Aerie

Fast Post-Quantum Signature Aggregation with Lattice SNARKs

Alberto Centelles
Imperial College London
a16z crypto research

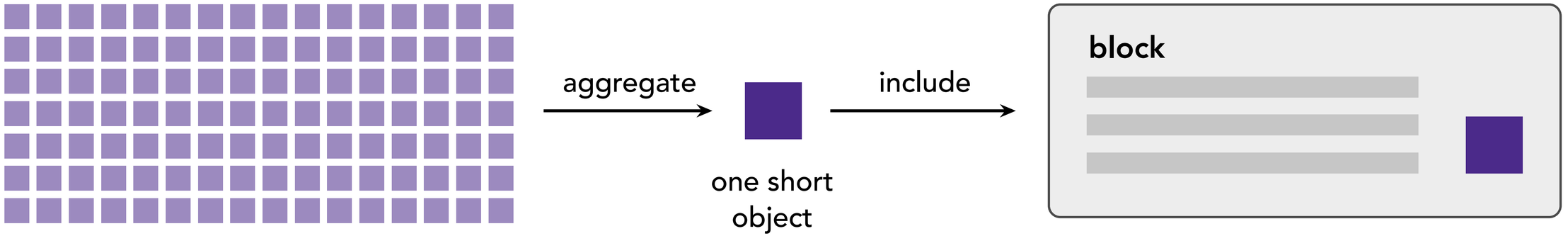
Quang Dao
Carnegie Mellon University

Mohammadtaghi Badakhshan
LayerZero Labs

Justin Thaler
a16z crypto research
Georgetown University

Ethereum PQ Workshop, Cambridge, October 2026

Signature Aggregation in Proof of Stake



one Ethereum slot: about 32,000 signatures

Today: BLS

- any number of signatures combine into one group element, verified with one pairing check
- not post-quantum, and standard post-quantum schemes cannot aggregate this way

The post-quantum route: a SNARK proving that every signature verifies

The Hash-Based Route

Ethereum's plan: hash-based signatures (XMSS), aggregated by a hash-based SNARK (leanVM)

