Aggregating ML-DSA Signatures Without Recursion: A Cross-Field Argument for the Commitment Hash

Jipeng Zhang and Jiaheng Zhang

National University of Singapore, Singapore, Singapore

Abstract. Consensus protocols must compress tens of thousands of signatures per block, yet ML-DSA (FIPS 204) signatures are kilobytes long and lack native aggregation. Aerie aggregates Falcon signatures into a single succinct proof per batch without recursion, using a sum-check-based polynomial interactive oracle proof (PIOP) over the Akita lattice-based polynomial commitment. Extending this approach to ML-DSA is challenging: the hash that its verifier recomputes from the response and hint takes seven Keccak-f permutations, or nine for ML-DSA-87, which are expensive over prime fields.

We address this bottleneck by proving ML-DSA's algebraic and Boolean components over separate fields, supporting all three FIPS 204 parameter sets. A PIOP over a 64-bit prime field handles the lattice relation, hint verification, and range checks, while Flock proves Keccak over GF(2). We introduce a cross-field parity bridge to enforce bitwise consistency between the two commitments. Using seeded sparse matrices, the bridge detects any fixed nonzero difference except with probability below $2^{-144.6}$, independently of batch size. The aggregate certifies that every signature passes FIPS 204 verification. In the classical random-oracle model, each Fiat–Shamir challenge incurs at most 2^{-128} soundness error per query, as in Aerie, and aggregate unforgeability reduces to the strong unforgeability of ML-DSA.

On 16 AMD Zen 5 cores with AVX2, our prover aggregates 32,768 ML-DSA-44 signatures in 2.68 s, at 49.2 bytes per signature. For ML-DSA-87, it takes 4.13 s at 82.7 bytes per signature. Against the Johnson-regime variant of Alloul26, a prime-field ML-DSA-87 aggregator, our prover is 43× faster and our aggregates are 1.8× smaller.

Keywords: ML-DSA · signature aggregation · post-quantum signatures · SNARK · sum-check · lattice-based commitments

1 Introduction

In every block, a proof-of-stake blockchain verifies tens of thousands of validator votes. Today such chains compress these votes with Boneh–Lynn–Shacham (BLS) aggregation [BGLS03], which a quantum adversary breaks. NIST's ML-DSA signatures take 2,420 to 4,627 bytes, against 64 bytes on elliptic curves, and do not aggregate natively [Nat24, BRL23, CDBT26]. Rather than transmit individual signatures, an aggregator can provide a succinct proof that every signature verifies [KCLM22, DKKW25, AAB+24].

Aerie [CDBT26] showed for Falcon [FHK+20] that batch proofs are fast when both signature scheme and proof system are lattice-based. It pairs a sum-check polynomial interactive oracle proof (PIOP) with the Akita lattice polynomial commitment [DBK+26]

E-mail: jp-zhang@outlook.com (Jipeng Zhang), jhzhang@nus.edu.sg (Jiaheng Zhang)

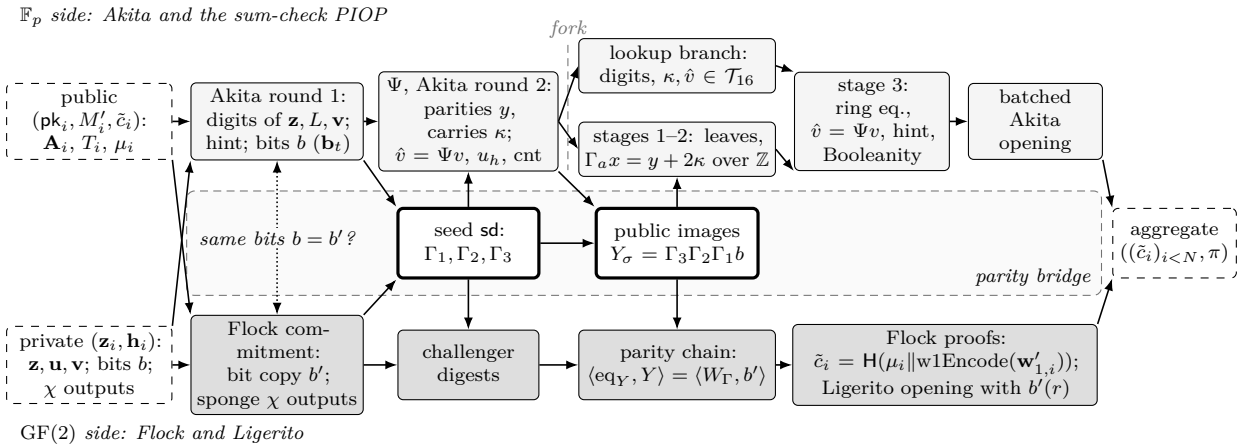


Figure 1: The prover (Section 3.1). The parity bridge ties the $\mathbb{F}_p$ and GF(2) copies b and b' of the w1Encode bits.

without recursion. On 16 AVX2 cores, Aerie proves 32,768 Falcon-512 signatures in about a second at 42 bytes per signature. Because each signature's random salt stays public, the verifier evaluates Falcon's hash-to-point directly.

ML-DSA does not fit this recipe. ML-DSA [Nat24] is NIST's primary post-quantum signature standard in FIPS 204, with parameter sets ML-DSA-44, -65 and -87 for security categories 2, 3 and 5. Most of its hashes take public inputs alone. The matrix $\mathbf{A}$ expands from the key seed ρ, while the key hash tr and message representative μ depend only on the public key and formatted message. The challenge c likewise expands from the $\lambda/4$-byte commitment hash $\tilde{c}$, for collision strength $\lambda \in \{128, 192, 256\}$. When an aggregate keeps $\tilde{c}$ public, as Aerie keeps the salt, the verifier computes $\mathbf{A}$, tr, μ and c directly.

The remaining checks read the response $\mathbf{z}$ and hint $\mathbf{h}$, which the aggregate omits. Alongside bounds on $\mathbf{z}$ and on the weight of $\mathbf{h}$, the verifier reconstructs the high-order commitment $\mathbf{w}'_1$ and checks $\tilde{c} = \mathsf{H}(\mu \,\|\, \mathrm{w1Encode}(\mathbf{w}'_1), \lambda/4)$. Here H is SHAKE256 [Nat15] with its output length, and w1Encode packs $\mathbf{w}'_1$ into bits. This Fiat–Shamir step hashes a witness-dependent input. Publishing $\mathrm{w1Encode}(\mathbf{w}'_1)$ costs 768 bytes per signature for ML-DSA-44 and -65, and 1,024 bytes for ML-DSA-87. A compact aggregator must therefore prove SHAKE256: seven Keccak-f[1600] permutations per signature for ML-DSA-44 and -65, or nine for ML-DSA-87.

Two fields, one witness. The hash and lattice checks suit different characteristics. Over GF(2), as in Flock [BRW26], Keccak is efficient to prove: XORs are linear, and each round commits only 1,600 AND outputs. Over a 64-bit prime field, every XOR costs a multiplication. Alloul26 [All26], which aggregates ML-DSA-87 over one prime field, spends over half of its prover time on Keccak at 2^{15} signatures.[1] Conversely, the lattice check is a linear identity modulo q with integer bounds, which a prime-field PIOP proves with a few committed digits per coefficient rather than bit by bit. As Figure 1 shows, we therefore prove the batched SHAKE256 rank-1 constraint system (R1CS) instances with Flock over GF(2) and the lattice relation with a PIOP over Akita's field $\mathbb{F}_p$, $p = 2^{64} - 59$. This split commits $\mathrm{w1Encode}(\mathbf{w}'_1)$ twice, in characteristics 2 and p: neither field embeds in the other, and neither commitment opens the other's polynomials.

The parity bridge. Mixed-mode conversions [RW19, EGK+20, BBMH+21, WYX+21] and composite-statement proofs [CGM16, AGM18, BHH+19] pay per shared bit or value, while commit-and-prove linking [CFQ19] works over a single field. Concurrent single-

[1] In one AVX2 run on 16 cores, Keccak took 101.2 of 179.2 s (56.5%).

Table 1: Aggregating 32,768 ML-DSA signatures on 16 Zen 5 cores.

System	Prove (s)		Verify (ms)	B/sig	Compression
	AVX2	AVX-512F			
Ours-44	2.68	2.35	75	49.2	49.2×
Ours-65	3.31	2.90	120	65.4	50.6×
Ours-87	4.13	3.62	177	82.7	55.9×
Alloul26 [All26]	179.2	176.3	5,849	151.0	30.6×

Note: Medians of five runs; Alloul26: ML-DSA-87, Johnson-regime variant, medians of three runs. Every row proves full FIPS 204 validity. Verification on AVX2 with uncached keys. B/sig includes the public $\tilde{c}_i$. Compression: signature bytes (Table 3) over B/sig.

signature ML-DSA proofs link the two characteristics via a polynomial message authentication code (MAC) with error linear in the chunk count [SG26] or in the exponent of $\mathbb{F}_{2^{128}}$ [BB26]. Our bridge instead compares a few thousand random sparse parity checks rather than individual bits. Once both commitments are fixed, a short challenge sd expands into three sparse binary matrices $\Gamma_1, \Gamma_2, \Gamma_3$. For each segment of up to 2^{13} signatures, both sides prove that $\Gamma_3\Gamma_2\Gamma_1$ maps their bits to one public 4,096-bit image over GF(2). Since modulo-2 reduction is nonlinear over $\mathbb{F}_p$, the $\mathbb{F}_p$ side proves $\Gamma_a x = y + 2\kappa$ over $\mathbb{Z}$ with Boolean y and range-checked carries κ. Over GF(2), the map is linear, so one sum-check proves the claim and commits nothing beyond the hash witness.

Contributions. We present a non-recursive signature aggregator for ML-DSA with four contributions:

- We introduce *a cross-field parity bridge* (Section 5) that establishes bitwise consistency between prime-field and binary-field commitments from evaluation openings alone, independent of either scheme's internal structure. We prove that composing random sparse binary parity-check matrices detects any fixed nonzero bit discrepancy with soundness error independent of batch size and parameter set (Theorem 1), and extend this guarantee across the decoding list of list-decoded binary commitments (Corollary 1 and Theorem 2).
- We design *an algebraic PIOP for full FIPS 204 verification* (Section 4). Our relation captures UseHint and the hint-weight bound with small integer witnesses, proving full validity of unmodified signatures. We commit the quotient by q and bound it via a random $\{0, 1\}$ projection, while proving all 16-bit digit and carry ranges in one lookup table via logarithmic derivatives and Goldwasser–Kalai–Rothblum (GKR) sum-checks.
- We provide *a concrete security analysis* (Section 6), proving knowledge soundness in the classical random-oracle model (ROM) under Module Short Integer Solution (Module-SIS) via round-by-round analysis of Flock and checkpoint extraction on our forked transcript (Lemma 5 and Theorem 2). We prove exactness of our SHAKE256 circuits and reduce aggregate unforgeability (Theorem 3) to the strong unforgeability of ML-DSA under chosen-message attacks (SUF-CMA).
- We build and evaluate *a unified implementation for ML-DSA* (Section 7) supporting all three parameter sets in one Rust codebase with AVX2 and AVX-512 kernels.

Our artifact is available at `https://github.com/Ji-Peng/AggregateMLDSA-code`.

Results and scope. Table 1 summarizes a batch of $N = 32{,}768$ signatures on 16 AMD Zen 5 cores with two threads each. With AVX2, supported on 95% of surveyed x86 machines [Val26], proving takes 2.68 s for ML-DSA-44 and 4.13 s for ML-DSA-87, fitting within Ethereum's 12-second consensus slot [Eth26] (Section 7). Aggregates take 49.2 to 82.7 bytes per signature with the public $\tilde{c}_i$ (49 to 56× compression, vs. 31× for Alloul26). Uncached verification takes 75 to 177 ms. On the same cores and batch, our ML-DSA-87 prover is 43 times faster than Alloul26 and our aggregate is 1.8 times smaller. Table 2 contrasts our design with Aerie.

Table 2: Our aggregator vs. Aerie.

	Aerie (Falcon) [CDBT26]	**This work (ML-DSA)**
Validity	ring eq., norm, format	full FIPS 204
Hash in proof	none (public salt)	SHAKE256 for $\tilde{c}$
Fields	$\mathbb{F}_p$	$\mathbb{F}_p$ and GF(2), bridged
Lattice shape	one ring element	$k \times \ell$ module, exact
Quotient by q	residual projection	$\{0,1\}$ projection
Range checks	leaf polynomials	LogUp-GKR into $\mathcal{T}_{16}$
Transcript	sequential	forked, two branches

Note: Both use Akita over $p = 2^{64} - 59$, batched sum-check stages, per-site proof of work and the classical ROM.

Security level. As in Aerie, proof of work bounds the soundness error of every challenge *site* by 2^{-128} per oracle query. Across Q queries and R sites, challenges add $(Q + R)\, 2^{-128}$ to the knowledge error.

2 Preliminaries

2.1 ML-DSA verification

ML-DSA [Nat24] operates over $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$ under the three parameter sets of Table 3. We write ring elements in plain lowercase (c), vectors in bold lowercase ($\mathbf{z} \in R_q^\ell$) and matrices in bold uppercase ($\mathbf{A} \in R_q^{k\times\ell}$). We identify a polynomial vector with its array of 256 coefficients per entry: $\mathbf{z}[s]$ denotes polynomial s of $\mathbf{z}$, $\mathbf{z}[s]_j$ its X^j coefficient and $\mathbf{A}[\alpha, s]$ entry (α, s) of $\mathbf{A}$. In a batch, a subscript i indexes record i, as in $\mathbf{z}_i$ and $\mathbf{A}_i$. A public key $\mathsf{pk} = (\rho, \mathbf{t}_1)$ consists of a seed ρ and $\mathbf{t}_1 \in [0, 2^{10})^{k\times 256}$, while a signature $(\tilde{c}, \mathbf{z}, \mathbf{h})$ comprises a $\lambda/4$-byte commitment hash, $\mathbf{z} \in R_q^\ell$ and a hint $\mathbf{h} \in \{0,1\}^{k\times 256}$ of weight at most ω. Verification reads M only through its *formatted message* M', which encodes the form (pure or pre-hash), a context of at most 255 bytes and M or its pre-hash [Nat24, Algorithms 3 and 5]. Algorithm 1 states verification on M' [Nat24, Algorithm 8], where H is SHAKE256 [Nat15] with byte output length and c has τ nonzero coefficients in $\{\pm 1\}$. Signature decoding sets $\mathbf{h} = \perp$ if the hint is malformed, non-canonically packed or of weight above ω, and ExpandA outputs $\hat{\mathbf{A}}$ in the number-theoretic transform (NTT) domain.

