Symphony

Scalable SNARKs from High-Arity Lattice Folding Schemes

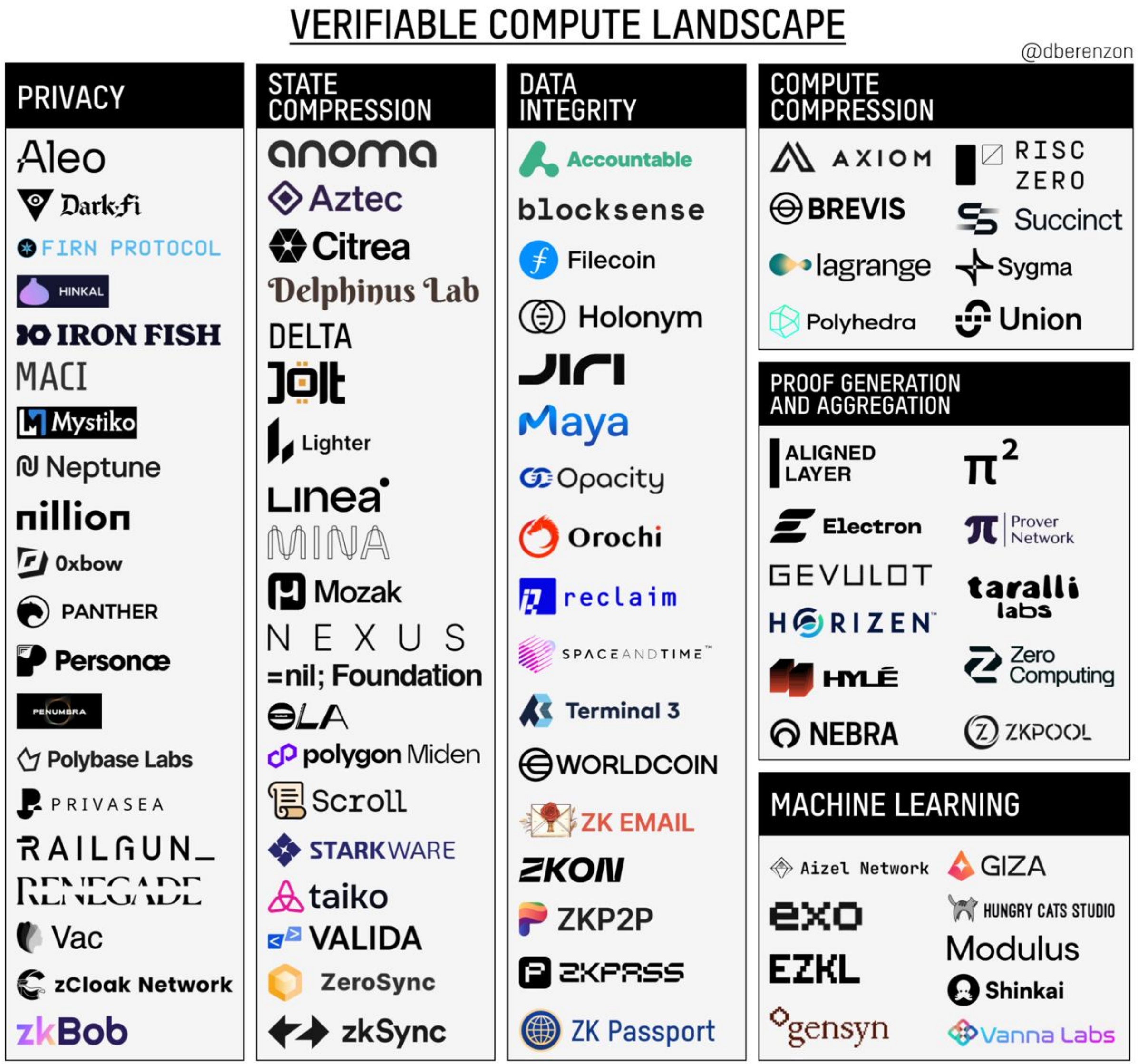
Binyi Chen

Stanford University

ZK-SNARKs: Advanced Applications

Applications

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



VERIFIABLE COMPUTE LANDSCAPE

@dberenzon

PRIVACY

Aleo, Dark.fi, FIRN PROTOCOL, HINKAL, IRON FISH, MACI, Mystiko, Neptune, nillion, 0xbow, PANTHER, Personae, PENUMBRA, Polybase Labs, PRIVASEA, RAILGUN, RENEGADE, Vac, zCloak Network, zkBob

STATE COMPRESSION

anoma, Aztec, Citrea, Delphinus Lab, DELTA, Jolt, Lighter, Linea, MINA, Mozak, NEXUS, =nil; Foundation, OLA, polygon Miden, Scroll, STARKWARE, taiko, VALIDA, ZeroSync, zkSync

DATA INTEGRITY

Accountable, blocksense, Filecoin, Holonym, JIRI, Maya, Opacity, Orochi, reclaim, SPACEANDTIME, Terminal 3, WORLDCOIN, ZK EMAIL, ZKON, ZKP2P, ZKPASS, ZK Passport

COMPUTE COMPRESSION

AXIOM, RISC ZERO, BREVIS, Succinct, lagrange, Sygma, Polyhedra, Union

PROOF GENERATION AND AGGREGATION

ALIGNED LAYER, π², Electron, Prover Network, GEVULOT, taralli labs, HORIZEN, Zero Computing, HYLE, NEBRA, ZKPOOL

MACHINE LEARNING

Aizel Network, GIZA, exo, HUNGRY CATS STUDIO, EZKL, Modulus, Shinkai, gensyn, Vanna Labs

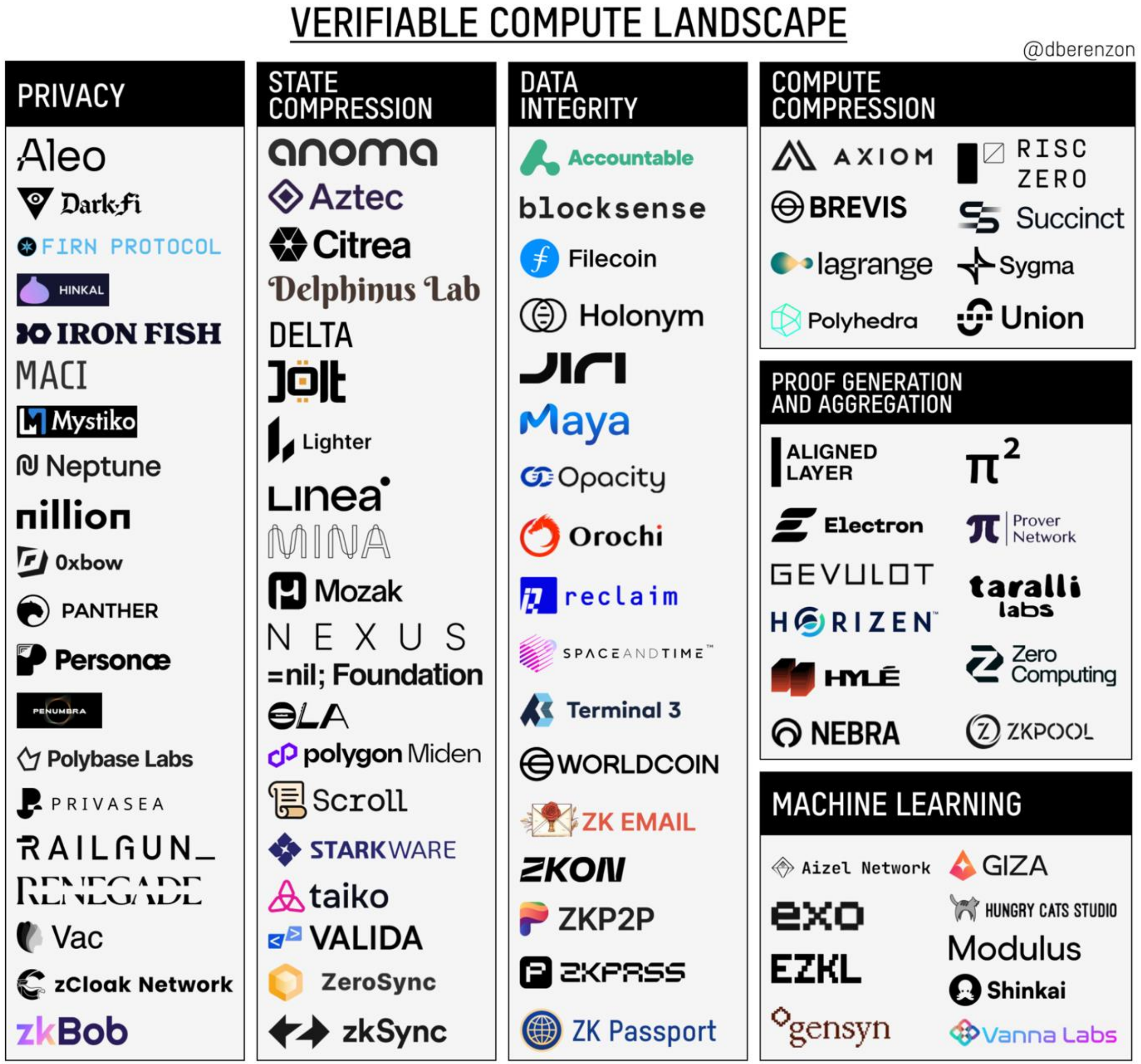
ZK-SNARKs: Advanced Applications

Applications

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

Common Theme

Large-scale computation



VERIFIABLE COMPUTE LANDSCAPE

@dberenzon

PRIVACY

- Aleo
- Dark.fi
- FIRN PROTOCOL
- HINKAL
- IRON FISH
- MACI
- Mystiko
- Neptune
- nillion
- 0xbow
- PANTHER
- Personae
- PENUMBRA
- Polybase Labs
- PRIVASEA
- RAILGUN
- RENEGADE
- Vac
- zCloak Network
- zkBob

STATE COMPRESSION

- anoma
- Aztec
- Citrea
- Delphinus Lab
- DELTA
- Jolt
- Lighter
- Linea
- MINA
- Mozak
- NEXUS
- =nil; Foundation
- OLA
- polygon Miden
- Scroll
- STARKWARE
- taiko
- VALIDA
- ZeroSync
- zkSync

DATA INTEGRITY

- Accountable
- blocksense
- Filecoin
- Holonym
- JIRI
- Maya
- Opacity
- Orochi
- reclaim
- SPACEANDTIME
- Terminal 3
- WORLDCOIN
- ZK EMAIL
- ZKON
- ZKP2P
- ZKPASS
- ZK Passport

COMPUTE COMPRESSION

- AXIOM
- RISC ZERO
- BREVIS
- Succinct
- lagrange
- Sygma
- Polyhedra
- Union

PROOF GENERATION AND AGGREGATION

- ALIGNED LAYER
- π^2
- Electron
- Prover Network
- GEVULOT
- taralli labs
- HORIZEN
- Zero Computing
- HYLE
- NEBRA
- ZKPOOL

MACHINE LEARNING

- Aizel Network
- GIZA
- exo
- HUNGRY CATS STUDIO
- EZKL
- Modulus
- gensyn
- Shinkai
- Vanna Labs

ZK-SNARKs: Advanced Applications

Applications

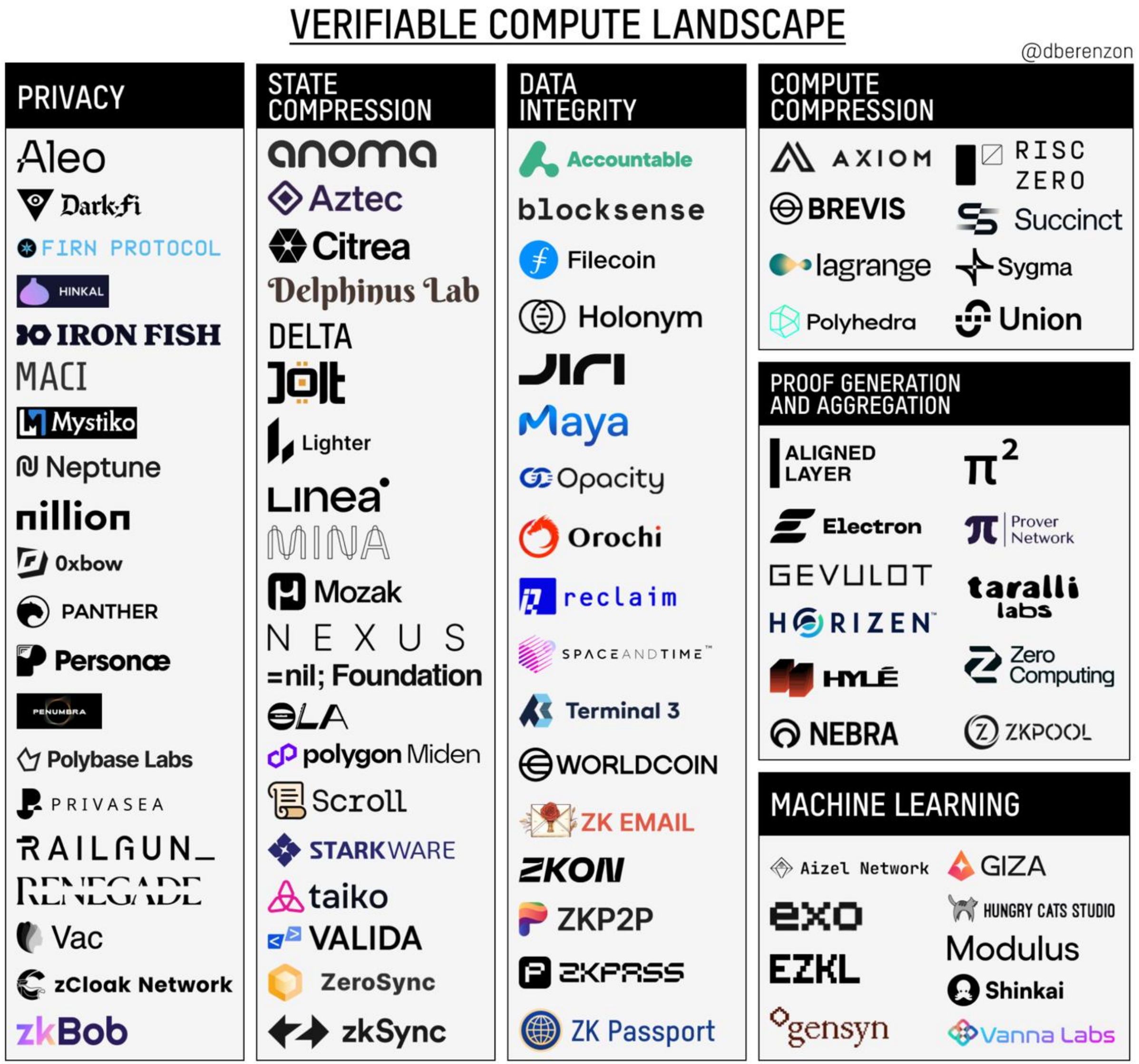
- zkVM/zkML
- Image/Video Provenance
- ZK Wallet/Passport
- Data Availability
- Decentralized Storage
- ……

Common Theme

Large-scale computation

Design Requirements

Speed + Memory + Streaming

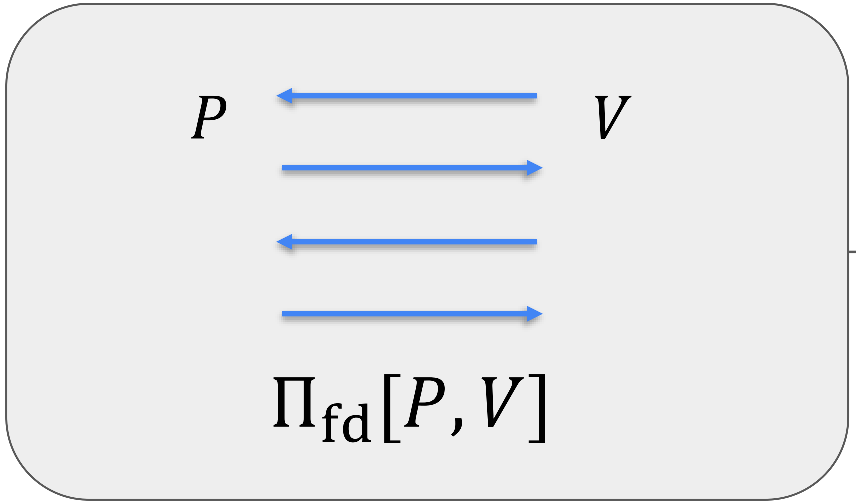


Folding Schemes [BGH'19, BCLMS'20, KST'21]

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

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

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



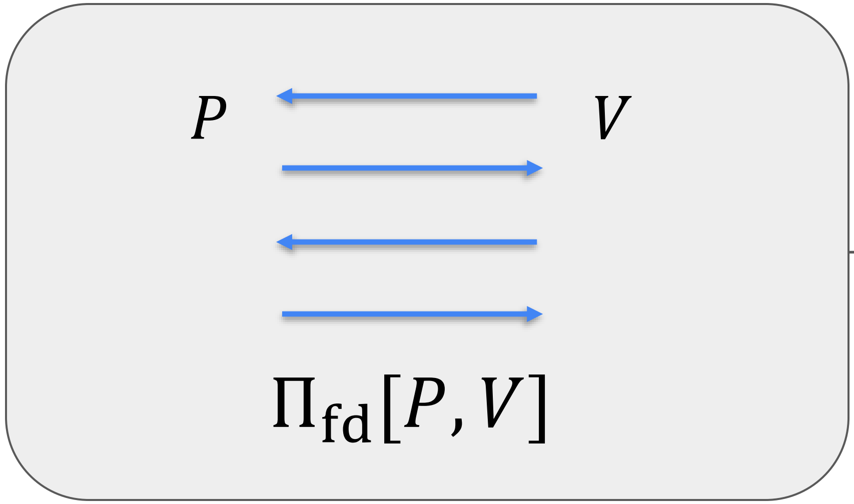
$(x, w) \in_? R_{acc}$

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

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

$(\mathrm{x}_1, \mathrm{w}_1) \in_? \mathrm{R_{NP}}$

$(\mathrm{x}_2, \mathrm{w}_2) \in_? \mathrm{R_{acc}}$



P V

$\Pi_{\mathrm{fd}}[P, V]$

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

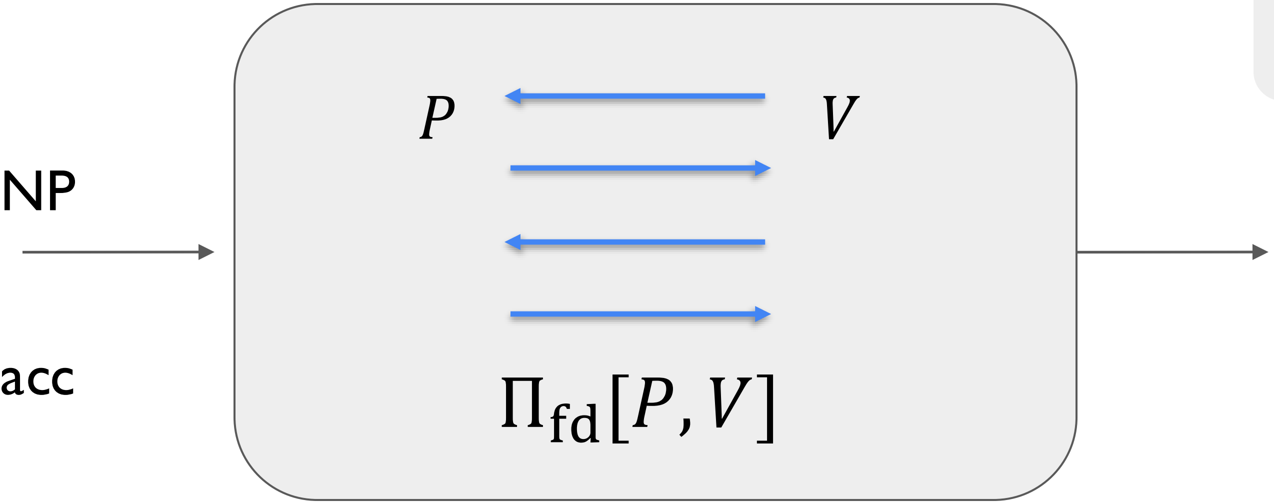
Non-Interactive Version:
$\mathrm{FS}^{\mathrm{H}}(\Pi_{\mathrm{fd}}[P, V])$

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

$|\mathsf{R}_{\mathsf{NP}}| \approx |\mathsf{R}_{\mathsf{acc}}|$

$(\mathsf{x}_1, \mathsf{w}_1) \in_? \mathsf{R}_{\mathsf{NP}}$

$(\mathsf{x}_2, \mathsf{w}_2) \in_? \mathsf{R}_{\mathsf{acc}}$



$(\mathsf{x}, \mathsf{w}) \in_? \mathsf{R}_{\mathsf{acc}}$

Non-Succinct

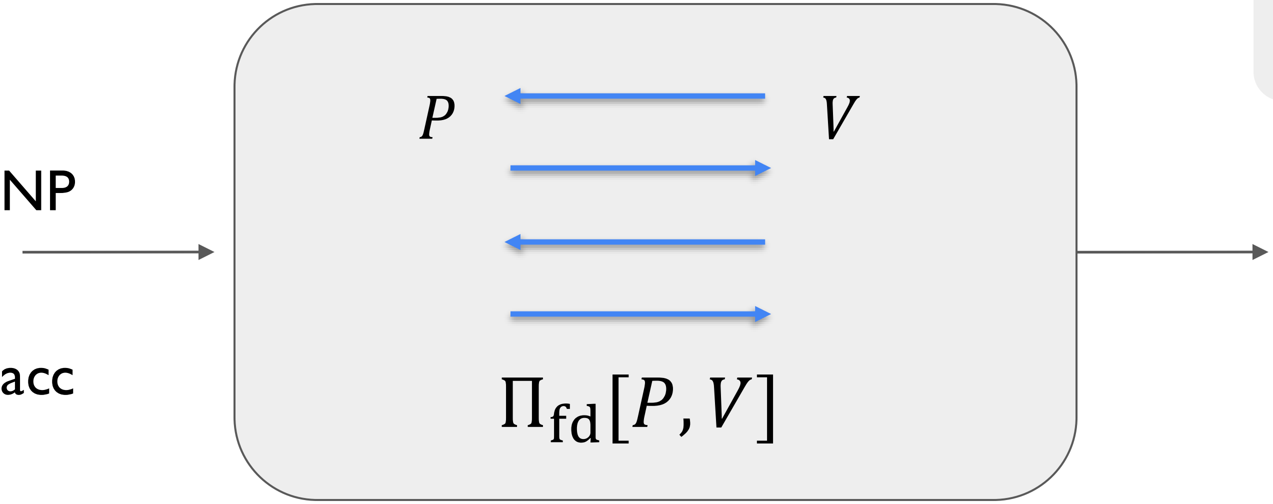
Non-Interactive Version:
$\mathrm{FS}^{\mathrm{H}}(\Pi_{\mathrm{fd}}[P, V])$

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

$|\mathrm{R}_{\mathrm{NP}}| \approx |\mathrm{R}_{\mathrm{acc}}|$

$(\mathrm{x}_1, \mathrm{w}_1) \in_? \mathrm{R}_{\mathrm{NP}}$

$(\mathrm{x}_2, \mathrm{w}_2) \in_? \mathrm{R}_{\mathrm{acc}}$



$(\mathrm{x}, \mathrm{w}) \in_? \mathrm{R}_{\mathrm{acc}}$

Non-Succinct

Advantages:

- Much faster than SNARKs
- Boost SNARK efficiency

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

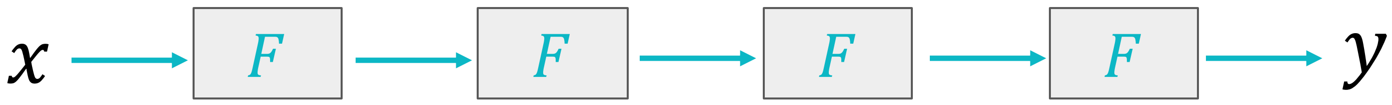
Recursive Folding [Val’08, BGH’19, BCLMS’20, KST’21, NDCTB’24…]

Computation:

$$\begin{tikzcd} x \arrow[r] & \boxed{F} \arrow[r] & \boxed{F} \arrow[r] & \boxed{F} \arrow[r] & \boxed{F} \arrow[r] & y \end{tikzcd}$$

Recursive Folding [Val'08, BGH'19, BCLMS'20, KST'21, NDCTB'24…]

Computation:



Proving:

