

Akita: A High-Performance



Lattice-Based PCS

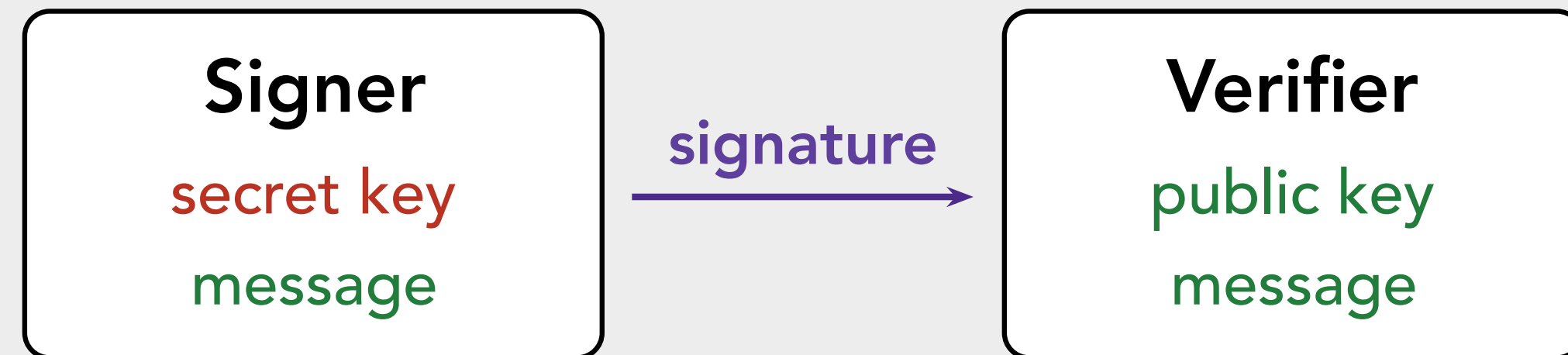
Quang Dao
CMU

Joint work with: Omid Bodaghi, Giuseppe Vitto, Amirhossein Khajepour, Taghi Badakhshan (LayerZero Labs), Fengrun Liu (CMU), Jiapeng Zhang (USC), Markos Georgiades (a16z crypto), and Justin Thaler (a16z crypto & Georgetown)

NYU CryptoSec Seminar, October 2026

From Signatures to SNARKs

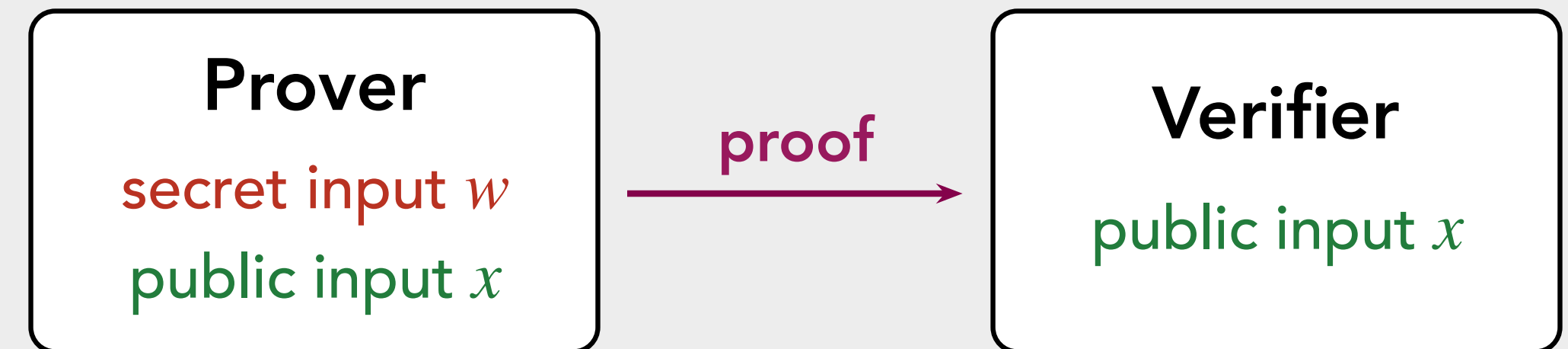
Digital signature



"I know the **secret key**"

a fixed statement, tied to the **message**

SNARK for a program or circuit P



"I know **w** such that $P(x, w)$ accepts"

any statement: P can be arbitrary

What (zk)SNARKs are used for

Scaling blockchains

rollups, zkVMs

compress validity of many transactions into one proof

Digital identity

Google Wallet, EU wallet

prove you are over 18 and reveal nothing else

SNARK: Succinct Non-interactive ARgument of Knowledge

Private payments

Zcash, Aztec

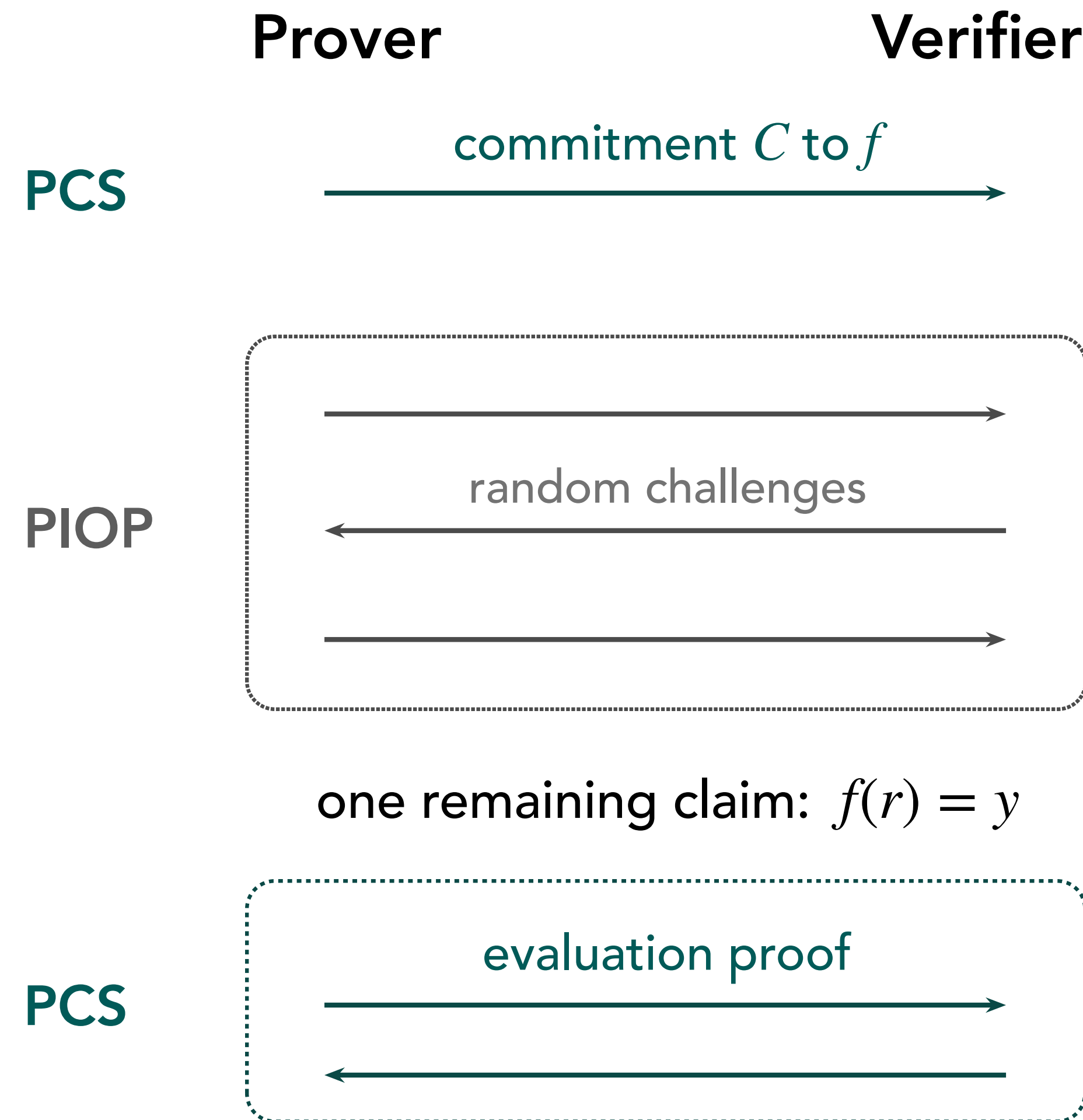
a valid payment, hiding who paid whom

Verifiable AI

Attestable, Lagrange, EZKL

prove which model produced an answer

SNARK = PIOP + PCS



Polynomial IOP (PIOP)

- the prover sends polynomials $f, \dots$ that encode the run
- the verifier checks them at a few random points
- **statistically sound**: no cryptographic assumption

Polynomial commitment scheme (PCS)

- a polynomial f is often too large to send in the clear
- **commit**: a short C stands in for f
- **open**: a short proof that $f(r) = y$
- **soundness** rests on a cryptographic assumption

All cryptographic assumptions are in the PCS: a **post-quantum** PCS gives a **post-quantum** SNARK

Post-Quantum Migration: Lattices versus Hashes

	elliptic curves	hashes	lattices
Signatures	Ed25519: 64 bytes not post-quantum	SLH-DSA: 8 to 50 KB NIST standard	ML-DSA, Falcon: 0.7 to 4.6 KB ML-DSA: NIST's primary standard
SNARKs	Groth16: 128 bytes not post-quantum	FRI, WHIR: 150 KB and up deployed: StarkWare, Succinct, ...	Greyhound [NS24]: 53 KB not deployed anywhere
Assumption	discrete log, pairings	a hash function, as a random oracle	Module-SIS, Module-LWE linear algebra with small numbers

Lattice-based signatures are the default. Why not lattice-based SNARKs?

