:
Decomp + Range-chk + Fold



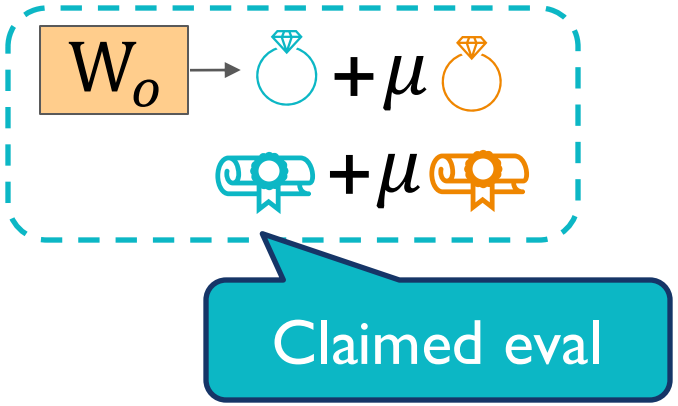
Folding Verifier:

- Batched Sumcheck: $O(\log n)$ Rq ops
- Rand. comb: $O(\lambda)$ field ops

[BC'25, NS'25]: Replace ring w/ field ops

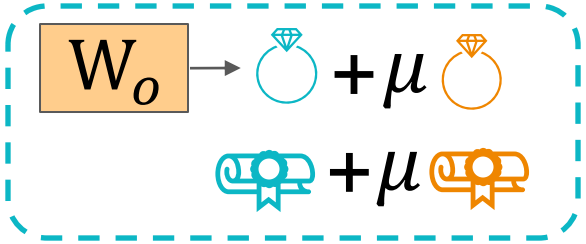
Knowledge Extraction

1st folded instance-witness pair:



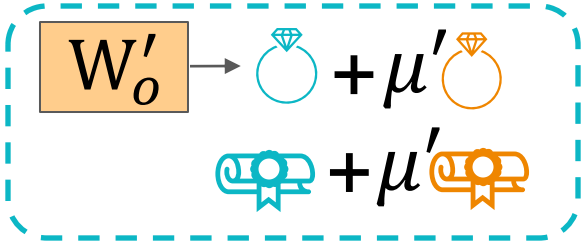
Knowledge Extraction

1st folded instance-witness pair:



Rewind*

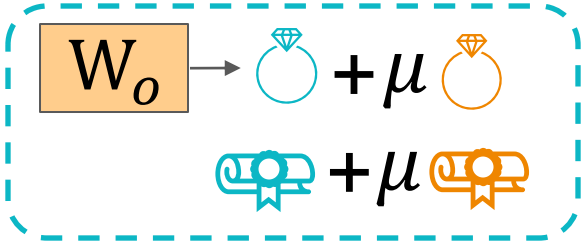
2nd folded instance-witness pair:



*: We actually rewind more than two times.

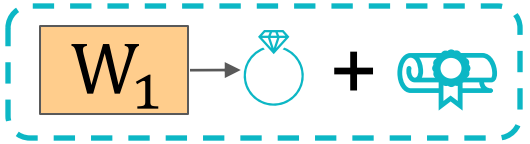
Knowledge Extraction

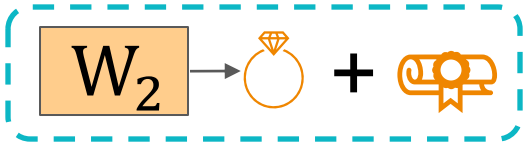
1st folded instance-witness pair:



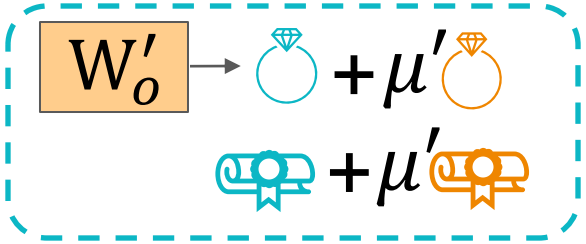
Rewind*

Interpolate





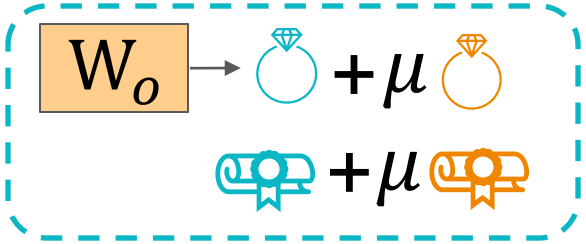
2nd folded instance-witness pair:



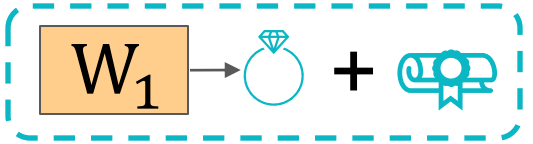
*: We actually rewind more than two times.