Decompose [Nat24, Algorithm 36] writes $r \in [0, q)$ as $2\gamma_2 r_1 + r_0$ with $r_0 \in (-\gamma_2, \gamma_2]$, except at $r - r_0 = q - 1$, where it sets $r_1 = 0$ and subtracts 1 from r_0. HighBits outputs r_1. UseHint [Nat24, Algorithm 40] returns r_1 when the hint bit is 0, and otherwise shifts r_1 modulo $m = (q-1)/2\gamma_2$ by $+1$ if $r_0 > 0$ and -1 if $r_0 \le 0$. Finally, w1Encode [Nat24, Algorithm 28] packs $\mathbf{w}_1' \in [0, m)^{k\times 256}$ into $n_w = \lceil \log_2 m \rceil$ bits per entry ($256kn_w$ bits in total), which SHAKE256 absorbs with μ in n_{perm} permutations (Table 3).

2.2 Multilinear polynomials and sum-check

We use one symbol for a table $f : \{0,1\}^n \to \mathbb{F}$ and its multilinear extension $f(r) = \sum_{x\in\{0,1\}^n} \mathrm{eq}(r, x) f(x)$, with $\mathrm{eq}(r, x) = \prod_{j=1}^n (r_j x_j + (1 - r_j)(1 - x_j))$. The sum-check protocol [LFKN92] reduces a claim on $\sum_{x\in\{0,1\}^n} g(x)$ to evaluating g at a random *terminal point*. Over a challenge field $\mathbb{K}$, its soundness error is at most n times the maximum individual degree of g, divided by $|\mathbb{K}|$ [Tha25, Section 3.2]. A random linear combination, or *batching vector*, merges claims on one domain into a single sum-check. A zero-check proves $g = 0$ on $\{0,1\}^n$ via sum-check on $\mathrm{eq}(\vartheta, x)\, g(x)$ at a random $\vartheta \in \mathbb{K}^n$, and a *virtual* polynomial is an uncommitted function of committed and public tables reduced by sum-check.

Algorithm 1: ML-DSA verification.

Input: key $\mathsf{pk} = (\rho, \mathbf{t}_1)$, message M', $(\tilde{c}, \mathbf{z}, \mathbf{h})$
Output: accept or reject
if $\mathbf{h} = \perp$ **then return** reject ▷ *malformed hint*
$\hat{\mathbf{A}} \leftarrow \mathrm{ExpandA}(\rho)$; $\mathbf{A} \leftarrow \mathrm{NTT}^{-1}(\hat{\mathbf{A}})$
$\mathsf{tr} \leftarrow \mathsf{H}(\mathsf{pk}, 64)$; $\mu \leftarrow \mathsf{H}(\mathsf{tr} \,\|\, M', 64)$ ▷ pk *as bytes*
$c \leftarrow \mathrm{SampleInBall}(\tilde{c})$ ▷ *challenge,* τ *entries* ± 1
$\mathbf{w}' \leftarrow \mathbf{A}\mathbf{z} - c\,\mathbf{t}_1 2^d \in R_q^k$ ▷ *approx. commitment*
$\mathbf{w}'_1 \leftarrow \mathrm{UseHint}(\mathbf{h}, \mathbf{w}')$ ▷ *its corrected high bits*
return $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ **and** $\tilde{c} = \mathsf{H}(\mu \,\|\, \mathrm{w1Encode}(\mathbf{w}'_1), \lambda/4)$

Table 3: ML-DSA parameter sets and derived constants.

Symbol	44	65	87	Symbol	44	65	87
NIST category	2	3	5	$n_w = \lceil \log_2 m \rceil$	6	4	4
(k, ℓ)	$(4, 4)$	$(6, 5)$	$(8, 7)$	n_{perm}	7	7	9
τ	39	49	60	$B_z = \gamma_1 - \beta - 1$	130,993	524,091	524,167
η	2	4	2	$B_u = 2\gamma_2 + 1$	190,465	523,777	523,777
$\beta = \tau\eta$	78	196	120	B_v	67,068,447	335,418,249	469,653,636
γ_1	2^{17}	2^{19}	2^{19}	B_Ψ, n_Ψ	2^{37}, 1	2^{39}, 1	2^{39}, 2
γ_2	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$	lift	$2^{60.9989}$	$2^{62.9990}$	$2^{62.9992}$
ω	80	55	75	d_{main}	$10 + d_{\mathrm{sig}}$	$8 + d_{\mathrm{sig}}$	$8 + d_{\mathrm{sig}}$
λ	128	192	256	cells/sig at 2^{15}	17,025.5	23,067.5	31,387.5
pk, sig (B)	1,312, 2,420	1,952, 3,309	2,592, 4,627	$\lvert\mathcal{F}\rvert$ at 2^{15}	79,970,304	105,136,128	147,079,168
$m = (q-1)/2\gamma_2$	44	16	16	sites R at 2^{15}	2,113	2,167	2,115

Note: All sets use $q = 8{,}380{,}417$ and $d = 13$. Left: FIPS 204 parameters. Right: constants under $\mathcal{R}_{\mathrm{FIPS}}$ at padded size $N^\star = 2^{d_{\mathrm{sig}}}$, with witness and quotient bounds B_z, B_u, B_v (Section 3.2), projection parameters B_Ψ, n_Ψ (Section 4.1), integer-lift bound (Lemma 3, below $p/2 \approx 2^{63}$) and main-domain variables d_{main} (Section 4.1). At $N = 2^{15}$, cells/sig counts committed $\mathbb{F}_p$ entries per signature, $\lvert\mathcal{F}\rvert$ counts all looked-up entries in $\mathcal{T}_{16}$ including padding (Section 4.2), and R counts challenge sites (Section 6).

2.3 Keccak and proof systems

Keccak and SHAKE256. SHAKE256 [Nat15] is a Keccak sponge over a 1,600-bit state. Starting from zero, it pads the input, XORs each 1,088-bit rate block into the state and applies Keccak-f[1600] before squeezing the digest. One permutation runs 24 rounds of GF(2)-linear mixing followed by the quadratic step χ of 1,600 bitwise ANDs (38,400 AND outputs per permutation).

Akita. Akita [DBK+26] is a lattice-based multilinear polynomial commitment scheme whose binding rests on Module-SIS. Following Aerie [CDBT26], we use its 64-bit profile over $\mathbb{F}_p$ ($p = 2^{64} - 59$) with challenges in $\mathbb{K} = \mathbb{F}_{p^2}$. Tables with entries in the balanced range $[-2^{w-1}, 2^{w-1})$ are committed at *width* $w \in \{4, 16\}$. One batched proof [DBK+26, Section 4.6] opens tables of different sizes and rounds, folding them with sparse challenges drawn one *fold coordinate* at a time. Extracted decommitments need not be short, so the width alone is not a range proof [DBK+26, Remark 10.41].

Proof of work per challenge. Each Fiat–Shamir challenge draw, called a *site* ι, reduces the current claim to the next. A false claim can become true only if the challenge falls in a *bad set* of probability at most ε_ι. Aerie attaches to site ι a work nonce whose hash starts with D_ι zero bits [CDBT26, Equation (17)], where

$$D_\iota = \min\{D \geq 0 : \; 2^{128}\varepsilon_\iota \leq 2^D\}. \tag{1}$$

Each oracle query then hits a bad set with probability at most $\varepsilon_\iota\, 2^{-D_\iota} \leq 2^{-128}$. Since $|\mathbb{K}| = p^2 < 2^{128}$, an error bound $\varepsilon_\iota = n/|\mathbb{K}|$ requires $D_\iota = \lfloor \log_2 n \rfloor + 1$ bits.

R1CS, Flock and Ligerito. A rank-1 constraint system (R1CS) [Set20, BRW26] over $\mathbb{F}$ uses sparse matrices $A, B, C \in \mathbb{F}^{m_c \times n_c}$ to constrain $w \in \mathbb{F}^{n_c}$ via the entrywise product $(Aw) \circ (Bw) = Cw$. Over GF(2), linear combinations evaluate XORs for free, so each row checks one AND gate. Flock [BRW26] proves batched GF(2) R1CS instances via a zerocheck and linear-combination check (lincheck) over $\mathbb{F}_{2^{128}}$, committing with Ligerito [NA25], a recursive hash-based Reed–Solomon commitment, via ring switching [DP25]. We run Flock at rate 1/2 in the proven Johnson list-decoding regime [BSCI+20, DKT26], relying on no proximity-gap conjecture. Each commitment binds the prover to the *Johnson list* of at most L_J codewords within its decoding radius.

2.4 Lattice problems and aggregate signatures

For uniform $\mathbf{A} \in R_q^{k \times \ell}$, Module-SIS (MSIS) with bound β' asks for a nonzero $\mathbf{x} \in R_q^{\ell+k}$ satisfying $[\mathbf{A} \mid I]\,\mathbf{x} = 0$ and $\|\mathbf{x}\|_\infty \le \beta'$ on centered representatives. SelfTargetMSIS [KLS18, DKL+18] models ML-DSA's Fiat–Shamir step: it asks for a short $\mathbf{x}$ and a message whose hash with $[\mathbf{A} \mid I]\,\mathbf{x}$ equals one block of $\mathbf{x}$. A signature scheme is strongly unforgeable under chosen-message attacks (SUF-CMA) if no adversary outputs a valid message-signature pair outside its signing oracle's responses. Prior proofs reduce ML-DSA's SUF-CMA security to module learning with errors (MLWE), SelfTargetMSIS and MSIS in the ROM [KLS18, DKL+18, BBD+23, DFPS23].

Definition 1 (Aggregate unforgeability)**.** An aggregate signature scheme consists of a setup outputting pp, an aggregator mapping records $\mathcal{L} = (\mathsf{pk}_i, M'_i)_{i<N}$ and their signatures to an aggregate Σ or rejecting, and a verifier $\mathsf{AggVerify}(\mathsf{pp}, \mathcal{L}, \Sigma) \in \{0, 1\}$. In $\mathsf{AggForge}(\mathcal{A})$, the challenger generates pp and an honest key pair $(\mathsf{pk}^\star, \mathsf{sk}^\star)$, and $\mathcal{A}(\mathsf{pk}^\star, \mathsf{pp})$ queries the random oracles and a signing oracle for $\mathsf{sk}^\star$. Given any formatted message M', the signing oracle runs the internal signer [Nat24, Algorithm 7] and records M' in $\mathcal{Q}$. Unless stated otherwise, signing is hedged with fresh randomness rnd per query [Nat24, Algorithm 2, line 5]. The adversary $\mathcal{A}$ wins if it outputs $(\mathcal{L}, \Sigma)$ with $\mathsf{AggVerify}(\mathsf{pp}, \mathcal{L}, \Sigma) = 1$ and $(\mathsf{pk}^\star, M') \in \mathcal{L}$ for some $M' \notin \mathcal{Q}$, and $\mathrm{Adv}^{\mathrm{agg}}(\mathcal{A})$ denotes its winning probability.

This game captures chosen-key aggregate unforgeability [BGLS03] as in Aerie [CDBT26, Definition 5.10], where the other keys in $\mathcal{L}$ require no proof of possession. Because our aggregate publishes $(\tilde{c}_i)_{i<N}$, $\mathcal{A}$ chooses those commitment hashes as well. Defining freshness on M' covers both forms and every context under one key, and Theorem 3 extends to the deterministic signer with $rnd = 0^{32}$.

3 Overview and Relation

3.1 System overview

We trace the proving pipeline of Figure 1 and formalize the target relation in Section 3.2.

Interface. The public input is a parameter set and N *live* records $(\mathsf{pk}_i, M'_i)_{i<N}$, each pairing a key of that set with a formatted message (Section 2.1). The aggregator verifies each signature [Nat24, Algorithm 8] and rejects on any failure. Its output is $(\tilde{c}_i)_{i<N}$ and one proof π that every live record has a witness for $\mathcal{R}_{\mathrm{FIPS}}$ (Definition 2). The batch is padded to the smallest power of two $N^\star = 2^{d_{\mathrm{sig}}}$ with $d_{\mathrm{main}} \ge 13$ (Table 3). *Padding* records $i \ge N$ set $\mathbf{z} = 0$ and zero hint weight, and the GF(2) side omits SHAKE256 [Nat15].

The hash boundary. Since ExpandA, $\mathsf{tr}_i = \mathsf{H}(\mathsf{pk}_i, 64)$, $\mu_i = \mathsf{H}(\mathsf{tr}_i \,\|\, M'_i, 64)$ and $c_i = \mathrm{SampleInBall}(\tilde{c}_i)$ depend only on $(\mathsf{pk}_i, M'_i, \tilde{c}_i)$, the verifier computes them directly and may cache $\hat{\mathbf{A}}_i$, $\mathbf{A}_i$ and tr_i per key across batches. The proof reads M'_i only through μ_i and proves the n_{perm} witness-dependent permutations of $\tilde{c}_i = \mathsf{H}(\mu_i \,\|\, \mathrm{w1Encode}(\mathbf{w}'_{1,i}), \lambda/4)$. With negacyclic multiplication $\circledast$ over $\mathbb{Z}$, the verifier also computes $T_i = 2^d(c_i \circledast \mathbf{t}_{1,i}) \in$

$\mathbb{Z}^{k\times 256}$. Lines 5 and 6 of Algorithm 1 then lift over $\mathbb{Z}$ to $\mathbf{A}_i\mathbf{z}_i - T_i = 2\gamma_2\mathbf{w}'_{1,i} + \mathbf{u}_i + q\mathbf{v}_i$, with UseHint residual $\mathbf{u}_i$ and quotient $\mathbf{v}_i$ by q.