Acc stmnts 🟦

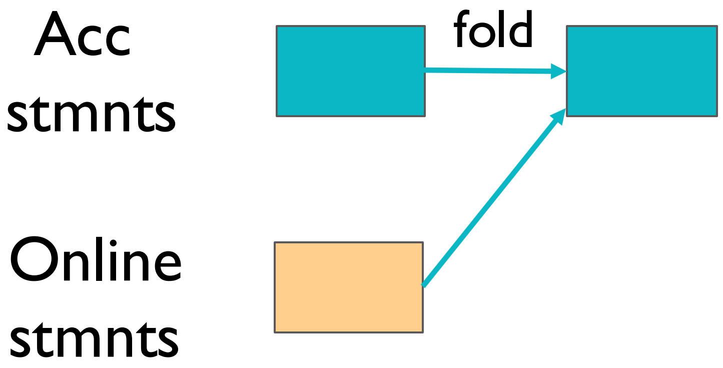
Online stmnts 🟧

Recursive Folding [Val'08, BGH'19, BCLMS'20, KST'21, NDCTB'24…]

Computation:

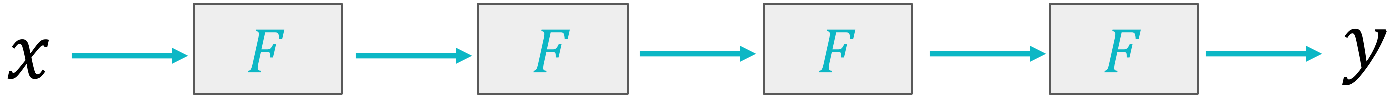
$$x \xrightarrow{} \boxed{F} \xrightarrow{} \boxed{F} \xrightarrow{} \boxed{F} \xrightarrow{} \boxed{F} \xrightarrow{} y$$

Proving:

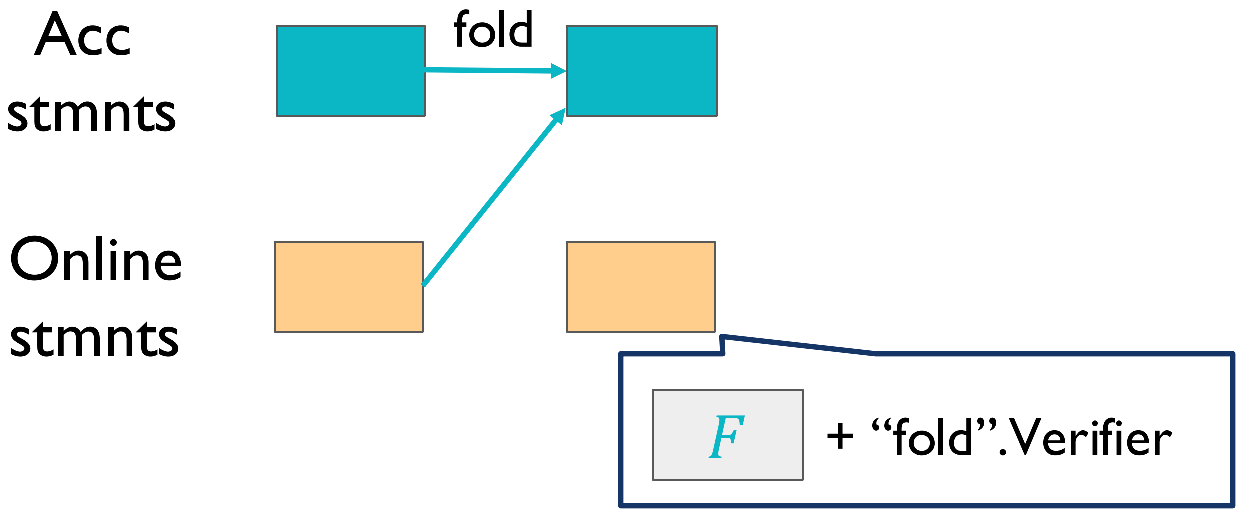


Recursive Folding [Val'08, BGH'19, BCLMS'20, KST'21, NDCTB'24…]

Computation:

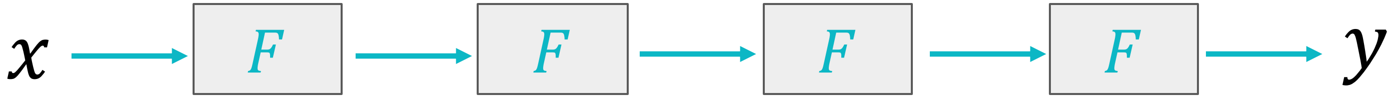


Proving:

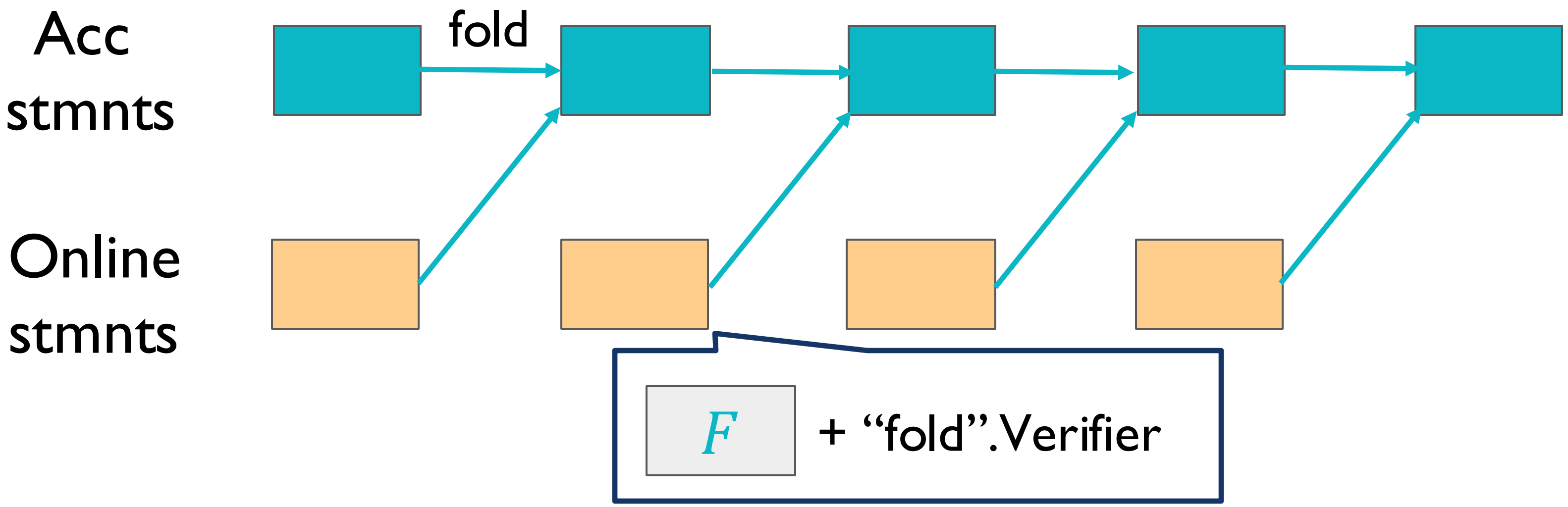


Recursive Folding [Val'08, BGH'19, BCLMS'20, KST'21, NDCTB'24…]

Computation:

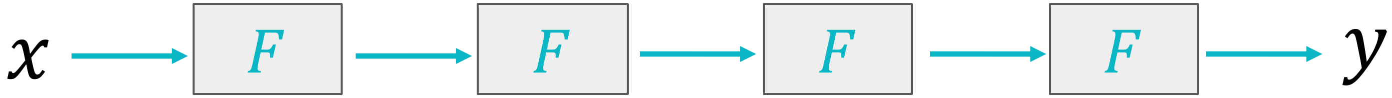


Proving:

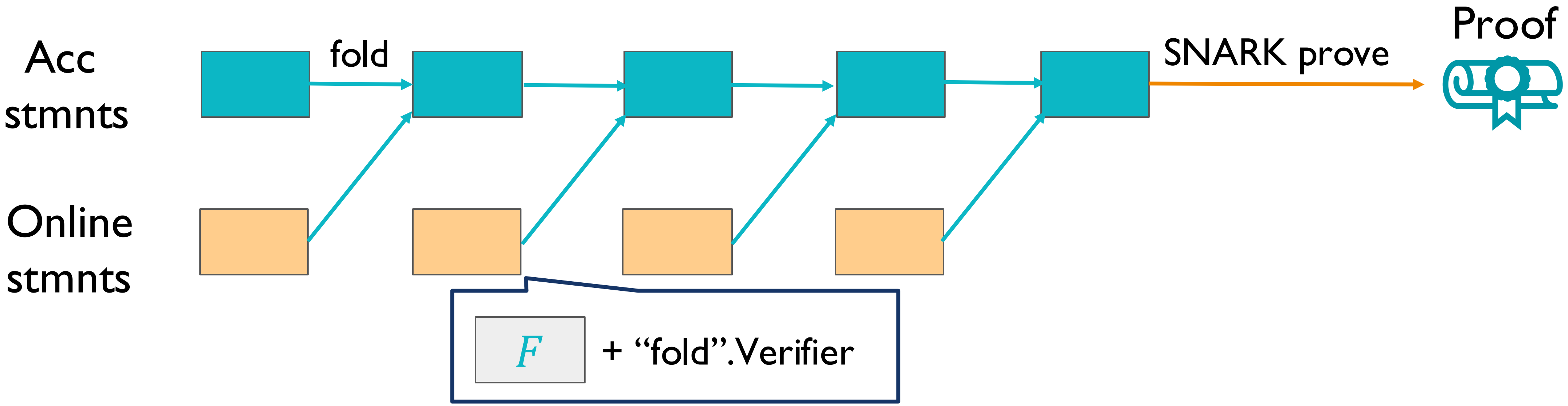


Recursive Folding [Val'08, BGH'19, BCLMS'20, KST'21, NDCTB'24…]

Computation:

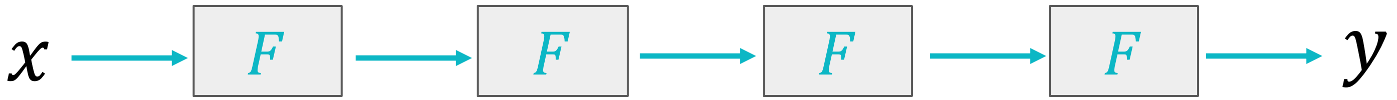


Proving:

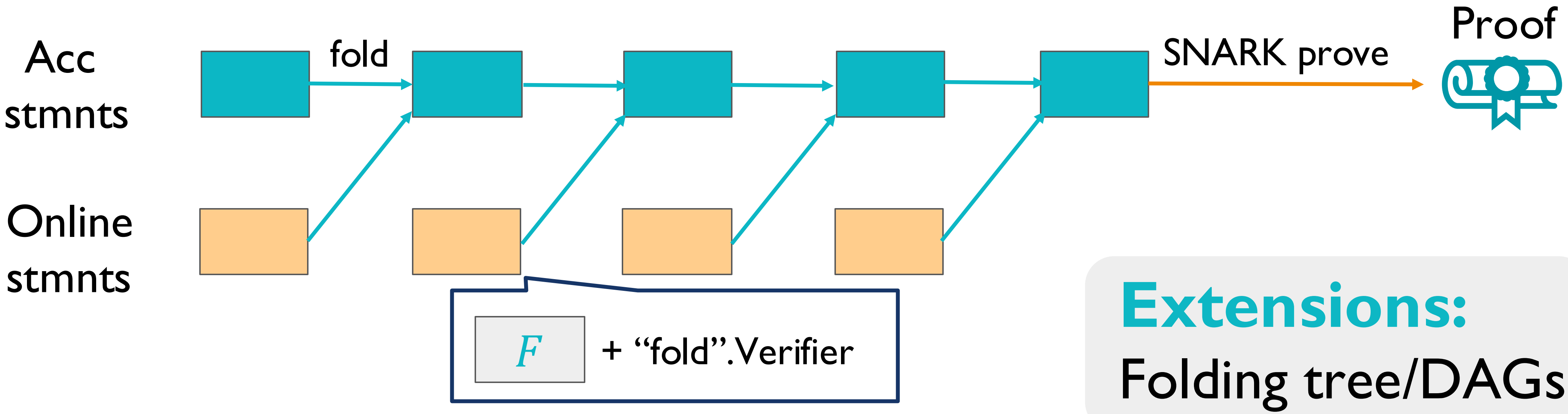


Recursive Folding [Val'08, BGH'19, BCLMS'20, KST'21, NDCTB'24…]

Computation:



Proving:



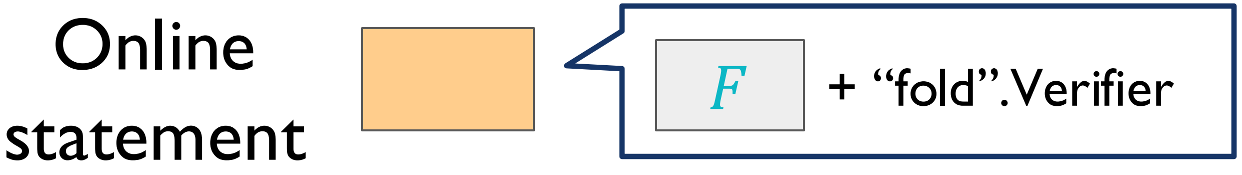
Extensions:

Folding tree/DAGs

Recursive Folding [Val’08, BGH’19, BCLMS’20, KST’21, NDCTB’24…]

Caveat: Embedding verifiers

Heuristics: Instantiating RO



Online statement

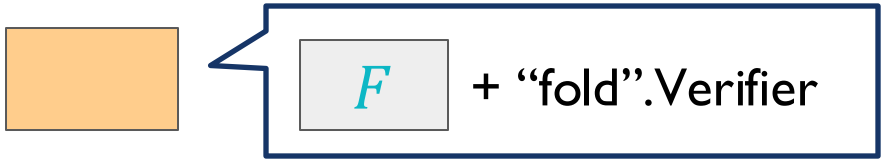
F + “fold”.Verifier

Recursive Folding [Val’08, BGH’19, BCLMS’20, KST’21, NDCTB’24…]

Caveat: Embedding verifiers

Heuristics: Instantiating RO

Online statement



F + “fold”.Verifier

Scientist

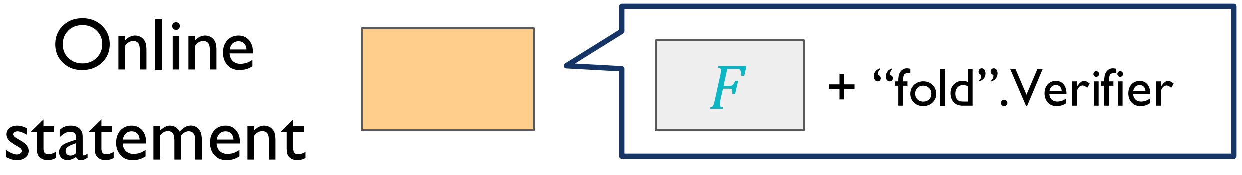
“Proving FS in recursive statement might be risky”

[KRS’25]

Recursive Folding [Val’08, BGH’19, BCLMS’20, KST’21, NDCTB’24…]

Caveat: Embedding verifiers

Heuristics: Instantiating RO



Engineer

“Proving hash is expensive and complex”

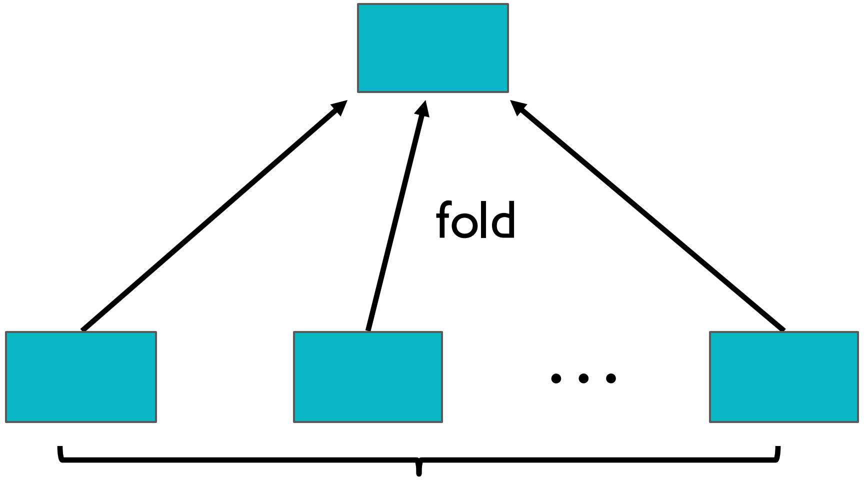
Scientist

“Proving FS in recursive statement might be risky”

[KRS’25]

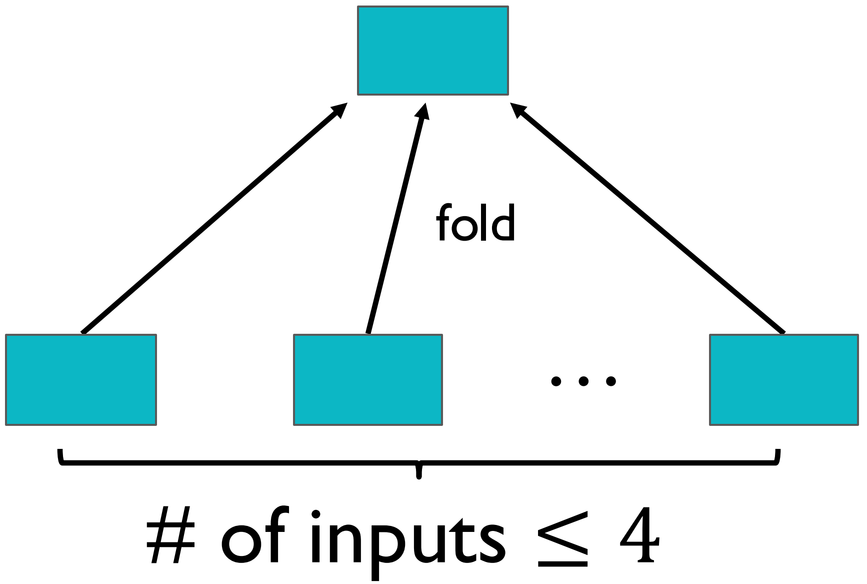
Q: Can we use folding schemes more efficiently and securely?

Typical Folding Schemes

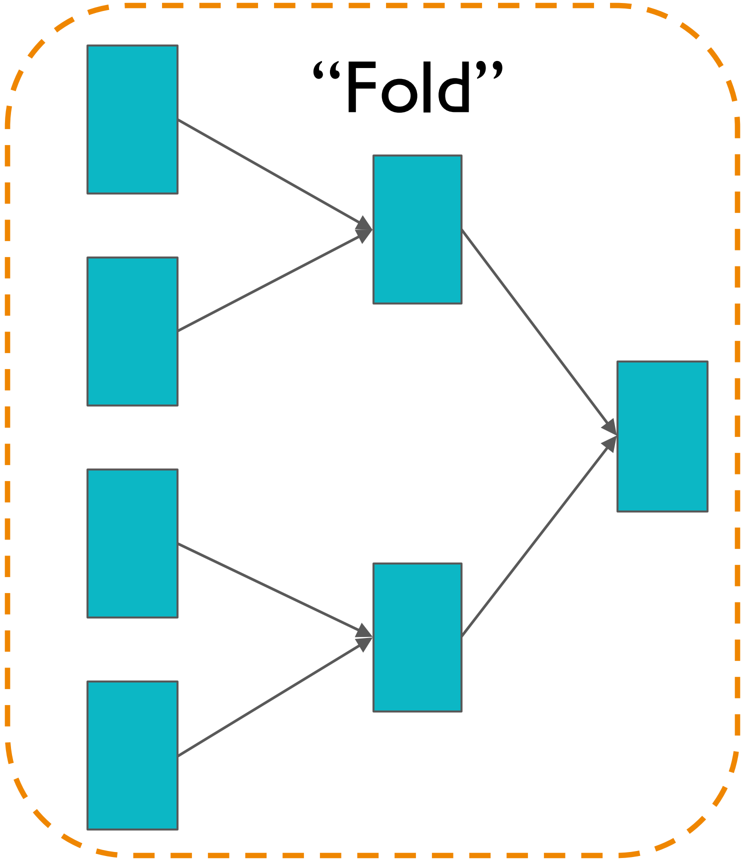


# of inputs ≤ 4

Typical Folding Schemes



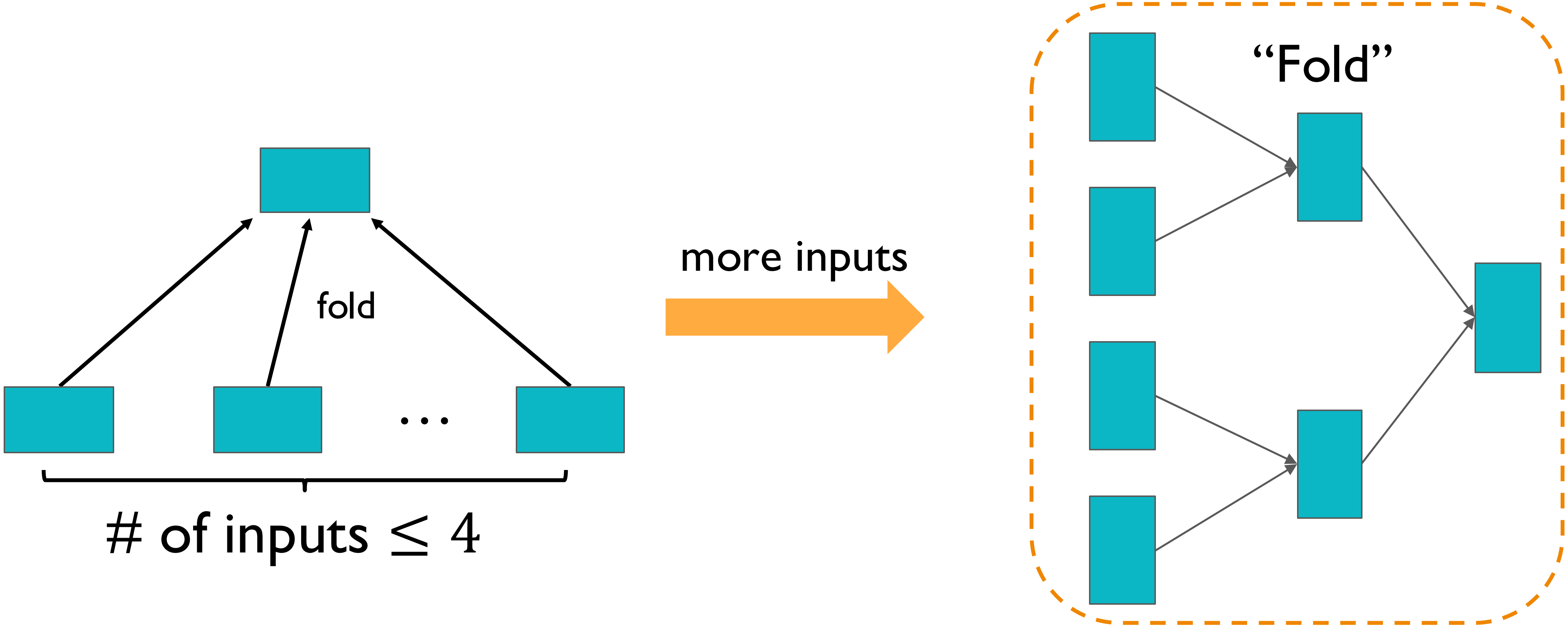
IVC/PCD Compiler



Deeper recursion + Higher latency

Typical Folding Schemes

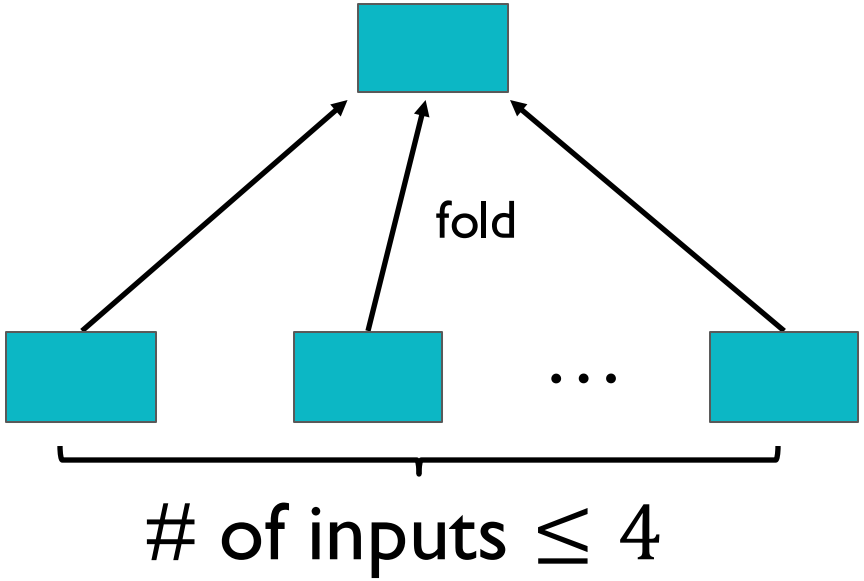
IVC/PCD Compiler



Deeper recursion + Higher latency

Q: Why not increasing folding arity?