Knowledge Extraction

1st folded instance-witness pair:

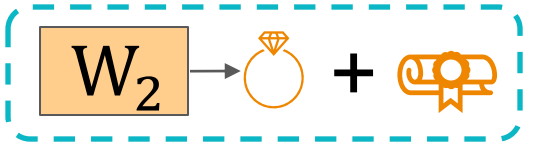


Relaxed binding

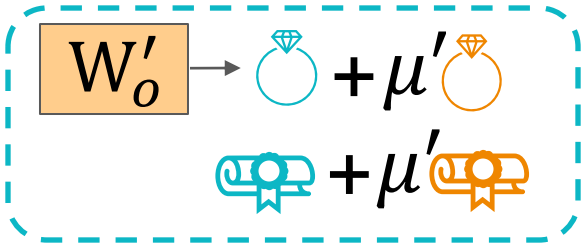


Rewind*

Interpolate



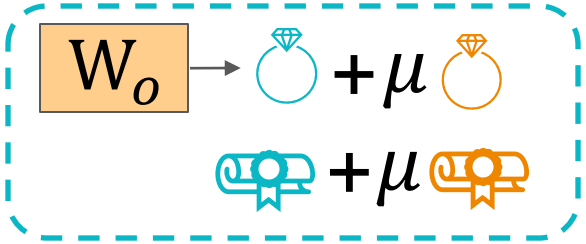
2nd folded instance-witness pair:



*: We actually rewind more than two times.

Knowledge Extraction

1st folded instance-witness pair:

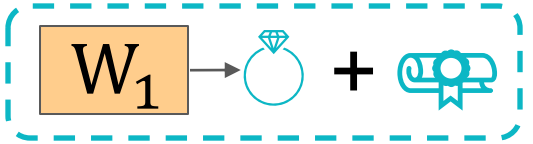


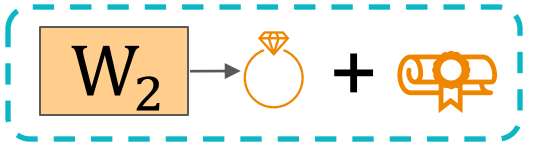
Rewind*

Interpolate

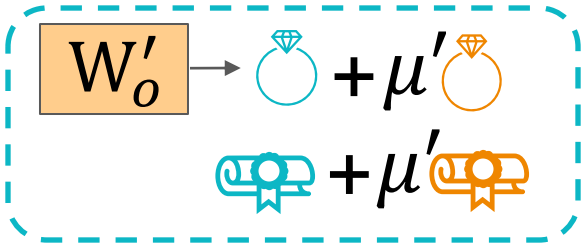
Relaxed binding

Satisfies eval claim





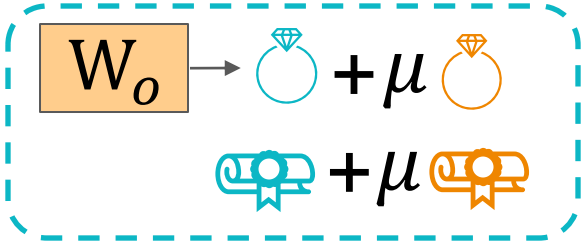
2nd folded instance-witness pair:



*: We actually rewind more than two times.