100s

of hash evaluations
to verify one signature

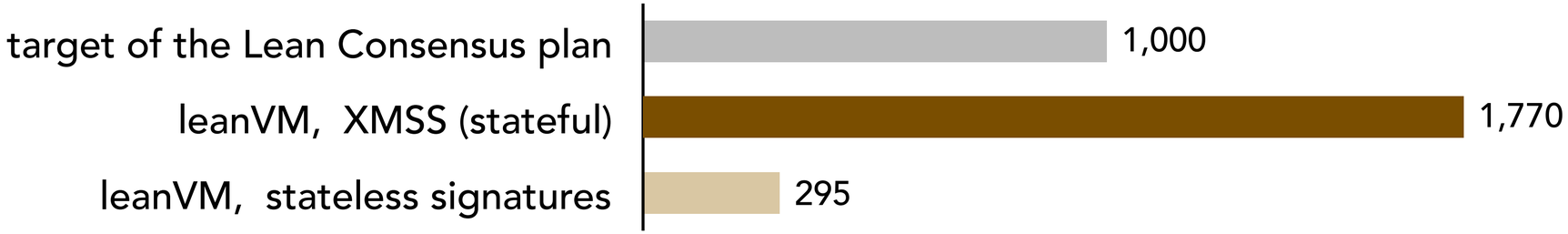
> 100×

to prove a hash,
compared with computing it

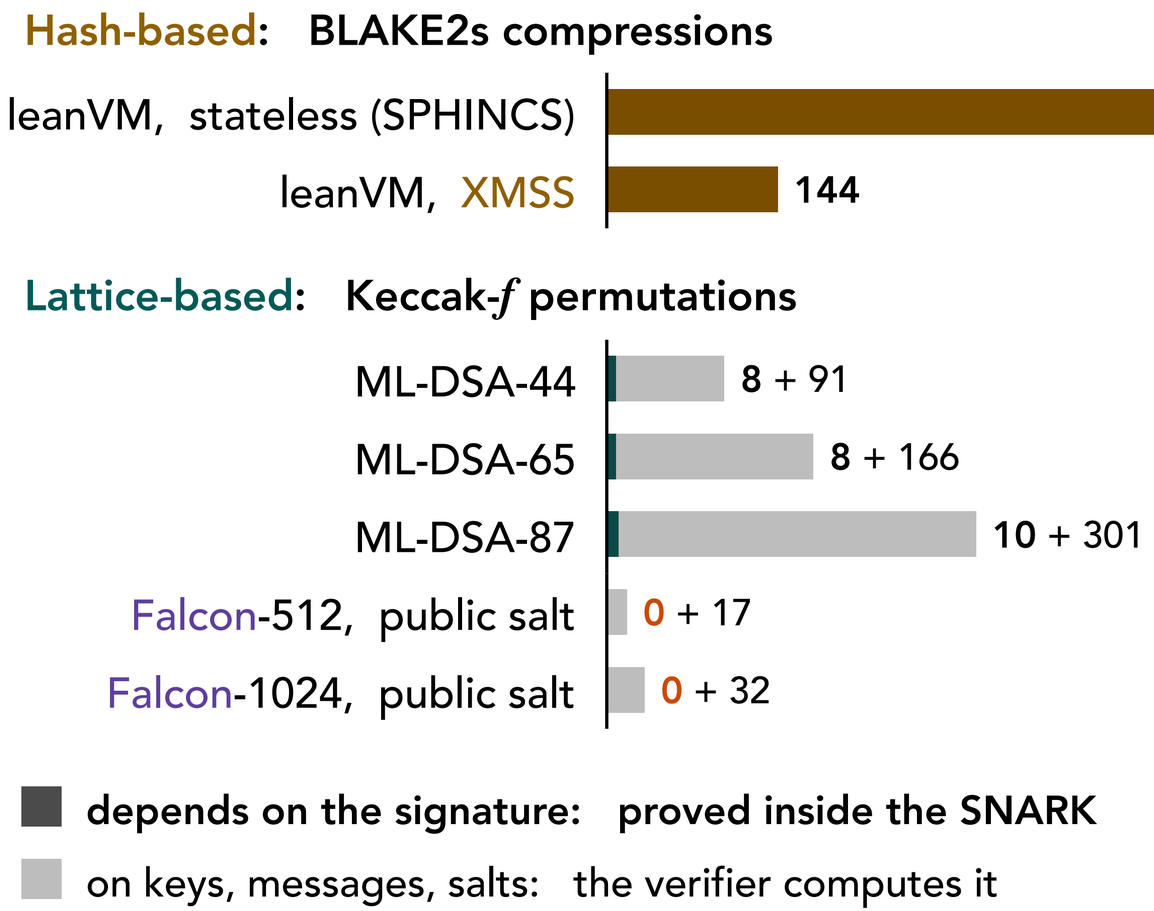
≈ 900

signatures per proof:
a slot needs recursion

Signatures aggregated per second



How Much Hashing Is in a Signature?



per signature, 32-byte message; leanVM as of October 2026

Hash-based

every hash depends on the signature:
hundreds to prove

Lattice-based

one hash call on the signature;
rest of verification is algebra & norm

Falcon with public salts

its one hash takes only public inputs:
nothing to hash inside the proof

a Keccak-*f* costs about twice a BLAKE2s compression to prove

The Lattice Route: Falcon with Public Salts

hash-and-sign over NTRU lattices · NIST's FN-DSA · signatures: 666 bytes (Falcon-512), 1,280 (Falcon-1024)

1. Hash to a point

$$c = \mathsf{HashToPoint}(r \,\|\, H(h) \,\|\, m)$$

r: salt, $H(h)$: key hash, m: message

the verifier computes this itself

2. Check a short lattice vector

$$s_1 = c - h \cdot s_2 \in \mathbb{Z}_q[X]/(X^n + 1)$$

$$\|s_1\|^2 + \|s_2\|^2 \leq \lfloor \beta^2 \rfloor$$

h: public key, s_2: the signature, $q = 12289$, $n = 512$ or 1024

the SNARK proves this: algebra & norm checks

How much salt is in an aggregated proof?

one salt per signature

unchanged signing: 40 bytes each, 1,280 KiB at 32K

one salt per slot, or deterministic Falcon

40 bytes for the whole batch, or none

Algorand deploys deterministic Falcon, Solana is considering to adopt Falcon

Aggregating Falcon: Prior Work

	signatures	proving	verifying	
LaBRADOR [BS22, AABKT24]	proof-size estimates; no end-to-end implementation			
LaZer [LSS24, NKS25]	10,000	5.96 s	2.65 s	restricted statement
Orthus [BBLMS26]	32,768	16.8 s	2.9 s	restricted statement
no aggregation, native verify	32,768		0.43 s	a single CPU core (M4)

Falcon-512; every number is single core, on its own machine

Proving and verifying both still take seconds

Can we aggregate one slot of Falcon signatures in about a second, with a verifier much cheaper than checking them one by one?

Aerie: Falcon Aggregation in Under a Second

32,768 Falcon-512 signatures: about one Ethereum slot of attestations

0.65 s

proving

50,800 signatures per second

66 KiB

proof

plus a 40-byte salt per batch, or per signature

54 ms

verification, one core

8× faster than native verification

Prover overhead

one core, against native Falcon

10× verifying them

1.4× signing them

peak memory: 2.1 GiB

Against Orthus [BBLMS26]

one core of the same AVX-512 server

1.5× faster proving

38× faster verification

1.7× smaller proof

Against leanVM

hash-based, on the same laptop

29× signatures per second

one proof per slot: **no recursion**

Apple M4 Max laptop, 16 threads, no GPU; medians of 7 runs

What Aerie Proves, and Why It Is Fast

The statement

- standard Falcon-512 and Falcon-1024 signatures (from the v1.2 specification)
- all of verification, including the canonical padded encoding (e.g., fits in 666 bytes)

Security

- knowledge soundness at 128 bits, in the classical random oracle model
- aggregate unforgeability from that of Falcon, with no loss in the batch size
- no recursion: avoid heuristic instantiation of the random oracle

Speed

- Akita PCS for commitments and openings; sum-check techniques from Lattice Jolt
- built for public-salt Falcon: **no hash proving, efficient ring equation & norm checks**

Roadmap

1. The Falcon Relation
2. The Aerie Proof System
3. Evaluation & Outlook

Roadmap

1. The Falcon Relation
2. The Aerie Proof System
3. Evaluation & Outlook

The Falcon Batch Relation

Public: keys h_i, messages m_i, salts r_i $\Rightarrow$ the verifier computes $c_i = \mathrm{HashToPoint}(r_i \,\|\, H(h_i) \,\|\, m_i)$

Witness: $s_{1,i},\ s_{2,i} \in \mathbb{Z}[X]/(X^n + 1)$ and $e_i \in \mathbb{Z}^4$, for every signature i