Typical Folding Schemes

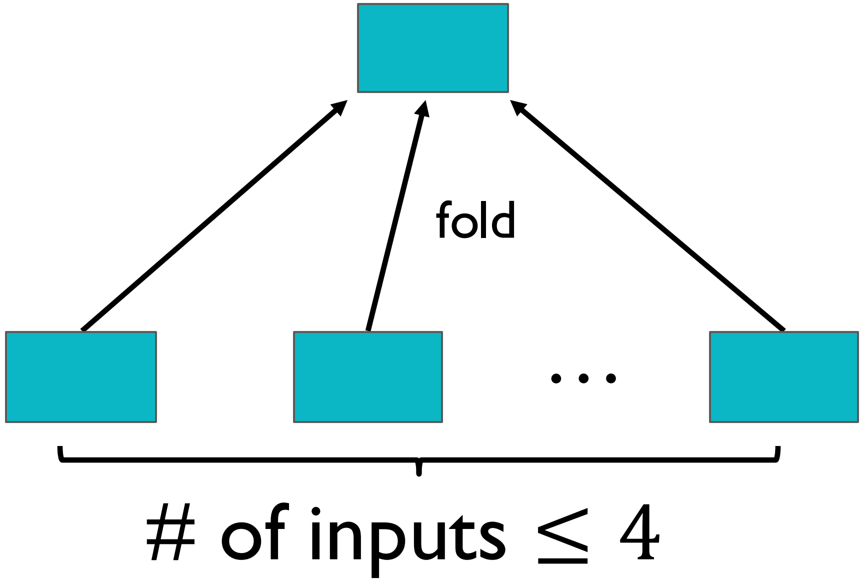


⚠ "fold".Verifier circuit size $\propto$ arity

- Lattice-based: ~100K for arity 2
- Hash-based: 1~10 millions

Q: Why not increasing folding arity?

Typical Folding Schemes



# of inputs ≤ 4

Hash computation is dominant

⚠ "fold".Verifier circuit size ∝ arity

- Lattice-based: ~100K for arity 2
- Hash-based: 1~10 millions

Q: Why not increasing folding arity?

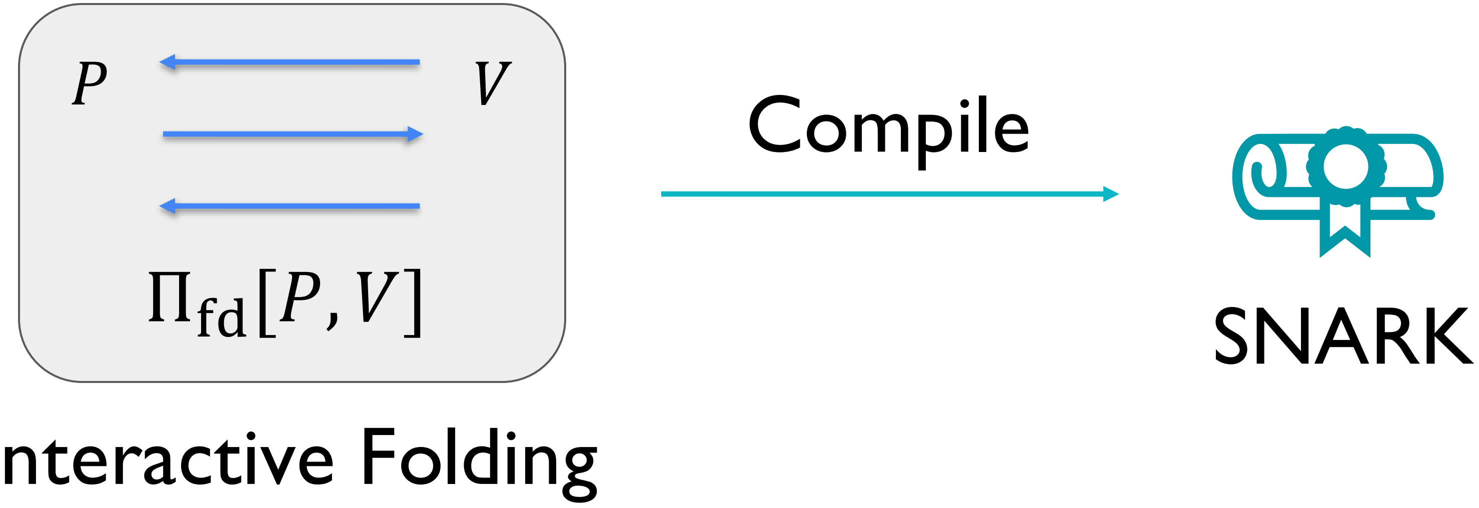
Q: What if we don't need to prove hash computations?

High-arity folding could be possible

Q: What if we don’t need to prove hash computations?

Our Contributions

A New Compiler:

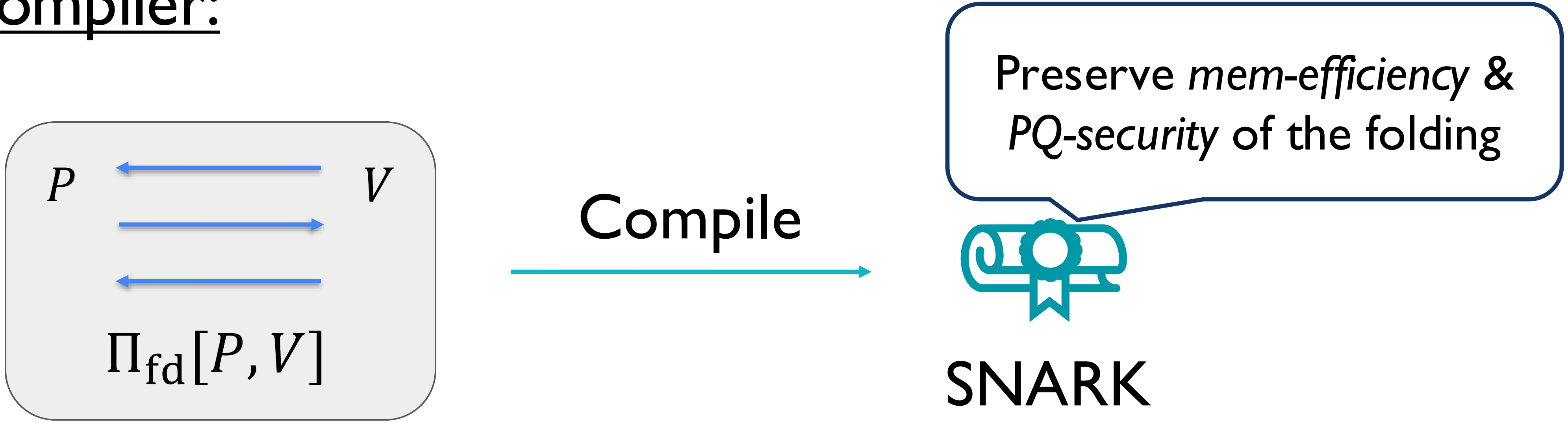


Advantages:

- No Fiat-Shamir circuit embedding
- Security in the random oracle model

Our Contributions

A New Compiler:



P V

$\Pi_{\mathrm{fd}}[P, V]$

Interactive Folding

Compile

Preserve *mem-efficiency* & *PQ-security* of the folding

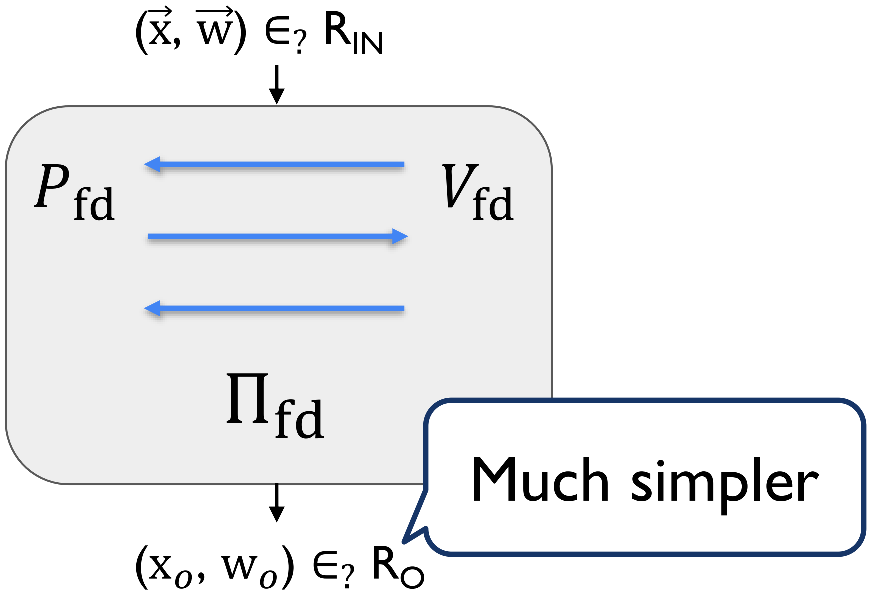
SNARK

Advantages:

- No Fiat-Shamir circuit embedding
- Security in the random oracle model

Folding Schemes to SNARKs

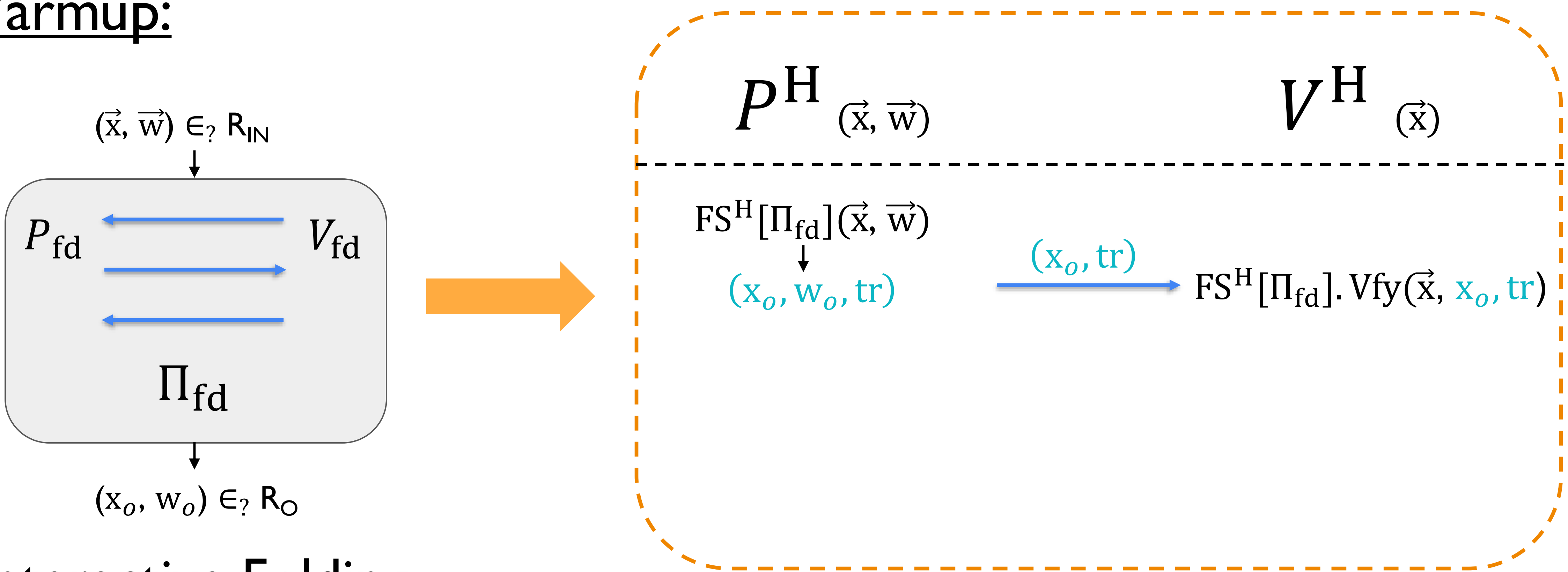
Warmup:



Interactive Folding

Folding Schemes to SNARKs

Warmup:



$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$

P_{fd}

V_{fd}

Π_{fd}

$(\mathrm{x}_o, \mathrm{w}_o) \in_? \mathrm{R_O}$

$P^{\mathrm{H}}\ (\vec{x}, \vec{w})$

$V^{\mathrm{H}}\ (\vec{x})$

$\mathrm{FS}^{\mathrm{H}}[\Pi_{\mathrm{fd}}](\vec{x}, \vec{w})$

$\downarrow$

$(\mathrm{x}_o, \mathrm{w}_o, \mathrm{tr})$

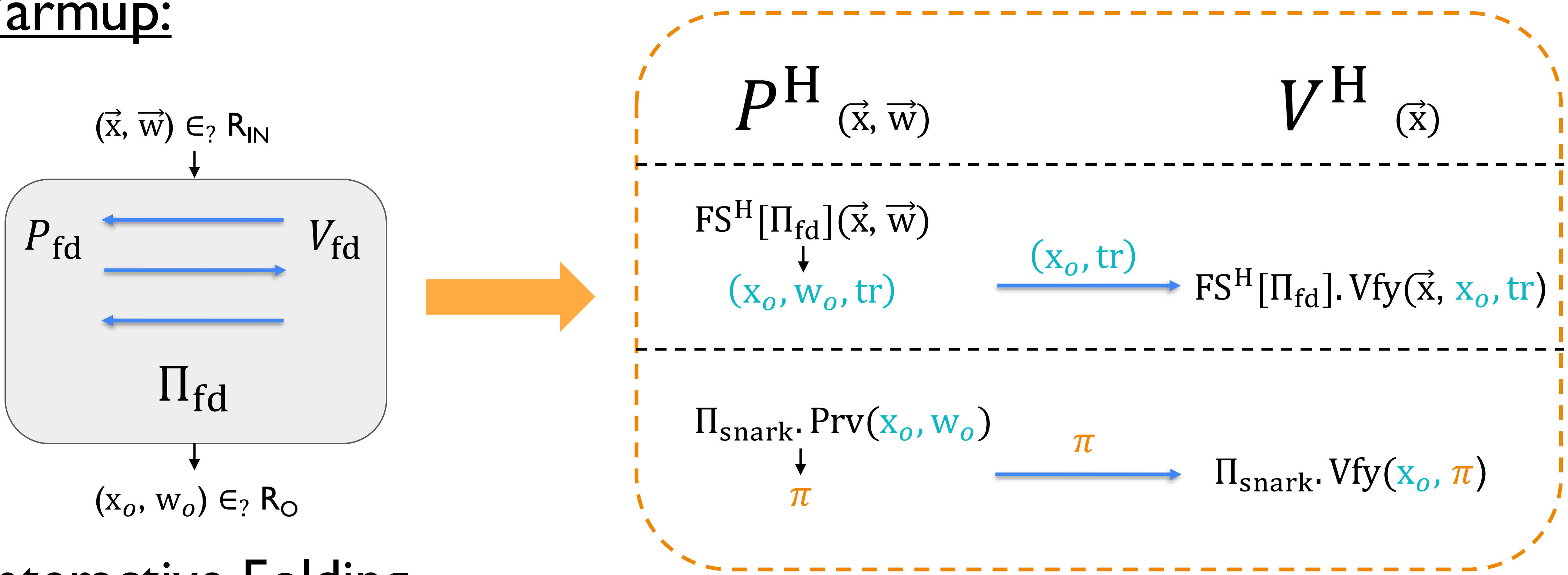
$\xrightarrow{(\mathrm{x}_o, \mathrm{tr})}$

$\mathrm{FS}^{\mathrm{H}}[\Pi_{\mathrm{fd}}].\mathrm{Vfy}(\vec{x}, \mathrm{x}_o, \mathrm{tr})$

Interactive Folding

Folding Schemes to SNARKs

Warmup:



$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$

P_{fd} V_{fd}

Π_{fd}

$(\mathrm{x}_o, \mathrm{w}_o) \in_? \mathrm{R_O}$

$P^{\mathrm{H}}(\vec{x}, \vec{w})$ $V^{\mathrm{H}}(\vec{x})$

$\mathrm{FS^H}[\Pi_{\mathrm{fd}}](\vec{x}, \vec{w})$

$(\mathrm{x}_o, \mathrm{w}_o, \mathrm{tr})$

$\xrightarrow{(\mathrm{x}_o, \mathrm{tr})}$

$\mathrm{FS^H}[\Pi_{\mathrm{fd}}].\mathrm{Vfy}(\vec{x}, \mathrm{x}_o, \mathrm{tr})$

$\Pi_{\mathrm{snark}}.\mathrm{Prv}(\mathrm{x}_o, \mathrm{w}_o)$

π

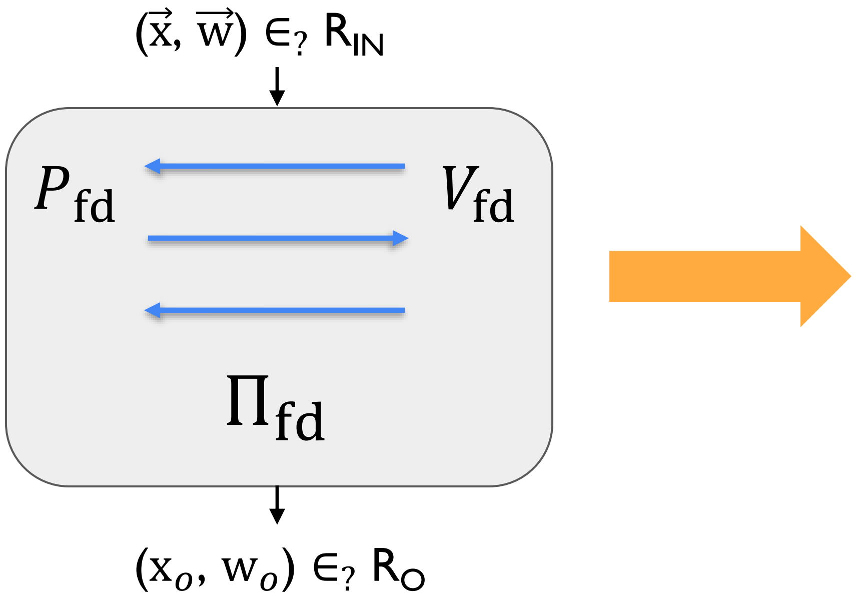
$\xrightarrow{\pi}$

$\Pi_{\mathrm{snark}}.\mathrm{Vfy}(\mathrm{x}_o, \pi)$

Interactive Folding

Folding Schemes to SNARKs

Warmup:



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

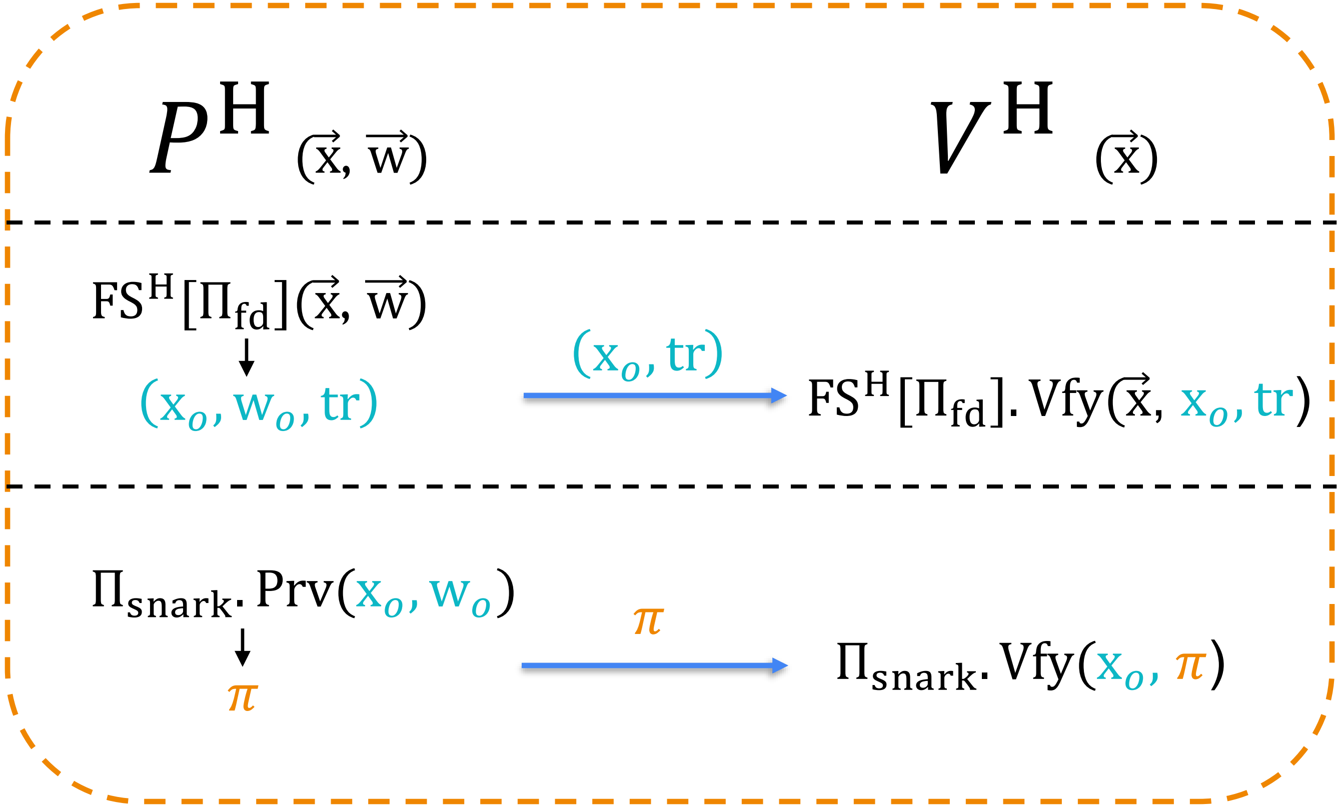
P_{fd}

V_{fd}

Π_{fd}

$(x_o, w_o) \in_? R_O$

Interactive Folding



P^H $(\vec{x}, \vec{w})$

V^H $(\vec{x})$

$\mathrm{FS}^H[\Pi_{fd}](\vec{x}, \vec{w})$

(x_o, w_o, tr)

(x_o, tr)

$\mathrm{FS}^H[\Pi_{fd}].\mathrm{Vfy}(\vec{x}, x_o, \mathrm{tr})$

$\Pi_{snark}.\mathrm{Prv}(x_o, w_o)$

π

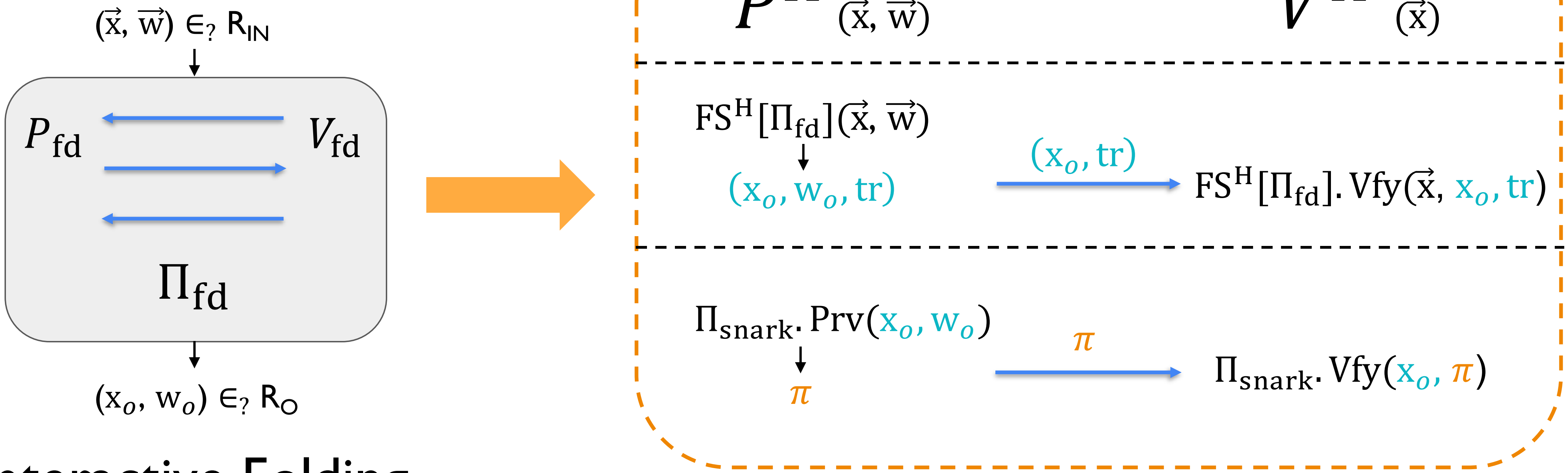
π

$\Pi_{snark}.\mathrm{Vfy}(x_o, \pi)$

Pros: No FS-hash proving

Folding Schemes to SNARKs

Warmup:



$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$

P_{fd} V_{fd}

Π_{fd}

$(\mathrm{x}_o, \mathrm{w}_o) \in_? \mathrm{R_O}$

$P^{\mathrm{H}}(\vec{x}, \vec{w})$ $V^{\mathrm{H}}(\vec{x})$

$\mathrm{FS}^{\mathrm{H}}[\Pi_{\mathrm{fd}}](\vec{x}, \vec{w})$

$\downarrow$

$(\mathrm{x}_o, \mathrm{w}_o, \mathrm{tr})$ $\xrightarrow{(\mathrm{x}_o,\, \mathrm{tr})}$ $\mathrm{FS}^{\mathrm{H}}[\Pi_{\mathrm{fd}}].\mathrm{Vfy}(\vec{x}, \mathrm{x}_o, \mathrm{tr})$

$\Pi_{\mathrm{snark}}.\mathrm{Prv}(\mathrm{x}_o, \mathrm{w}_o)$

$\downarrow$

π $\xrightarrow{\pi}$ $\Pi_{\mathrm{snark}}.\mathrm{Vfy}(\mathrm{x}_o, \pi)$

Interactive Folding

$(> 30\mathrm{MB}$ for $\ell = 1000)$

Pros: No FS-hash proving

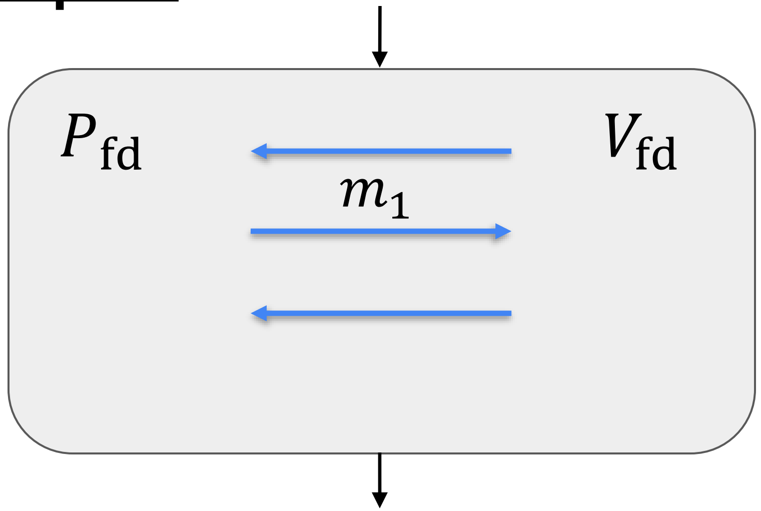
Cons: tr's size is large

Folding Schemes to SNARKs

Cons: tr's size is large ($> 30\text{MB for } \ell = 1000$)

Idea: compress tr via a commitment

Step 1: $(\vec{x}, \vec{w}) \in_? \mathsf{R}_{\mathsf{IN}}$



$(x_o, w_o) \in_? \mathsf{R}_{\mathsf{O}}$

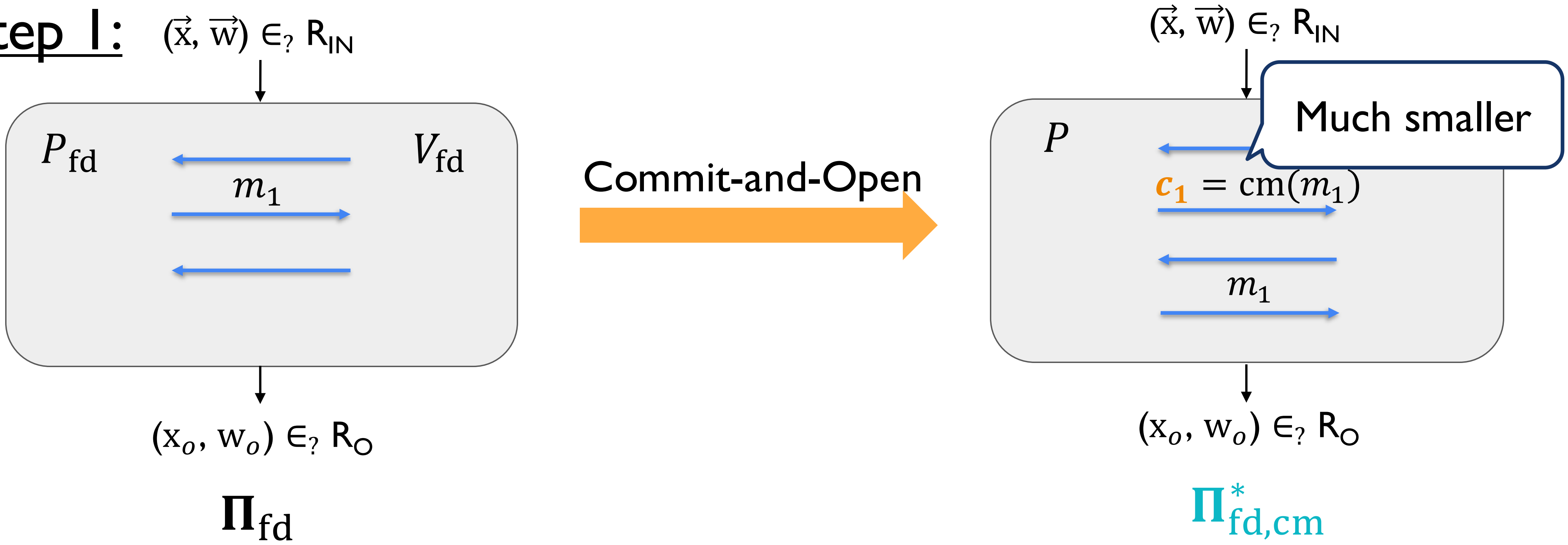
$\boldsymbol{\Pi}_{\mathrm{fd}}$

Folding Schemes to SNARKs

Cons: tr's size is large (> 30MB for $\ell = 1000$)

Idea: compress tr via a commitment

Step 1:

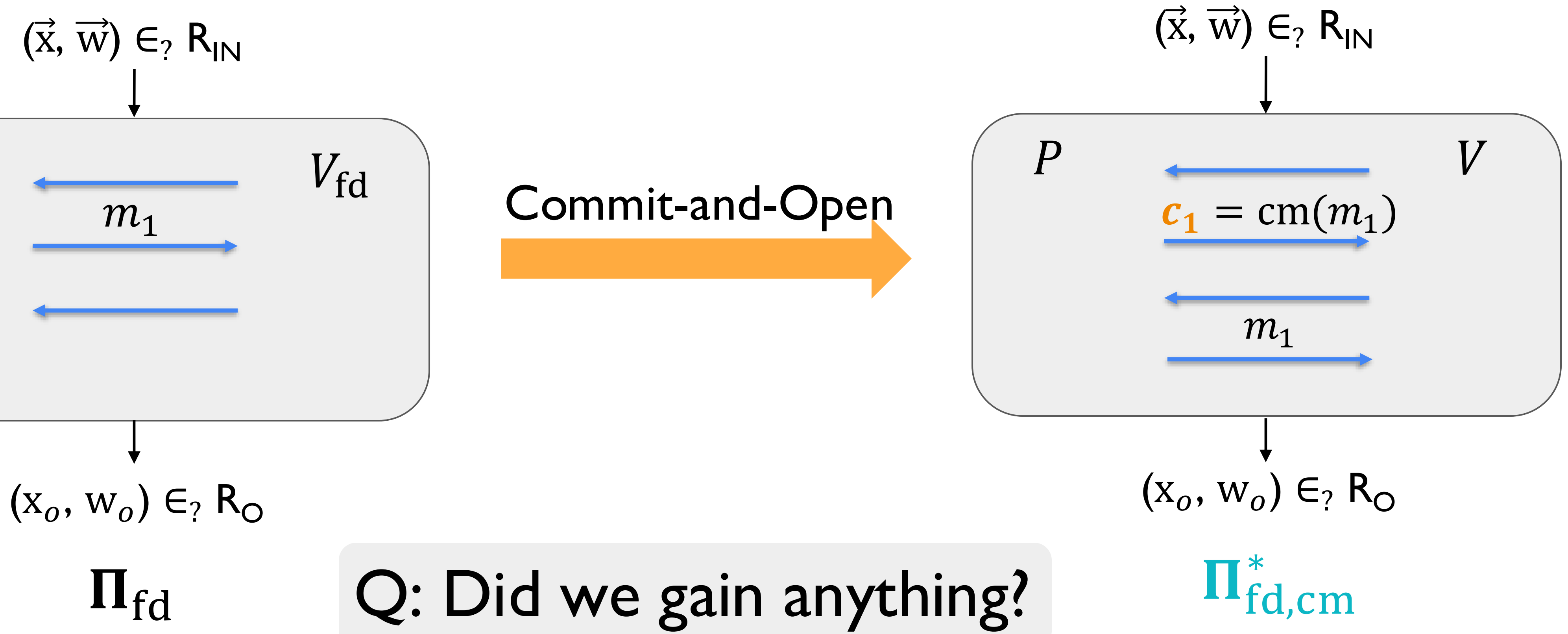


Folding Schemes to SNARKs

Cons: tr's size is large $(> 30\text{MB for } \ell = 1000)$

Idea: compress tr via a commitment

Step 1:

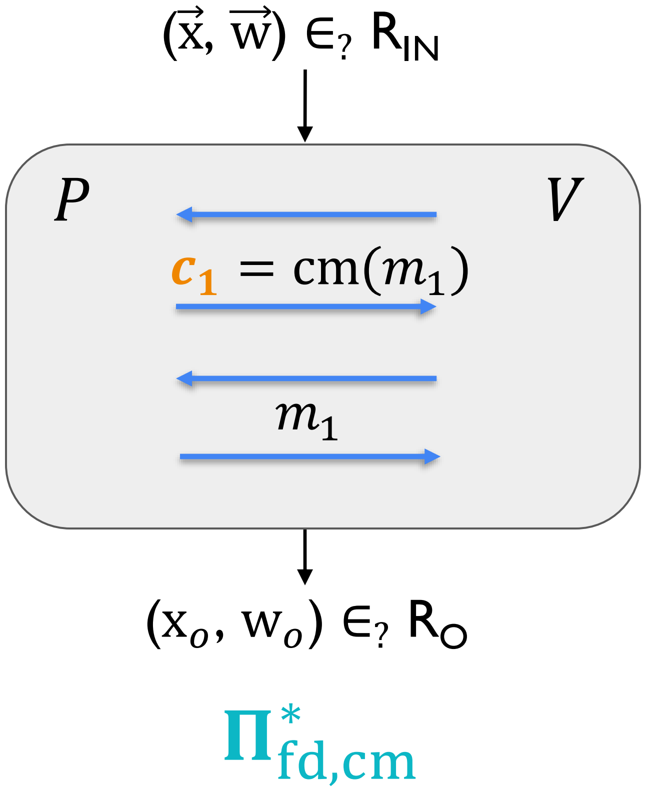


Q: Did we gain anything?

Folding Schemes to SNARKs

Idea: Commit-and-Prove SNARKs [Kil'89, CLOS02, CFQ'19]

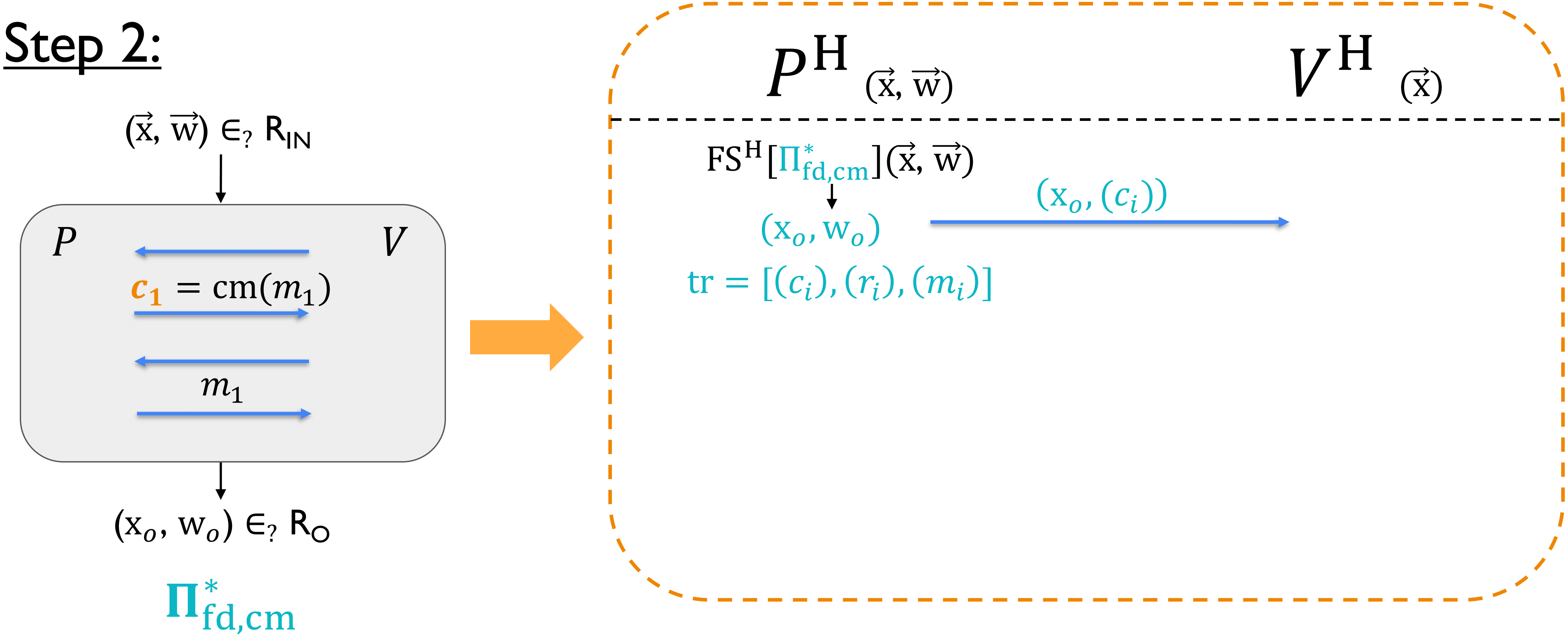
Step 2:


$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$
P
V
$c_1 = \mathrm{cm}(m_1)$
m_1
$(\mathrm{x}_o, \mathrm{w}_o) \in_? \mathrm{R_O}$
$\Pi^*_{\mathrm{fd,cm}}$

Folding Schemes to SNARKs

Idea: Commit-and-Prove SNARKs [Kil'89, CLOS02, CFQ'19]

Step 2:



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

P V

$c_1 = \mathrm{cm}(m_1)$

m_1

$(x_o, w_o) \in_? R_O$

$\Pi^*_{\mathrm{fd,cm}}$

$P^{\mathrm{H}}\ (\vec{x}, \vec{w})$ $V^{\mathrm{H}}\ (\vec{x})$

$\mathrm{FS}^{\mathrm{H}}[\Pi^*_{\mathrm{fd,cm}}](\vec{x}, \vec{w})$

(x_o, w_o)

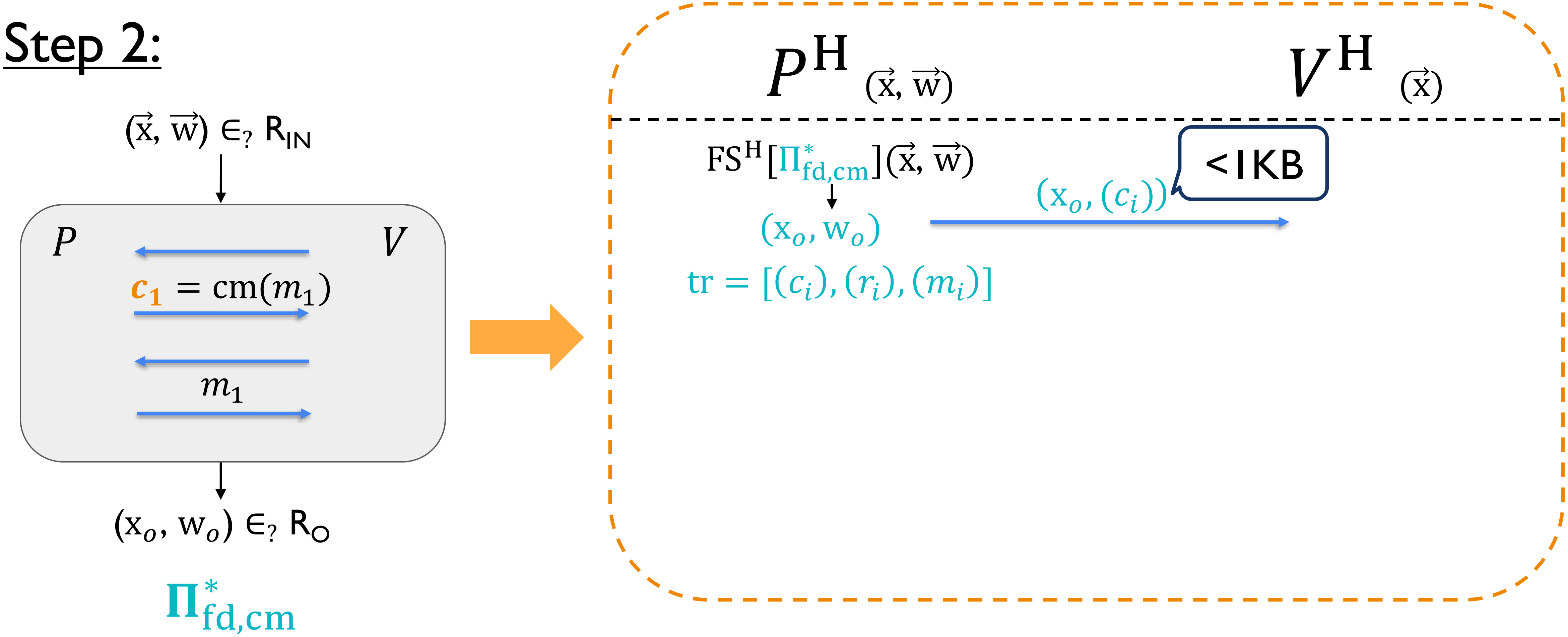
$(x_o, (c_i))$

$\mathrm{tr} = [(c_i), (r_i), (m_i)]$

Folding Schemes to SNARKs

Idea: Commit-and-Prove SNARKs [Kil'89, CLOS02, CFQ'19]

Step 2:

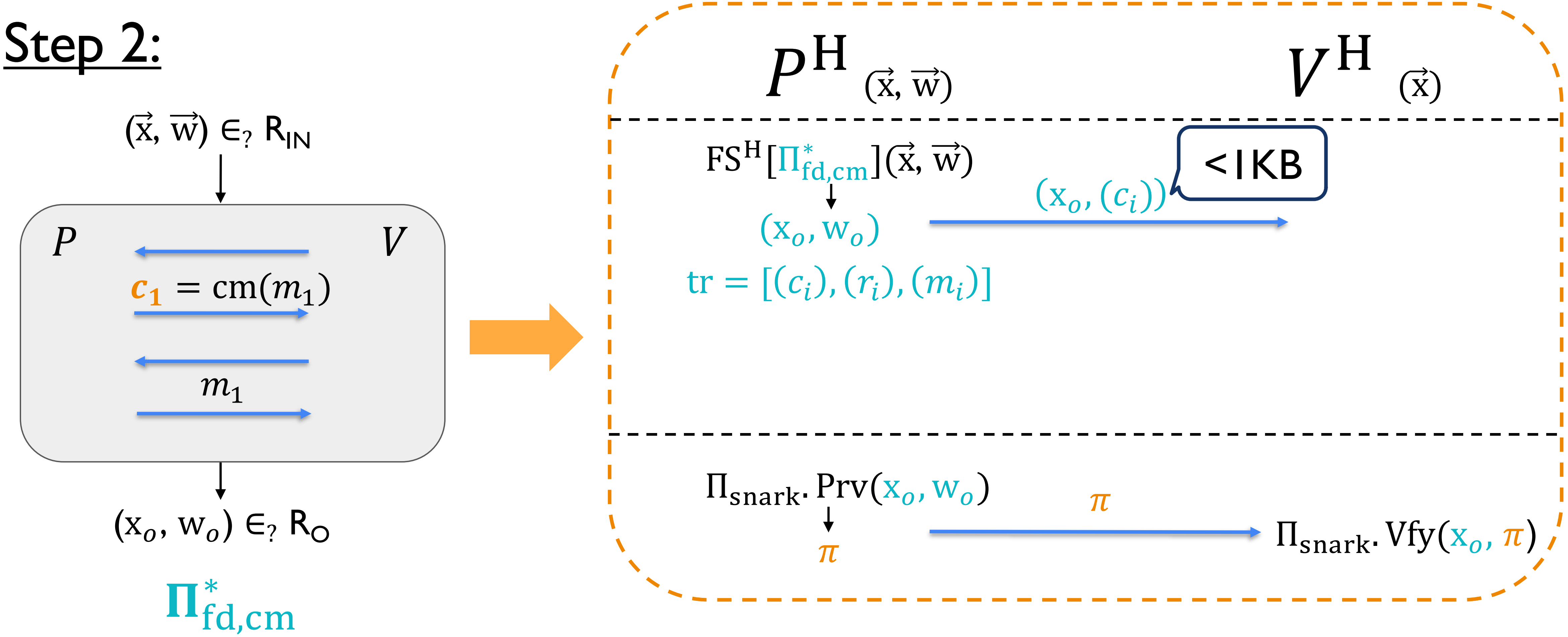


$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$
P
V
$c_1 = \mathrm{cm}(m_1)$
m_1
$(x_o, w_o) \in_? \mathrm{R_O}$
$\Pi^*_{\mathrm{fd,cm}}$
$P^{\mathrm{H}}(\vec{x}, \vec{w})$
$V^{\mathrm{H}}(\vec{x})$
$\mathrm{FS}^{\mathrm{H}}[\Pi^*_{\mathrm{fd,cm}}](\vec{x}, \vec{w})$
(x_o, w_o)
$\mathrm{tr} = [(c_i), (r_i), (m_i)]$
$(x_o, (c_i))$
<1KB

Folding Schemes to SNARKs

Idea: Commit-and-Prove SNARKs [Kil'89, CLOS02, CFQ'19]

Step 2:



$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$

P V

$c_1 = \mathrm{cm}(m_1)$

m_1

$(\mathrm{x}_o, \mathrm{w}_o) \in_? \mathrm{R_O}$

$\Pi^*_{\mathrm{fd,cm}}$

$P^{\mathrm{H}}(\vec{x}, \vec{w})$ $V^{\mathrm{H}}(\vec{x})$

$\mathrm{FS}^{\mathrm{H}}[\Pi^*_{\mathrm{fd,cm}}](\vec{x}, \vec{w})$

$(\mathrm{x}_o, \mathrm{w}_o)$

$(\mathrm{x}_o, (c_i))$

<1KB

$\mathrm{tr} = [(c_i), (r_i), (m_i)]$

$\Pi_{\mathrm{snark}}.\mathrm{Prv}(\mathrm{x}_o, \mathrm{w}_o)$

π

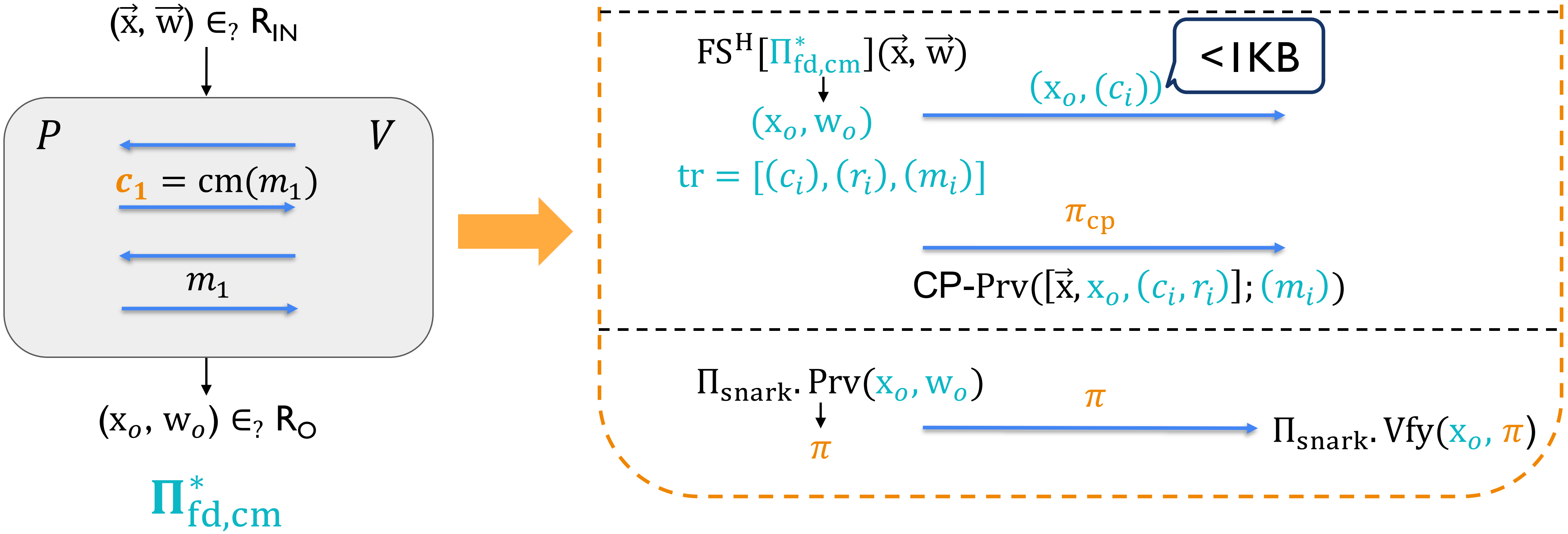
π

$\Pi_{\mathrm{snark}}.\mathrm{Vfy}(\mathrm{x}_o, \pi)$

Folding Schemes to SNARKs

Idea: Commit-and-Prove SNARKs [Kil'89, CLOS02, CFQ'19]