Why Lattice-Based PCSs Are Not Deployment-Ready

	proof size	verifier at $N = 2^{30}$	assumption	code
Greyhound [NS24]	53 KB	$\sqrt{N}$: 2.8 s	Module-SIS	AVX-512 only
Hachi [NOZ26]	55 KB, estimated	$\sqrt{N}$: 227 ms, estimated	Module-SIS	one fold only
Jindo [HLSS26]	1.2 MB at 2^{20}	$N^{1/3}$: 14 ms at 2^{20}	Module-SIS*	yes
RoKoKo [KLNOT26]	157 KB	polylog: 13 ms	vanishing SIS*	yes
Grand Danois [KK26]	80 KB, estimated	polylog: n/a	vanishing SIS	none
Maltese [CNT26]	335 KB, estimated	polylog: n/a	Module-SIS	none

* with a heuristic challenge set

Deployment requires more than just performance:

- one proof for commitments made at different times; proving across many machines
- code that withstands review: mistakes were found, and since fixed, in LaBRADOR, Greyhound, RoKoKo

Can a **lattice-based** PCS have small proofs, a fast verifier, and security from Module-SIS?

Akita: A Production-Ready Lattice-Based PCS

Opening a polynomial with 2^{24} to 2^{30} coefficients

32-bit field, one thread, AMD Ryzen 9 9950X

58 to 64 KB

proofs

Greyhound*: 60 to 67 KB

7 to 14 ms

verification

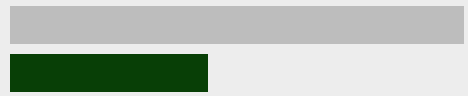
Greyhound*: 148 to 1,537 ms

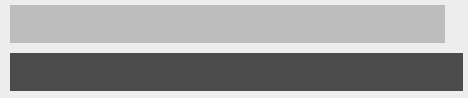
Module-SIS

standard assumption

1. Setup offloading verifier: $\sqrt{N} \rightarrow N^{1/K}$

the prover applies the commitment matrix, and proves it

verification 32 ms $\rightarrow$ **14 ms** 

proof size 61 KB $\rightarrow$ **64 KB** 

opening time 5.7 s $\rightarrow$ **8.4 s** 

with and without offloading, at 2^{30} coefficients

2. A leaner fold

each fold is cheaper to prove, smaller to send

commitment 1 to 8 KB $\rightarrow$ **128 bytes**

range check, prover $O(b^2L) \rightarrow$ **$O(bL)$**

evaluation check 1 element $\rightarrow$ **none sent**

compared with the folds of Greyhound and Hachi

* when tuned to the same lattice security level as Akita

First Application: Post-Quantum Jolt

Jolt is a zkVM: a SNARK for “this RISC-V program ran correctly”

What **Jolt** commits to: one-hot polynomials

0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

a single 1 per step of the program

- Dory, in **Jolt** today: pays only for the 1s, but rests on elliptic curves
- a hash-based PCS: post-quantum, but pays for every 0
- **Akita**: pays only for the 1s, and is post-quantum

Jolt with **Akita** in place of Dory

2.2 to 4.2×

faster prover

2.8 to 8.5×

faster verifier

under 100 KB

proofs

Roadmap

1. The Lattice-Based PCS Template
2. Akita and Its Core Techniques
3. Evaluation & Future Work

Roadmap

1. The **Lattice-Based PCS** Template
2. **Akita** and Its Core Techniques
3. **Evaluation & Future Work**

Polynomial Commitment Schemes

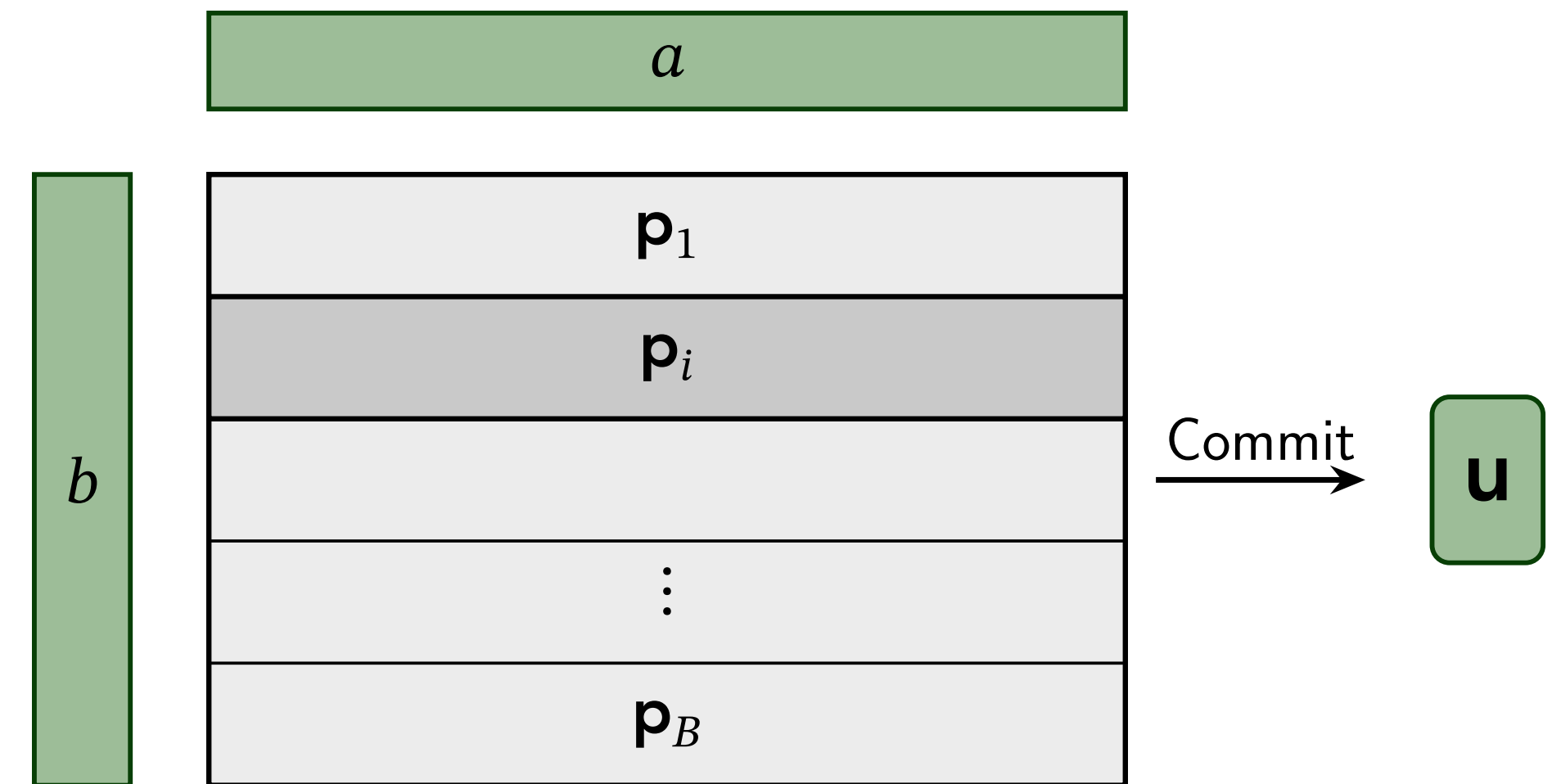
p : a multilinear polynomial, $N = 2^\mu$ evaluations in $\{0,1\}^\mu \subset \mathbb{F}_q^\mu$

private, and far too large to send

Public: a short commitment $\mathbf{u}$, a point x , a value y

Claim: $p(x) = y$ and $\mathbf{u} = \text{Commit}(p)$

Succinctness: proof size and verifier time far below N



the table of p : B rows of length M

Re-size p into a table, and leverage vector-matrix-vector form:

- the weights a and b are determined by the halves of x : $a = \bigotimes_j (1 - x_{L,j}, x_{L,j})$, $b = \bigotimes_i (1 - x_{R,i}, x_{R,i})$
- each row has a partial evaluation $e_i = \langle \mathbf{p}_i, a \rangle$, and $y = \sum_i b_i \cdot e_i$

Succinctness by Folding

Transform p into blocks of ring elements, folded into one using short randomness

$$\begin{array}{l} \boxed{\hat{\mathbf{p}}_1} \times c_1 \stackrel{\$}{\leftarrow} \text{Chal}(R_q) \\ \boxed{\hat{\mathbf{p}}_2} \times c_2 \stackrel{\$}{\leftarrow} \text{Chal}(R_q) \\ \vdots \\ \boxed{\hat{\mathbf{p}}_B} \times c_B \stackrel{\$}{\leftarrow} \text{Chal}(R_q) \end{array}$$

$$\mathbf{z} = c_1 \hat{\mathbf{p}}_1 + \cdots + c_B \hat{\mathbf{p}}_B$$

- Digit-decompose each row $\mathbf{p}_i$ into $(\hat{\mathbf{p}}_{i,j})_{j \in [\ell]}$
- B blocks of length $\ell \cdot M$, in short digits
- the verifier picks short ring challenges $c_1, \dots, c_B$
- $\mathbf{z}$ has the length of one block, and is still short

1. How can we transform p into R_q ?
2. How can we check $\mathbf{z} \in R_q^m$ is consistent with the PCS relation $p(x) = y$?