Commitments. A Fiat–Shamir transcript orders messages and challenges in numbered steps (events 0 to 6). Event 0 binds the parameter set, the relation, the records with their $\tilde{c}_i$ and the proof-of-work schedule. In event 1, the $\mathbb{F}_p$ side uses Akita [DBK+26] to commit the digits of $\mathbf{z}$ and $\mathbf{v}$, the hint witnesses for $\mathbf{u}$ and the n_w bit planes $\mathbf{b}_t \in \{0,1\}^{k\times 256}$ of $\mathbf{w}'_1 = \sum_{t=0}^{n_w-1} 2^t\mathbf{b}_t$. In event 2, the GF(2) side uses Flock [BRW26] to commit a copy b' of those bits and the χ outputs.

The parity bridge. After both bit copies are committed, event 3 draws the 32-byte bridge seed sd, which expands into sparse binary matrices $\Gamma_1, \Gamma_2, \Gamma_3$. A batch splits into $S = \max(1, N^\star/2^{13})$ *segments* σ of at most 2^{13} records, and per segment both sides prove the public image $Y_\sigma = \Gamma_3\Gamma_2\Gamma_1 b$ over GF(2) of their bits. At each level $a \in \{1,2,3\}$, the $\mathbb{F}_p$ side evaluates $\Gamma_a x$ over $\mathbb{Z}$ and proves $\Gamma_a x = y + 2\kappa$ for bits y and range-checked carries κ, so $y = \Gamma_a x \bmod 2$. The GF(2) side reduces the *parity chain* $\langle \mathrm{eq}_Y, Y\rangle = \langle W_\Gamma, b'\rangle$, for the equality table eq_Y at a random point and $W_\Gamma = (\Gamma_3\Gamma_2\Gamma_1)^\top \mathrm{eq}_Y$, to $b'(r)$ (Section 5). The sponges of all segments form one Flock batch, proved with the parity chain in one *joint* GF(2) *argument* sharing a Ligerito [NA25] opening and Johnson list.

The forked transcript. After sd, a separate query draws a $\{0,1\}$ matrix Ψ to bound $\mathbf{v}$. Akita round 2 (event 4) commits the level-2 parities y, the carries κ, the projections of $\mathbf{v}$ by Ψ and the multiplicities cnt of a LogUp-GKR [Hab22, PH23] lookup into $\mathcal{T}_{16} = [0, 2^{16})$ (Section 4). Transcript digests for the Flock challengers (event 5) and the images Y_σ (event 6) follow. Since the lookup and sum-check stages 1 and 2 share no data, the transcript forks into a *lookup branch* and a *stage branch* that run concurrently. Stage 3 absorbs both branch digests, proves the ring identity and ends in one batched Akita opening. Events 5 and 6 initialize the Flock challengers before the fork, so Flock runs beside the PIOP and no PIOP or Akita challenge reads a Flock message.

3.2 The relation

Our relation $\mathcal{R}_{\mathrm{FIPS}}$ expresses FIPS 204 validity of a live record as integer identities in small witnesses, so every extracted witness yields a standard signature (Lemma 1). Rather than decomposing $\mathbf{w}'$ into bits for UseHint, it witnesses the low-order part of Decompose, a signed hint step and a modular wrap indicator. Fix a parameter set and a record with centered $\mathbf{A}_i \in R_q^{k\times\ell}$, T_i of Section 3.1, μ_i and $\tilde{c}_i$. We drop i when clear, set $m = (q-1)/2\gamma_2$ and let $B_z = \gamma_1 - \beta - 1$ be the bound on $\|\mathbf{z}\|_\infty$ in Algorithm 1.

Definition 2 (Relation $\mathcal{R}_{\mathrm{FIPS}}$)**.** The witness consists of $\mathbf{z}_i \in \mathbb{Z}^{\ell\times 256}$, $\mathbf{v}_i \in \mathbb{Z}^{k\times 256}$, bit planes $\mathbf{b}_{i,t} \in \{0,1\}^{k\times 256}$ for $0 \le t < n_w$, an integer slack $u_h \ge 0$ and, at each $(\alpha, j) \in [0,k) \times [0,256)$, a tuple $(\theta, L, \delta, e) \in \{\pm 1\} \times [0,\gamma_2) \times \{0, \pm 1\}^2$. Let $\mathbf{w}'_{1,i} = \sum_{t=0}^{n_w-1} 2^t\mathbf{b}_{i,t}$. At each (α, j), set $w'_1 = \mathbf{w}'_{1,i}[\alpha]_j$ and

$$r_0 = \gamma_2\tfrac{1+\theta}{2} + L + 1 - \gamma_2, \qquad u = r_0 - 2\gamma_2\delta - e,$$

and collect the values u into $\mathbf{u}_i \in \mathbb{Z}^{k\times 256}$. The relation holds if, for every row $\alpha < k$ and coefficient $j < 256$,

$$\begin{aligned}
&\textstyle\sum_{s<\ell} \big(\mathbf{A}_i[\alpha, s] \circledast \mathbf{z}_i[s]\big)_j - T_i[\alpha]_j - 2\gamma_2\mathbf{w}'_{1,i}[\alpha]_j \\
&\qquad - \mathbf{u}_i[\alpha]_j - q\,\mathbf{v}_i[\alpha]_j = 0 \quad \text{over } \mathbb{Z}, && (2)\\
&\|\mathbf{z}_i\|_\infty \le B_z, \quad \mathbf{w}'_{1,i} \in [0,m)^{k\times 256}, \\
&\delta \in \{0,\theta\}, \quad e \in \{0,1,\delta\}, \quad \textstyle\sum \delta^2 + u_h = \omega, \\
&e = 1 \Rightarrow w'_1 = 0, \quad e = -1 \Rightarrow w'_1 = m-1, && (3)\\
&\tilde{c}_i = \mathsf{H}\big(\mu_i \,\|\, \mathrm{w1Encode}(\mathbf{w}'_{1,i}), \lambda/4\big). && (4)
\end{aligned}$$

Here $(\theta, L) \mapsto r_0$ is a bijection onto $(-\gamma_2, \gamma_2]$, the range of the low-order part of Decompose. As in UseHint, $\theta = 1$ iff $r_0 > 0$, and the step δ is nonzero only where the hint bit is 1. The modular wrap indicator e records reduction modulo m, also at the $q-1$ corner case of Decompose. Since $2\gamma_2 m = q-1$, each modular wrap shifts u by one, so $u = \mathrm{ctr}_q(w' - 2\gamma_2 w_1')$ is the UseHint residual of the coefficient w' of $\mathbf{w}'$, where ctr_q maps $\mathbb{Z}_q$ to $(-q/2, q/2]$. As δ never opposes r_0 in sign, $|u| \le 2\gamma_2 + 1 = B_u$. The sum $\sum \delta^2$ counts active hints, leaving slack u_h.

For a valid signature, bounding the first four terms of (2) by $128\ell(q-1)B_z$, $2^d\tau(2^{10}-1)$, $q-1-2\gamma_2$ and B_u and dividing by q gives $\|\mathbf{v}\|_\infty \le B_v$. Write $b_t = \mathbf{b}_{i,t}[\alpha]_j \in \{0,1\}$ so $w_1' = \sum_{t=0}^{n_w-1} 2^t b_t$. For ML-DSA-65 and -87, $m = 16 = 2^{n_w}$, so Booleanity of b_t alone enforces $w_1' \in [0, m)$. For ML-DSA-44, $w_1' < 44$ reduces to the degree-3 identity $b_5(b_4 + b_3 b_2) = 0$, since $w_1' \ge 44 = 101100_2$ iff $b_5 = 1$ and either $b_4 = 1$ or $b_3 = b_2 = 1$.

The PIOP proves (3) by $\theta^2 = 1$, $\delta(\delta - \theta) = 0$, $e(e-1)(e-\delta) = 0$ and $2ew_1' + (m-1)e(e-1) = 0$ per coefficient, and $\sum \delta^2 + u_h = \omega$ per record. The first three fix the allowed values of θ, δ and e, while the fourth enforces both modular wrap conditions. Since u_h sums at most 2^{10} entries in $\mathcal{T}_{16}$ per record and $256k + 2^{26} < p$, this sum holds over $\mathbb{Z}$.

Lemma 1 (Equivalence with FIPS 204 verification)**.** *Fix* $\mathsf{pk} = (\rho, \mathbf{t}_1)$ *and a formatted message* M'*. (a) If Algorithm 1 accepts* $(\tilde{c}, \mathbf{z}, \mathbf{h})$*, then* $\mathcal{R}_{\mathrm{FIPS}}$ *holds with* $\sum \delta^2$ *equal to the weight of* $\mathbf{h}$*,* $\|\mathbf{u}\|_\infty \le B_u$ *and* $\|\mathbf{v}\|_\infty \le B_v < 2^{29}$ *(Table 3). (b) If* $\mathcal{R}_{\mathrm{FIPS}}$ *holds, then Algorithm 1 accepts* $(\tilde{c}, \mathbf{z}, \mathbf{h}^\star)$*, where* $\mathbf{w}' = 2\gamma_2 \mathbf{w}_1' + \mathbf{u} \bmod q$ *and* $\mathbf{h}^\star \in \{0,1\}^{k\times 256}$ *marks the coefficients with* $\mathrm{HighBits}(\mathbf{w}') \ne \mathbf{w}_1'$*.*

Proof sketch. Both parts reduce to one coefficient w' of $\mathbf{w}'$ and w_1' of $\mathbf{w}_1'$. Write $r = w' \bmod q = 2\gamma_2 \bar{r}_1 + \bar{r}_0$ with $\bar{r}_0 \in (-\gamma_2, \gamma_2]$, and let $\mathrm{sgn}(x) = 1$ if $x > 0$ and $\mathrm{sgn}(x) = -1$ otherwise. The $q-1$ corner case of Decompose changes neither $\bar{r}_1 \bmod m$ nor $\mathrm{sgn}(\bar{r}_0)$, so $\mathrm{HighBits}(r) = \bar{r}_1 \bmod m$ and

$$\mathrm{UseHint}(h, r) = \big(\bar{r}_1 + h\,\mathrm{sgn}(\bar{r}_0)\big) \bmod m. \tag{5}$$

With $\tilde{r}_1 = w_1' - \delta + em$, the identity $2\gamma_2 m = q-1$ gives $2\gamma_2 w_1' + u = 2\gamma_2 \tilde{r}_1 + r_0 - eq$. For (a), the honest tuple sets $r_0 = \bar{r}_0$, $\delta = h\theta$ and the modular wrap indicator e that reduces $\bar{r}_1 + \delta$ into $[0, m)$, so $\tilde{r}_1 = \bar{r}_1$. For (b), (3) gives $\tilde{r}_1 \in [-1, m+1]$, hence $r = 2\gamma_2(\tilde{r}_1 - \zeta m) + (r_0 - \zeta)$ with $\zeta \in \{0, \pm 1\}$. Unless $r_0 - \zeta$ leaves $(-\gamma_2, \gamma_2]$ or changes sign, this decomposes r. Then $w_1' \equiv \bar{r}_1 + \delta \pmod m$ with $\delta \in \{0, \mathrm{sgn}(\bar{r}_0)\}$, and (5) gives $\mathrm{UseHint}(h^\star, r) = w_1'$ with $h^\star \le \delta^2$. The remaining boundary cases have $r \in \{0, \gamma_2, q-\gamma_2\}$ and $w_1' = 0 = \mathrm{HighBits}(r)$, so $h^\star = 0$. As $\mathbf{h}^\star$ has weight at most $\sum \delta^2 \le \omega$, Algorithm 1 accepts $\mathrm{sigEncode}(\tilde{c}, \mathbf{z}, \mathbf{h}^\star)$ [Nat24, Algorithm 26] (Appendix A). □

A hint-free variant. Applications that require only unforgeability may use the relaxed relation $\mathcal{R}_{\mathrm{ex}}$. It omits the tuples and u_h, commits $\mathbf{u}$ directly with $\|\mathbf{u}\|_\infty \le B_u$, and retains (2), (4) and the bounds on $\mathbf{z}$ and $\mathbf{w}_1'$. Without a hint of weight at most ω, an extracted witness need not form a standard signature. Even so, Dilithium's reduction [KLS18, DKL+18] ignores the hint, so a forged witness still yields the set's own SelfTargetMSIS instance or a Module-SIS solution. $\mathcal{R}_{\mathrm{ex}}$ proves faster (Section 7), while below we use $\mathcal{R}_{\mathrm{FIPS}}$.

4 Algebraic PIOP and Range Checks

4.1 The algebraic PIOP

The $\mathbb{F}_p$ side proves the ring identity (2) and every range of (3) with one sum-check PIOP over Akita [DBK+26] commitments. Plain letters z, u, v, w_1', b_t denote the coefficient tables of $\mathbf{z}_i$, $\mathbf{u}_i$, $\mathbf{v}_i$, $\mathbf{w}_{1,i}'$, $\mathbf{b}_{i,t}$ across all $N^\star = 2^{d_{\mathrm{sig}}}$ records on a *main domain* of d_{main}

Boolean variables (Table 3). Likewise, θ, L, δ and e denote the hint-tuple tables. We first fix the table layout and commitments, then bound the quotient v via a random binary projection, lift the ring identity to $\mathbb{Z}$ and reduce it to public weights. Section 4.2 describes the range checks and sum-check stages.