1. Ring equation

$$s_{1,i} + h_i \cdot s_{2,i} \equiv c_i \pmod{q}$$

$q = 12289$, product mod $X^n + 1$

2. Norm

$$\|s_{1,i}\|^2 + \|s_{2,i}\|^2 + \|e_i\|^2 = \lfloor \beta^2 \rfloor$$

Lagrange four-square theorem

3. Encoding

$$|s_{2,i,j}| \le 2047, \qquad \sum_j \left\lfloor |s_{2,i,j}|/128 \right\rfloor \le L$$

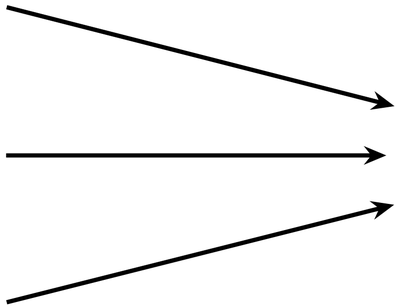
fits the 666-byte signature

Rough size of the computation:

32,768 signatures $\times$ 512 coefficients $=$ 2^{24} coefficients in every polynomial

Three Conditions, One Field

1. **Ring equation** modulo q and $X^n + 1$
2. **Norm** ℓ_2 bound on (s_1, s_2)
3. **Encoding** ℓ_∞ and ℓ_1 bounds on s_2



the proof system works over

$$\mathbb{F}_p, \qquad p = 2^{64} - 59$$

challenges from $\mathbb{F}_{p^2}$

Different moduli: the ring equation is modulo q and $X^n + 1$, the proof system modulo $p \gg q$

$$s_1 + h \cdot s_2 \;=\; c \;+\; (X^n + 1) \cdot t \;+\; q \cdot k \qquad \text{over } \mathbb{Z}[X]$$

standard: commit to both quotients, t for $X^n + 1$ and k for q

Norm bounds: over the integers; checking same equations over $\mathbb{F}_p$ can overflow

standard: range-check every committed coefficient, the quotients included

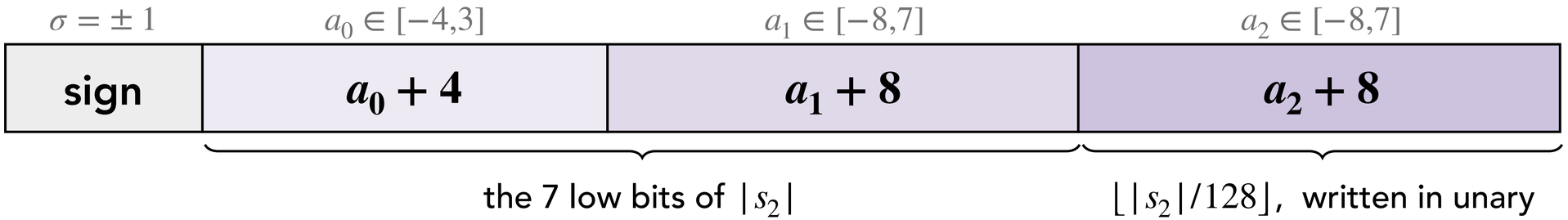
Aerie: **no committed quotients**; range checks on **small digits** only; the rest from **random projections**

Roadmap

1. The Falcon Relation
2. The Aerie Proof System
3. Evaluation & Outlook

Committing to s_2 in Falcon's Own Digits

One coefficient of s_2 in the padded signature: a sign, seven low bits, the high part in unary

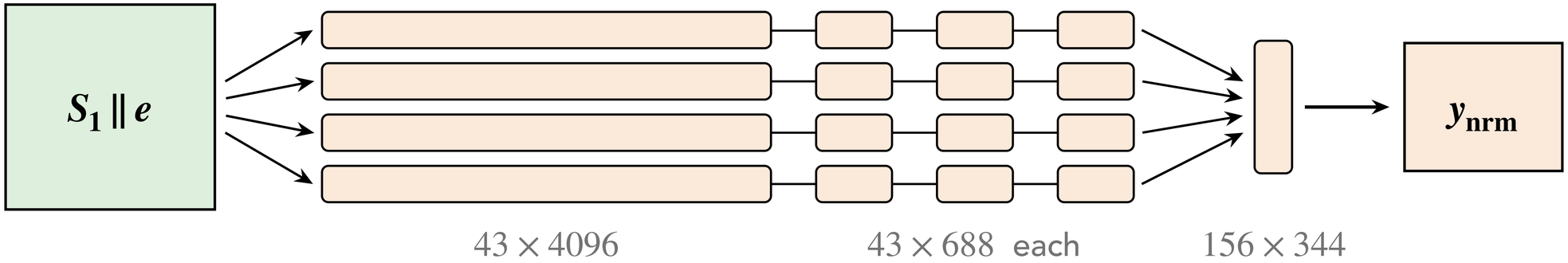


$$M = (a_0 + 4) + 8\,(a_1 + 8) + 128\,(a_2 + 8), \qquad S_2 = \sigma \cdot M$$

- four committed tables of small digits; M and S_2 are **virtual**: determined by them, never committed
- range-check the digits $\Rightarrow$ $0 \le M \le 2047$: S_2 is bounded by construction
- the same digits give the encoding length: $\sum_j (a_2 + 8) + u = L$, unused budget $u \ge 0$ also in digits
- S_1 and e have no digits in the format: committed directly, as arbitrary field elements

The Norm: A Random Projection in the Clear

Needed: a bound on the total squared norm of S_1 and e, not digits for each entry (up to 6144 and 8191)



sizes: rows × columns, at 32K signatures

The verifier checks, over the integers: $y_{\text{nrm},1}^2 + \ldots + y_{\text{nrm},156}^2 \leq \Theta$

squared norm of (S_1, e) at least $E_\star$: the check passes with probability at most 2^{-136}