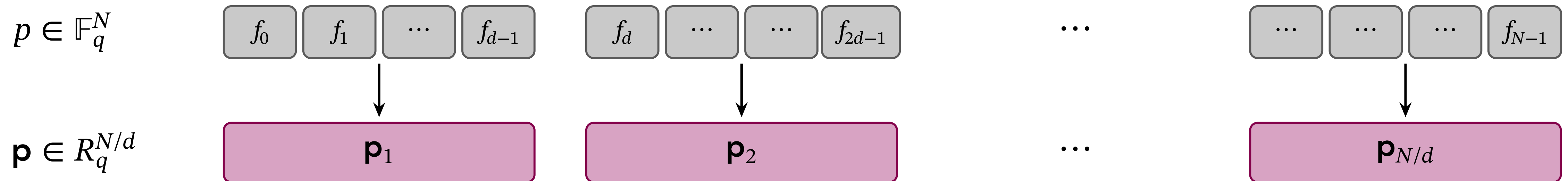
$R_q := \mathbb{F}_q[X]/(X^d + 1)$ is the cyclotomic ring with $d = 2^k$

$\text{Chal}(R_q)$ is the ring challenge set: low norm, invertible pairwise differences

($d = 128$: 31 random entries are -1 or 1 , the rest are zero; R_q must not split too much)

PCS Evaluation: From Field to Ring

Coefficient packing: d field values are the coefficients of one ring element



$\mathbf{p}_1 = f_0 + f_1X + \dots + f_{d-1}X^{d-1}$ in $R_q = \mathbb{F}_q[X]/(X^d + 1)$: the same data, d times fewer entries

An inner product over $\mathbb{F}_q$ is the constant term of a ring product:

$$\langle f, g \rangle = \text{ct}(\mathbf{f} \cdot \sigma_{-1}(\mathbf{g})), \quad \sigma_{-1} : X \mapsto X^{-1}$$

$$\begin{aligned} & (f_0 + f_1X + \dots + f_{d-1}X^{d-1}) \\ & \cdot (g_0 - g_{d-1}X + \dots - g_1X^{d-1}) \\ & = \langle f, g \rangle + \dots \cdot X + \dots \end{aligned}$$

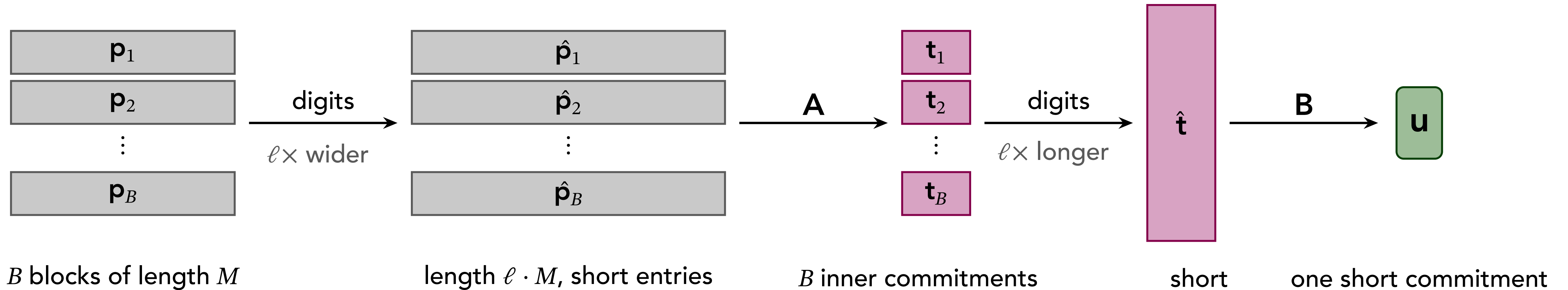
The claim $p(x) = y$ becomes a claim over R_q :

$$\mathbf{e}_i = \langle \mathbf{p}_i, \mathbf{a} \rangle \text{ for each block, } y = \text{ct} \left(\left(\sum_i b_i \mathbf{e}_i \right) \cdot \sigma_{-1}(\mathbf{o}) \right) \quad \text{since } X^d = -1$$

$\mathbf{a}, \mathbf{b}, \mathbf{o}$ are fixed by the point x

Two-Tier Commitments

$\mathbf{u} = \text{Commit}(p)$, computed in two tiers of Ajtai commitments



- **Ajtai:** $\mathbf{A} \cdot \mathbf{s}$ binds only a short $\mathbf{s}$ (Module-SIS) $\implies$ every input is first decomposed in small digits
- **Inner tier:** one matrix $\mathbf{A}$ for every block $\implies$ a combination of blocks has the same combination of the t_i
- **Outer tier:** the t_i are too many to send $\implies \mathbf{B}$ binds all of them in one short $\mathbf{u}$

$\mathbf{A}$ and $\mathbf{B}$ are random matrices in the public parameters, can be chosen to be prefixes of a seed-derived setup

Ring Relations and Norm Checks

The verifier has u, x, y and the challenges; $z = \sum_i c_i \hat{p}_i$

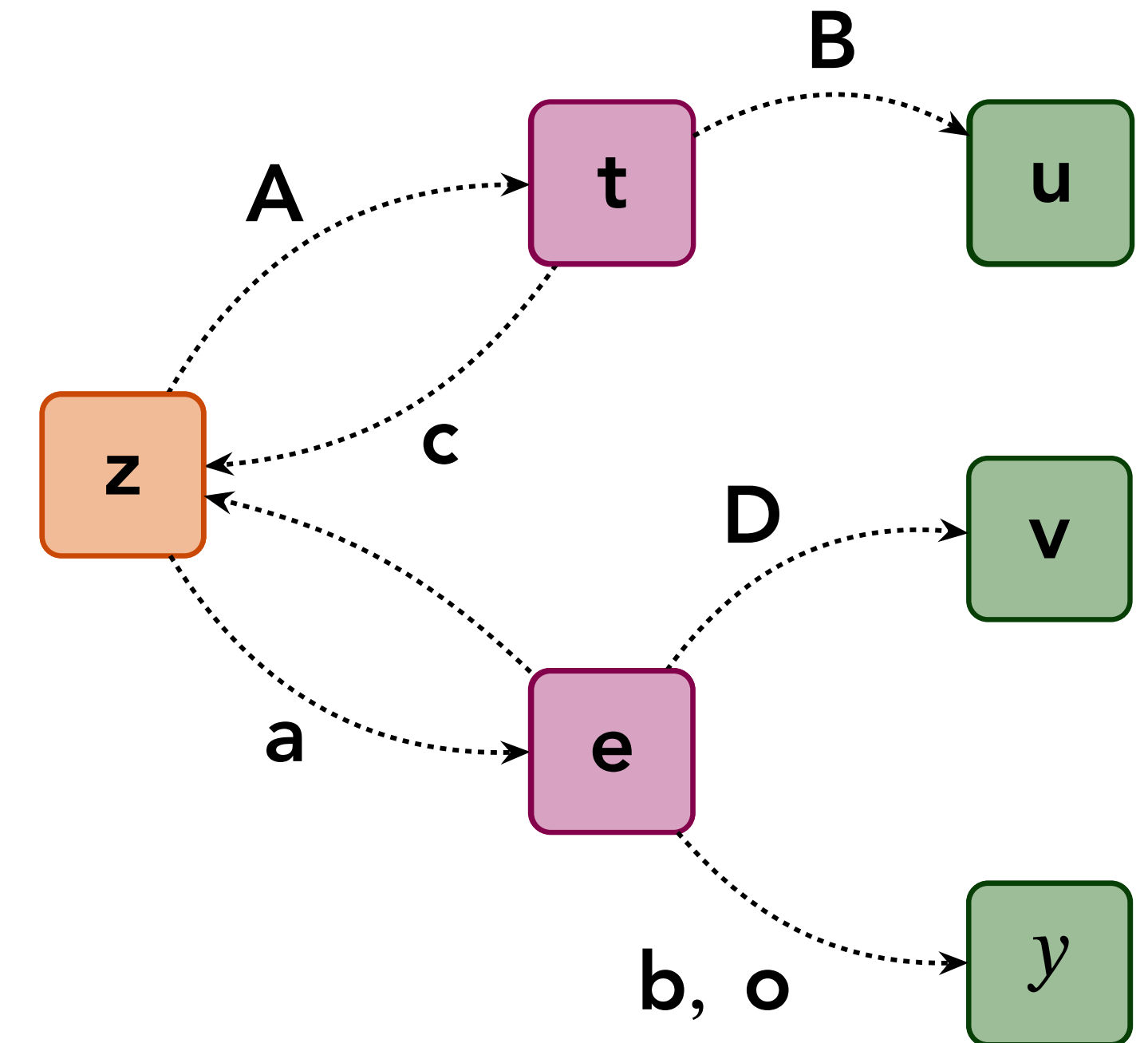
z folds the committed blocks:

$$\mathbf{A} \mathbf{z} = \sum_i c_i \mathbf{t}_i \quad \mathbf{u} = \mathbf{B} \hat{\mathbf{t}}$$

z agrees with the partial evaluations $\mathbf{e}_i = \langle \mathbf{p}_i, \mathbf{a} \rangle$:

$$\mathbf{a}^\top \mathbf{G} \mathbf{z} = \sum_i c_i \mathbf{e}_i \quad \mathbf{v} = \mathbf{D} \hat{\mathbf{e}}$$

$\mathbf{v}$ is sent before the challenges c_i



the partial evaluations give y :

$$y = \text{ct}\left(\sum_i \mathbf{b}_i \mathbf{e}_i \cdot \sigma_{-1}(\mathbf{o})\right)$$

Alternatively: prove smallness in L_2 norm

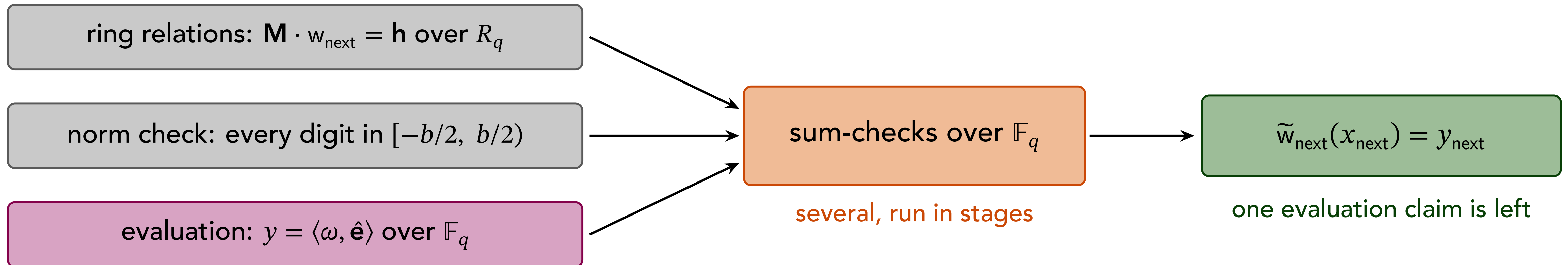
Norm check: every digit of $\hat{\mathbf{z}}, \hat{\mathbf{t}}, \hat{\mathbf{e}}$ lies in $[-b/2, b/2)$

Needed in security proof, since commitments only bind short vectors under Module-SIS.

z grows in the fold, so it is decomposed again: $\hat{\mathbf{z}}, \hat{\mathbf{t}}, \hat{\mathbf{e}}$ enter the witness of the next fold.