Exact module dimensions. A multilinear table has 2^n entries. Each record has 256ℓ coefficients in $\mathbf{z}_i$ and $256k$ in $\mathbf{v}_i$. For ML-DSA-65 and -87, (k, ℓ) is $(6, 5)$ and $(8, 7)$, so ℓ is not a power of two. We therefore commit one table per polynomial. For each column $s < \ell$ of $\mathbf{A}$, the table $z[s]$ holds $\mathbf{z}_i[s]_j$ for all $j < 256$ and $i < N^\star$. It has $256N^\star = 2^{8+d_{\text{sig}}}$ entries, so $d_{\text{main}} = 8 + d_{\text{sig}}$. Likewise, v, each bit plane b_t and each entry of the hint tuple have one table per row $\alpha < k$. For ML-DSA-44, $k = \ell = 4$, so the four tables of each witness merge into one table with two extra variables indexing s or α, giving $d_{\text{main}} = 10 + d_{\text{sig}}$.

Committed tables. Round 1, before the bridge challenge sd, commits the witness of $\mathcal{R}_{\text{FIPS}}$: the digits of z, L and v, the bit planes b_t, the sign bit $(1 + \theta)/2$ and the ternary columns δ and e. Round 2, after sd and Ψ, commits the derived tables: the level-a carries $\kappa^{(a)}$, level-2 parities $y_t^{(2)}$ of bit plane t, projections $\hat{v}$, hint slack u_h and base-16 lookup multiplicities cnt (Figure 2). Every other table is virtual, including $w_1' = \sum_{t=0}^{n_w-1} 2^t b_t$, u of Definition 2, $\mathbf{A}z$ and the level-1 parities $y_t^{(1)}$. Padding records set $z = v = 0$ and zero hint weight, so padded cells pass every check.

A quotient bounded by a projection. Aerie's divisibility projection [CDBT26] requires projected outputs below $p/2$, which fails for ML-DSA. For ML-DSA-44, the honest residual $\Delta_0 = \mathbf{A}z - T - 2\gamma_2 w_1' - u$ has root-mean-square size $2^{42.4}$, about 2^{19} times Falcon-512's. A projection over 2^{25} positions then has standard deviation about $2^{68.4}$ and wraps modulo $p \approx 2^{64}$. We therefore commit the quotient v as two 16-bit digits v_{lo} and v_{hi} and prove that the virtual table $\Delta = \mathbf{A}z - T - 2\gamma_2 w_1' - u - qv$ vanishes on the hypercube. An extracted v is an arbitrary field element, whose centered representative enters (2).

Without a bound on v, setting $v = q^{-1}\Delta_0 \bmod p$ would satisfy $\Delta = 0$ for any z. A random $\{0, 1\}$ projection bounds v without looking up its digits. After round 1, a dedicated challenge site samples one random matrix $\Psi \in \{0, 1\}^{128\times 256k}$ shared across all records. Each record's $256k$ positions split into n_Ψ row blocks, two for ML-DSA-87 and one otherwise. Let $v_{b,i}$ be the quotient of record i on block $b < n_\Psi$, and Ψ_o be row o of Ψ on that block. Round 2 commits $\hat{v}_{o,b,i} = \langle \Psi_o, v_{b,i} \rangle$. The lookup proves $-B_\Psi \le \hat{v} < B_\Psi$ (Section 4.2), and (8) ties $\hat{v}$ to v.

By Lemma 1, a valid signature satisfies $\|\mathbf{v}_i\|_\infty \le B_v$. Each output sums at most $256k/n_\Psi$ entries, so $|\hat{v}_{o,b,i}| \le (256k/n_\Psi)B_v < B_\Psi$ (Table 3), and the honest prover never aborts here. Conversely, since v is committed before Ψ, any entry exceeding $2B_\Psi$ in centered magnitude pushes an output out of range (Lemma 2).

Lemma 2 (Projection bound)**.** *Let $v_{b,i} \in \mathbb{F}_p^{256k/n_\Psi}$ be fixed before Ψ is drawn with independent uniform bits. Suppose $|v_{b,i}(x_0)|_c > 2B_\Psi$ for some x_0, where $|\cdot|_c$ is the centered absolute value in $\mathbb{F}_p$. Then $|\langle \Psi_o, v_{b,i} \rangle|_c \le B_\Psi$ in $\mathbb{F}_p$ holds for all $o < 128$ with probability at most 2^{-128}. If all $v_{b,i}$ are fixed before Ψ and some has such a coordinate, then every record passes with probability at most 2^{-128}, for every N.*

Proof. Fix a row o and its bits other than $\Psi_o(x_0)$. The $\mathbb{F}_p$ inner product takes two values that differ by $v_{b,i}(x_0) \bmod p$. Since $2B_\Psi < p/2$, if both lay within B_Ψ of zero, then $|v_{b,i}(x_0)|_c \le 2B_\Psi$. So each of the independent rows passes with probability at most $1/2$. For the batch, the first record with a large coordinate is fixed before Ψ, so its bound of 2^{-128} holds for the batch without a union bound. □

Lemma 3 lifts $\Delta = 0$ to $\mathbb{Z}$ under the ranges of z (Section 4.2), u and w_1' (Section 3.2), and v (Lemma 2).

Lemma 3 (Integer lift)**.** *Let* $\|z\|_\infty \le B_z$, $\|u\|_\infty \le B_u$, $|v|_c \le 2B_\Psi$ *and* $w'_1 \in [0,m)^{k\times 256}$. *With* $\mathbf{A}$ *in centered form, every coefficient of* Δ *computed over* $\mathbb{Z}$ *satisfies*

$$\begin{aligned}|\Delta^{\mathbb{Z}}| \le 128\ell(q-1)B_z + 2^d\tau(2^{10}-1)\\ + (q-1-2\gamma_2) + B_u + 2qB_\Psi < p/2,\end{aligned} \tag{6}$$

so $\Delta = 0$ *in* $\mathbb{F}_p$ *implies the ring identity* (2) *over* $\mathbb{Z}$.

Proof. The first four terms are the bounds of Section 3.2 on $|\mathbf{A}z|$, $|T|$, $2\gamma_2 w'_1$ and $|u|$, and the last bounds $|qv|$. Their sum is the lift row of Table 3, below $p/2$. An integer of magnitude below $p/2$ that vanishes modulo p is zero. □

Ring identity and public weights. Stage 3 tests the k row identities of (2) at the stage-2 terminal point $r = (r_{\text{coef}}, r_{\text{sig}})$, combined with row coefficients $\zeta_\alpha \in \mathbb{K}$:

$$\begin{aligned}\textstyle\sum_\alpha \zeta_\alpha T_\alpha(r) = \sum_{s,j',i} w_A^{(s)}(j',i)\, z^{(s)}(j',i)\\ - \textstyle\sum_{\alpha,x} \zeta_\alpha \operatorname{eq}(r,x)\big(2\gamma_2 w'_1 + u + qv\big)^{(\alpha)}(x).\end{aligned} \tag{7}$$

The verifier evaluates $T_\alpha(r)$ from the T_i, and $z^{(s)}$ denotes column table s of z. For ML-DSA-65 and -87, $d_{\text{main}} = 8 + d_{\text{sig}}$ has no row variables, so stage 3 batches the k row claims with verifier challenges $\zeta_\alpha \in \mathbb{K}$. For ML-DSA-44, the stage-2 point on $d_{\text{main}} = 10 + d_{\text{sig}}$ variables also carries two row coordinates r_{row} beyond r, fixing $\zeta_\alpha = \operatorname{eq}(r_{\text{row}}, \alpha)$. The public weight $w_A^{(s)}(j',i) = \operatorname{eq}(r_{\text{sig}}, i)\, \Phi_{\tilde{\mathbf{A}}_{i,s}}[j']$ pulls $\operatorname{eq}(r,\cdot)$ back through $z \mapsto \mathbf{A}z$, where $\tilde{\mathbf{A}}_{i,s} = \sum_\alpha \zeta_\alpha \mathbf{A}_i[\alpha, s]$ and Φ_f is the negacyclic pull-back of $\operatorname{eq}(r_{\text{coef}},\cdot)$ under multiplication by f [CDBT26]. Stage 3 also proves the projection identity at the suffix $r' = (r'_o, r'_{b,i})$ of the stage-2 point,

$$\textstyle\sum_x W_\Psi(x)\, v(x) = \sum_{(o,b,i)} \operatorname{eq}(r', (o,b,i))\, \hat{v}(o,b,i), \tag{8}$$

where $W_\Psi = \operatorname{eq}(r'_{b,i}, (b,i)) \sum_o \operatorname{eq}(r'_o, o)\Psi_o$ on block (b,i).

4.2 Range checks by lookup

Why LogUp-GKR. Akita commits only dense tables of small digits. Plain LogUp [Hab22] commits full-field helper columns, while Lasso [STW24] and Shout [STZ26] commit one-hot tables of 2^{16} cells per lookup. Aerie's leaf check [CDBT26] on a 16-bit digit has degree 2^{15}. LogUp-GKR [PH23] commits only the looked-up columns and table multiplicities, proving the fractional sum via the Goldwasser–Kalai–Rothblum (GKR) protocol [GKR08] at 0.9 to 1.2 B/sig for $N = 2^{15}$ (Table 8).

Decomposition and layout. The lookup table $\mathcal{T}_{16} = [0, 2^{16})$ covers all 16-bit digits. Each of z and L is written as

$$x + B = x_{\text{lo}} + s_1 x_{\text{hi}} + s_2 a_x \tag{9}$$

with a digit $x_{\text{lo}} \in \mathcal{T}_{16}$, a small top digit $x_{\text{hi}} \in [0, n_{\text{hi}})$ and, where needed, a slack bit a_x (Figure 2). For L and ML-DSA-44's z, the interval width stays below 2^{18}, so $n_{\text{hi}} = 2$ suffices. For ML-DSA-65 and -87, $\gamma_1 = 2^{19}$ raises $2B_z$ near 2^{20}, so z takes $n_{\text{hi}} = 8$. The shifts s_1 and s_2 make the translated copies of $\mathcal{T}_{16}$ overlap to cover $[0, 2B_z]$ for z ($B = B_z$) and $[0, \gamma_2)$ for L ($B = 0$) exactly. Non-unique decompositions in the overlaps are harmless, since downstream checks read only the reconstructed value x. One table thus enforces the exact bounds $|z| \le B_z$ and $L \in [0, \gamma_2)$, although neither bound is a power of two. Each shifted projection $\hat{v} + B_\Psi$ splits into $\hat{v}_0, \hat{v}_1 \in \mathcal{T}_{16}$ and top digits $\hat{v}_2$ to $\hat{v}_4$ of at most three bits, and $u_h \in \mathcal{T}_{16}$.

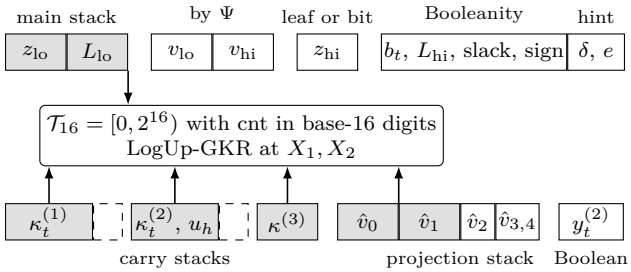


Figure 2: Committed $\mathbb{F}_p$ columns ($\mathcal{T}_{16}$ shaded, padding dashed).

Leaves and Booleanity. Top digits of four or eight balanced values, including those of $\hat{v}$, use Akita's leaf check [DBK+26, Section 6.1]. For a balanced digit D, a polynomial of degree at most four in $P_D = D(D+1)$ vanishes over $\mathbb{F}_p$ precisely on D's alphabet. Stage 2 zero-checks this polynomial, and stage 3 proves $P_D = D(D+1)$. The slack bits, binary top digits ($n_{\mathrm{hi}} = 2$) and sign bit pass a *Booleanity* zero-check $x(x-1) = 0$. The ternary columns δ and e need no lookup, as $\delta(\delta - \theta) = 0$ and $e(e-1)(e-\delta) = 0$ fix their values.

One table, one identity. Merging tables into one identity via per-table offsets is unsound over $\mathbb{F}_p$. LogUp's lemma [Hab22, Lemma 5] guarantees only that the multiset of all looked-up values lies in the merged table. Since an extracted cell is an arbitrary field element, a cell meant for one table can match an entry of another table's shifted copy. We therefore use $\mathcal{T}_{16}$ alone for every digit and carry. Multiplicities need no range check, as LogUp's lemma holds for any cnt when p exceeds the number of looked-up cells.

Fraction trees and GKR. Let $\mathcal{F}$ be the multiset of looked-up values, including padding, and let $\mathrm{cnt}(\psi)$ be the committed multiplicity of $\psi \in \mathcal{T}_{16}$. At a post-commitment challenge $X \in \mathbb{K}$, the lookup proves

$$\sum_{f \in \mathcal{F}} \frac{1}{X - f} = \sum_{\psi \in \mathcal{T}_{16}} \frac{\mathrm{cnt}(\psi)}{X - \psi}.$$

The looked-up columns of each domain form a *stack* padded with public zeros: round 1 fills the main stack, and round 2 fills the three carry stacks and the projection stack (Figure 2). Each stack and $\mathcal{T}_{16}$ form binary fraction trees with leaves $(1, X - f)$ and $(\mathrm{cnt}(\psi), X - \psi)$. A node ϕ with children ϕ_0, ϕ_1 holds $(\mathrm{n}_\phi, \mathrm{d}_\phi) = (\mathrm{n}_{\phi_0}\mathrm{d}_{\phi_1} + \mathrm{n}_{\phi_1}\mathrm{d}_{\phi_0},\ \mathrm{d}_{\phi_0}\mathrm{d}_{\phi_1})$. Per challenge, the verifier checks that all root denominators are nonzero and that the stack roots sum to the $\mathcal{T}_{16}$ root.