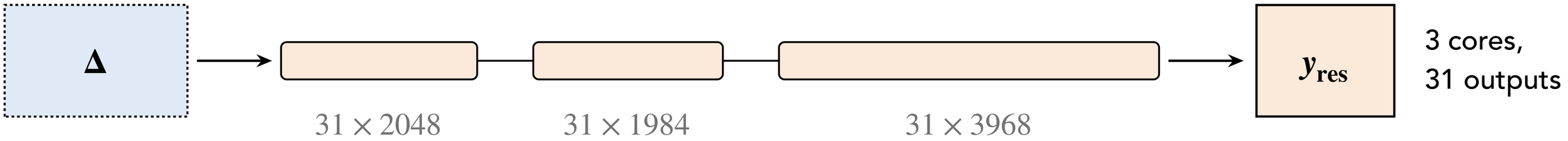
below $E_\star$: the squared norm of every signature is below p ⇒ **the norm identity holds over $\mathbb{Z}$**

- Improved efficiency & tightness over prior works; no union bound loss [Osa26, GP26]
- The projection theorem is checked in Lean 4 (formalization public soon)

The Ring Equation Without a Quotient

The residual $\Delta = S_1 + h \cdot S_2 - C$ must vanish modulo q

after the norm check its entries are integers below $p/2$: the centered value of Δ is the integer residual



sizes: rows × columns, at 32K signatures

matrix entries are uniform in $\{-12,\ldots,12\}$

The verifier checks: q divides each of the 31 outputs

$\Delta \not\equiv 0 \pmod{q}$: each output is divisible by q with probability at most $1/25$

all 31 pass with probability below 2^{-142}

- **no quotient is committed:** the proof never contains a multiple of q
- establish norm boundedness from norm projection guarantees of S_1 and e

Linking the Outputs to the Commitments

The verifier holds 187 field elements; the tables they were computed from are only committed

187 outputs $y_{\text{nrm}}, y_{\text{res}}$ —random μ→ **one linear identity** in $S_1,\ e,\ \Delta$ —sumcheck→ **claims at r_Π** v_{S_1}, v_e, v_Δ

Δ is virtual: its claim becomes a claim on the product

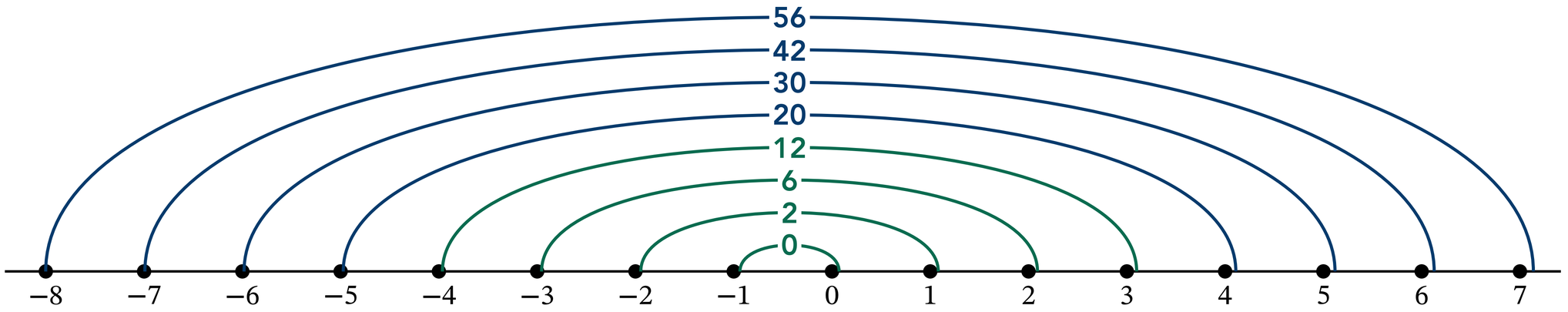
$$(h \cdot S_2)(r_\Pi) \;=\; v_\Delta - v_{S_1} + C(r_\Pi)$$

Ring identity

$$\sum_x w_{\text{ring}}(x)\,\sigma(x)\,M(x) \;=\; v_\Delta - v_{S_1} + C(r_\Pi)$$

- w_{ring}: the transpose of multiplication by h, applied to the evaluation weights at r_Π
- public, and it includes the reduction modulo $X^n + 1$: no quotient here either
- M is affine in the digits: the sum-check ends in evaluations of committed tables

Range Checks on Balanced Digits



The folded digit $P_Z = Z(Z+1)$ takes the same value at Z and $-1-Z$: $P_Z \in \{0,2,6,12\}$ for $Z \in [-4,3]$

$$Z \in [-4,3] \iff Q_8(P_Z) = 0, \quad \text{with} \quad Q_8(P) = P\,(P-2)(P-6)(P-12)$$