Proving the Fold with Sum-Checks

A sum-check-based PIOP for the committed witness $w_{\text{next}} = (\hat{z}, \hat{t}, \hat{e}, \hat{r})$



Ring switching [HMZ25, NOZ26]: from ring relations to field relations

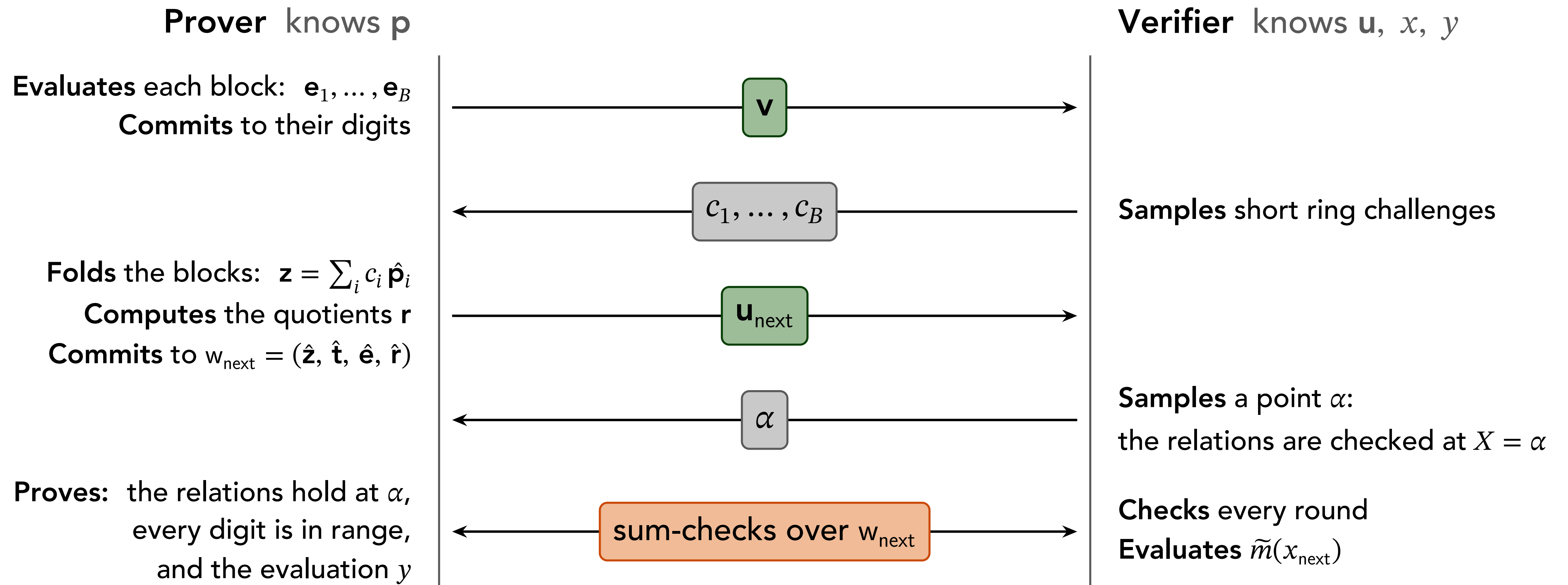
- an equation modulo $X^d + 1$ is a polynomial identity with a quotient $\mathbf{r} \implies$ check it at a random $X = \alpha$
- the quotient joins the witness; every sum-check runs over $\mathbb{F}_q$, not over the ring

Evaluation, proved on the digits (an **Akita** optimization)

- Hachi sends the ring element $\sum_i b_i \mathbf{e}_i$, about 1-4 KB per fold, and checks its constant term
- **Akita** sends no ring element: an inner-product sum-check proves $y = \sum_x \tilde{\omega}(x) \cdot \tilde{\mathbf{e}}(x)$

One PCS Folding Step

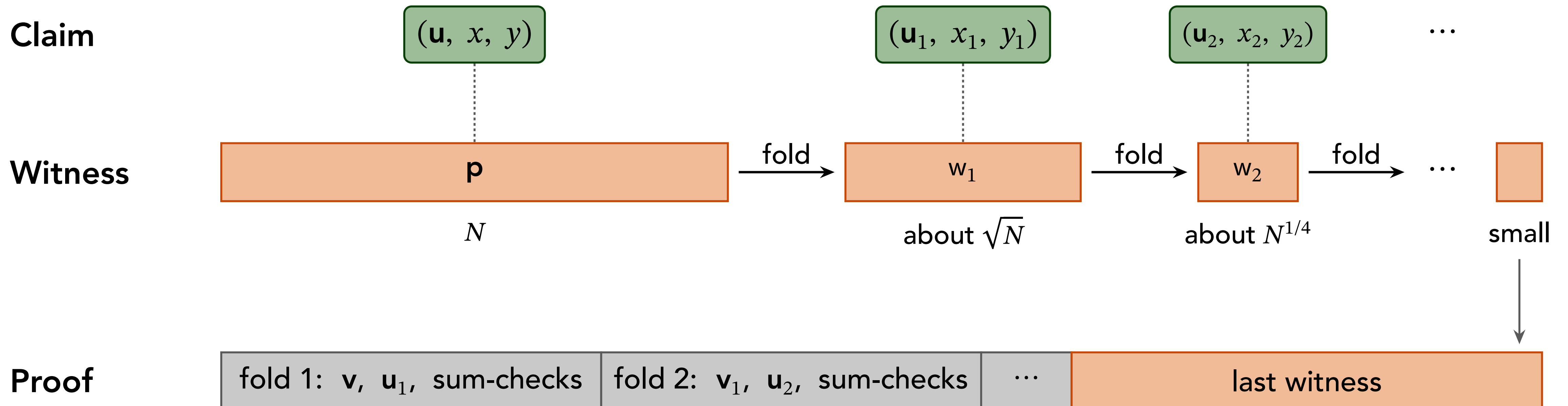
PCS claim: $p(x) = y$, for the polynomial committed in u



What is left: $\tilde{w}_{\text{next}}(x_{\text{next}}) = y_{\text{next}}$, for the witness committed in u_{next}

The Whole PCS Protocol

We fold until the witness is small, then send it in full



- Each fold: B blocks of length N/B leave a witness of about $N/B + B$, so $B = \sqrt{N}$
- The proof stays short: each fold adds two commitments and its sum-check messages
- The first fold is the largest: main contributor to prover and verifier time

Roadmap

1. The Lattice-Based PCS Template
2. **Akita** and Its Core Techniques
3. Evaluation & Future Work

PCS Fold: Bottlenecks and Improvements

Akita introduces new techniques to handle each downside of the folding template:

Verifier time

the verifier evaluates the commitment matrices itself:
about $B + N/B$ ring operations, at best $\sqrt{N}$



Technique 1:
Setup Offloading

Norm check

digits in base b requires a single sum-check of degree $b + 1$



Technique 2:
(Multi-Stage)
Digit Range Check

Commitment size

every fold sends two commitments of several kilobytes



Technique 3:
Compressed Commitments

... and more!

The Verifier's Work in One Fold

The last sum-check of a fold proves a sum over the next witness: m : the rows of the matrix, batched

$$\sum_x \widetilde{w}_{\text{next}}(x) \cdot \widetilde{m}(x) = \text{claimed value}$$

$$\boxed{\widetilde{w}_{\text{next}}(x_{\text{next}})} \cdot \boxed{\widetilde{m}(x_{\text{next}})} = \text{last sum-check message}$$

the claim for the next fold

the verifier computes this itself

Folding requires generating challenges $c_1, \dots, c_B$ Other $O(\log N)$ sum-check rounds are cheap.