Step 2:



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

P V

$c_1 = \mathrm{cm}(m_1)$

m_1

$(x_o, w_o) \in_? R_O$

$\Pi^*_{\mathrm{fd,cm}}$

$P^{H}(\vec{x}, \vec{w})$ $V^{H}(\vec{x})$

$\mathrm{FS}^{H}[\Pi^*_{\mathrm{fd,cm}}](\vec{x}, \vec{w})$

(x_o, w_o)

$(x_o, (c_i))$

<1KB

$\mathrm{tr} = [(c_i), (r_i), (m_i)]$

π_{cp}

$\text{CP-Prv}([\vec{x}, x_o, (c_i, r_i)]; (m_i))$

$\Pi_{\mathrm{snark}}.\mathrm{Prv}(x_o, w_o)$

π

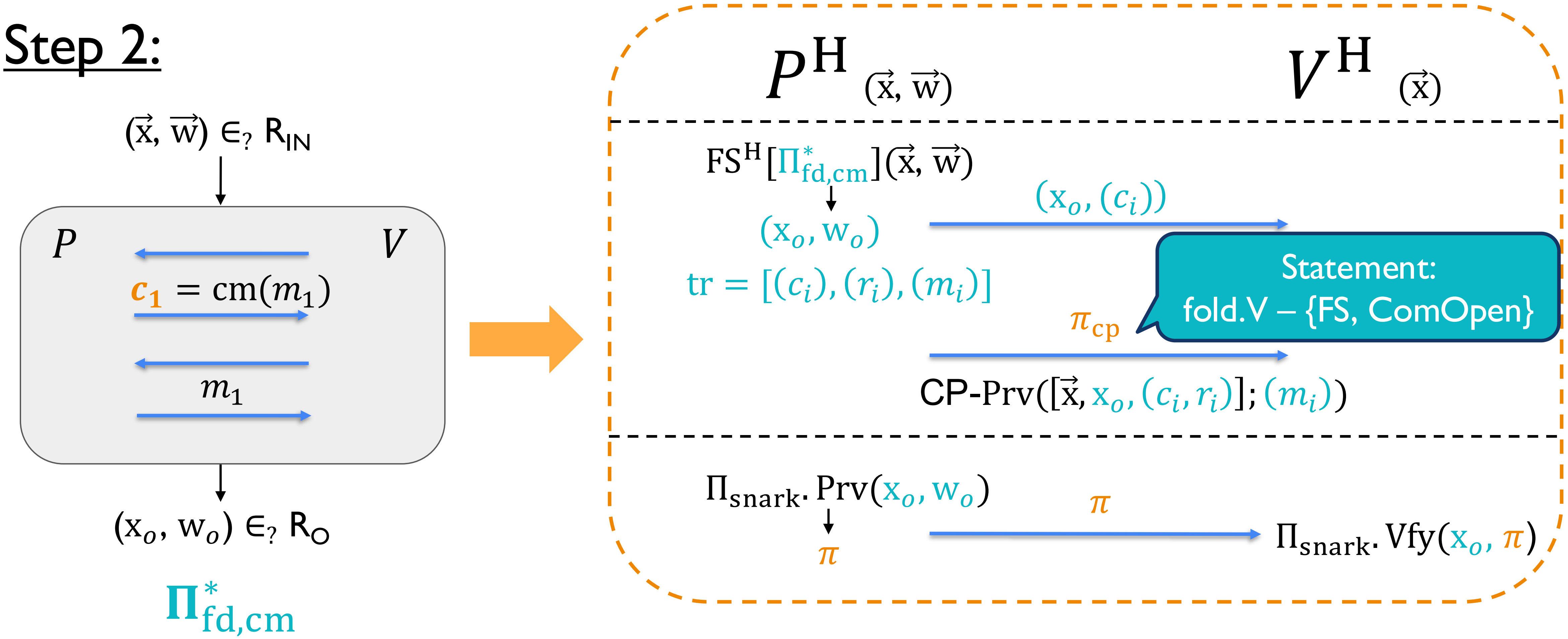
π

$\Pi_{\mathrm{snark}}.\mathrm{Vfy}(x_o, \pi)$

Folding Schemes to SNARKs

Idea: Commit-and-Prove SNARKs [Kil'89, CLOS02, CFQ'19]

Step 2:



$(\vec{x}, \vec{w}) \in_? \mathrm{R_{IN}}$

P V

$c_1 = \mathrm{cm}(m_1)$

m_1

$(x_o, w_o) \in_? \mathrm{R_O}$

$\Pi^*_{\mathrm{fd,cm}}$

$P^{\mathrm{H}}(\vec{x}, \vec{w})$ $V^{\mathrm{H}}(\vec{x})$

$\mathrm{FS^H}[\Pi^*_{\mathrm{fd,cm}}](\vec{x}, \vec{w})$

(x_o, w_o)

$\mathrm{tr} = [(c_i), (r_i), (m_i)]$

$(x_o, (c_i))$

Statement: fold.V – {FS, ComOpen}

π_{cp}

CP-Prv$([\vec{x}, x_o, (c_i, r_i)]; (m_i))$

$\Pi_{\mathrm{snark}}.\mathrm{Prv}(x_o, w_o)$

π

π

$\Pi_{\mathrm{snark}}.\mathrm{Vfy}(x_o, \pi)$

High-arity folding ⇒ Succinct Arg in the ROM? ✓

High-arity folding ⇒ Succinct Arg in the ROM? ✓

Q: How to build a high-arity folding scheme?

High-arity folding ⇒ Succinct Arg in the ROM? ✓

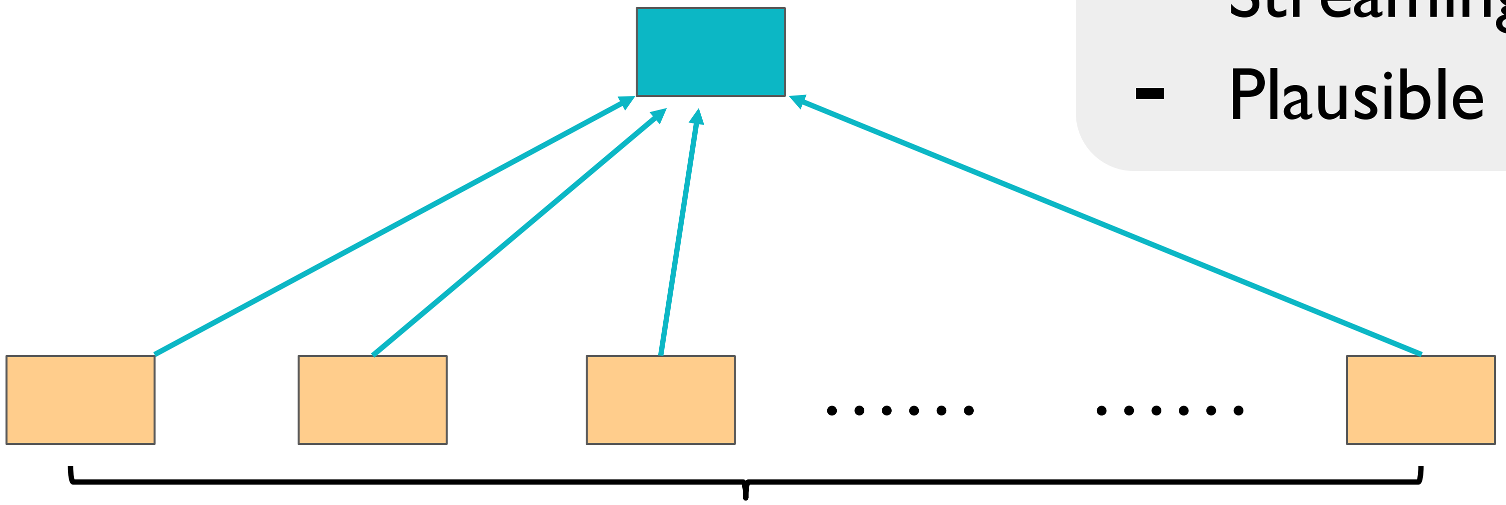
Q: How to build a high-arity folding scheme?

New Design Requirements

- Efficient prover for high-arity setting
- Minimize |fold.V – Fiat-Shamir|

Lattice-based High-Arity Folding Scheme

- Memory efficient
- Streaming friendly
- Plausible post-quantum security



$\ell \geq 2^{10}$ stmnts (each w/ witness length n)