Knowledge Extraction

1st folded instance-witness pair:

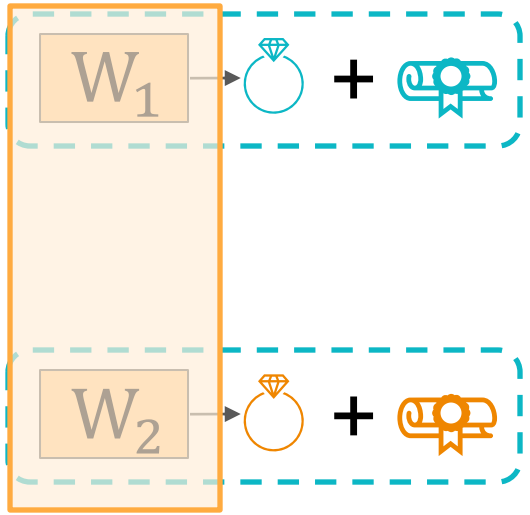


Rewind*

Interpolate

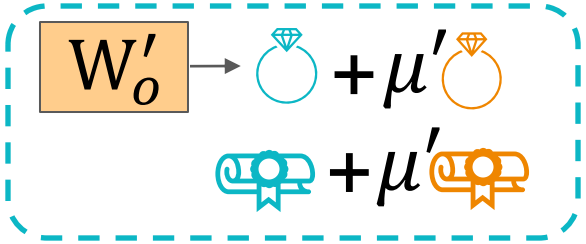
Relaxed binding

Satisfies eval claim



Low-norm by
Sumcheck soundness

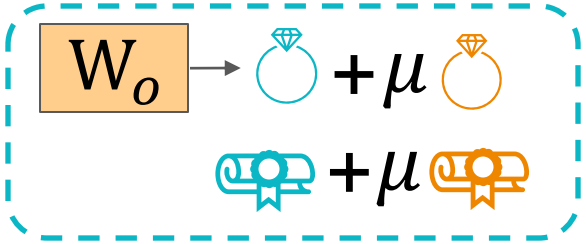
2nd folded instance-witness pair:



*: We actually rewind more than two times.

Knowledge Extraction

1st folded instance-witness pair:

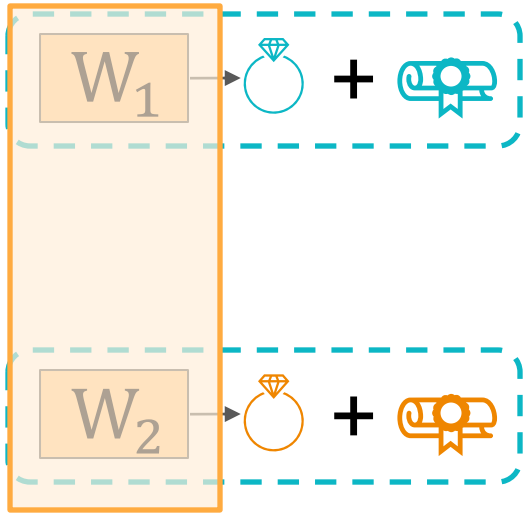


Rewind*

Interpolate

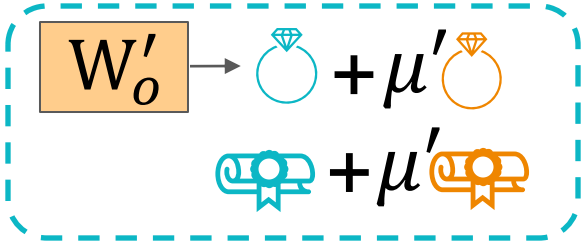
Relaxed binding

Satisfies eval claim



Low-norm by
Sumcheck soundness

2nd folded instance-witness pair:



More subtleties on supporting witness vectors over R_q

*: We actually rewind more than two times.

Summary & Future Work

Summary & Extensions

- First plausibly PQ-secure folding scheme from Module-SIS
- Extensions: Small modulus & high-degree constraint system

Summary & Future Work

Summary & Extensions

- First plausibly PQ-secure folding scheme from Module-SIS
- Extensions: Small modulus & high-degree constraint system

Follow-up

- [BC'25, C'25]: Efficient range-proofs w/o decomposition
- [NS'25]: Pay-per-bit commitment + better support to small fields
- [FKNP'24]: Security from SIS

Summary & Future Work

Summary & Extensions

- First plausibly PQ-secure folding scheme from Module-SIS
- Extensions: Small modulus & high-degree constraint system

Follow-up

- [BC'25, C'25]: Efficient range-proofs w/o decomposition
- [NS'25]: Pay-per-bit commitment + better support to small fields
- [FKNP'24]: Security from SIS

Future work

QROM Security Analysis

THANK YOU

Eprint 2024/257