$\Rightarrow$ the verifier needs to balance the costs: evaluating $\widetilde{m}(x_{\text{next}})$ & processing ring challenges

Technique 1: Setup Offloading

Where the time goes: $\widetilde{m}(x_{\text{next}})$, for a witness of B blocks of length N/B

$$\begin{bmatrix} \hat{\mathbf{z}} & \hat{\mathbf{t}} & \hat{\mathbf{e}} & \hat{\mathbf{r}} \\ 0 & \boxed{\mathbf{B}_\alpha} & 0 & \boxed{-(\alpha^d+1)\mathbf{G}_\alpha} \\ 0 & 0 & \boxed{\mathbf{D}_\alpha} & \boxed{-(\alpha^d+1)\mathbf{G}_\alpha} \\ \boxed{(\mathbf{A}\mathbf{G})_\alpha} & \boxed{(-\mathbf{c}^\top \otimes \mathbf{G})_\alpha} & 0 & \boxed{-(\alpha^d+1)\mathbf{G}_\alpha} \\ \boxed{(-\mathbf{a}^\top \mathbf{G}\mathbf{G})_\alpha} & 0 & \boxed{(\mathbf{c}^\top \otimes \mathbf{G})_\alpha} & \boxed{-(\alpha^d+1)\mathbf{G}_\alpha} \\ 0 & 0 & \boxed{\omega_{\text{eval}}} & 0 \end{bmatrix}$$

 structured: gadgets, evaluation points

 folding challenges $c_1, \dots, c_B$

 setup matrices $\mathbf{A}, \mathbf{B}, \mathbf{D}$: random

$O(\log N)$

$O(B)$

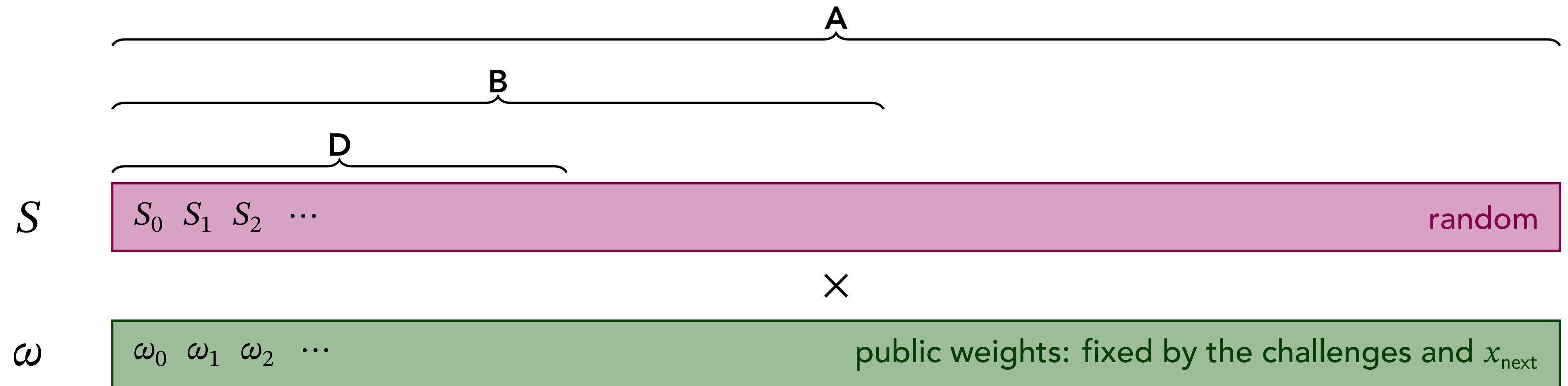
$O(N/B + B)$

Total: $B + N/B \geq 2\sqrt{N}$. Fewer blocks mean fewer challenges, but a wider $\mathbf{A}$.

Polylog verification is possible with structured matrices, but relies on the vanishing-SIS assumption.

Technique 1: Setup Offloading

A, B, D are prefixes of one random sequence S , expanded from a seed



$$\tilde{m}(x_{\text{next}}) = \boxed{\text{structured}} + \boxed{\text{challenges}} + \boxed{\sigma} \quad \text{where } \sigma = \sum_p S_p \cdot \omega_p$$

- The verifier computes the structured and challenge terms itself: $O(\log N + B)$
- The prover sends σ , the setup term; it remains to prove that σ is correct

Technique 1: Setup Offloading

A sum-check reduces σ to one evaluation of S

View S and ω as multilinear polynomials:

$$\sigma = \sum_{p \in \{0,1\}^v} \tilde{S}(p) \cdot \tilde{\omega}(p) \xrightarrow{\text{sum-check}} \boxed{\tilde{S}(\rho)} \cdot \boxed{\tilde{\omega}(\rho)}$$

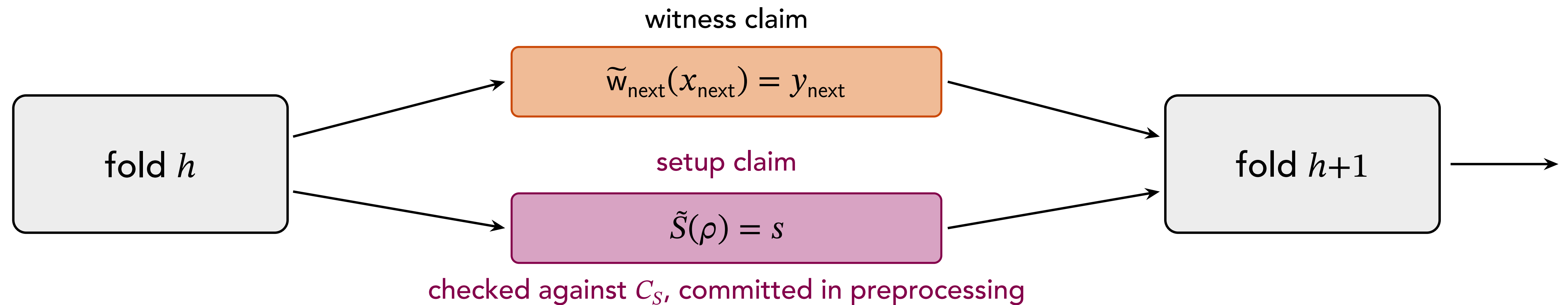
random: computing it is the scan again structured: cheap

But S is fixed at setup: commit to it once, in preprocessing: $C_S = \text{Commit}(S)$

$\tilde{S}(\rho) = s$ is an evaluation claim for our own PCS

Technique 1: Setup Offloading

The setup claim joins the next fold



- The next fold opens w_{next} and S together, batched by a random challenge
- S is around the size of w_{next} : the next fold grows by a constant factor
- The next fold(s) can offload again, until it is no longer profitable to offload

Technique 1: Setup Offloading

Wide folds: $B = N^{1/K}$ blocks in each of the first $K - 1$ folds

Template

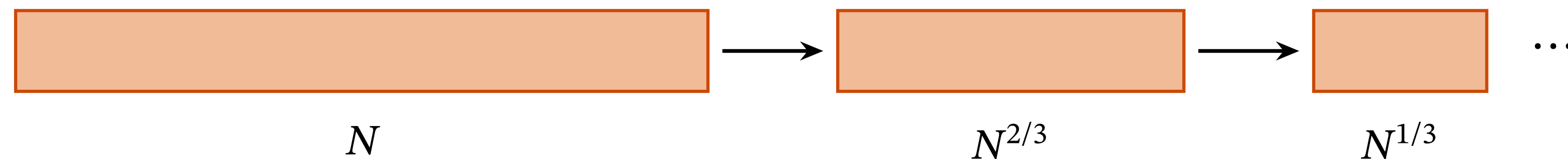
$$B = \sqrt{N}$$



verifier: $\sqrt{N}$

Akita, $K = 3$

$$B = N^{1/3}$$



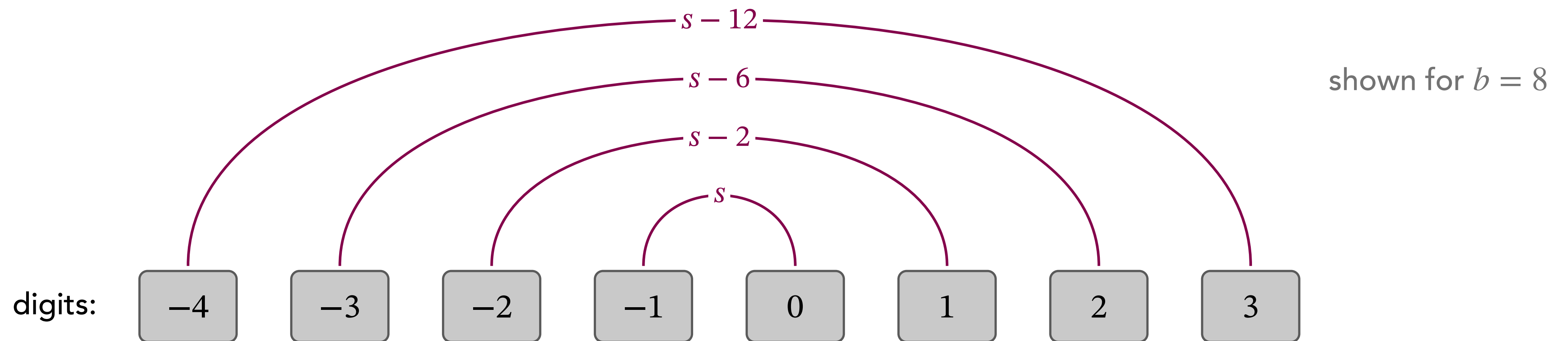
verifier: $N^{1/3}$

- Verifier time $\widetilde{O}(N^{1/K})$ for any constant K ; prover $\widetilde{O}(N)$ and proof size as before
- At 4 GiB over 32-bit field: verification 32 ms to 14 ms; opening 5.7 s to 8.4 s, proof +2.7 KB
- A planner offloads only in the folds where it pays

Technique 2: Digit Range Check

Every digit of the witness must lie in $\{-b/2, \dots, b/2 - 1\}$

Template: $\prod_{a=-b/2}^{b/2-1} (w - a) = 0$ at every digit w : an equality-weighted sum-check of degree $b + 1$