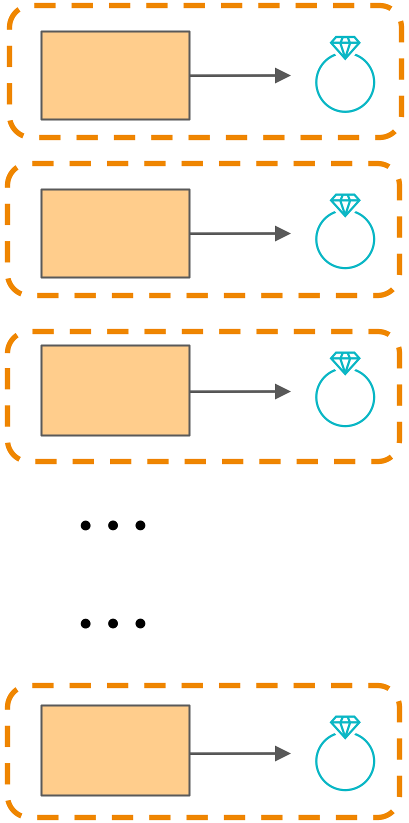
Excluding FS

<u>Prover:</u> $O(\ell n)$ R_q-ops

<u>Verifier:</u> $O(\ell)$ R_q-ops

A Standard Lattice-Folding Framework

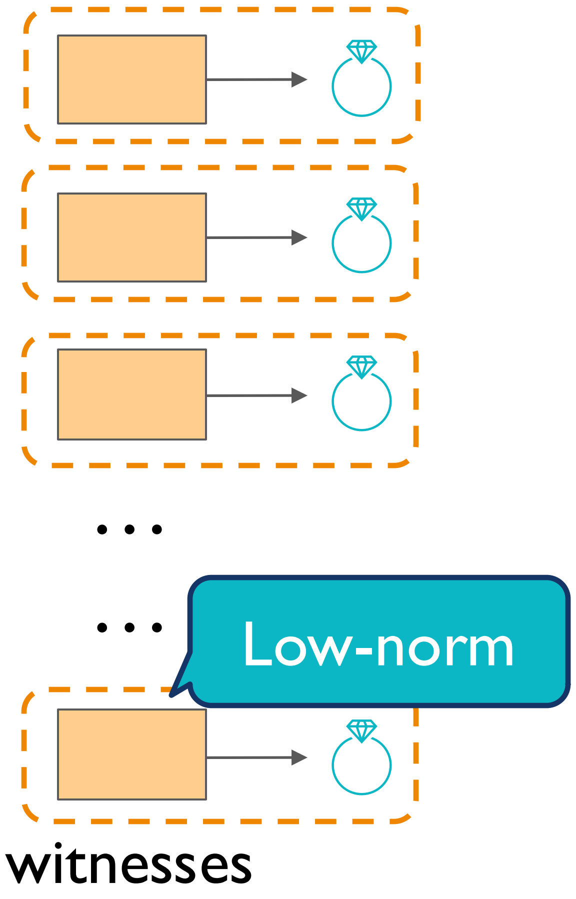
Step 1: Commit



witnesses

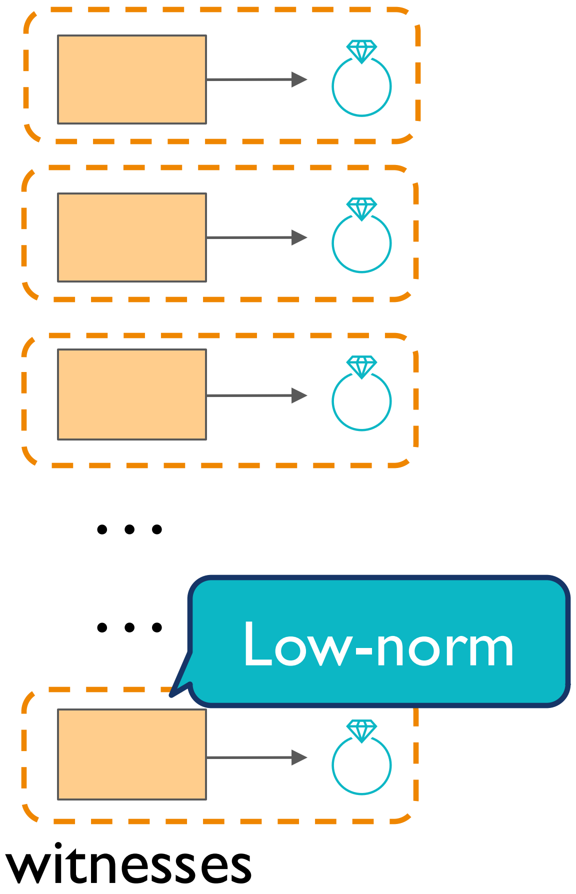
A Standard Lattice-Folding Framework

Step 1: Commit

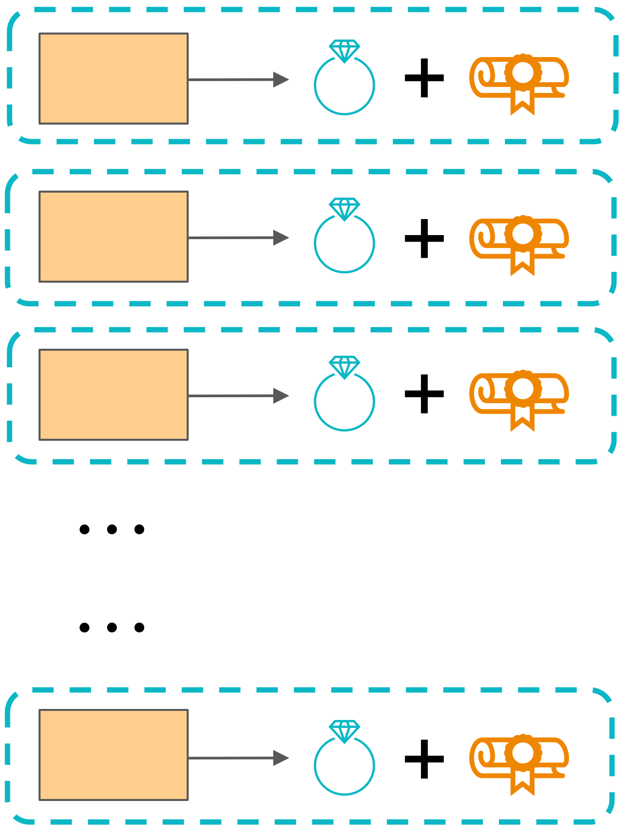


A Standard Lattice-Folding Framework

Step 1: Commit

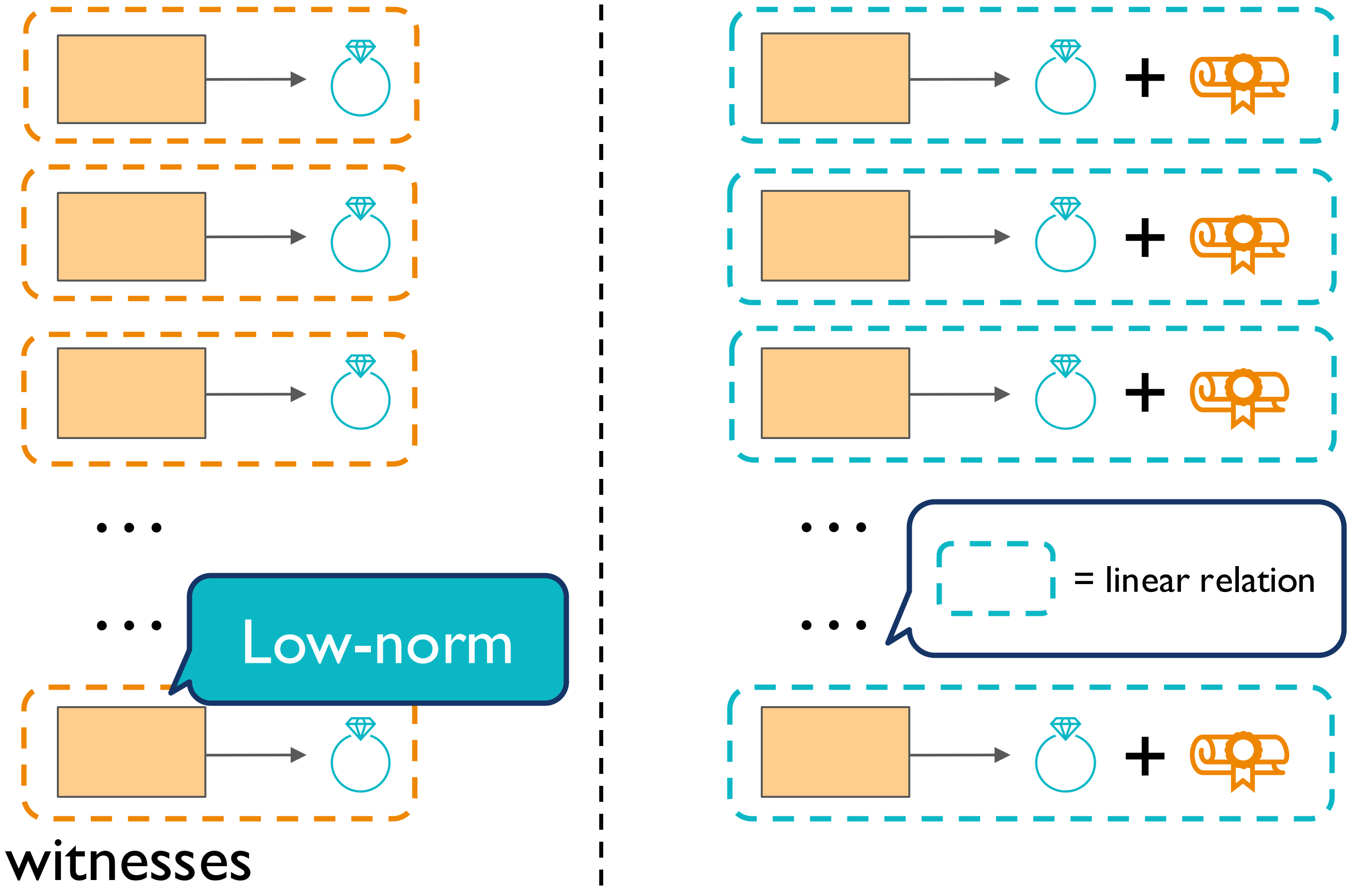


Step 2: Linearize + Range-chk



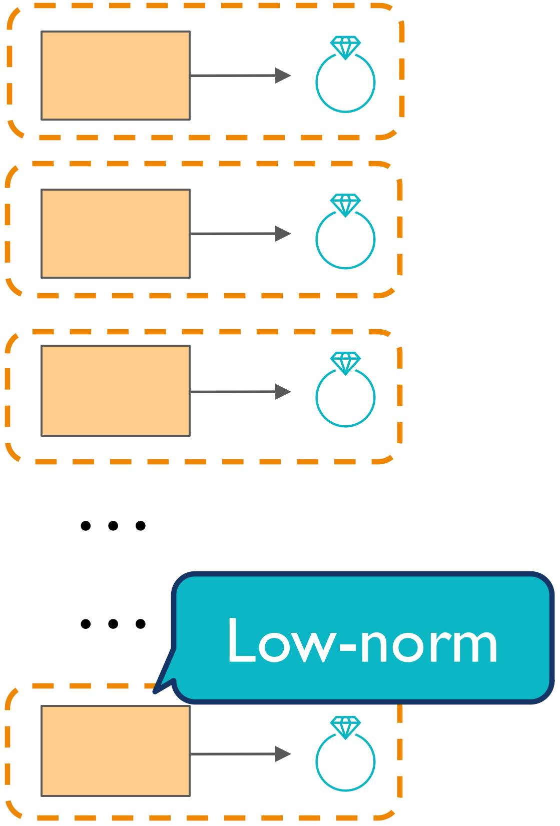
A Standard Lattice-Folding Framework

Step 1: Commit | Step 2: Linearize + Range-chk



A Standard Lattice-Folding Framework

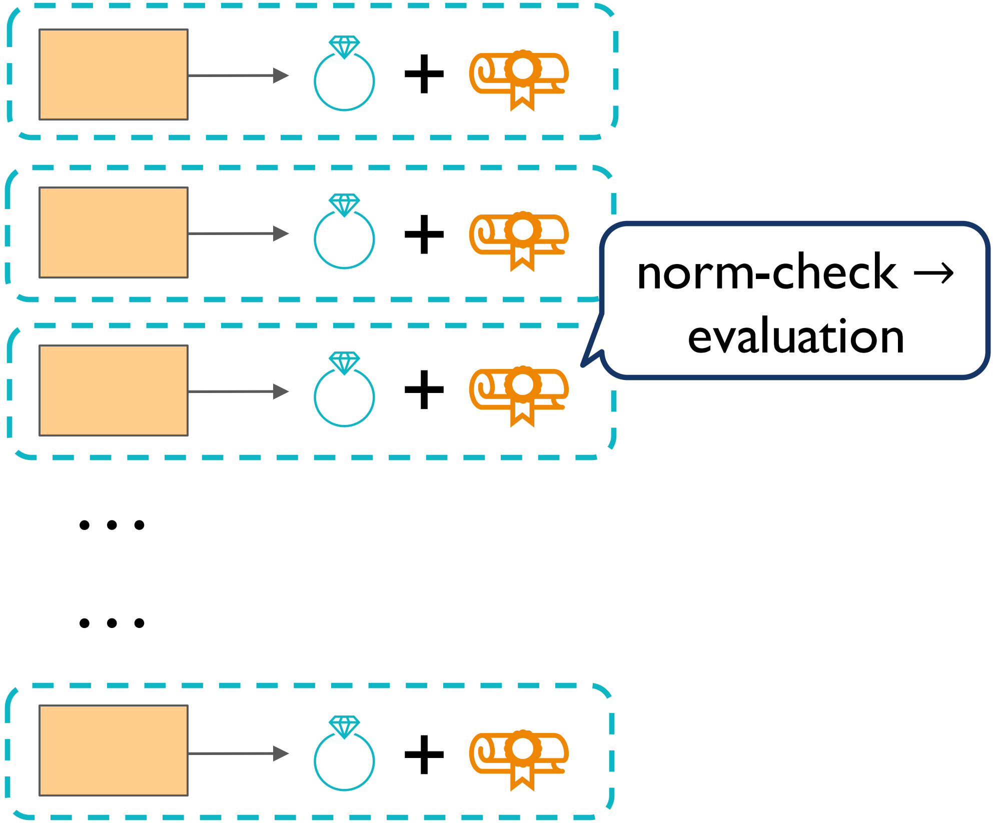
Step 1: Commit



Low-norm

witnesses

Step 2: Linearize + Range-chk



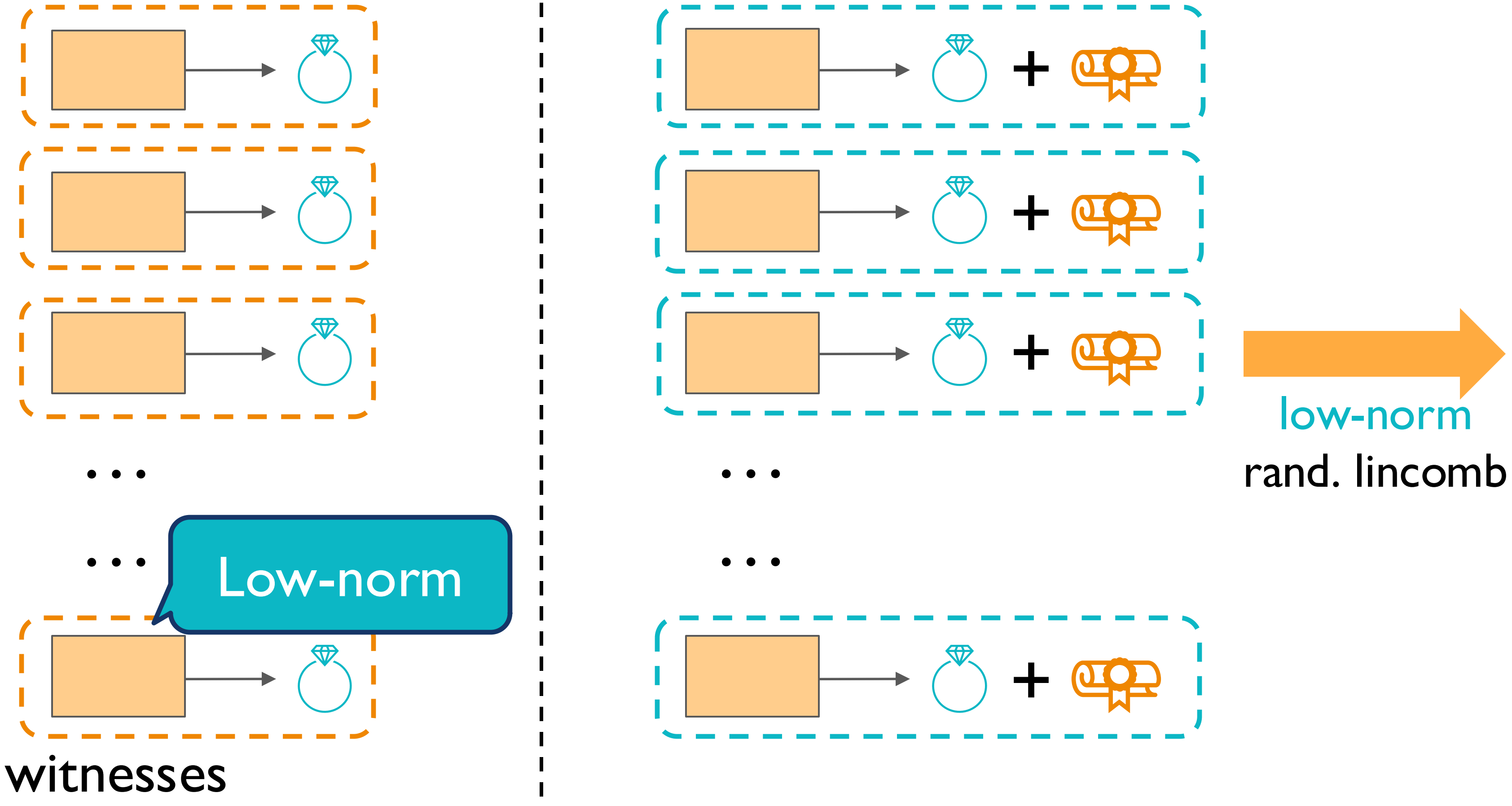
norm-check → evaluation

A Standard Lattice-Folding Framework

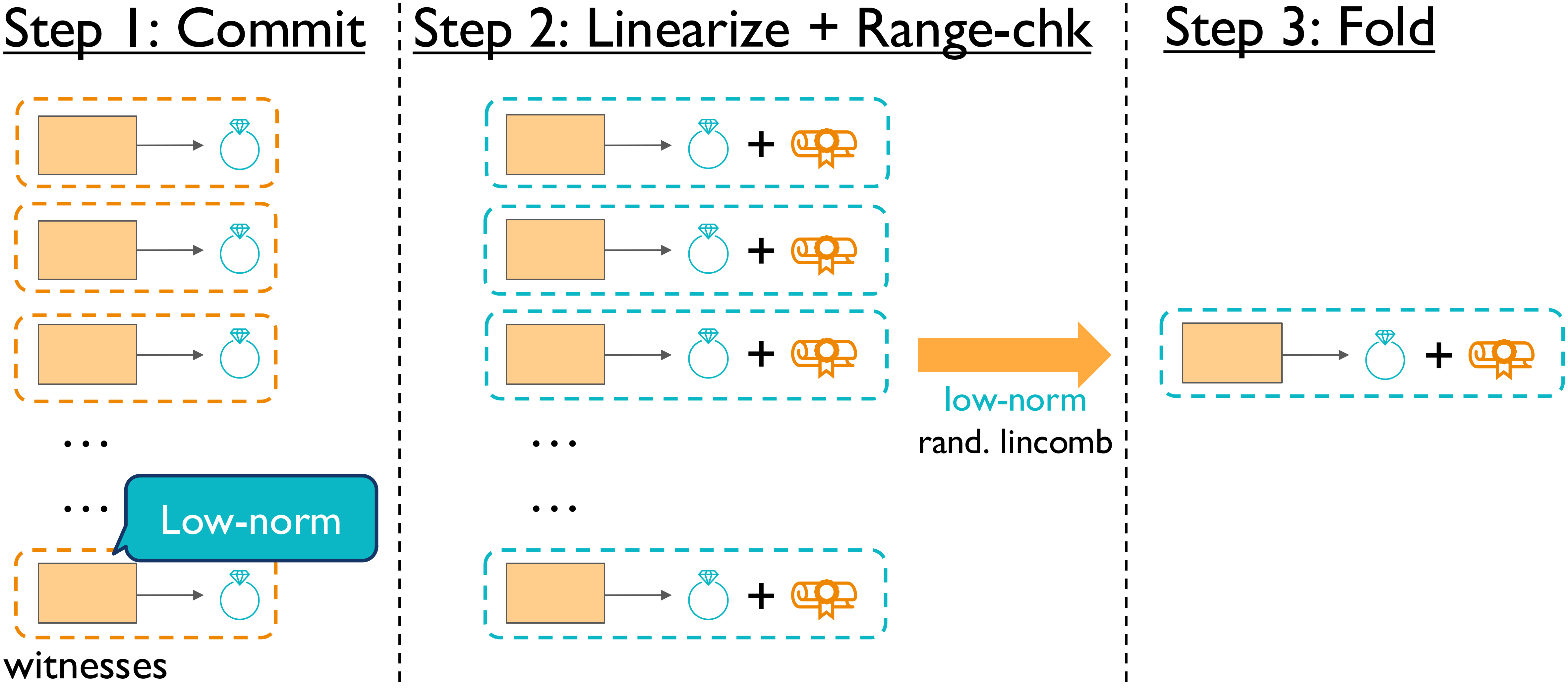
Step 1: Commit

Step 2: Linearize + Range-chk

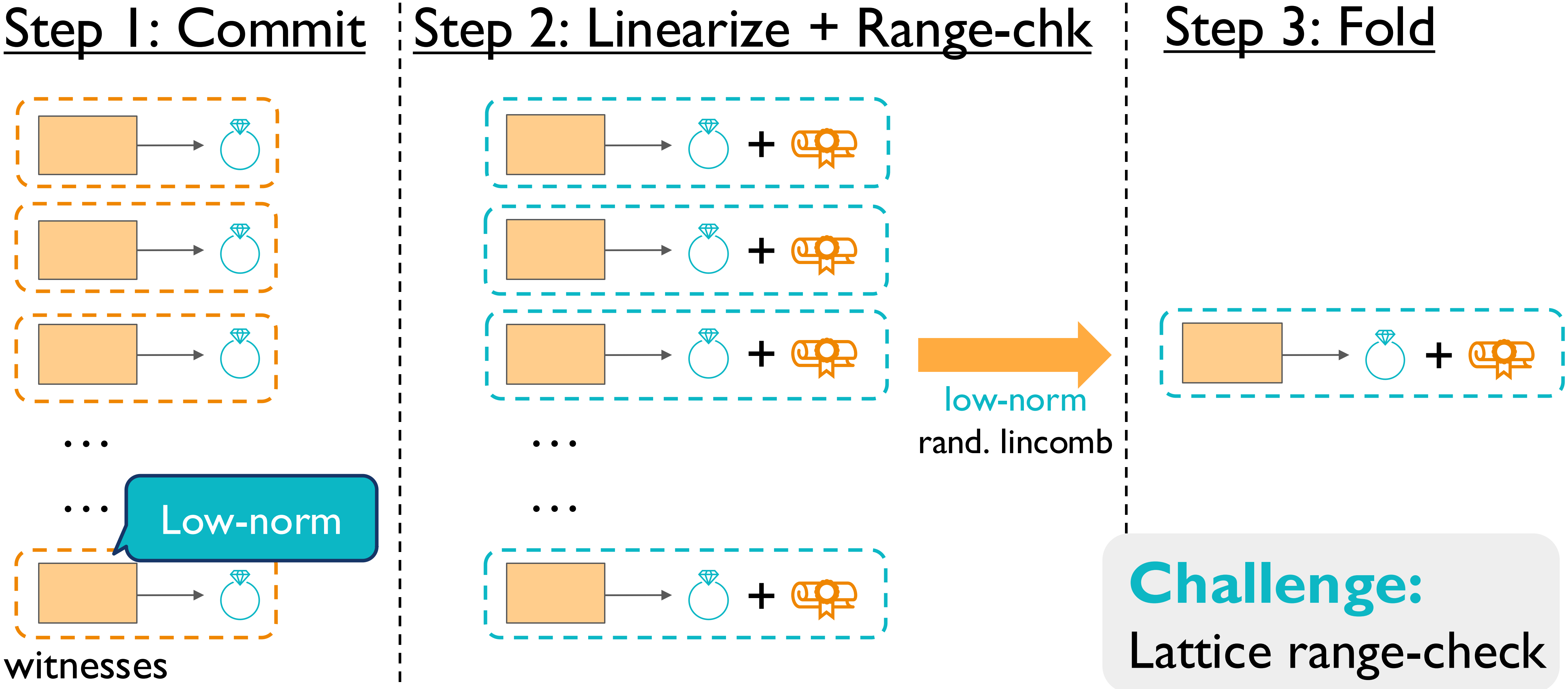
Step 3: Fold



A Standard Lattice-Folding Framework



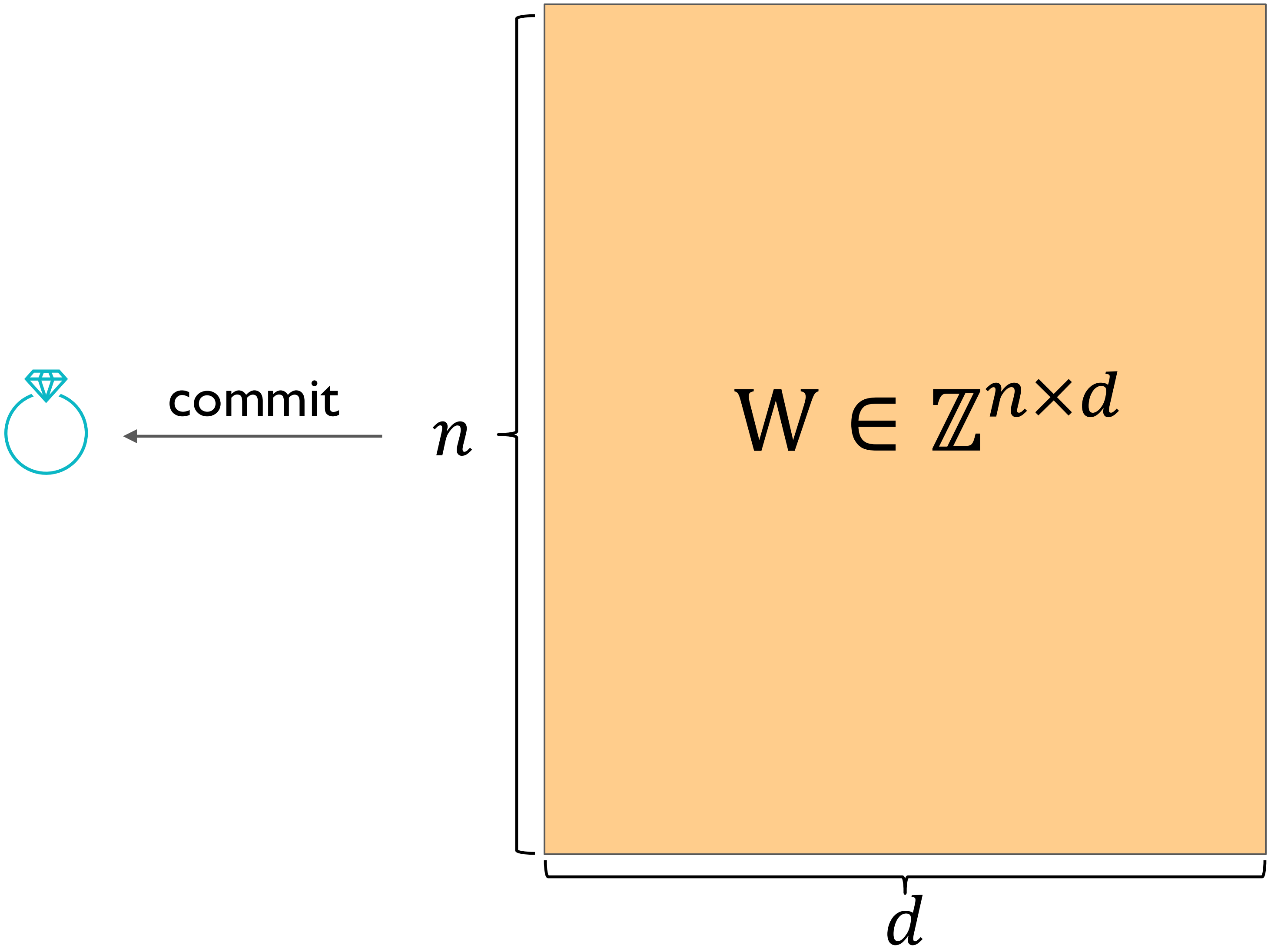
A Standard Lattice-Folding Framework



Challenge:
Lattice range-check

Idea: Approximate range-proof is enough in the high-arity setting!

Lattice-Based Range Proofs



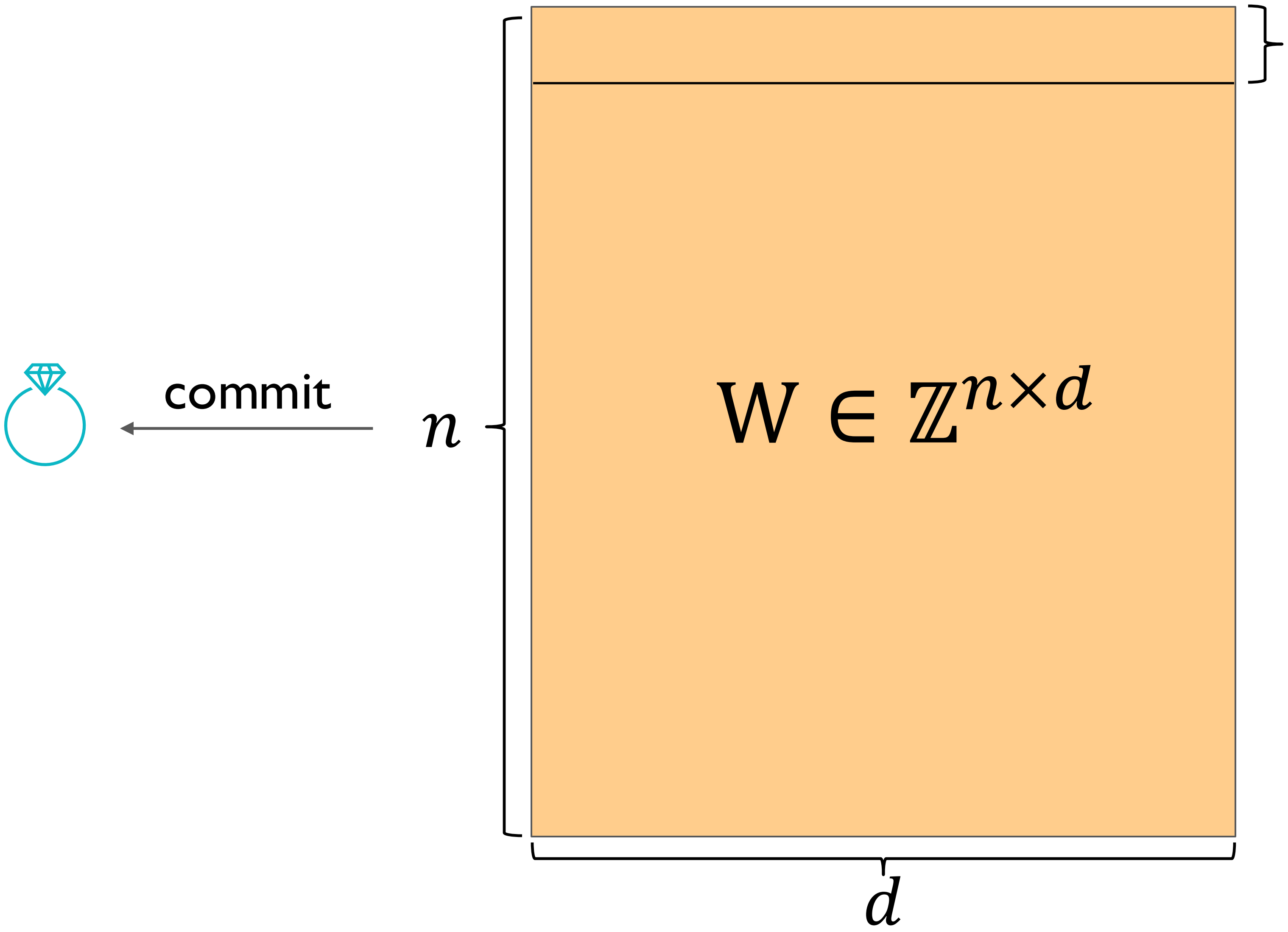
commit

n

$W \in \mathbb{Z}^{n\times d}$

d

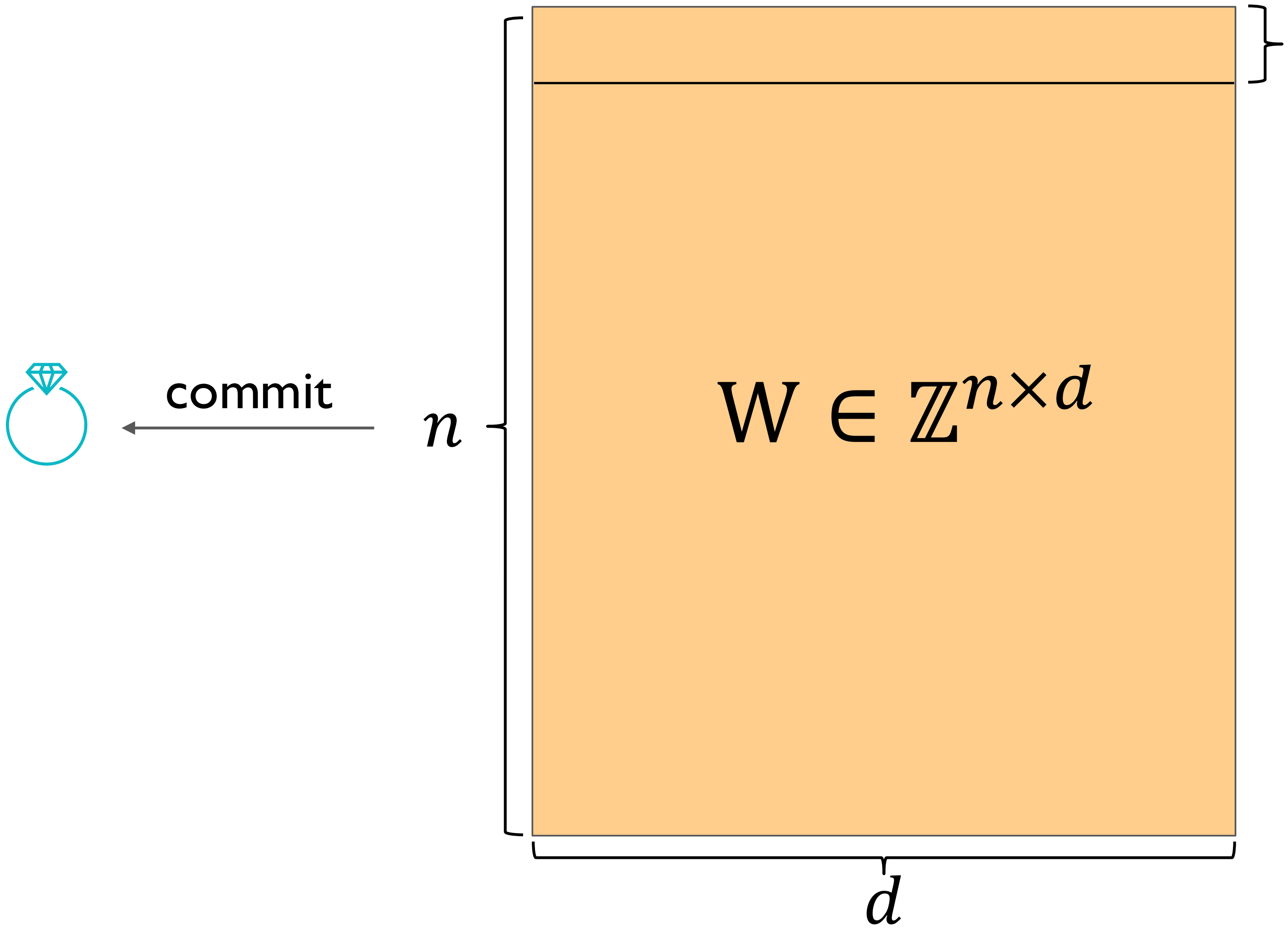
Lattice-Based Range Proofs



d coefficients of each witness elem over $R = \mathbb{Z}[X]/(X^d + 1)$

Lattice-Based Range Proofs

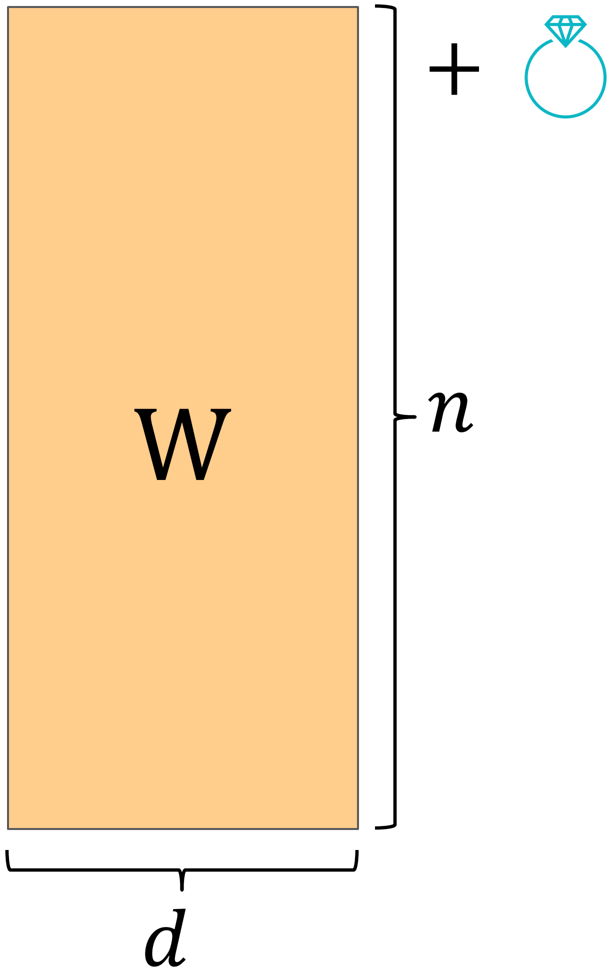
d coefficients of each witness elem over $R = \mathbb{Z}[X]/(X^d + 1)$



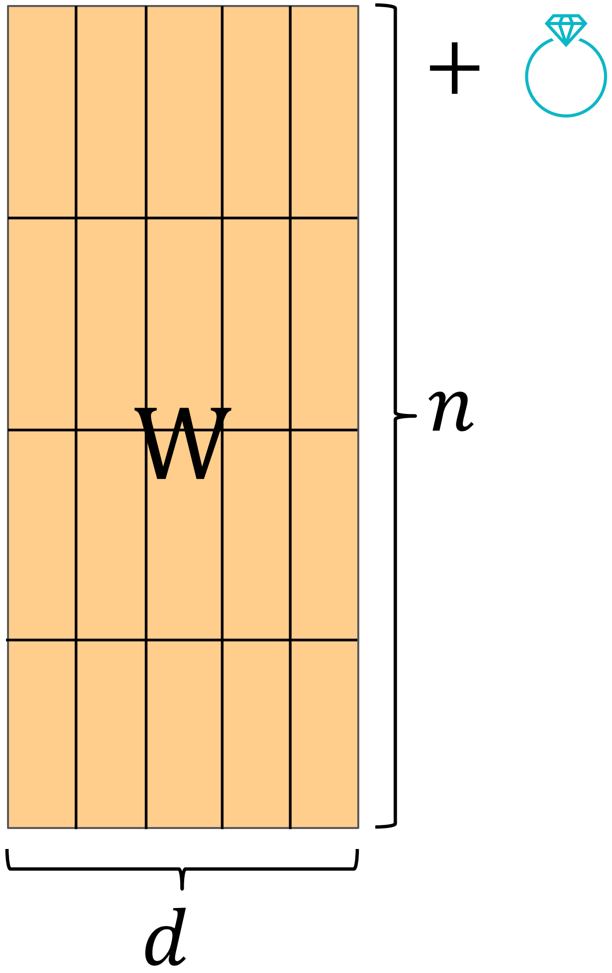
Goal check:

$\|\mathrm{W}\| < B$

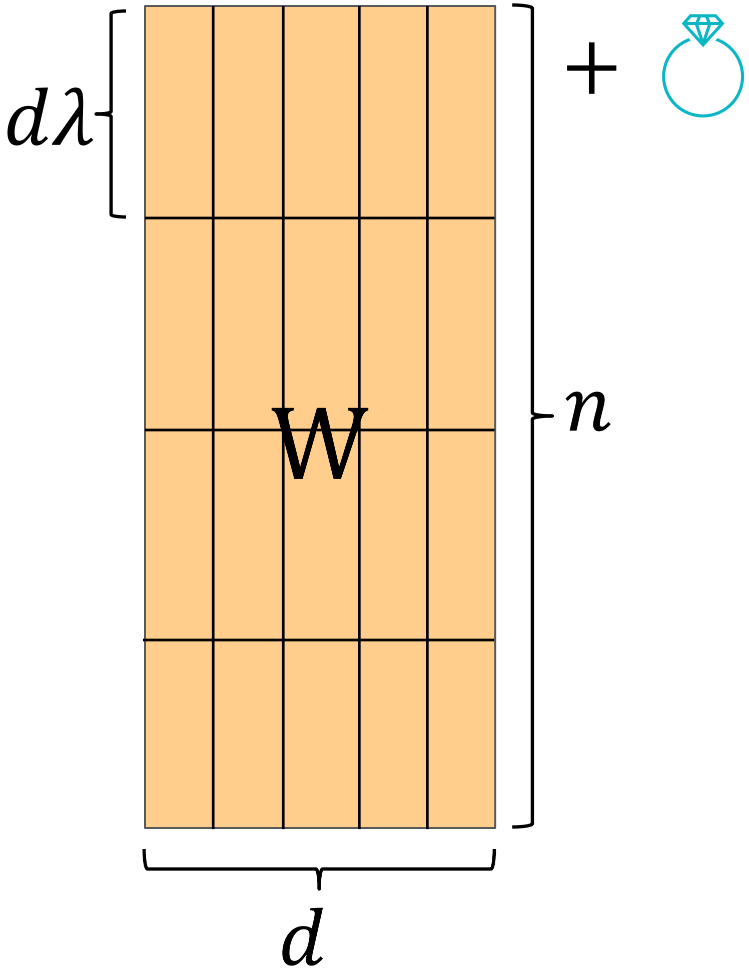
Step 1: Structured Random Projection[KLNO'25]



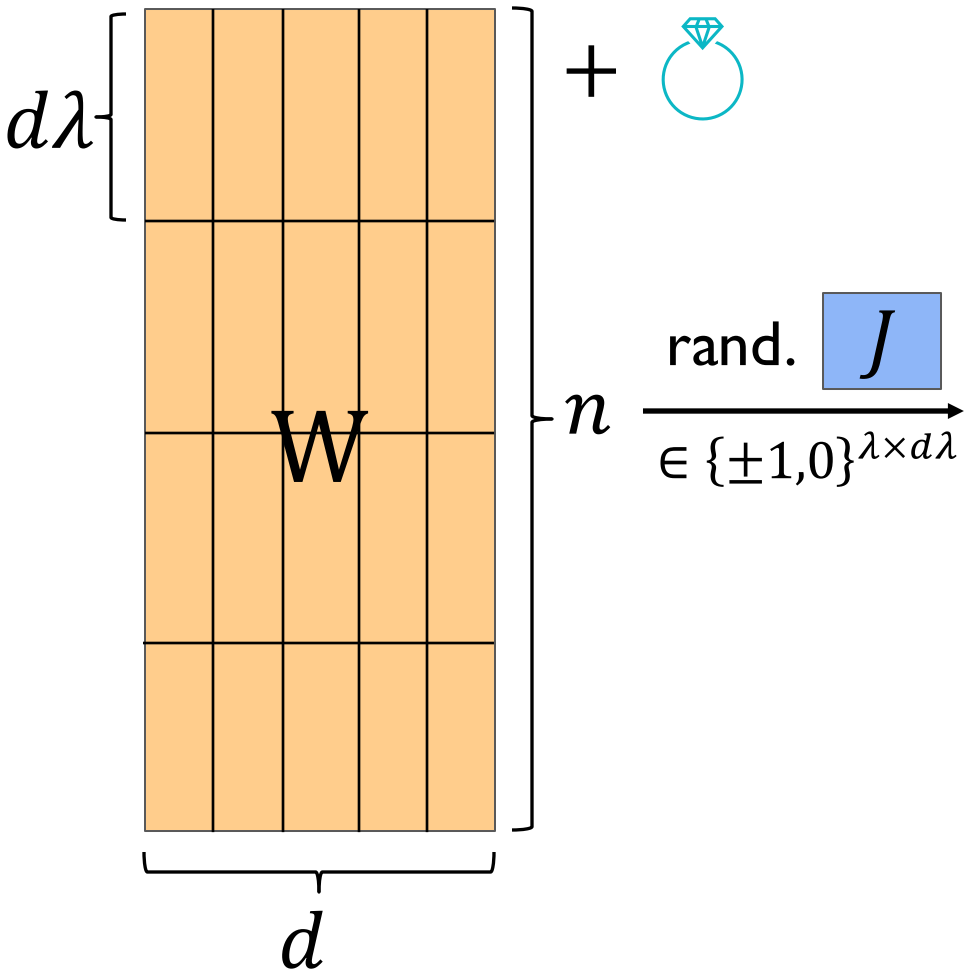
Step 1: Structured Random Projection[KLNO’25]



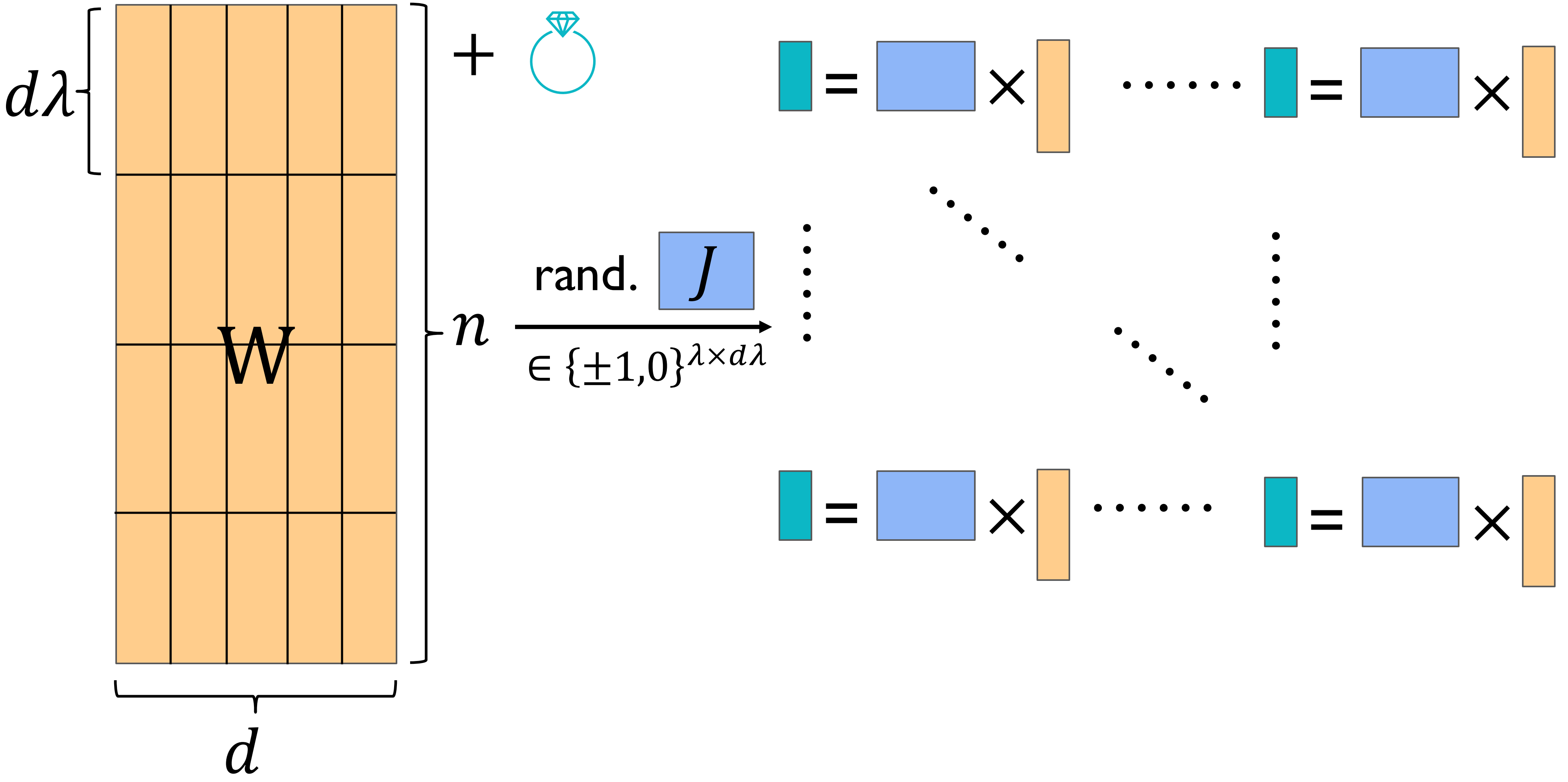
Step 1: Structured Random Projection[KLNO’25]



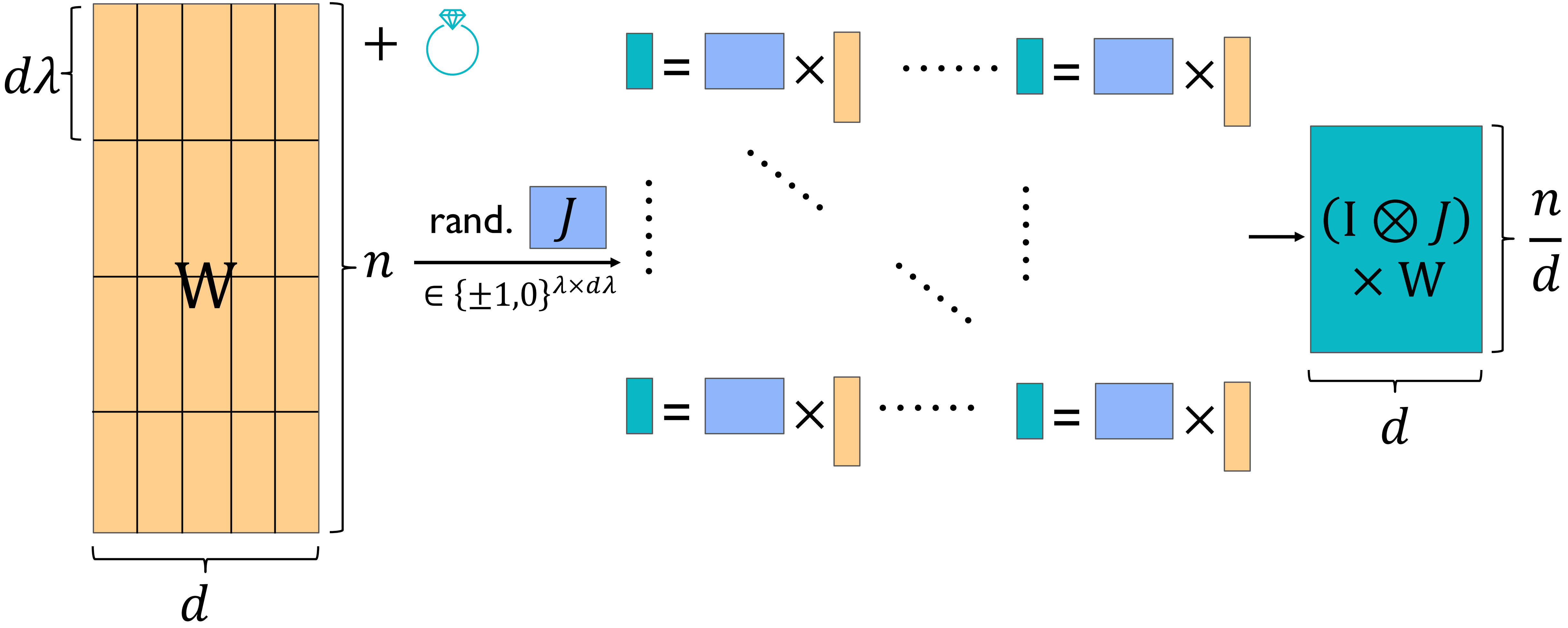
Step 1: Structured Random Projection[KLNO'25]



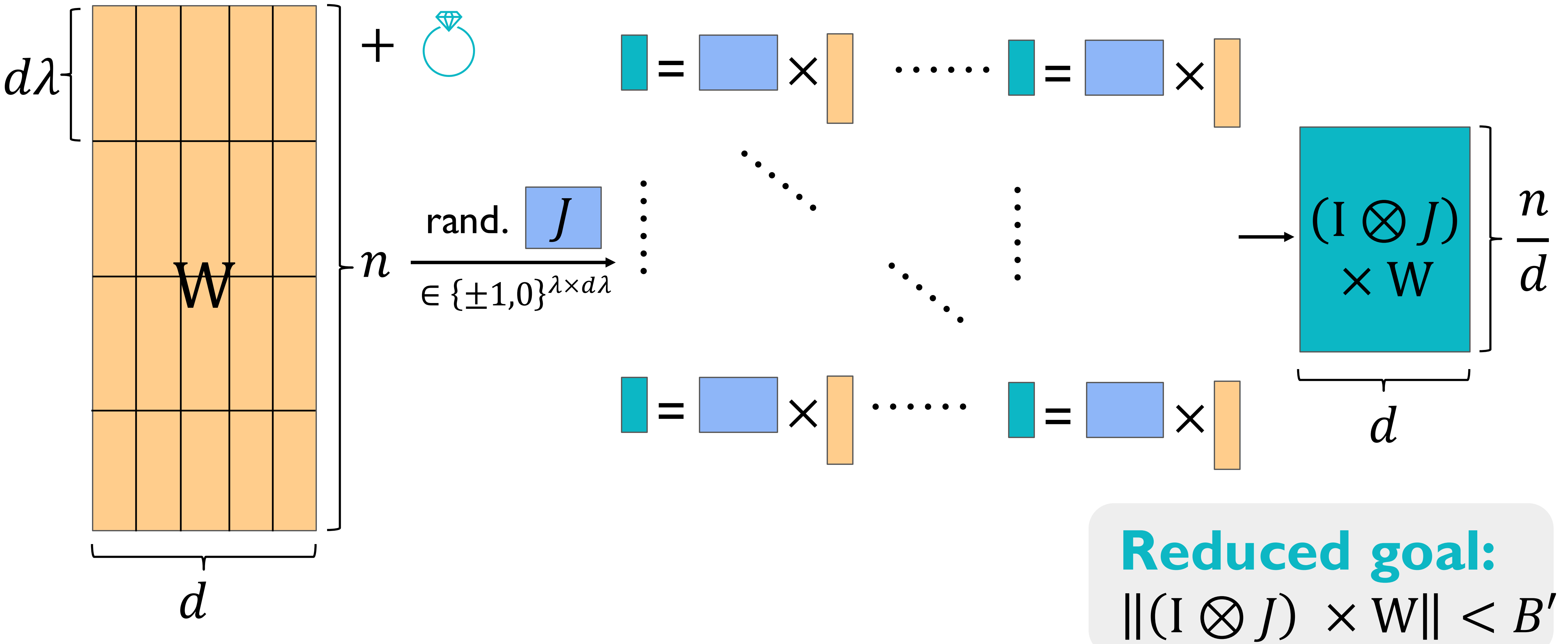
Step 1: Structured Random Projection[KLNO'25]



Step 1: Structured Random Projection[KLNO'25]



Step 1: Structured Random Projection[KLNO'25]

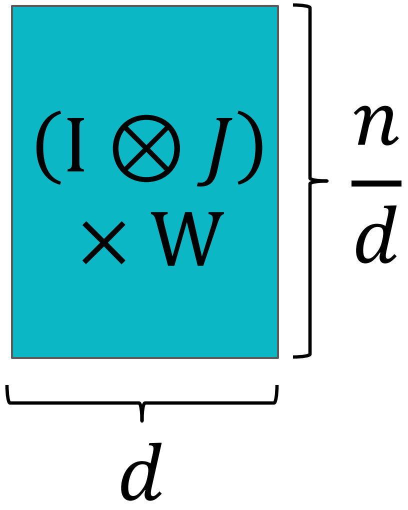


Reduced goal:

$\|(\mathrm{I} \otimes J) \times \mathrm{W}\| < B'$

Step 2: Monomial Lookup [BC'25]

$+$ 💍 $\mathrm{CM}(W)$

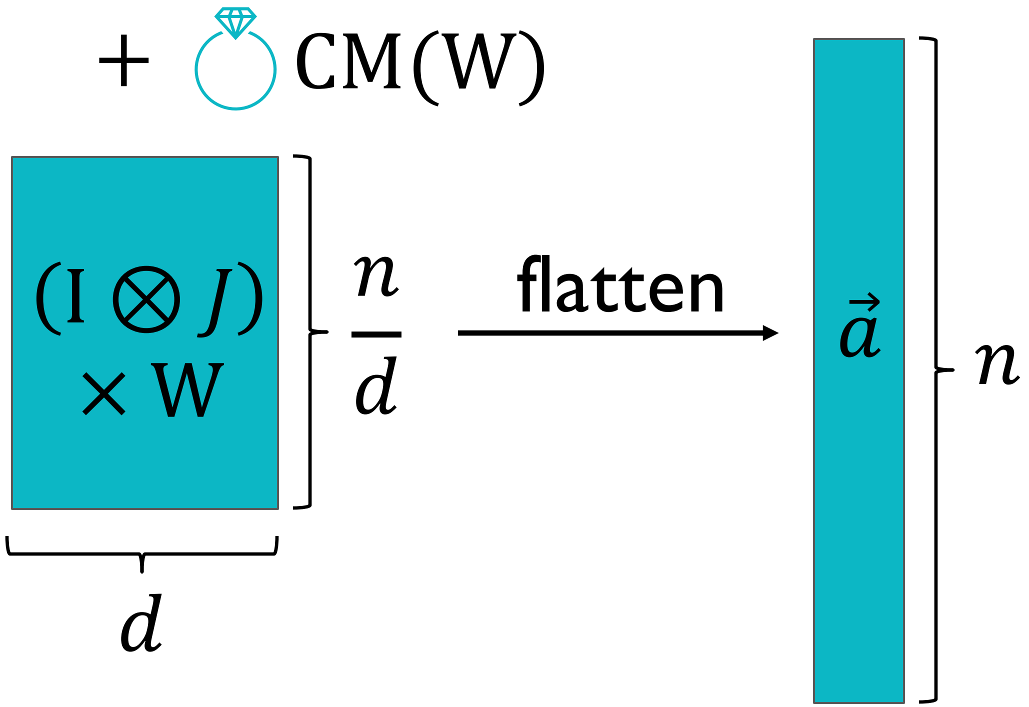


$(I \otimes J) \times W$

$\frac{n}{d}$

d

Step 2: Monomial Lookup [BC'25]



$+$ 💍 $\mathrm{CM}(\mathrm{W})$

$(\mathrm{I} \otimes J) \times \mathrm{W}$

$\frac{n}{d}$

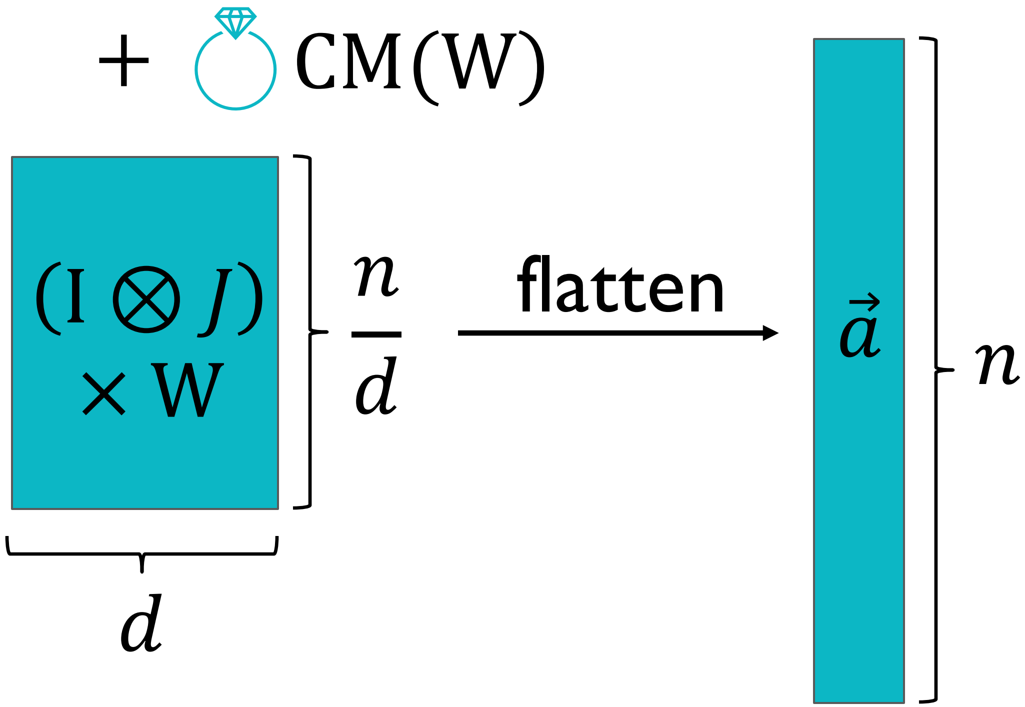
d

$\xrightarrow{\text{flatten}}$

$\vec{a}$

n

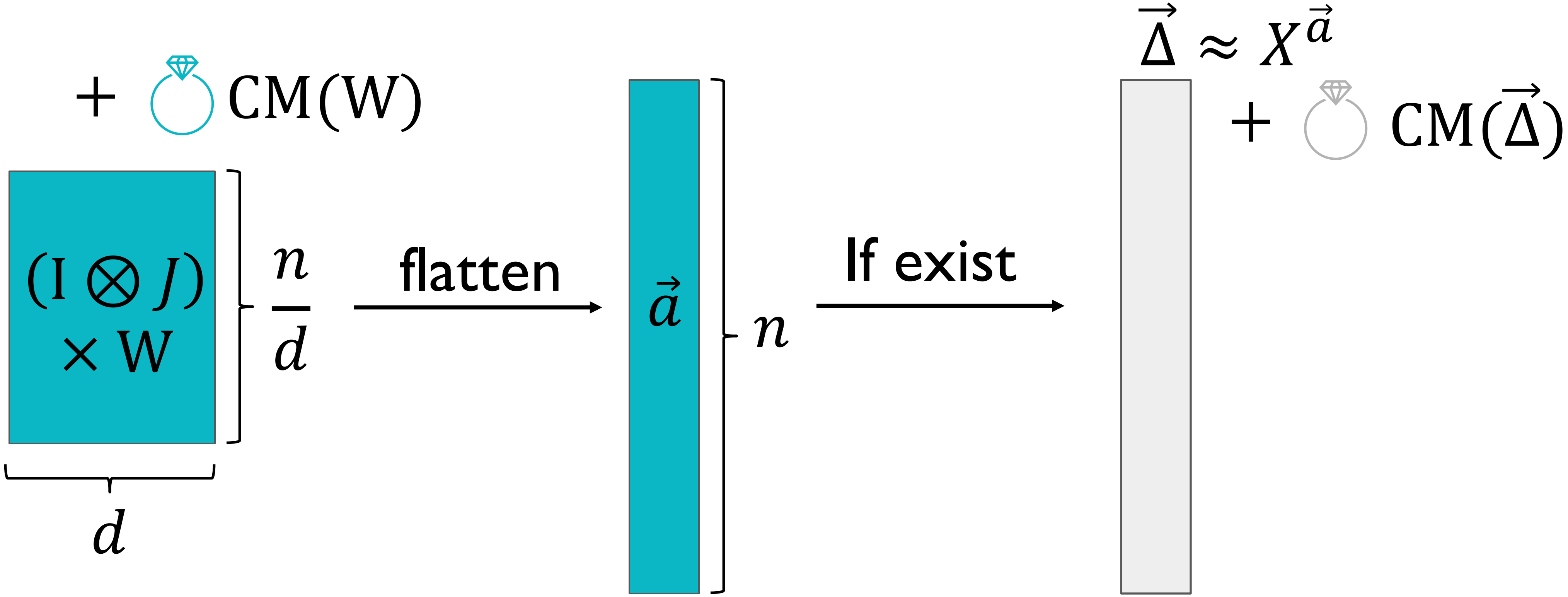
Step 2: Monomial Lookup [BC'25]



Simpler goal:

Each elem of $\vec{a}$ is in $[0, d-1]$

Step 2: Monomial Lookup [BC'25]