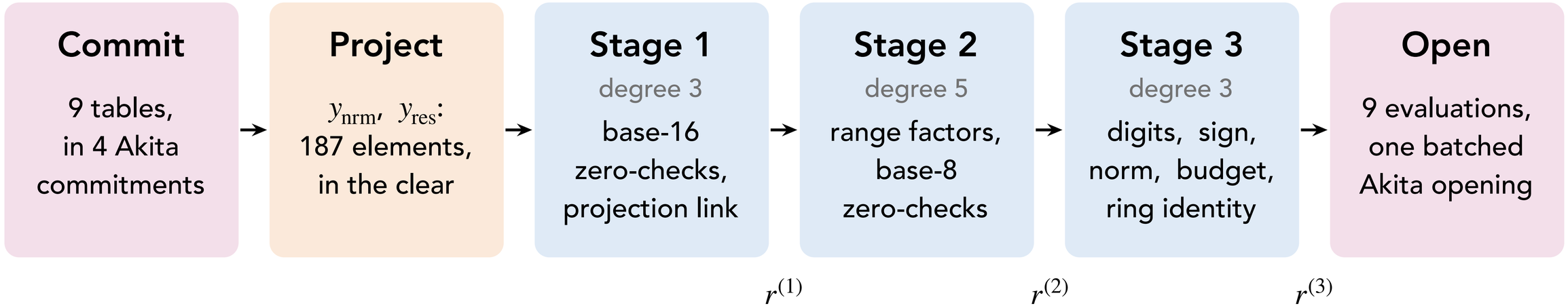
$$Z \in [-8,7] \iff Q_{16}^{\mathrm{lo}}(P_Z) \cdot Q_{16}^{\mathrm{hi}}(P_Z) = 0, \quad \text{a product of two quartics}$$

zero-check at τ ⟶ **$Q_Z^{\mathrm{lo}}, Q_Z^{\mathrm{hi}}$ at $r^{(1)}$** ⟶ **P_Z at $r^{(2)}$** ⟶ **Z at $r^{(3)}$**

Note: virtual tables are multilinear extensions, not polynomial products: $P_Z(r) \neq Z(r)\,(Z(r)+1)$

For details, see Akita paper, Section 6.1

Three Stages, One Batched Opening



- the range chain forces three sequential sum-checks; every other check is placed within these stages

The proof: 66 KiB



Roadmap

1. The Falcon Relation
2. The Aerie Proof System
3. Evaluation & Outlook

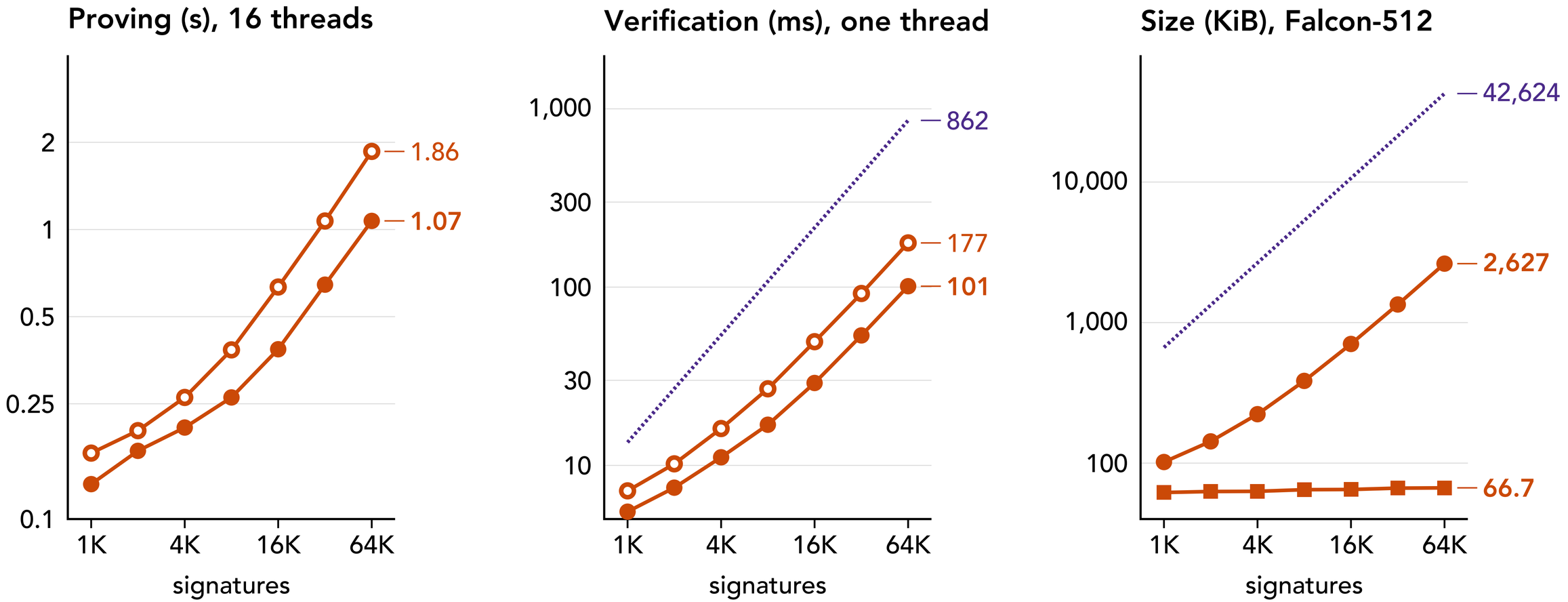
Evaluation: Aerie and Native Falcon

	Falcon-512 × 32K	Falcon-1024 × 16K	
Native signing, one core	3.2 s	3.1 s	
Native verification, one core	0.43 s	0.43 s	
Aerie proving, one core	4.3 s	4.3 s	10× native verification
Aerie proving, 16 cores	**0.65 s**	**0.63 s**	50,800 signatures / s
Aerie verification, one core	**54 ms**	**49 ms**	8× faster than native
Peak prover memory	2.1 GiB	2.1 GiB	
The signatures themselves	20.8 MiB	20.0 MiB	
Proof with one salt per signature	1,346 KiB	706 KiB	16× and 29× smaller
Proof with common or no salt	**66 KiB**	**66 KiB**	

Apple M4 Max laptop, 64 GiB, no GPU; medians of 7 runs; both batches have 2^{24} coefficients