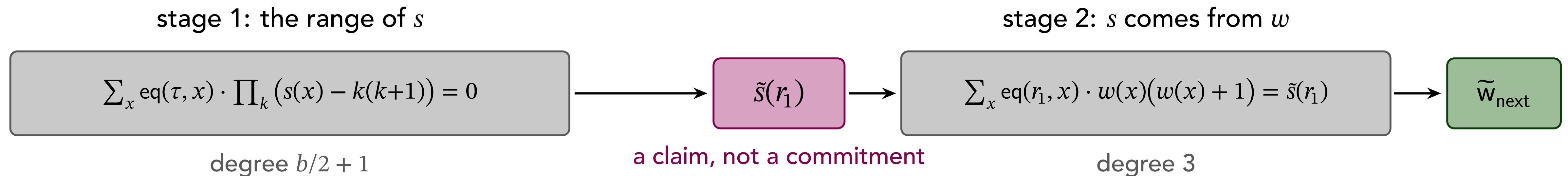


Pair the digit k with $-(k + 1)$: $(w - k)(w + k + 1) = s - k(k + 1)$, where $s = w(w + 1)$

The check becomes $\prod_{k=0}^{b/2-1} (s - k(k + 1)) = 0$: half the degree, in the variable s

Technique 2: Digit Range Check

Two or more sum-checks of low degree in place of one of high degree



- s is never committed: stage 2 reduces its claim to the witness, inside the relation sum-check of the fold
- Large bases, $b \geq 16$: stage 1 splits again, as a product tree with two or four children per node
- Prover: w and s take few, small values, so most multiplications are cheap

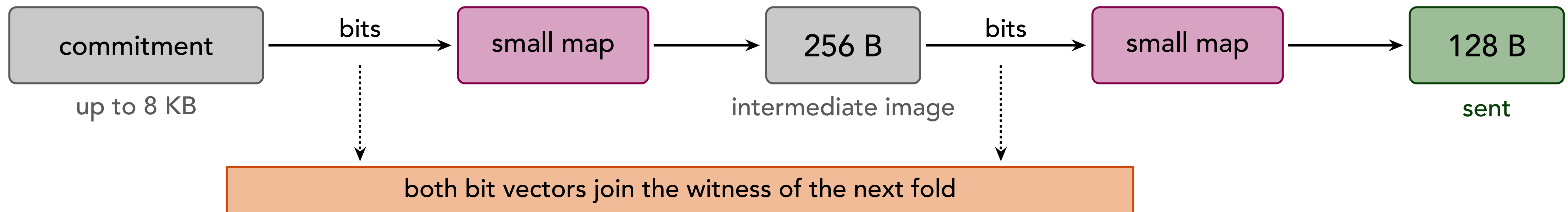
Sum-check messages of the range check, 2^{20} digits, baseline vs. optimized:

$b = 16$: 5.1 KB to **2.2 KB**

$b = 32$: 10.2 KB to **2.9 KB**

Technique 3: Compressed Commitments

Each fold sends two commitments, v and u_{next} : usually 1 KB, up to 8 KB



- **Small output:** bits give the smallest collision norm, so Module-SIS allows the shortest output
- **Two-stage compression:** SIS security depends on the input width, so a wide commitment is absorbed gradually
- Bits in $\{-1, 0\}$ satisfy $w(w+1) = 0 \implies$ fused into the range check of Technique 2
- This adds to the next witness $\implies$ compression is used in the early folds, where the witness is large

Every compressed commitment is **128 bytes**, the polynomial commitment included.

Orthus [\[BBLMS26\]](#) uses the same idea with a single compression step: its commitments are more than twice as large.

Roadmap

1. The Lattice-Based PCS Template
2. *Akita* and Its Core Techniques
3. Evaluation & Future Work

Evaluation: Setting & Methodology

How does **Akita** compare with other **post-quantum** PCSs?

How much better is **Jolt** with **Akita**?

Each scheme has its own field. For a fair comparison, all commit the same amount of data, measured in bits:

bits committed = field elements × bits per element

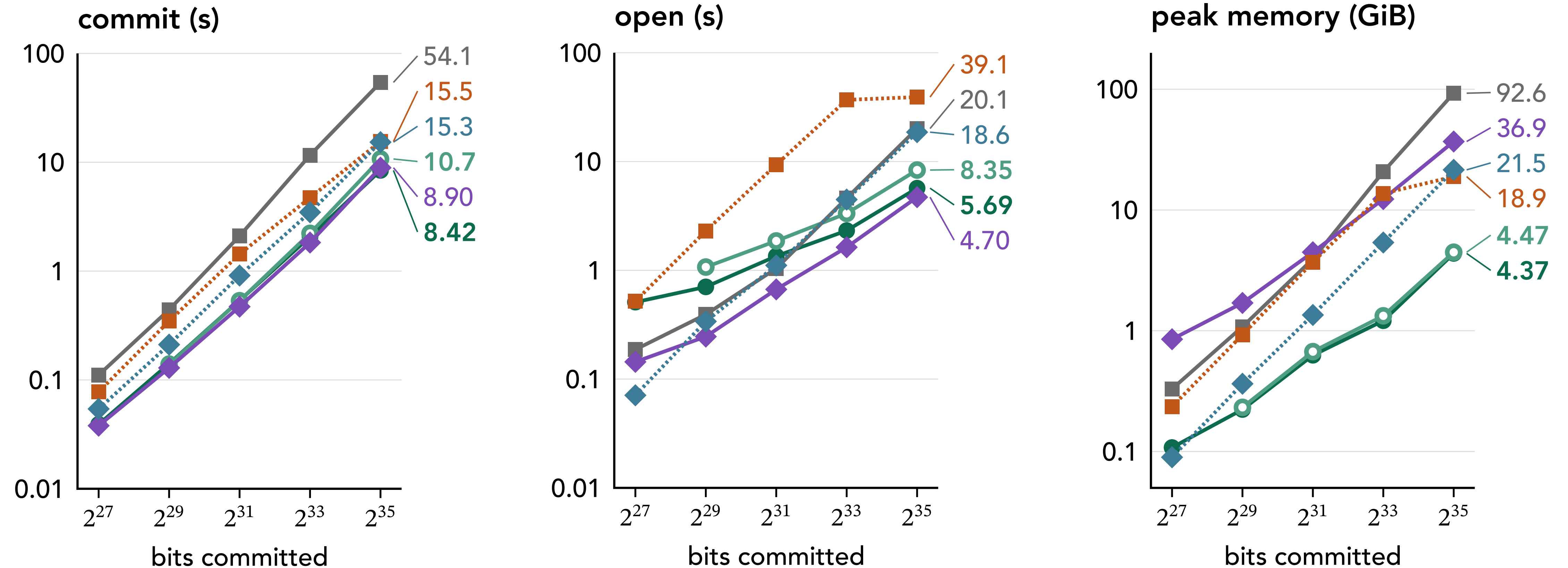
scheme	implementation	security	field order	elements at 2 ³⁵ bits
Akita	ours, in Rust	128-bit, Module-SIS	2 ³² – 99	2 ³⁰
Greyhound	reference code, plus our fixes	128-bit, Module-SIS *	2 ³² – 99	2 ³⁰
RoKoKo	authors' code and parameters	100-bit, vanishing SIS	2 ⁵⁰ – 2687	2 ³⁰ , 31-bit values
WHIR	Plonky3, rate 1/2	128-bit, Johnson bound	2 ³¹ – 2 ²⁴ + 1	2 ³⁰
Ligerito	Flock, default fast profile	128-bit, Johnson bound	2 ¹²⁸	2 ²⁸

* The implementation we test uses an optimization that requires a non-standard assumption, truncated Ring-SIS, not pure Module-SIS.