$+$ 💍 $\mathrm{CM}(\mathrm{W})$

$(\mathrm{I} \otimes J) \times \mathrm{W}$

$\frac{n}{d}$

d

flatten

$\vec{a}$

n

If exist

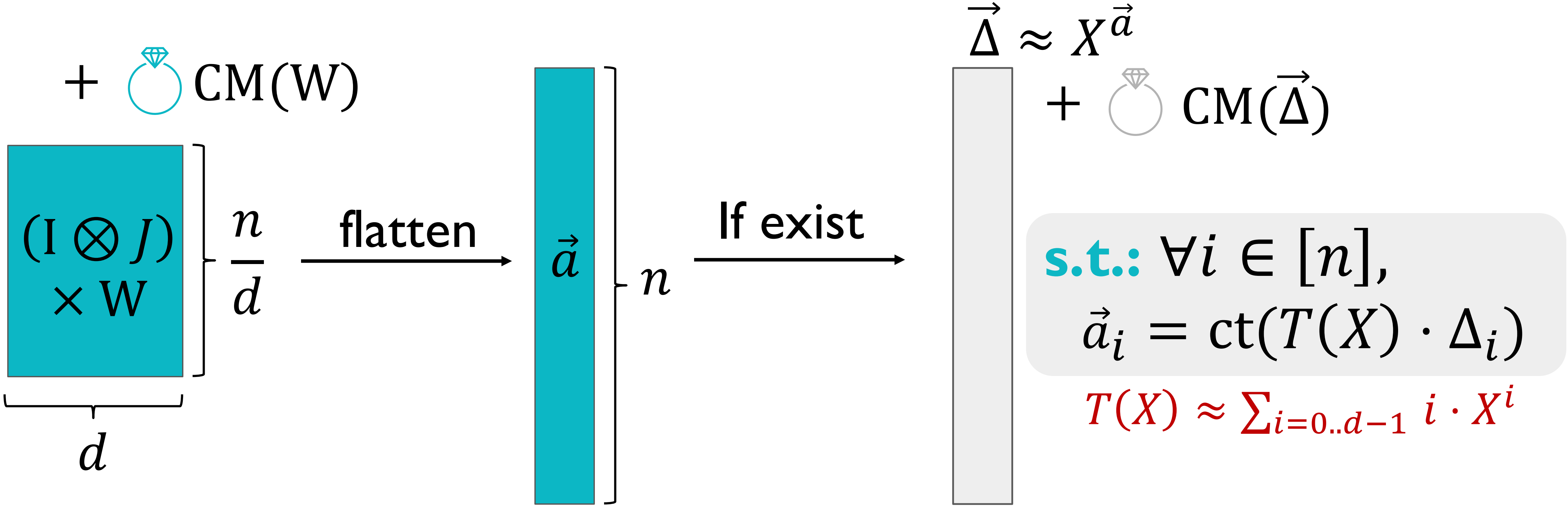
$\vec{\Delta} \approx X^{\vec{a}}$

$+$ 💍 $\mathrm{CM}(\vec{\Delta})$

Simpler goal:

Each elem of $\vec{a}$ is in $[0, d-1]$

Step 2: Monomial Lookup [BC'25]



+ 💍 $\mathrm{CM}(\mathrm{W})$

$(\mathrm{I} \otimes J) \times \mathrm{W}$

$\frac{n}{d}$

d

flatten

$\vec{a}$

n

If exist

$\vec{\Delta} \approx X^{\vec{a}}$

+ 💍 $\mathrm{CM}(\vec{\Delta})$

s.t.: $\forall i \in [n],$

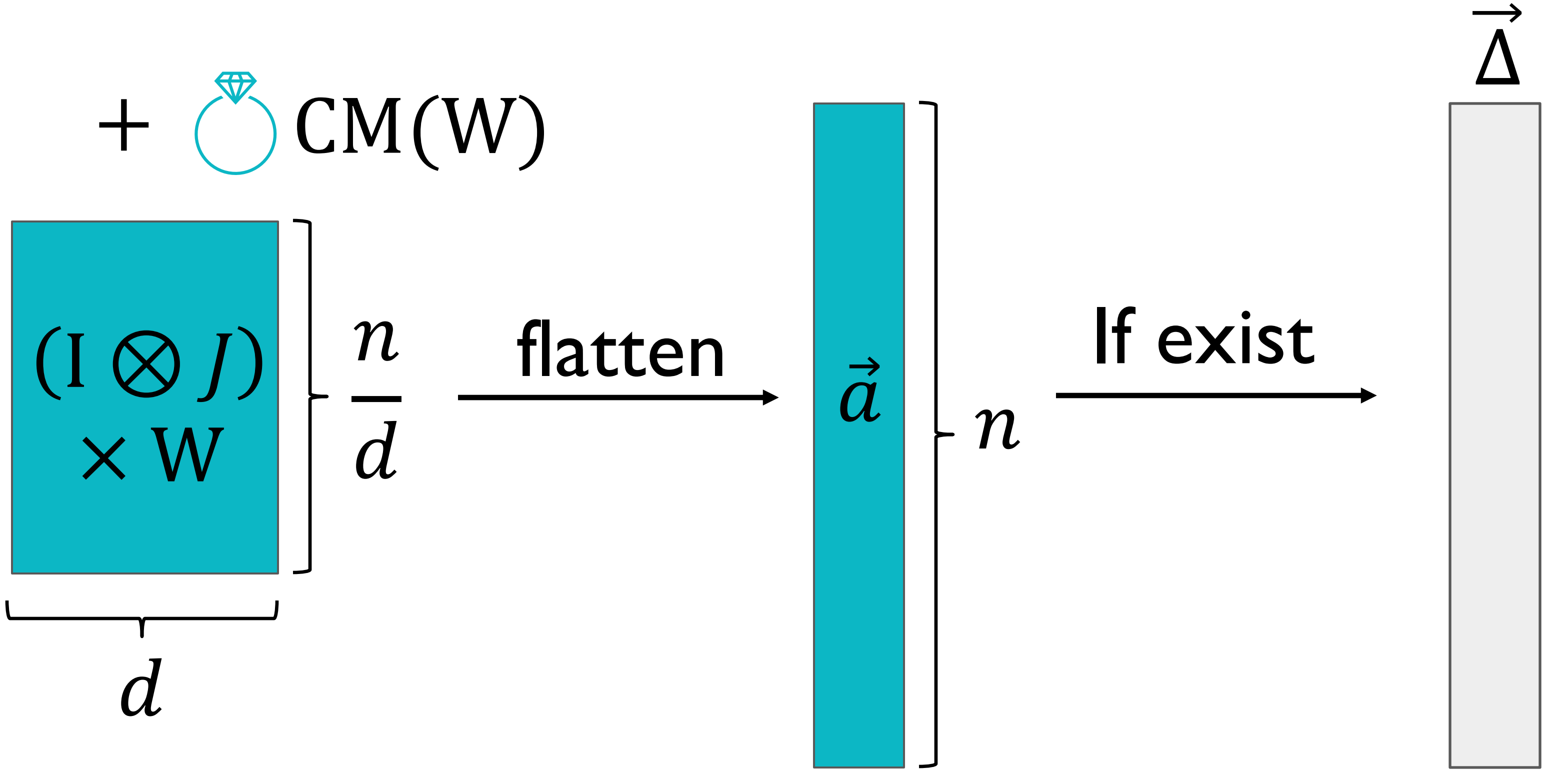
$\vec{a}_i = \mathrm{ct}(T(X) \cdot \Delta_i)$

$T(X) \approx \sum_{i=0..d-1} i \cdot X^i$

Simpler goal:

Each elem of $\vec{a}$ is in $[0, d-1]$

Step 2: Monomial Lookup [BC’25]


+ 💍 $\mathrm{CM}(\mathrm{W})$
$(\mathrm{I} \otimes J) \times \mathrm{W}$
$\frac{n}{d}$
d
flatten
$\vec{a}$
n
If exist
$\vec{\Delta} \approx X^{\vec{a}}$
+ 💍 $\mathrm{CM}(\vec{\Delta})$

s.t.: $\forall i \in [n]$,
$\vec{a}_i = \mathrm{ct}(T(X) \cdot \Delta_i)$

$T(X) \approx \sum_{i=0..d-1} i \cdot X^i$

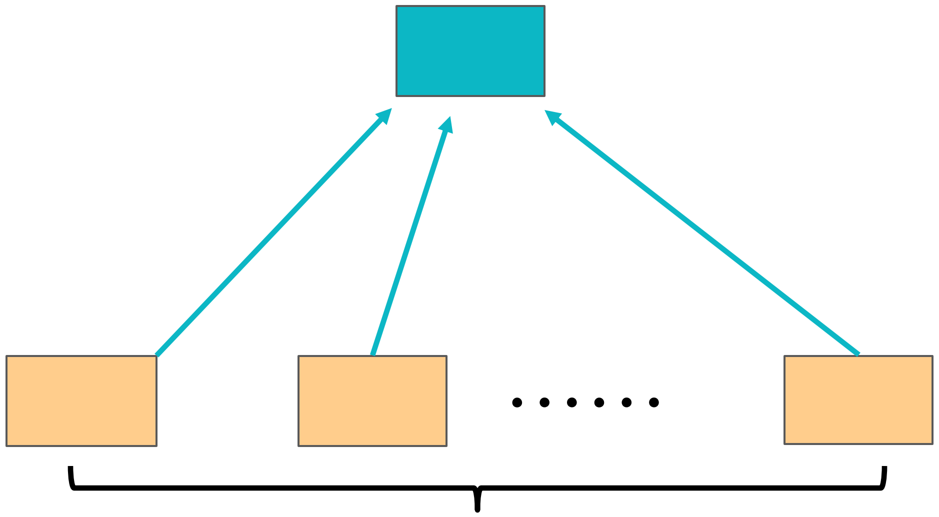
Simpler goal:
Each elem of $\vec{a}$ is in $[0, d-1]$

Efficiency [BC’25]
Prover: $O(n)$ R_q-add
Verifier: $O(d + \log(n))$ $\mathbb{F}$-ops

d times cheaper than [BC’25]

Summary & Extensions

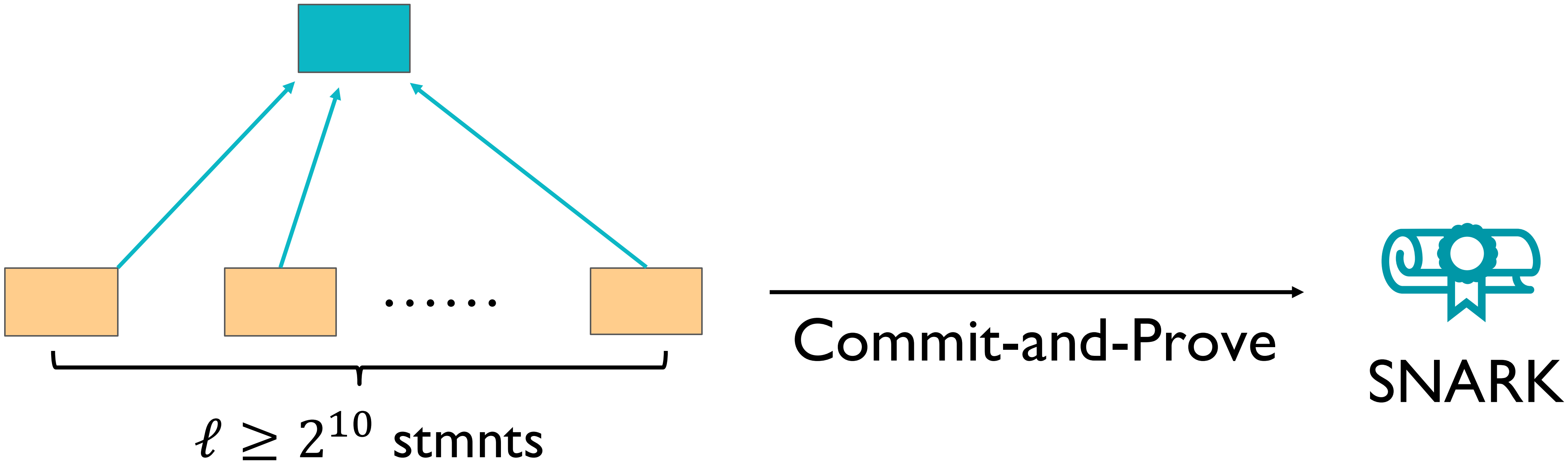
Lattice High-Arity Folding:



$\ell \geq 2^{10}$ stmnts

Summary & Extensions

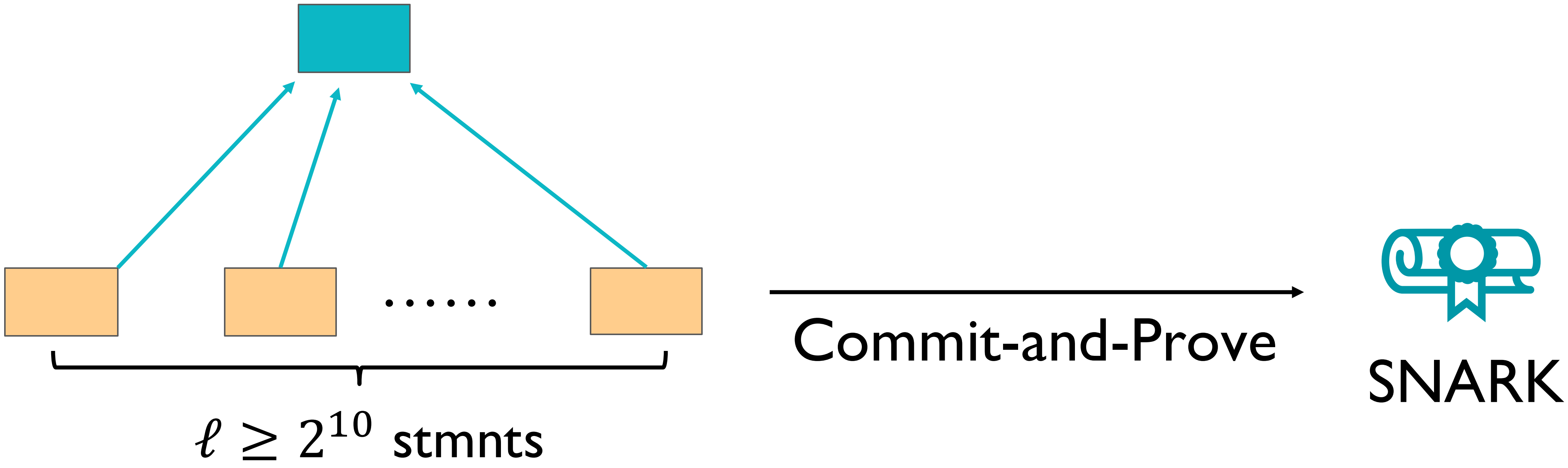
Lattice High-Arity Folding:



……
$\underbrace{\qquad\qquad}_{\ell \geq 2^{10} \text{ stmnts}}$
Commit-and-Prove
SNARK

Summary & Extensions

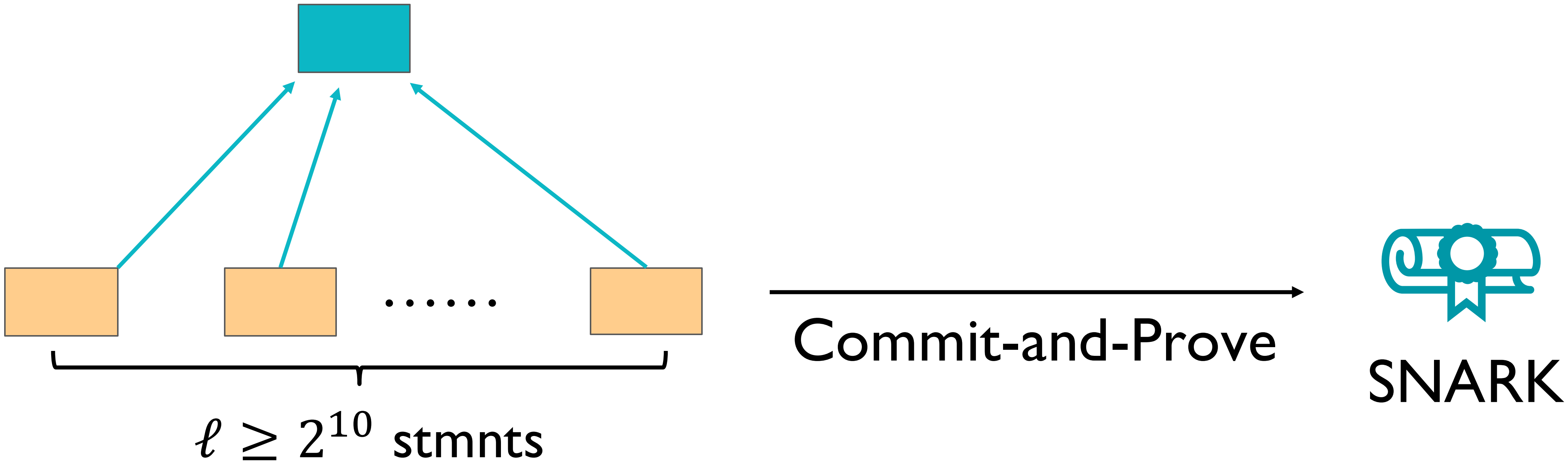
Lattice High-Arity Folding:



No FS circuit + Security in ROM

Summary & Extensions

Lattice High-Arity Folding:



$\underbrace{\qquad\qquad\qquad}_{\ell \geq 2^{10} \text{ stmnts}}$

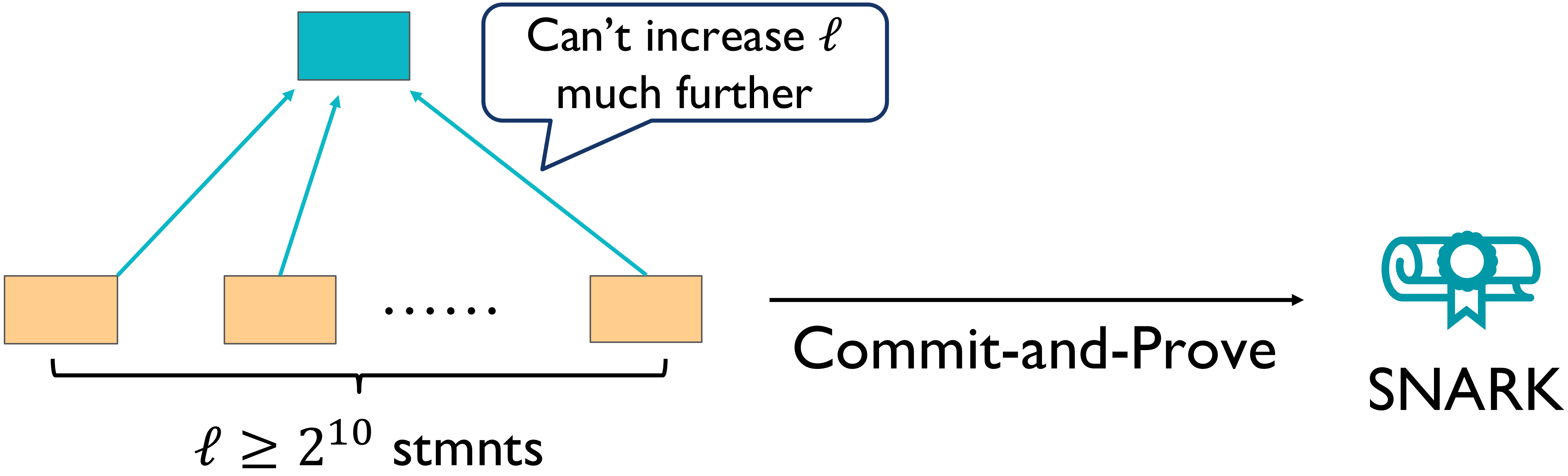
$\xrightarrow{\text{Commit-and-Prove}}$ SNARK

Verifier circuit: $\approx O(\ell)$ R_q-ops

No FS circuit + Security in ROM

Summary & Extensions

Lattice High-Arity Folding:



Can’t increase ℓ much further

$\ell \geq 2^{10}$ stmnts

Commit-and-Prove

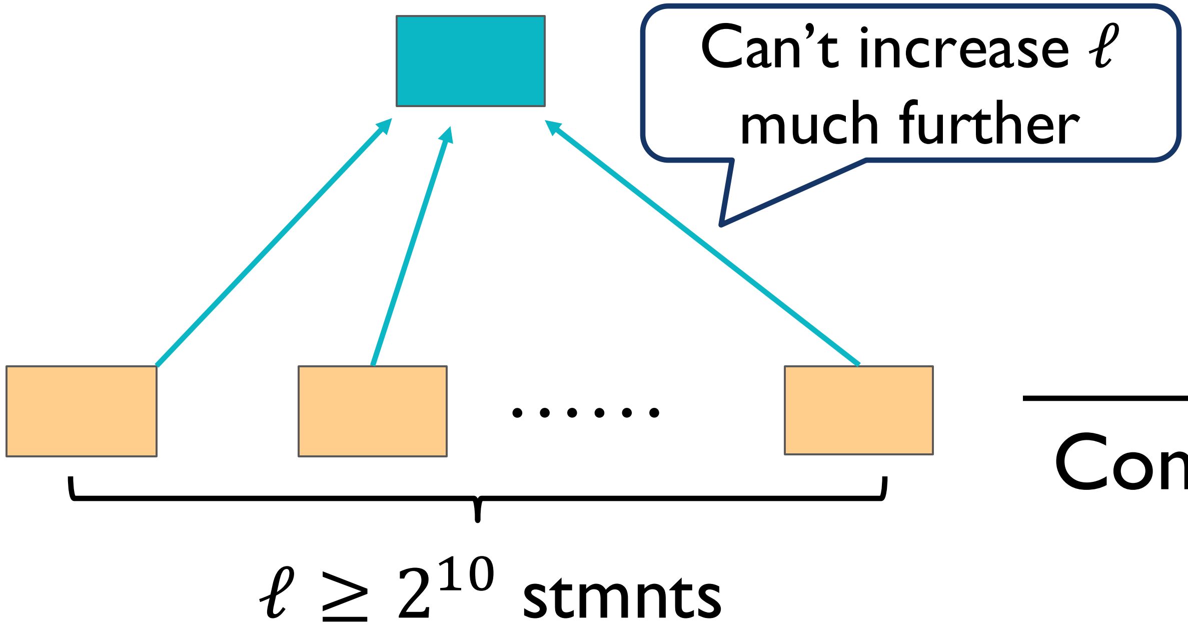
SNARK

Verifier circuit: $\approx O(\ell)$ R_q-ops

No FS circuit + Security in ROM

Summary & Extensions

Lattice High-Arity Folding:



Extension:

- Boosting:
  ℓ-folding → SNARK for ℓ^2-inputs
- Tradeoff: MSIS matrix $\mathrm{A} = \vec{r}^{\top} \otimes \mathrm{A}'$

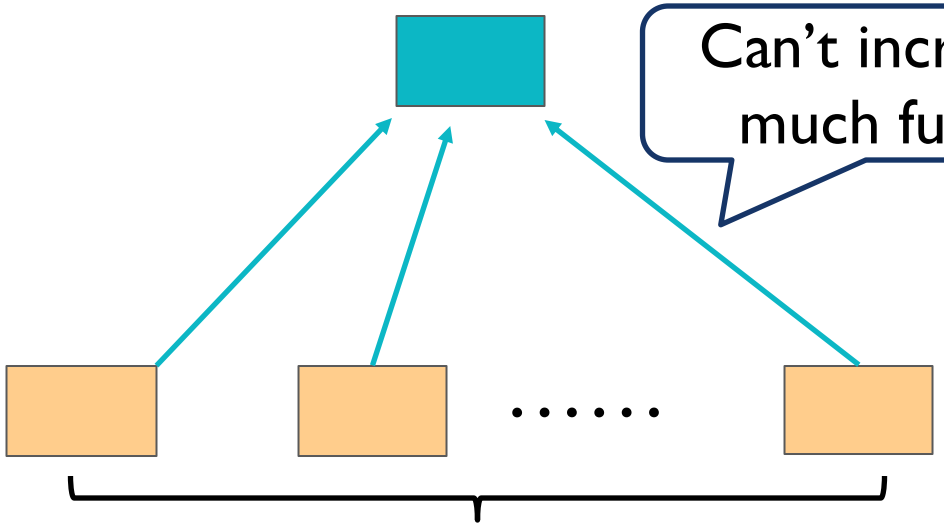
Commit-and-Prove → SNARK

Verifier circuit: $\approx O(\ell)$ R_q-ops

No FS circuit + Security in ROM

Summary & Extensions

Lattice High-Arity Folding:



Can't increase ℓ much further

$\ell \geq 2^{10}$ stmnts

Extension:

norm blowup & verifier circuit $\approx O(\ell)$

- Boosting:
  ℓ-folding → SNARK for ℓ^2-inputs
- Tradeoff: MSIS matrix $\mathrm{A} = \vec{r}^{\top} \otimes \mathrm{A}'$

$\xrightarrow{\text{Commit-and-Prove}}$ SNARK

Verifier circuit: $\approx O(\ell)$ R_q-ops

No FS circuit + Security in ROM

Open Problems

Symphony: $2 + \log\log(n)$ passes

One-pass small-memory prover without recursion

Arity boosting based on standard MSIS assumption

Folding verifier w/ $o(\ell)$-complexity

QROM analysis?

THANK YOU