Scaling from 1K to 64K Signatures

Aerie, Falcon-512 — Aerie, Falcon-1024 — no aggregation: native Falcon-512

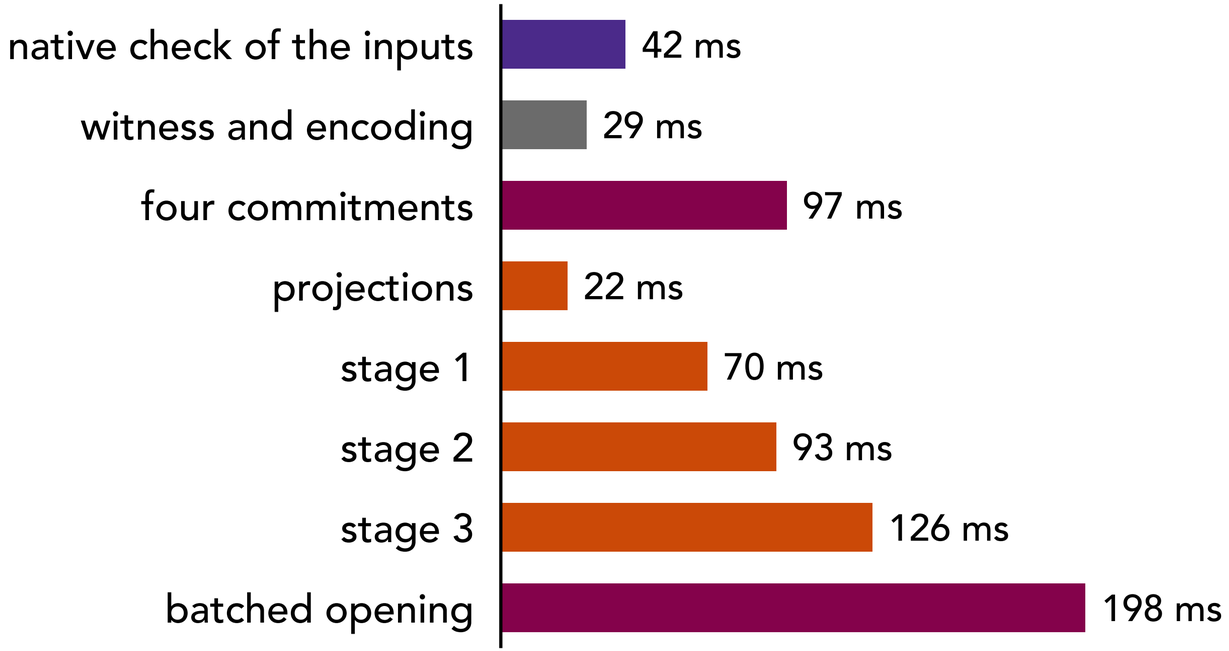


native verification is measured at 32K and scaled linearly; size panel: circles with one salt per signature, squares the proof alone

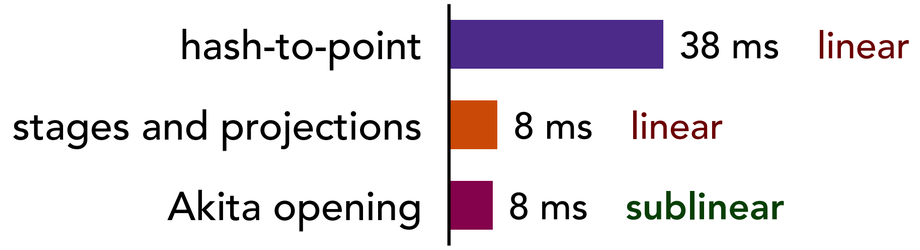
Where the Time Goes

32,768 Falcon-512 signatures

Prover: 0.65 s on 16 threads



Verifier: 54 ms on one thread



Linear in the batch:

- hashing every message to a point (70%)
- inner product on public data inside stage 3

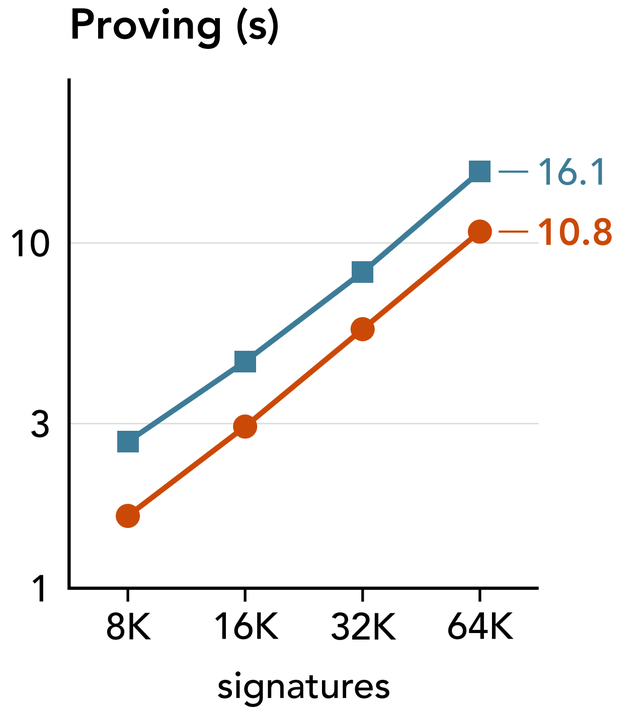
Sublinear: the Akita opening (14%)

The batched opening and stage 3 take 50% of the proving time; checking the inputs natively, 7%

median wall time of each phase; phases overlap, so the prover bars add up to more than 0.65 s; verifier on 16 threads: 11.6 ms