Across both challenges, all trees are aligned at the root and proved layer by layer [PH23]. Each layer runs one batched degree-3 sum-check and merges each pair of child claims to a single point by a random linear combination. At the bottom layer, each stack leaves a claim on its committed columns for stage 3, while the $\mathcal{T}_{16}$ tree leaves a claim on the digits of cnt.

Soundness with two challenges. With one challenge, a cheating prover passes with probability up to $(|\mathcal{F}| + 2^{16} - 1)/|\mathbb{K}|$, which is $2^{-101.7}$ for ML-DSA-44 at $N = 2^{15}$ (Table 3) and would need 27 bits of proof of work. We therefore sample two independent challenges $X_1, X_2 \in \mathbb{K}$ at one site. The GKR layers add the round errors of [PH23, Proposition 1], each covered by proof of work (Section 6).

Lemma 4 (Two independent challenges)**.** *Let the $|\mathcal{F}| < p$ looked-up values and* cnt *be committed before independent uniform $X_1, X_2 \in \mathbb{K}$ are drawn. If a looked-up value lies outside $\mathcal{T}_{16}$, the identity holds at both challenges, with nonzero root denominators, with probability at most $\big((|\mathcal{F}| + 2^{16} - 1)/|\mathbb{K}|\big)^2$. This bound stays below 2^{-128} for $|\mathcal{F}| \le p/2$. At $N = 2^{15}$, it is at most $2^{-201.7}$ across sets.*

Table 4: Parity-bridge levels per segment.

Level a	J_a	m_a	n_a	$1/\binom{m_a}{n_a}$
1	2^{19}	2^{13}	14	$2^{-145.64}$
2	2^{19}	2^{13}	14	$2^{-145.64}$
3	$2^{13}n_w$	4,096	18	$2^{-163.44}$
sum				$2^{-144.64}$

Proof. By LogUp's lemma [Hab22, Lemma 5], the two sides differ by a nonzero rational function whose numerator, fixed by the commitments, has degree at most $|\mathcal{F}|+2^{16}-1$. The verifier rejects a pole, so acceptance at X_j requires a root of the numerator. Independence squares the bound, and the event $X_1 = X_2$ needs no check. □

Sum-check stages. Three batched sum-checks of d_{main} rounds follow Akita round 2. On the stage branch, stage 1 starts at a zero-check point $\vartheta \in \mathbb{K}^{d_{\text{main}}}$ and checks bridge levels 2 and 3 along with Booleanity of $y_t^{(1)}$ and $y_t^{(2)}$. At stage 1's terminal point, stage 2 checks bridge level 1, the leaf polynomials in P_D, $e(e-1)(e-\delta) = 0$ and, for ML-DSA-44, $b_5(b_4+b_3b_2) = 0$. Stage 3 joins both branches to check the ring and projection identities (7) and (8), $P_D = D(D+1)$ and the remaining Booleanity and hint identities. It also reduces every earlier claim on committed tables and lookup stacks to its terminal point $r^{(3)}$. One batched Akita opening then verifies the multiplicity digits at the $\mathcal{T}_{16}$ point and every other committed table at $r^{(3)}$.

5 Keccak Argument and Parity Bridge

The GF(2) side proves the commitment-hash check $\tilde{c}_i = \mathsf{H}(\mu_i \,\|\, \text{w1Encode}(\mathbf{w}'_{1,i}), \lambda/4)$ of (4) for every live record. We describe its sponge circuit, then the bridge that ties the circuit's input bits to the $\mathbb{F}_p$ side: the random sparse matrices, the proof on each side and the soundness bound.

The sponge circuit. Per record, the circuit chains n_{perm} Keccak-f[1600] permutations and a $\lambda/4$-byte squeeze [Nat15]. Its input is μ_i and the w1Encode stream, a fixed permutation of the bits $\mathbf{b}_{i,t}$ [Nat24, Algorithms 16 and 28]. One Flock [BRW26] R1CS instance per record encodes the chain. It commits the input bits, a constant column cst, the output bits and each round's χ outputs, restricted in the final round to those that reach the output. Every other value is a virtual XOR (Lemma 6). One evaluation claim at a random *tie point* fixes the public inputs μ_i, $\tilde{c}_i$ and the constant $\mathsf{cst} = 1$.

Two commitments to the same bits. Over GF(2), multiplying by a public $\{0,1\}$ matrix Γ_a requires no multiplication constraints. Over $\mathbb{F}_p$, evaluating $\Gamma_a x$ as an integer sum adds no committed column in a sum-check, whereas modulo-2 reduction is nonlinear. The $\mathbb{F}_p$ side therefore writes each entry of $\Gamma_a x$ as $y + 2\kappa$ with $y \in \{0,1\}$ and $\kappa \in \mathcal{T}_{16}$, so $y = \Gamma_a x \bmod 2$.

Random sparse matrices. We expand the 32-byte verifier challenge sd with customizable SHAKE256 (cSHAKE256) [KCP16], NIST's domain-separated variant of SHAKE256, into three sparse binary matrices $\Gamma_a \in \{0,1\}^{m_a \times J_a}$ for $a \in \{1,2,3\}$. Each column of Γ_a indicates an independent uniform n_a-subset of $[m_a]$ (Table 4). In each segment σ, both sides apply Γ_a to blocks of at most J_a inputs to prove $Y_\sigma = \Gamma_3\Gamma_2\Gamma_1 b$ on their copy of b. Level 1 maps each block of a bit plane to 2^{13} parities. Level 2 maps a segment's level-1 parities per bit plane to 2^{13}, and level 3 maps all n_w bit planes to the public $Y_\sigma \in \{0,1\}^{4096}$. The last column of Table 4 gives each level's error (Theorem 1).

The $\mathbb{F}_p$ side. For each level a, block and bit plane t, the prover shows

$$\textstyle\sum_\xi \Gamma_a[o,\xi]\, x^{(a-1)}(\xi) = y^{(a)}(o) + 2\kappa^{(a)}(o), \tag{10}$$

with $x^{(0)} = b_t$, $x^{(a)} = y^{(a)}$ and $y^{(3)} = Y_\sigma$. We omit the subscript t on $y_t^{(a)}$ and $\kappa_t^{(a)}$ at levels 1 and 2. It also proves Booleanity of b_t and the committed parities $y^{(2)}$, and looks up every carry $\kappa^{(a)}$ in $\mathcal{T}_{16}$. The level-1 parities $y^{(1)}(o) = \sum_\xi \Gamma_1[o,\xi] b_t(\xi) - 2\kappa^{(1)}(o)$ are virtual: stage 1 zero-checks their Booleanity and uses them at level 2, while stage 2 proves the level-1 identity. Each identity is a degree-2 sum-check whose public weight applies $\Gamma_a^\top$ to an equality table.

For any sd, Boolean $x^{(a-1)}$ and $\kappa^{(a)} \in \mathcal{T}_{16}$, the left side of (10) lies in $[0, J_a] \subseteq [0, 2^{19}]$ and the right side in $[0, 2^{17})$, both below p. Equation (10) thus holds over $\mathbb{Z}$ and fixes the parity $y^{(a)}(o)$, whereas an unbounded carry modulo p could fit either bit. For completeness, the honest prover aborts if a row of some Γ_a exceeds its row-weight bound, an event below 2^{-200} over sd. Otherwise $\kappa^{(a)} < 2^{10} \in \mathcal{T}_{16}$.

The GF(2) side: a virtual parity chain. The GF(2) side commits only the hash witness. In characteristic two, $\Gamma_3\Gamma_2\Gamma_1$ is linear over $\mathbb{F}_{2^{128}}$. At a random *chain point* with equality table eq_Y, the verifier's target $\langle \mathrm{eq}_Y, Y\rangle$ equals $\langle W_\Gamma, b'\rangle$ for $W_\Gamma = (\Gamma_3\Gamma_2\Gamma_1)^\top \mathrm{eq}_Y$. One degree-2 sum-check over the variables of b' reduces this claim to $b'(r)$, which the sponge opening proves via ring switching [DP25]. The prover applies the sparse Γ_a directly without forming W_Γ, and the verifier evaluates $W_\Gamma(r)$ in about 10^7 additions for any N.

Soundness. Let $\Delta_b = b \oplus b'$ over all positions, padding included, fixed by events 1 and 2 before sd. Matching images on both sides give $\Gamma_3\Gamma_2\Gamma_1\Delta_b = 0$ in every segment.

Theorem 1 (Parity bridge)**.** *Let $\Gamma_1, \Gamma_2, \Gamma_3$ be drawn as above with parameters (m_a, n_a). For every fixed $\Delta_b \neq 0$, $\Pr_{\mathsf{sd}}[\Gamma_3\Gamma_2\Gamma_1\Delta_b = 0] \le \sum_{a=1}^{3} \binom{m_a}{n_a}^{-1}$, independently of the block widths J_a, the number of blocks, the batch size and the parameter set. For Table 4 it is $2^{-144.64}$.*

Proof. Consider one level Γ_a with columns E_ξ, independent uniform n_a-subsets of $[m_a]$, and one block input $x \neq 0$ with support U. If $U = \{\xi\}$, then $\Gamma_a x = 1_{E_\xi} \neq 0$. If $|U| \ge 2$, fix $\xi_0 \in U$ and condition on the other columns of U. Then $\Gamma_a x = 0$ iff E_{ξ_0} is the support of their fixed sum, which has probability at most $1/\binom{m_a}{n_a}$. A level applies the same Γ_a to every block. Its output vanishes only if every nonzero block maps to zero, in particular one fixed nonzero block, so the bound for one block suffices without a union bound. For the composition let $\Delta_b^{(0)} = \Delta_b$ and $\Delta_b^{(a)} = \Gamma_a \Delta_b^{(a-1)}$. The events "$a$ is the first level with $\Delta_b^{(a)} = 0$" are disjoint. Each requires $\Delta_b^{(a-1)} \neq 0$, so the bound above applies to it. Here $\Delta_b^{(a-1)}$ depends only on $\Gamma_1, \ldots, \Gamma_{a-1}$, which are independent of Γ_a since the levels use distinct hash inputs. Summing per-level bounds gives the claim. □

The GF(2) commitment binds the prover only to a Johnson list, and the extractor of Theorem 2 keeps the seed-expansion oracle fixed while redrawing sd. At rate 1/2, the level-0 code decodes at agreement 0.02 above the Johnson threshold, inside the proven regime [DKT26, Theorem 4.4], with list size at most $L_J = 35.36$ [BRW26, Equation (10)].

Corollary 1 (List-weighted bound)**.** *If the batch is proven in one joint* GF(2) *argument whose Ligerito commitment has Johnson list size L_J, then the probability over sd and the seed-expansion oracle that some codeword of the list differs from b yet shares its image in every segment is at most $L_J \cdot 2^{-144.64}$, which is $2^{-139.4}$ at rate 1/2. Except with probability $\varepsilon_{\mathrm{seed}} < 2^{-2^{110}}$ over the choice of that oracle, the probability over sd alone for the fixed oracle is at most twice this bound, $2^{-138.4}$ (Claim 1).*

Claim 1 (Uniform seed density)**.** *A segment's level-1 space has $J = n_w 2^{13} 2^{\lceil \log_2 256k \rceil} \le 2^{26}$ bits, for n_w bit planes, 2^{13} records and $256k$ positions padded to a power of two. Except with probability $\varepsilon_{\mathrm{seed}} < 2^{-2^{110}}$ over the choice of the seed-expansion oracle, every nonzero Δ_b in this space satisfies $\Gamma_3\Gamma_2\Gamma_1\Delta_b = 0$ for at most a $2 \cdot 2^{-144.64} = 2^{-143.64}$ fraction of the 2^{256} seeds.*

Proof. Injective framing makes distinct seeds' matrix triples independent [CDBT26, Lemma C.2]. For a fixed nonzero Δ_b, Theorem 1 bounds the expected fraction of seeds with $\Gamma_3\Gamma_2\Gamma_1\Delta_b = 0$ by $2^{-144.64}$. By a Chernoff bound over the 2^{256} seeds, this fraction exceeds twice that mean with probability at most $\exp(-2^{111.36}/3)$. A union bound over $2^{2^{26}}$ vectors gives $\varepsilon_{\text{seed}} < 2^{-2^{110}}$, covering adaptive Δ_b. □

Broader applicability. The parity bridge interacts with both commitment schemes solely through linear evaluation claims, independent of their internal structure. Any system that commits a shared bit vector over $\mathbb{F}_p$ with $p > 2^{19}$ and over a binary extension field can use this three-level check. List-decoded binary commitments multiply the soundness error by the list size, and the GF(2) side commits no extra cells. Beyond ML-DSA, the bridge pairs bitwise primitives such as Keccak, SHA-2 or AES with lattice or prime-field arithmetic. This covers post-quantum credentials and ring signatures, as well as virtual machines with separate bitwise and arithmetic execution.

6 Security

We state our security model, extend Akita's [DBK+26] extractor to the forked transcript (Lemma 5), and analyze the SHAKE256 [Nat15] circuit and GF(2) list decoding (Lemmas 6 and 7). With our proof-of-work schedule, these give knowledge soundness (Theorem 2), reducing aggregate unforgeability (Theorem 3) to SUF-CMA of ML-DSA [Nat24].

Security model. Knowledge soundness (Theorem 2, proved in Appendix B) and aggregate unforgeability (Theorem 3) hold in the classical ROM. Extraction reduces to Module-SIS via Akita's binding [DBK+26], with conditional resampling given a prefix of decoded fold coordinates [DBK+26, Lemma 10.2] (Lemma 5 (v)). The GF(2) side relies on no conjecture and adds no hardness term (Lemma 7). Aggregate unforgeability reduces to FIPS 204 SUF-CMA [Nat24, Section 3.1], with advantage $\mathrm{Adv}^{\mathrm{SUF}}_{\text{ML-DSA}}$ on M'.