Evaluation: PCS Prover

One thread (AMD Ryzen 9 9950X, AVX-512); value at 2^{35} bits next to each line

● Akita ○ Akita, offloading ■ Greyhound ◆ RoKoKo ■ WHIR ◆ Ligerito

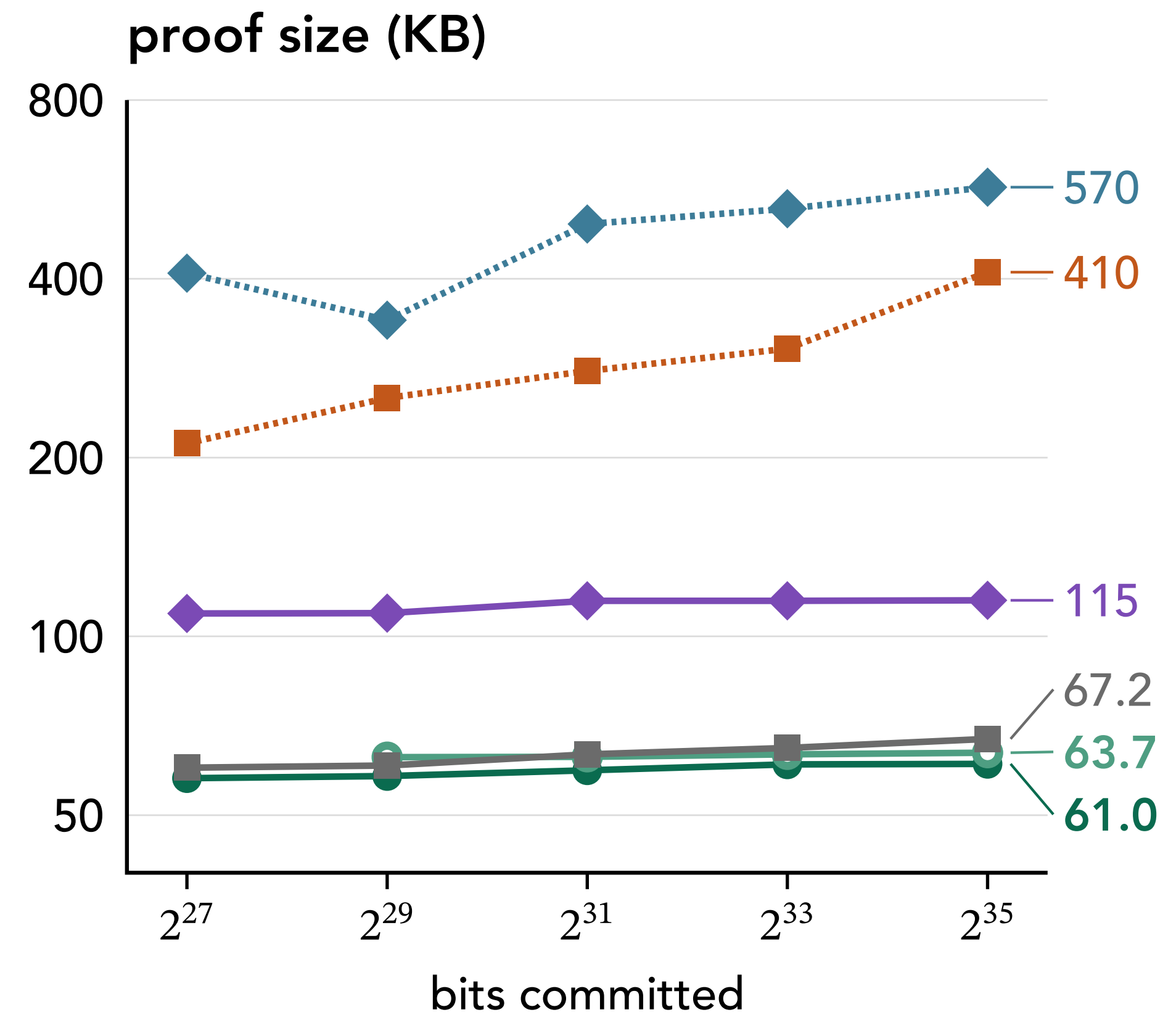
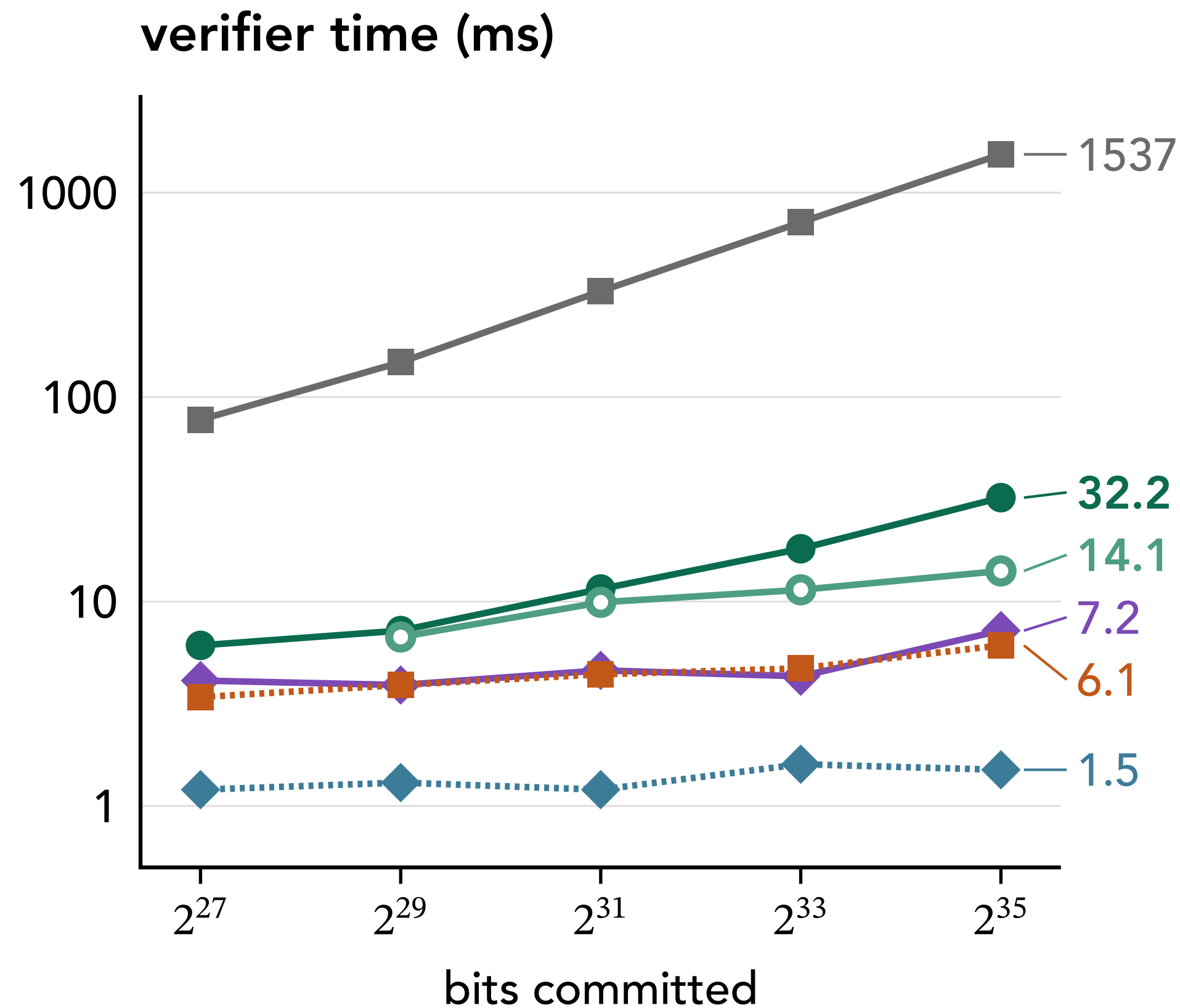


Median of ten runs. Commit and open are timed separately.

Evaluation: PCS Verifier and Proof Size

One thread (AMD Ryzen 9 9950X, AVX-512); value at 2^{35} bits next to each line

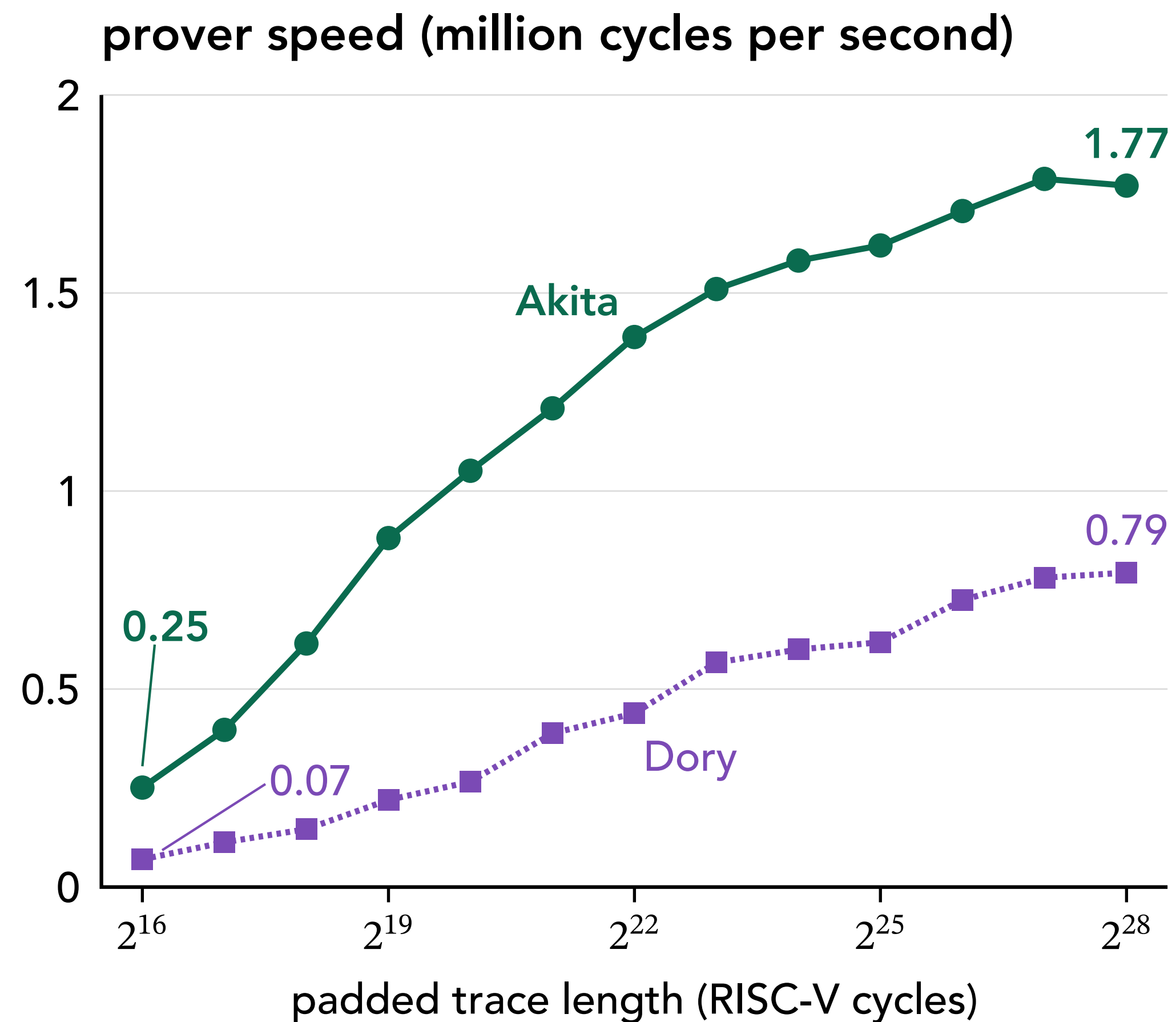
● Akita ○ Akita, offloading ■ Greyhound ◆ RoKoKo ■ WHIR ◆ Ligerito



Median of ten runs. Proof size excludes the commitment.