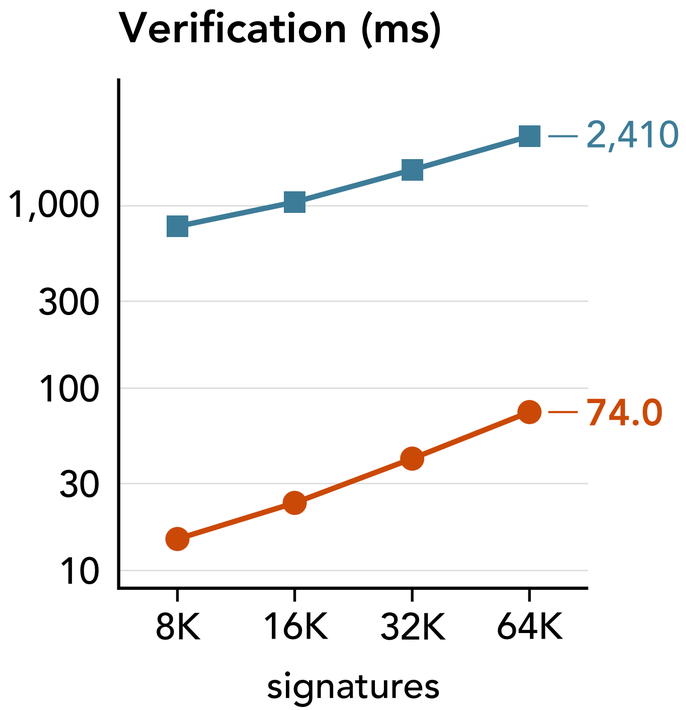
Against Orthus: Same Server, One Core

Aerie — Orthus + LaBRADOR [BBLMS26] — one core of an AMD EPYC 9554P; Falcon-512



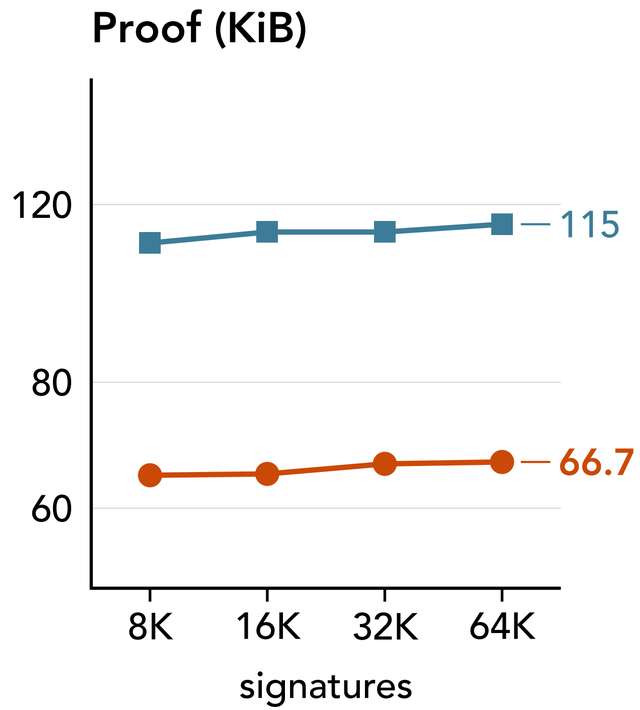
1.5×

faster proving, at 32K



38×

faster verification, at 32K



1.7×

smaller proof

Orthus proves a restricted statement (one shared salt, no encoding check) and needs AVX-512; Aerie uses a third of its memory

Against leanVM: The Hash-Based Route

Signatures aggregated per second, all cores



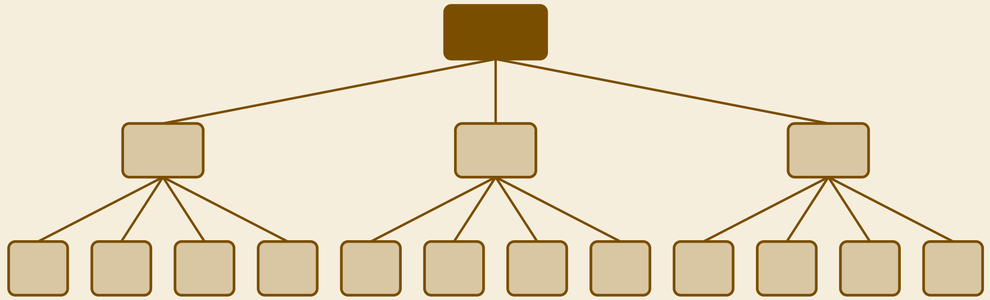
One slot: about 32K signatures	proofs	machines	latency	total work
Aerie	1	1	**0.65 s**	**0.65 s**
leanVM, recursion with fan-in 4	37 leaves + 3 levels	one per proof	2.0 s 3.1×	30 s
leanVM, recursion with fan-in 16	37 leaves + 2 levels	one per proof	2.9 s	29 s 44×

leanVM verifies faster (6 ms against 54 ms, one thread), and its proofs can be merged; Aerie needs no recursion

Apple M4 Max; stateless leanVM rate as reported; the leanVM tree is assembled from measured leaf and merge times

A Simpler Design Space

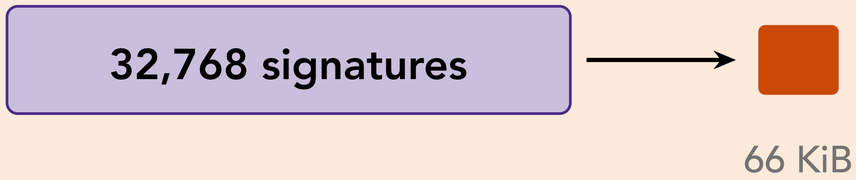
Recursive aggregation



37 proofs of 900 signatures, then proofs of proofs

- several rounds of merging, across nodes
- intricate, and prone to bugs
- proves statements about the hash itself:
  - random-oracle analyses do not directly apply