Checkpoint extraction. Akita's extractor [DBK+26, Theorem 10.17] rewinds the prover below two stage types: at a *fold stage* (an opening fold coordinate), a *combiner* merges sibling runs, while at a *checkpoint* it tests a candidate witness from later openings. A failed test leaves a false claim that later stages preserve outside bad sets. Whereas Akita hashes the full prefix at every stage, our transcript forks and fold coordinates omit the Keccak proof, so each stage queries a *challenge input* on its causal history.

Lemma 5 (Projected challenge inputs)**.** *Let every stage query a challenge input determined by its transcript prefix. Suppose (i) each input fixes every earlier checkpoint and fold input, (ii) each fold input holds all data read by Akita's combiner and by checks and checkpoints from that stage onward, (iii) each checkpoint test and (iv) each bad set depend on the prefix through this input alone, and (v) fold stages admit conditional resampling [DBK+26, Lemma 10.14]. Then Akita's Theorem 10.17 holds, each queried input adding its stage error.*

Proof sketch. By (i), later challenge inputs partition by a checkpoint's input, and a prover querying none in a part runs identically when that part is redrawn (Akita's Claim 2). A rerun from a failed checkpoint ending with a passing test must, by (iii) and (iv), hit a bad set fixed before its answer is drawn (Claim 1). By (ii), a combined fold opening serves the reference run, re-deriving Akita's Appendix E.2. □

The hash check. Flock [BRW26] proves (4) on the sponge circuit of Section 5. While Flock binds only to a Johnson list, the circuit admits only honest evaluations.

Lemma 6 (Sponge circuit exactness)**.** *Let an assignment to the columns of one sponge instance satisfy every R1CS constraint with its constant column* $\mathsf{cst} = 1$*. Then its* χ *columns and its* 2λ *output bits match the honest evaluation. The output bits are*

$\mathsf{H}(\mu \| \text{w1Encode}(\mathbf{w}'_1), \lambda/4)$ *for the input* μ *and* $\mathbf{w}'_1$*, and each input has a unique such assignment.*

By induction over the $24n_{\text{perm}}$ rounds, an honest input forces every committed AND bit through the χ constraints. Flock's lincheck and the tie point enforce $\mathsf{cst} = 1$, letting Theorem 2 apply Lemma 6 via Lemma 7.

Proven Johnson regime for GF(2)**.** We stay inside the proven Johnson regime [BSCI+20, DKT26] and avoid beyond-Johnson conjectures, which fail on binary additive domains [DKTZ26]. Ligerito [NA25] folds row pairs of a committed matrix by random linear combinations. Mutual correlated agreement (MCA) states that if a fold agrees with codewords on a column set, so do both rows except on a few *exceptional* challenges. Flock needs MCA up to the Johnson bound [BRW26, Theorem 8], formerly conjectured in WHIR [ACFY25, Conjecture 4.12] and proven with weaker bounds in [Hab25, Theorem 2], [BCGM25, Theorem 9.2]. Dao, Kominers and Thaler prove MCA over all fields and evaluation sets [DKT26, Theorem 4.4], leaving fewer than $2^{37.1}$ exceptional $\mathbb{F}_{2^{256}}$ challenges per row pair on Flock's $\mathbb{F}_{2^{128}}$ domains at $N \le 2^{15}$ (Appendix B). Calling a rate-1/2 Flock configuration *sound* when every internal site stays within 2^{-128} after work, Lemma 7 tracks a failing claim per listed codeword.

Lemma 7 (Flock round by round)**.** *For the Flock batch of a joint* GF(2) *argument under a sound rate-*1/2 *configuration, every codeword* C' *of the Johnson list of its commitment and every challenge input carry a predicate "*C' *fails". Specifically, (i) at the first input,* C' *fails iff it violates the R1CS constraints or* $\mathsf{cst} = 1$*, or a claim in the input is false, (ii) every input fixes a bad set outside which every failing* C' *still fails at the next input, (iii) if the verifier accepts, some* C' *does not fail at the end, and (iv) the work nonces compose [DBK+26, Proposition 10.3] to* 2^{-128} *per challenge input.*

Proof sketch. Following Flock's site-by-site analysis, each bad set before the Ligerito folds is the zero set of a degree-≤ 384 polynomial by Schwartz–Zippel, unioned over at most L_J codewords. Across the folds, MCA transports the list codeword by codeword [ACFY25, Lemma 4.13]. At each recursive level, the verifier draws two out-of-domain points before the prover sends the folded word, leaving at most one passing listed codeword, and proximity queries reject any codeword beyond the decoding radius. □

The proof-of-work schedule. Every challenge site ι of the PIOP, tie point and parity chain uses difficulty D_ι from (1). Sites with $D_\iota > 0$ carry a $(D_\iota + 7)$-bit nonce, bounding the honest search at $2^{D_\iota+7}$ trials, while sites with $D_\iota = 0$ omit it. As in Aerie [CDBT26], work bits and challenges come from disjoint slices of one oracle output, bounding every site by 2^{-128} per query. The lookup and bridge-seed sites need no work (Lemma 4 and Corollary 1), whereas we set $D_\iota = 1$ at Ψ despite (1) giving 0 (Lemma 2). At $N = 2^{15}$, a transcript has $R = 2{,}113$, 2,167 and 2,115 sites for ML-DSA-44, -65 and -87 (Appendix B).

Digests in the transcript. Akita's challenge *stream* is initialized from a digest of the PIOP transcript. Hashed statement and transcript entries [DBK+26, Lemma 10.47], both branch outputs and the S Flock challenger digests enter the transcript as digests. With $\bar{Q} = Q + R$, expanding them adds $\varepsilon_{\text{hash}} = \sum_o \big(\binom{Q_o}{2} + Q_o V_o\big) 2^{-256} + (5 + 3S^2)\bar{Q}^2/2^{257} + \bar{Q}/2^{191} + \varepsilon_{\text{seed}}$, where o ranges over digest types of hashed entries and Merkle trees with Q_o queries and V_o digest positions [DBK+26, Equation (212)], and $\varepsilon_{\text{seed}}$ is from Claim 1.

Knowledge soundness. Following Aerie [CDBT26], we compile a round-by-round PIOP [CCH+19, BGK+23] with Akita and Fiat–Shamir [AFK23]. The extractor outputs round-1 tables f_1 (z, v, b and the tuples) in $\mathbb{F}_p$ [DBK+26, Remark 10.41] and no Flock witness, while Ligerito's trees extract straight-line from the query log [BSCS16] at event 2. Setting $u_h = \omega - \sum \delta^2$ makes $\mathcal{R}_{\text{FIPS}}$ a function of f_1 alone, with $\sum \delta^2 \le \omega$ enforced by round 2's range check on u_h. Outside the errors below, the PIOP gives (3), Booleanity,

Table 5: Aggregation performance and baselines at $N = 32{,}768$.

System	Cat.	ISA	Prove (s)			Verify (ms)			B/sig	Compression
			16C32T	8C16T	1T	16C32T	8C16T	1T		
Aerie-512 [CDBT26]	1	AVX2	0.99	1.23	7.64	15.2	17.3	89.6	42.1	15.8×
		AVX-512F	0.99	1.24	7.59	12.6	13.2	51.2		
Aerie-1024 [CDBT26]	5	AVX2	1.67	2.22	14.58	21.3	24.4	154.8	42.1	30.4×
		AVX-512F	1.72	2.20	14.48	16.2	16.8	82.2		
Ours-44	2	AVX2	2.68	4.20	32.49	75	113	904	49.2	49.2×
		AVX-512F	2.35	3.45	24.59	61	84	632		
		IFMA	2.35	3.42	24.48	61	86	630		
Ours-65	3	AVX2	3.31	4.95	38.71	120	192	1,598	65.4	50.6×
		AVX-512F	2.90	4.09	29.48	85	126	1,000		
		IFMA	2.89	4.04	29.36	81	120	952		
Ours-87	5	AVX2	4.13	6.51	50.95	177	301	2,602	82.7	55.9×
		AVX-512F	3.62	5.33	38.53	117	187	1,558		
		IFMA	3.61	5.25	38.31	112	174	1,483		
Alloul26-87 [All26]	5	AVX2	179.2	308.5	2,637.7	5,849	8,164	58,431	151.0	30.6×
		AVX-512F	176.3	300.5	2,548.5	5,974	8,091	51,727		

Note: Cat.: NIST category. Uncached keys, excluding keys and messages from B/sig. AVX-512F omits IFMA and includes BW and VBMI (or x86-64-v4 for Alloul26). Aerie: medians of five processes, prove time without setup (205 and 410 ms on 16C32T), B/sig with 40 B salt, against 666 and 1,280 B Falcon signatures. Alloul26: Johnson-regime WHIR, medians of three processes. Native verification (pq-crystals, AVX2): 31, 47 and 74 ms on 16C32T and 636, 919 and 1,451 ms on 1T for ML-DSA-44, -65 and -87.

the integer bridge identities, $|v|_c \le 2B_\Psi$ (Lemma 2) and (2) (Lemma 3). If (4) fails on extracted bits b of a live record, every listed sponge codeword violates the Flock statement or differs from b (Lemma 6). The sites of the joint GF(2) argument rule out the first case, and the bridge rules out the second except with probability $2^{-138.4}$ under a fixed seed-expansion oracle (Corollary 1). An Akita schedule [DBK+26, Definition 9.2, Theorem 10.46] is *sound* when extraction-admissible with internal sites within 2^{-128} after work. Our Flock configurations and Akita schedules are sound up to $N = 2^{15}$ (Appendix B), where $|\mathcal{F}| < 2^{28}$ (Table 3).

Theorem 2 (Knowledge soundness)**.** *Fix a parameter set, a batch size N with $|\mathcal{F}| \le p/2$, the relation $\mathcal{R}_{\mathrm{FIPS}}$, the batch proven in one joint* GF(2) *argument under a sound rate-*1/2 *Flock configuration, the proof-of-work schedule above and a sound Akita schedule sch. Model the transcript sponge, the Flock challengers, Akita's setup and challenge streams, the digests of hashed transcript entries, the Merkle hash and the seed-expansion function as independent random oracles. For every prover with auxiliary environment independent of these oracles [DBK+26, Section 10.3] and at most Q oracle queries, an extractor preserves the first run. It runs the prover at most $4(Q + R + 2)\,\mathrm{Tree}^0_{\mathrm{ck}}(sch)$ times in expectation, for the tree size of sch [DBK+26, Proposition 10.44]. Whenever the verifier accepts the first run, the extractor outputs a witness of $\mathcal{R}_{\mathrm{FIPS}}$ for every live record of that run's statement, except with probability at most*

$$\varepsilon_{\mathrm{ext}} \le (Q + R)\,2^{-128} + \varepsilon_{\mathrm{hash}} + \textstyle\sum_{\mathcal{I}} \mathrm{Adv}^{\mathrm{MSIS}}_{\mathcal{I}},$$

with R the transcript sites, $\varepsilon_{\mathrm{hash}}$ as above and $\mathcal{I}$ over the Module-SIS instances of Akita's schedule [DBK+26, Lemma 10.20].

Proof sketch. Game hops expand every digest in a challenge input ($\varepsilon_{\mathrm{hash}}$). Lemma 5 applies with checkpoints at sd, testing $\mathcal{R}_{\mathrm{FIPS}}$ on f_1, and at X, testing the PIOP identities. □

Publishing w1Encode($\mathbf{w}'_{1,i}$) (Section 7) omits the joint GF(2) argument and Lemmas 6 and 7.

The total error. As in Aerie [CDBT26], each oracle query contributes at most 2^{-128}, so success probability ε requires about $\varepsilon\, 2^{128}$ queries, matching classical AES-128 key search. At $N = 2^{15}$, the R sites add at most $2^{-116.91}$ (for ML-DSA-65). Every Module-SIS instance of Akita's schedule reaches at least 2^{128} security in quantum core-SVP, the cost of one sieving call at the reduction's block size [DBK+26, Appendix G]. We make no claim in the quantum ROM (QROM), where Grover search on bad sets and the quadratic query loss of [CMS19] roughly halve per-query security. As in Aerie, QROM analysis is future work.

Security level across parameter sets. Whereas NIST categories classify the signature scheme, whose unforgeability an aggregate inherits via Theorem 3, the proof system gives 2^{-128} soundness per query for every set. This matches ML-DSA-44, whose category 2 rests on collision search on a 256-bit hash [Nat24, Section 4]. ML-DSA-65 and -87 aggregates stay below categories 3 and 5, as Aerie on Falcon-1024 stays below category 5, since matching higher categories requires a larger challenge field.

Completeness. In the ROM, for $|\mathcal{F}| \le p/2$, the honest prover fails on a valid batch with probability below $2^{-110} + \varepsilon_{\text{Akita}}$, where $\varepsilon_{\text{Akita}}$ is Akita's completeness error [DBK+26, Theorem 10.9]. The lookup challenges hit $\mathcal{T}_{16}$ with probability about 2^{-111}, Flock's zero-check hits a pole below 2^{-122}, each work search exceeds its trial bound below 2^{-184}, and a bridge row exceeds its bound below 2^{-200}.

Aggregate unforgeability. By Lemma 1, an extracted witness yields a valid ML-DSA signature, turning an aggregate forger $\mathcal{A}$ into a SUF-CMA adversary $\mathcal{B}$ against ML-DSA on a fresh honest-key record. To keep Theorem 2 applicable, $\mathcal{B}$ answers signing queries via *lazy forwarding*: a table indexed by formatted message and per-run query count is filled on first access via $\mathcal{B}$'s challenger and never rolled back. Any query on the extracted message occurs only in a rewind, after event 0 fixes its $\tilde{c}$, matching that $\tilde{c}$ with probability at most 2^{-H_∞} for fresh min-entropy H_∞. Because a rewind may sign the extracted message, the reduction needs strong unforgeability. ML-DSA lacks a salt, ruling out Aerie's presampling, while lazy forwarding avoids the $e(Q_S + 1)$ loss over Q_S signing queries in Aerie's common-salt reduction [CDBT26, Theorem C.9].