Evaluation: Jolt Prover

Jolt with Akita against Jolt with Dory; SHA-256 chains, Apple M4 Max (16 cores)



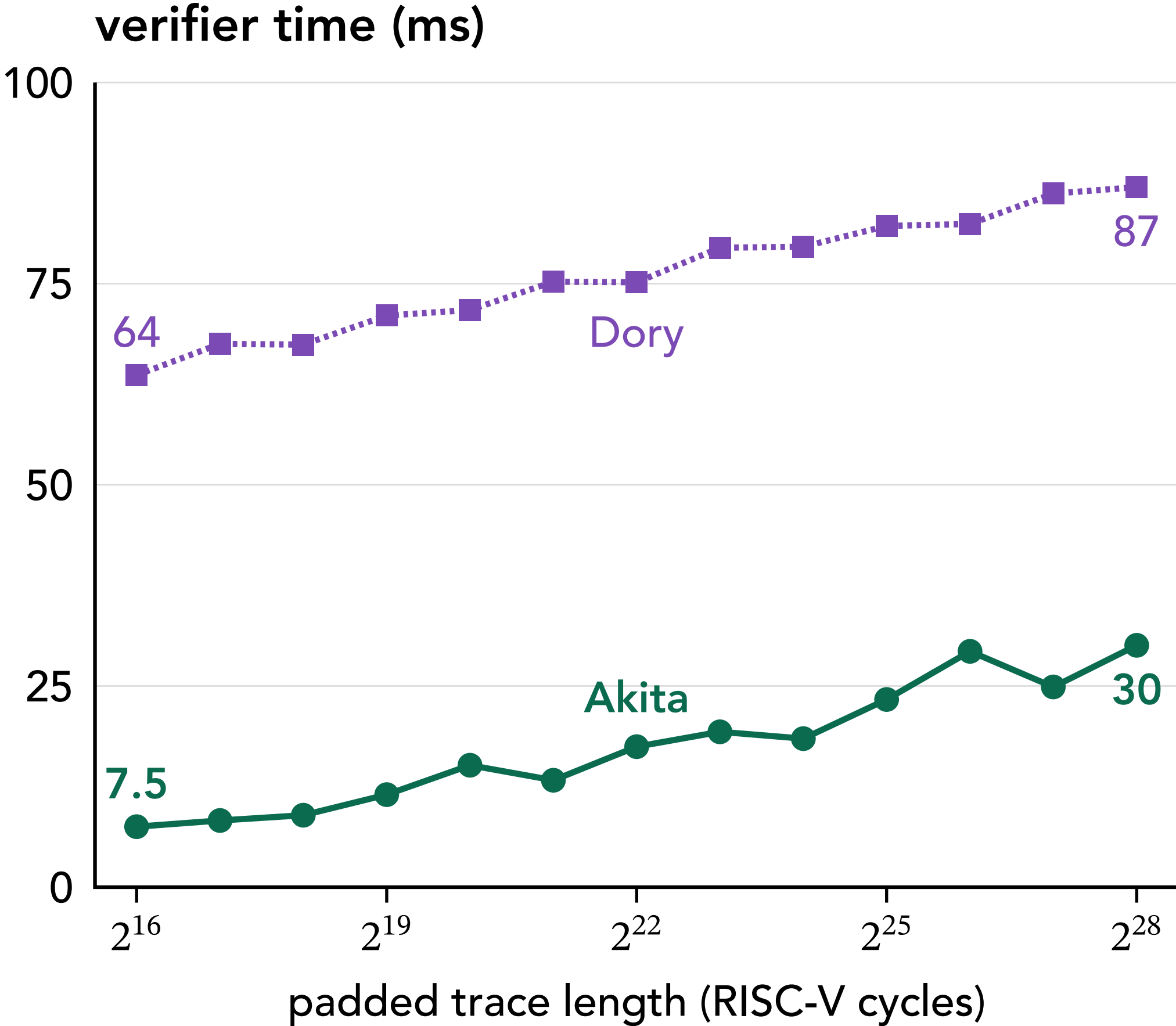
proving time

cycles	Akita	Dory	speedup
2^{16}	0.26 s	0.93 s	3.6×
2^{19}	0.59 s	2.38 s	4.0×
2^{22}	3.02 s	9.56 s	3.2×
2^{25}	20.7 s	54.3 s	2.6×
2^{28}	152 s	338 s	2.2×

Median of three runs. Proving time excludes program preprocessing, PCS setup, and execution tracing.

Evaluation: Jolt Verifier and Proof Size

One thread; SHA-256 chains, Apple M4 Max



cycles	verifier (ms)		proof size (KB)	
	Akita	Dory	Akita	Dory
2^{16}	7.5	63.6	80.5	78.4
2^{19}	11.5	71.0	84.4	86.2
2^{22}	17.5	75.2	89.5	91.4
2^{25}	23.3	82.2	91.5	91.8
2^{28}	30.1	87.0	94.1	96.6

AI in the Making of Akita

Protocol design ideas from conversations with AI models

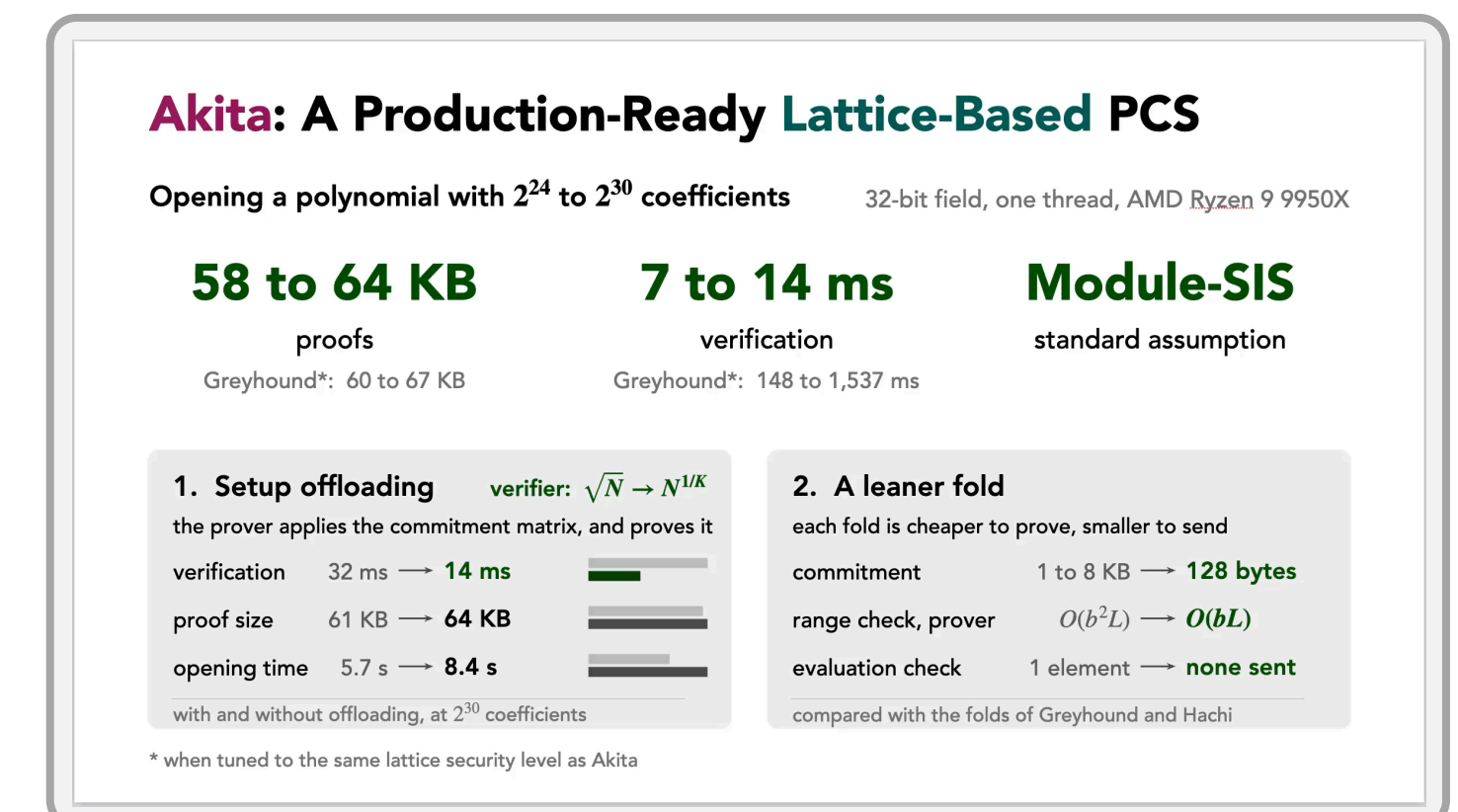
- embedding the evaluation claim from the field into the ring
- leaving the last block out of the final opening

Implementation written, reviewed and tuned with AI

- codebase tuned for AI contributions: spec, implement, review
- autoresearch loops: commit and prove **2× faster** in two weeks

Presentation these slides

- each slide is a script, written by Claude (Opus 5.5)
- very fast iterations, higher-quality final results



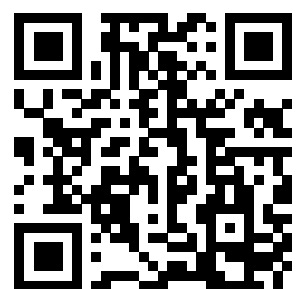
Summary and Future Work

Takeaways

- setup offloading: a verifier below $\sqrt{N}$
- proofs of **58 to 64 KB**, verified in **7 to 14 ms**
- security from **Module-SIS** alone
- one-hot friendly: **2.2 to 4.2×** faster **Jolt** prover

Future work

- zero knowledge; binary fields [OS26]
- **lattice** signature aggregation [AABKT24, BBLMS26]
- verifiable OPRF: Beasley [LWZ26]
- verifiable FHE [ABPS24, WWX+25, DT26]



Code

github.com/LayerZero-Labs/akita



Paper

eprint.iacr.org/2026/1983

Thank you!