One node, one proof



- more of the slot to collect and relay votes
- any node holding the signatures can prove
- analysis in the classical random-oracle model

No recursion: aggregation that one laptop finishes well within the slot

Discussion

Limitations of Aerie today

- aggregates cannot be merged: one prover must hold the whole batch
- verification is linear: one hash and one pkey inner product per signature
- 40-byte salt per signature in the proof with standard signing
- proved in the classical, not quantum, random oracle model

Signing with Falcon

- floating point "exists" only in signing: verification, and all of Aerie, is integer arithmetic
- signing does not use it: fixed point, about 2× slower than floating point [BFLP26]
- NIST's plan for FN-DSA (Sept 2026): fixed point only; the standard is not final

Security of the whole system matters: recursion and custom schemes carry risk too

Summary and Future Work

Takeaways

- one slot of Falcon signatures in **0.65 s**, on a laptop
- a **66 KiB** proof, verified in **54 ms**
- no hashing, no quotients, no recursion
- 128-bit knowledge soundness in the ROM

Future work

- private salts: hash proofs with LaBinius [OS26]
- ML-DSA; hybrid lattice + elliptic-curve
- merging proofs; distributed proving
- GPU proving (Metal / CUDA)

Code
github.com/a16z/aerie (public soon)

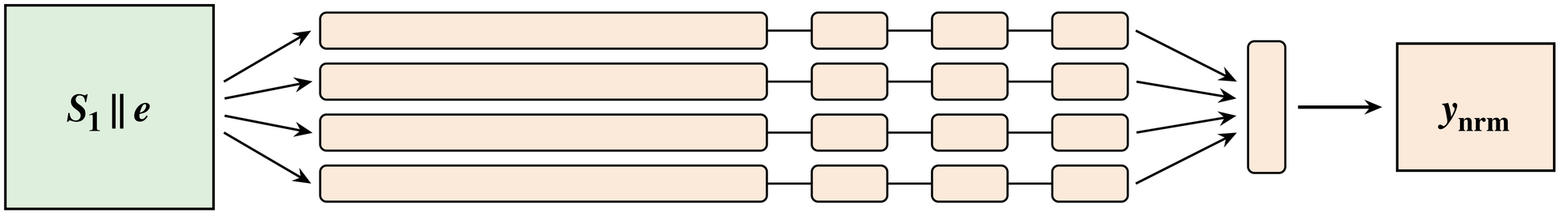
Paper
coming soon

Thank you!

Extra Slides

Inside the Norm Projection

A core is a small random matrix with entries in $\{-1,0,1\}$, applied to every block of its input



4 chains of 4 cores,
one terminal core;
all from one seed

sizes: rows × columns,
at 32K signatures

No wrap-around. A core with c rows maps a large entry to a large output, except with prob. 2^{-c}; small entries are never reduced modulo p $\Rightarrow$ an accepted chain is a chain over $\mathbb{Z}$

Lower bound on the output. Over $\mathbb{Z}$, each core multiplies the squared norm by at least a random Z_c:

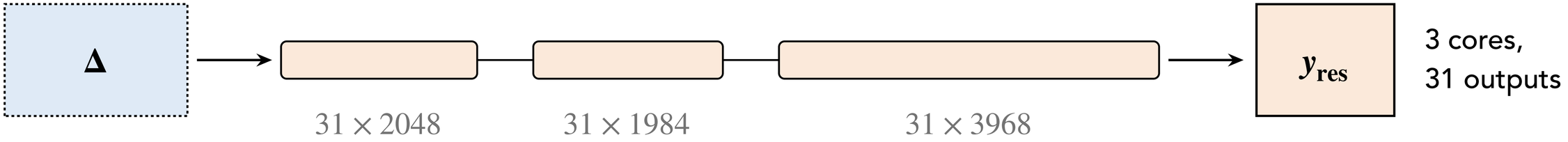
$$\Theta \;\geq\; \|y_{\mathrm{nrm}}\|^2 \;\geq\; \|(S_1, e)\|^2 \cdot Z_{156} \cdot \sum_{g=1}^{4} \prod_{i=1}^{4} Z_{43}^{(g,i)}$$

squared norm at least $E_\star$: the right side falls below Θ with probability at most 2^{-136}

computed exactly from the distribution of the independent Z_c; no union bound over signatures or coordinates

Inside the Residual Projection

Core entries are uniform in $\{-12,\ldots,12\}$; the verifier checks that q divides each of the 31 outputs



sizes: rows × columns, at 32K signatures

After the norm projection, $\Delta = S_1 + h \cdot S_2 - C$ is an integer table with entries below $p/2$

No wrap-around. The outputs are below $p/2$, except with probability 2^{-153} $\Rightarrow$ $y_{\text{res}} = \Pi_{\text{res}}\,\Delta$ over $\mathbb{Z}$

One core. Fix an entry of the input that is nonzero modulo q. In each row, its coefficient takes 25 values modulo q; at most one gives output zero modulo q. All 31 rows: probability 25^{-31}

$\Delta \not\equiv 0 \pmod{q}$: some core maps a nonzero input to zero modulo q, or the outputs wrap:

$$\Pr[\, q \text{ divides all 31 outputs} \,] \;\le\; 3 \cdot 25^{-31} + 2^{-153} \;<\; 2^{-142}$$