Theorem 3 (Aggregate unforgeability)**.** *In the setting of Theorem 2 with $\mathcal{R}_{\text{FIPS}}$, let $\mathcal{A}$ in the chosen-key game of Definition 1 make Q proof-oracle and at most Q_S signing queries per run. Then a SUF-CMA adversary $\mathcal{B}$ against ML-DSA runs $\mathcal{A}$ at most $T_{\text{ext}} = 4(Q + R + 2)\,\text{Tree}^0_{\text{ck}}(sch)$ times in expectation. For the hedged signer,*

$$\text{Adv}^{\text{agg}}(\mathcal{A}) \le \text{Adv}^{\text{SUF}}_{\textit{ML-DSA}}(\mathcal{B}) + \varepsilon_{\text{ext}} + \mathbb{E}[Q_{\text{re}}]\, 2^{-H_\infty},$$

where $Q_{\text{re}} \le T_{\text{ext}} Q_S$ in expectation counts the rerun-only signing queries on the extracted message. Here H_∞ is the min-entropy of $\tilde{c}$ in a fresh signature, minimized over (sk, M'). For the deterministic signer, the bound adds $\text{Adv}^{\text{PRF}} + \text{Adv}^{\text{coll}}_{\mu}$, with H_∞ over uniform ρ''. Here Adv^{PRF} is the pseudorandom function (PRF) advantage on $\mu \mapsto \rho'' = \mathsf{H}(K \| 0^{32} \| \mu, 64)$ [Nat24, Algorithm 7] under key K, and $\text{Adv}^{\text{coll}}_{\mu}$ bounds a μ-collision.

Proof. The reduction. Given $\mathsf{pk}^\star$ from its challenger, $\mathcal{B}$ generates pp and runs $\mathcal{A}$ inside the extractor of Theorem 2. It answers the n-th query on M' in a run with $\Theta[(M', n)]$, querying its challenger on first use and never rolling Θ back.

Step 1: extraction applies. Lazy forwarding gives $(\mathsf{pk}^\star, \Theta)$ the law of an auxiliary environment independent of setup and proof oracles [DBK+26, Section 10.3], so Theorem 2 applies with Q proof-oracle queries. On a winning first run, take the first $i^\star$ in $\mathcal{L}$ under $\mathsf{pk}^\star$ with $M'_{i^\star} \notin \mathcal{Q}$. Except with ε_{ext}, extraction gives a $\mathcal{R}_{\text{FIPS}}$ witness at $i^\star$, and Lemma 1 yields a valid $sig^\star = \text{sigEncode}(\tilde{c}_{i^\star}, \mathbf{z}_{i^\star}, \mathbf{h}^\star)$ on $M'_{i^\star}$.

Step 2: the output is a strong forgery. Though a rewind may query $M'_{i^\star}$, the first run does not, and $sig^\star$ is fixed beforehand ($\tilde{c}_{i^\star}$ at event 0, the rest at round 1). Fresh queries

Table 6: Ablation of the GF(2) side for ML-DSA-44.

Variant	Prove (s)	Verify (ms)	B/sig
full aggregate	2.68	75	49.2
Flock after the PIOP	3.09	–	49.2
public $\mathbf{w}'_1$ bits	1.88	113	803.1

Note: $N = 32{,}768$, AVX2, 16C32T, uncached. GF(2) side alone: 1.32 s. Prover parity chain: 36.5 ms. Verifier bridge: 9 ms (matrices 5.7 ms).

on $M'_{i^\star}$ in rewinds sample signer randomness afterwards and hit $\tilde{c}_{i^\star}$ with probability at most 2^{-H_∞}, so all miss $sig^\star$ except with $\mathbb{E}[Q_{\mathrm{re}}]2^{-H_\infty}$ ($\mathbb{E}[Q_{\mathrm{re}}] \le T_{\mathrm{ext}}Q_S$).

Step 3: the deterministic signer. Every query on $M'_{i^\star}$ reads K via $\rho'' = \mathsf{H}(K\|0^{32}\|\mu_{i^\star}, 64)$. Replacing it by a random function adds $\mathrm{Adv}^{\mathrm{PRF}}$. Barring a μ-collision ($\mathrm{Adv}^{\mathrm{coll}}_{\mu}$), ρ'' is uniform and independent of $\tilde{c}_{i^\star}$, so Step 2 applies. □

For ML-DSA-44, a fresh $\tilde{c}$ has min-entropy about 253.9 bits (estimate) in the ROM, so each rerun-only query adds about 2^{-254}. Like Aerie, our Module-SIS and SUF-CMA bit estimates omit the rewind factor: at $N = 2^{15}$, $\mathrm{Tree}^0_{\mathrm{ck}} \le 2^{40.8}$ [DBK+26, Equation (197)] gives at most $2^{43}(Q + R + 2)$ expected runs. SHAKE256 serves two roles: Theorems 2 and 3 treat it as a concrete circuit (Lemma 6) separated from proof oracles by input framing, whereas ML-DSA's SUF-CMA proofs [KLS18, DKL+18, BBD+23, DFPS23] model the signer's H and ExpandA as random oracles.

7 Implementation and Evaluation

7.1 Implementation and setup

We implement in Rust the witness generator, the $\mathbb{F}_p$ PIOP, the LogUp-GKR lookup, the parity bridge, the matrix-free Keccak lincheck and the AVX2, AVX-512F and AVX-512 integer fused multiply-add (IFMA) kernels. We reuse the crates of Aerie [CDBT26] and Akita [DBK+26], Flock's core crate [BRW26] and a Spongefish transcript over BLAKE2b [ark26].

The GF(2) side. We express the sponge R1CS through Flock's lincheck interface. We adapt Flock's verifier in four ways to support our security proof. At sites before Ligerito's opening, it sets proof-of-work difficulties to account for the Johnson list size and rejects nonces exceeding $D_\iota + 7$ bits. The bridge's claim on b' enters the sponge opening, and each recursive Ligerito [NA25] level draws its out-of-domain points before the prover answers any.

Setup. All measurements ran on dual 32-core AMD EPYC 9355 nodes (Zen 5, up to 4.4 GHz, 256 GB RAM). Peak prover memory is at most 20.0 GiB on any tier (ML-DSA-87 at $N = 2^{15}$), fitting in 32 GB. We test 16 cores with simultaneous multithreading (SMT, 16C32T), 8 cores with SMT (8C16T) and one thread (1T). One binary dispatches across three instruction-set architecture (ISA) tiers: AVX2 without AVX-512 (with 256-bit VPCLMULQDQ and GFNI where present), AVX-512F without IFMA, and IFMA. We default to $\mathcal{R}_{\mathrm{FIPS}}$ on AVX2, required by RHEL 10's x86-64-v3 baseline [Red26]. Valve's October 2026 survey reports 95% support for AVX2 versus 24% for AVX-512F [Val26]. Tables give medians of five in-process runs after one warm-up unless noted.

Variation and accounting. The fastest run lies within 2.3% of the median at $N = 2^{15}$ and 4.8% at smaller sizes, so we treat differences under 3% at $N = 2^{15}$ as noise. Each record uses a distinct key and an empty context. Ratios use unrounded medians.

Table 7: Scaling with the batch size.

Set	N	B/sig	Compression	Prove (s)	Verify (ms)	Memory (GiB)
44	2^{10}	418.1	5.8×	0.35	24 / 23	1.9
	2^{13}	94.4	25.6×	0.93	38 / 34	4.5
	2^{15}	49.2	49.2×	2.68	75 / 57	15.8
65	2^{10}	439.6	7.5×	0.43	26 / 24	2.0
	2^{13}	111.2	29.8×	1.13	51 / 42	4.3
	2^{15}	65.4	50.6×	3.31	120 / 82	15.1
87	2^{10}	478.0	9.7×	0.51	31 / 27	2.2
	2^{13}	132.5	34.9×	1.43	67 / 49	5.7
	2^{15}	82.7	55.9×	4.13	177 / 110	19.8

Note: AVX2, 16C32T. Verify: uncached / cached. Memory: prover peak. Compression: sig. over B/sig.

7.2 Prover time

At $N = 32{,}768$ (Table 5), proving takes 2.68 to 4.13 s on 16C32T. ML-DSA-87 commits 1.84 times as many cells per signature as ML-DSA-44 (Table 3) yet takes 1.54 times as long. Its witness, commitments, lookup and stages 2 and 3 grow 1.5 to 2.2 times, while the Akita opening grows by 19%. Running beside the PIOP, Flock spans 90% to 100% of their joint wall time. Over AVX2, IFMA cuts proving time by 12.5% to 12.8% on 16C32T and 24% to 25% on 1T. AVX-512F stays within 2% of IFMA, and $\mathcal{R}_{\text{ex}}$ proves faster while saving at most 0.02 B/sig.

Ablation. Table 6 isolates the GF(2) side for ML-DSA-44. Publishing w1Encode($\mathbf{w}'_1$) directly lets the verifier compute every hash without a Flock proof, cutting proving time by 30% while making the aggregate 16 times larger. Verification rises to 113 ms, including 63 ms to hash those bits. Running Flock after the PIOP adds 0.41 s, and the GF(2) side alone takes 1.32 s. Committing the 268,800 AND bits in $\mathbb{F}_p$ would add 15.8× the 17,025.5 lattice cells (estimate).

7.3 Scaling, bytes and baselines

Table 7 measures scaling from $N = 2^{10}$ to 2^{15}. Quadrupling N from 2^{13} to 2^{15} multiplies proving time by 2.9. Fixed costs dominate at 2^{10}, where the sets differ by 14% in B/sig and 43% in proving time.

Aggregate bytes. In Table 8, the public hashes $\tilde{c}_i$ ($\lambda/4$ B/sig) form most of each aggregate. The proof π adds 17.2 to 18.7 B/sig. Flock and the GF(2) bridge dominate π, growing solely for ML-DSA-87's nine permutations.

Batches above 2^{15}. Batches exceeding 2^{15} signatures can be proved either as a single batch or as independent chunks of 2^{15}. A single proof requires only generating offline Akita schedules and Ligerito configurations beyond $d_{\text{sig}} = 15$, with peak memory growing linearly in N. Chunking keeps memory within 20.0 GiB and distributes proving across machines.

Native verification. Without cached keys, our verifier takes 2.4 to 2.5 times native AVX2 verification (Table 5 note) on 16C32T and 1.4 to 1.8 times on 1T. Caching keys on both sides shifts these ratios to 4.5 to 5.6 and 2.0 to 4.3. We omit Winterfell's STARK aggregator [KCLM22], which needs 87.6 s on 1T for 1,024 one-time signatures.

Aerie. On 16C32T, our category-5 ML-DSA-87 prover takes 2.5 times Aerie v0.2.0 [CDBT26] on Falcon-1024 with AVX2 and 2.1 times with AVX-512F, which changes Aerie's time by at most 3%. Comparing ML-DSA-44 with Falcon-512 gives ratios of 2.7 and 2.4. On AVX2, 16C32T verification takes 4.9 and 8.3 times Aerie's for ML-DSA-44 and -87, spent mainly expanding $\mathbf{A}_i$ and checking Flock.

Table 8: Composition of the aggregate at $N = 32{,}768$.

Component	**44**	**65**	**87**
hashes $\tilde{c}_i$	32.0	48.0	64.0
Flock proof	13.9	13.9	15.1
Akita opening	1.9	1.9	1.9
lookup	0.9	1.1	1.2
PIOP rounds and claims	0.2	0.2	0.2
commitments	0.2	0.2	0.2
other	0.1	0.1	0.1
total	49.2	65.4	82.7

Note: B/sig, rounded. Commitments: Akita, Flock. Other: bridge, nonces.

Prime-field aggregate. Across $N = 2^{10}$ to 2^{15} on 16C32T, Alloul26 [All26] takes 18 to 43 times our ML-DSA-87 proving time and up to 4.1 times our peak memory. Its aggregates take 1.8 to 2.4 times our B/sig. At 2^{15}, the time ratio is 47 on 8C16T and 52 on 1T, and AVX-512F saves Alloul26 at most 3.4%.

Verifier. At $N = 2^{15}$ on 16C32T, the verifier checks the GF(2) side in 19 to 21 ms beside the PIOP, decodes keys and computes $\mathsf{tr}_i, \mu_i, c_i, T_i$ in 26 to 43 ms, and runs ExpandA with w_A in 22 to 110 ms (cached keys save 13 to 67 ms).

7.4 Deployment

Our aggregate saves bandwidth rather than verification time, shrinking 32,768 signatures from 79.3 to 1.6 MB for ML-DSA-44 and from 151.6 to 2.7 MB for ML-DSA-87 (versus about 4 KB (estimate) for BLS with a participation bitfield). Without recursion, an aggregator collects the batch directly rather than merging sub-aggregates. In Ethereum, each 12 s slot splits into three 4 s phases for proposal, attestation and aggregation across roughly 32,000 validator votes [Eth26, DKKW25, CDBT26]. Proving $N = 32{,}768$ signatures takes 2.68 to 3.31 s on AVX2 (2.35 to 2.89 s with IFMA) for ML-DSA-44 and -65, fitting inside one 4 s aggregation window, while ML-DSA-87 takes 3.61 s with IFMA (4.13 s on AVX2) or 1.43 s per 2^{13}-signer subnet. Registered keys and a shared block message let the aggregate omit both and let verifiers cache expanded matrices. Layer-2 rollups [ZWW25], cross-chain bridges and certificate or authorization logs [All26] likewise batch thousands of signatures over multi-second or asynchronous intervals, cutting storage 49 to 56 times with 75 to 177 ms verification.

8 Related Work

Aggregating signatures. Lattice signatures [GPV08, DKL+18] lack native aggregation [BRL23, BT23], while half-aggregation [BT23], Fiat–Shamir-with-aborts aggregation [ZMLH25] and synchronized multi-signatures [FSZ22, FHSZ23] modify the scheme or stay linear in size. STARKs aggregate hash-based signatures [KCLM22] and custom-gate arguments aggregate Falcon [FHK+20, ZWW25], whereas XMSS variants [DKKW25] and CAPSS [FR26] design proof-friendly schemes. Quan aggregates Dilithium with a STARK [Qua22], and zkDilithium swaps SHAKE for Poseidon and drops the hint in 85 to 175 KB single-signature STARKs [PWFW23], whereas we aggregate unmodified FIPS 204 [Nat24] ML-DSA.

Aggregating Falcon and ML-DSA. Aardal et al. estimate 417 KB for 8,192 Falcon-512 signatures via recursive LaBRADOR [BS23] and public salts [AAB+24, Table 2]. Aerie [CDBT26] pairs a sum-check PIOP with Akita [DBK+26], and Orthus verifies LaZer's

Falcon-512 aggregates [LSS24] nine times faster [BBL+26]. Hekate proves up to 128 ML-DSA verifications in 1,341 to 1,604 KiB over a binary field at 100-bit security [Oum26]. Alloul26 [All26] aggregates ML-DSA-87 over a 64-bit prime field using WHIR [ACFY25] and GKR [GKR08] for SHAKE256. Against its Johnson-regime variant, we prove 43 times faster with 1.8 times smaller aggregates (Table 5).

Proofs of one ML-DSA signature. Anonymous credentials prove one hidden ML-DSA signature in 0.56 to 1.38 MB via two-field Ligero in the quantum ROM [SG26], VOLE-in-the-head [BB26] or Longfellow over $\mathbb{F}_q$ [AHKPKT26]. Omitting zero knowledge and amortizing across 2^{15} signatures yields 49.2 to 82.7 B/sig here.

Lattice arguments and commitments. Module-SIS succinct arguments include LaBRADOR [BS23], Greyhound [NS24], LatticeFold [BC25] and SALSAA [KLOT25], and Aerie's norm projection follows PikkuFold [CDBT26, Osa26]. Lattice polynomial commitments include Akita [DBK+26], Grand Danois [KK26] and binary-polynomial LaBinius [OS26]. Using one commitment for both sides would still need a bridge for the $\mathbb{F}_p$ PIOP to read bits as field elements.

Proving Keccak. Over binary-field towers [DP25], Binius64 proves 64-bit word circuits and commits every intermediate value [The26, BRW26], whereas Flock commits only inputs, outputs and AND outputs of binary R1CS [BRW26], opened with Ligerito [NA25]. GKR provers commit only the inputs, as in prime-field Keccacheck [KKA+25] and binary GKR over Boolean circuits [HLQ+25], with round counts that grow with circuit depth. On an M4 Pro, Keccacheck proves 16,384 permutations in 401 s including its outer proof [KKA+25, Table 1]. Run alone on 16C32T, our GF(2) side proves all 229,376 permutations of 2^{15} ML-DSA-44 signatures in 1.32 s (Table 6).

Sum-check and mixed-mode proofs. Like Spartan [Set20] and Jolt [AST24], our PIOP rests on sum-check [LFKN92] and GKR [GKR08, XZZ+19]. Whereas binary-field systems stay in characteristic two [DP25, BRW26] and prior linking methods (Section 1) pay per bit or share one field, our bridge links commitments across two characteristics.

9 Conclusion

We prove ML-DSA's commitment hash over GF(2) and lattice relation over $\mathbb{F}_p$, linked by a cross-field parity bridge, establishing full FIPS 204 validity without recursion. On 16 Zen 5 cores with AVX2, we aggregate 32,768 signatures of any set in at most 4.13 s (3.61 s with IFMA) at 49.2 to 82.7 B/sig.

References

[AAB+24] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. Aggregating Falcon signatures with LaBRADOR. In *Advances in Cryptology (CRYPTO)*, pages 71–106, 2024. doi:10.1007/978-3-031-68376-3_3.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-Solomon proximity testing with super-fast verification. In *Advances in Cryptology (EUROCRYPT)*, pages 214–243, 2025. doi:10.1007/978-3-031-91134-7_8.

[AFK23] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat–Shamir transformation of multi-round interactive proofs (extended version). *Journal of Cryptology*, 36(4), August 2023. URL: http://dx.doi.org/10.1007/s00145-023-09478-y, doi:10.1007/s00145-023-09478-y.

[AGM18] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *Advances in Cryptology (CRYPTO)*, pages 643–673, 2018. doi:10.1007/978-3-319-96878-0_22.

[AHKPKT26] Calvin Abou Haidar, Ryo Kikuchi, Octavio Perez-Kempner, and Marcel Tiepelt. Nowhere man: Device binding for post-quantum anonymous credentials. Cryptology ePrint Archive, Paper 2026/2472, 2026. URL: https://eprint.iacr.org/2026/2472.

[All26] Yacine Alloul. Archive certificates for ML-DSA-87 authorizations from an exact bounded relation. Cryptology ePrint Archive, Paper 2026/2739, 2026. URL: https://eprint.iacr.org/2026/2739.

[ark26] arkworks-rs. Spongefish: Fiat–Shamir for the masses. https://github.com/arkworks-rs/spongefish, 2026.

[AST24] Arasu Arun, Srinath Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. In *Advances in Cryptology (EUROCRYPT)*, pages 3–33, 2024. doi:10.1007/978-3-031-58751-1_1.

[BB26] Carsten Baum and Marvin Beckmann. edaBits in the exponent (edaBitE) and issuer-hiding blind signatures from ML-DSA and VOLE-in-the-head. Cryptology ePrint Archive, Paper 2026/2473, 2026. URL: https://eprint.iacr.org/2026/2473.

[BBD+23] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and mechanizing the security proof of Fiat-Shamir with aborts and Dilithium. In *Advances in Cryptology (CRYPTO)*, pages 358–389, 2023. doi:10.1007/978-3-031-38554-4_12.

[BBL+26] Madalina Bolboceanu, Jonathan Bootle, Vadim Lyubashevsky, Antonio Merino-Gallardo, and Gregor Seiler. Orthus: Practical sublinear batch-verification of lattice relations from standard assumptions. In *Advances in Cryptology (CRYPTO)*, pages 100–132, 2026. URL: http://dx.doi.org/10.1007/978-3-032-35377-1_4, doi:10.1007/978-3-032-35377-1_4.

[BBMH+21] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoît Razet, and Peter Scholl. Appenzeller to Brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and $\mathbb{Z}_{2^k}$. In *ACM Conference on Computer and Communications Security*, pages 192–211, 2021. doi:10.1145/3460120.3484812.

[BC25] Dan Boneh and Binyi Chen. LatticeFold: A lattice-based folding scheme and its applications to succinct proof systems. In *Advances in Cryptology (ASIACRYPT)*, pages 330–362, 2025. URL: http://dx.doi.org/10.1007/978-981-95-5099-9_11, doi:10.1007/978-981-95-5099-9_11.

[BCGM25] Sarah Bordage, Alessandro Chiesa, Ziyi Guan, and Ignacio Manzur. All polynomial generators preserve distance with mutual correlated agreement. Cryptology ePrint Archive, Paper 2025/2051, 2025. URL: https://eprint.iacr.org/2025/2051.

[BGK+23] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michal Zajac. Fiat-Shamir security of FRI and related SNARKs. In *Advances in Cryptology (ASIACRYPT)*, pages 3–40, 2023. doi:10.1007/978-981-99-8724-5_1.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology (EUROCRYPT)*, pages 416–432, 2003. URL: https://doi.org/10.1007/3-540-39200-9_26.

[BHH+19] Michael Backes, Lucjan Hanzlik, Amir Herzberg, Aniket Kate, and Ivan Pryvalov. Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In *Public-Key Cryptography (PKC)*, pages 286–313, 2019. doi:10.1007/978-3-030-17253-4_10.

[BRL23] Katharina Boudgoust and Adeline Roux-Langlois. Overfull: Too large aggregate signatures based on lattices. *The Computer Journal*, 67(2):719–727, 2023. doi:10.1093/comjnl/bxad013.

[BRW26] Benedikt Bünz, Ron D. Rothblum, and William Wang. Flock: Fast proving for batch Boolean computations. Cryptology ePrint Archive, Paper 2026/1329, 2026. URL: https://arxiv.org/abs/2607.27491.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from Module-SIS. In *Advances in Cryptology (CRYPTO)*, pages 518–548, 2023. doi:10.1007/978-3-031-38554-4_17.

[BSCI+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for Reed-Solomon codes. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 900–909, 2020. doi:10.1109/FOCS46700.2020.00088.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography (TCC)*, pages 31–60, 2016. doi:10.1007/978-3-662-53644-5_2.

[BT23] Katharina Boudgoust and Akira Takahashi. Sequential half-aggregation of lattice-based signatures. In *Computer Security (ESORICS)*, pages 270–289, 2023. doi:10.1007/978-3-031-50594-2_14.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: From practice to theory. In *Annual ACM Symposium on Theory of Computing*, 2019. doi:10.1145/3313276.3316380.

[CDBT26] Alberto Centelles, Quang Dao, Mohammadtaghi Badakhshan, and Justin Thaler. Aerie: Efficient aggregation of Falcon signatures. Manuscript, 2026.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anais Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In *ACM Conference on Computer and Communications Security*, pages 2075–2092, 2019. doi:10.1145/3319535.3339820.

[CGM16] Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *Advances in Cryptology (CRYPTO)*, pages 499–530, 2016. doi:10.1007/978-3-662-53015-3_18.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In *Theory of Cryptography (TCC)*, pages 1–29, 2019. doi:10.1007/978-3-030-36033-7_1.

[DBK+26] Quang Dao, Omid Bodaghi, Amirhossein Khajehpour, Giuseppe Vitto, Mohammadtaghi Badakhshan, Markos Georghiades, Fengrun Liu, Jiapeng Zhang, and Justin Thaler. Akita: A high-performance lattice-based polynomial commitment scheme. Cryptology ePrint Archive, Paper 2026/1983, 2026. URL: https://eprint.iacr.org/2026/1983.

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of Fiat–Shamir with aborts. In *Advances in Cryptology (CRYPTO)*, pages 327–357, 2023. doi:10.1007/978-3-031-38554-4_11.

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum Ethereum. *IACR Communications in Cryptology*, 2(1), 2025.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018. URL: https://tches.iacr.org/index.php/TCHES/article/view/839, doi:10.13154/tches.v2018.i1.238-268.

[DKT26] Quang Dao, Scott Duke Kominers, and Justin Thaler. Reed–Solomon codes beyond Johnson: Efficient decoding and smaller cryptographic proofs. Cryptology ePrint Archive, Paper 2026/2056, 2026. URL: https://eprint.iacr.org/2026/2056.

[DKTZ26] Quang Dao, Scott Duke Kominers, Justin Thaler, and Kai Zhe Zheng. Counterexamples to beyond-Johnson proximity gaps over binary fields. Cryptology ePrint Archive, Paper 2026/2424, 2026. URL: https://eprint.iacr.org/2026/2424.

[DP25] Benjamin E. Diamond and Jim Posen. Succinct arguments over towers of binary fields. In *Advances in Cryptology (EUROCRYPT)*, pages 93–122, 2025. doi:10.1007/978-3-031-91134-7_4.

[EGK+20] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In *Advances in Cryptology (CRYPTO)*, pages 823–852, 2020. doi:10.1007/978-3-030-56880-1_29.

[Eth26] Ethereum Foundation. Phase 0 – the Beacon Chain. Ethereum Consensus Specs, 2026. Accessed May 24, 2026. https://ethereum.github.io/consensus-specs/specs/phase0/beacon-chain/.

[FHK+20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. NIST Post-Quantum Cryptography Project, Round 3 submission, 2020. URL: https://falcon-sign.info/falcon.pdf.

[FHSZ23] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In *ACM Conference on Computer and Communications Security*, pages 386–400, 2023. doi:10.1145/3576915.3623219.

[FR26] Thibauld Feneuil and Matthieu Rivain. CAPSS: A framework for SNARK-friendly post-quantum signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(4):731–758, 2026. doi:10.46586/tches.v2026.i4.731-758.

[FSZ22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In *ACM Conference on Computer and Communications Security*, pages 1109–1123, 2022. doi:10.1145/3548606.3560655.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Annual ACM Symposium on Theory of Computing*, pages 113–122, 2008. doi:10.1145/1374376.1374396.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Annual ACM Symposium on Theory of Computing*, pages 197–206, 2008. doi:10.1145/1374376.1374407.

[Hab22] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Paper 2022/1530, 2022. URL: https://eprint.iacr.org/2022/1530.

[Hab25] Ulrich Haböck. A note on mutual correlated agreement for Reed–Solomon codes. Cryptology ePrint Archive, Paper 2025/2110, 2025. URL: https://eprint.iacr.org/2025/2110.

[HLQ+25] Yuncong Hu, Chongrong Li, Zhi Qiu, Tiancheng Xie, Yue Ying, Jiaheng Zhang, and Zhenfei Zhang. GKR for Boolean circuits with sub-linear RAM operations. Cryptology ePrint Archive, Paper 2025/717, 2025. URL: https://eprint.iacr.org/2025/717.

[KCLM22] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. Aggregating and thresholdizing hash-based signatures using STARKs. In *ACM Asia Conference on Computer and Communications Security*, 2022. doi:10.1145/3488932.3524128.

[KCP16] John Kelsey, Shu-jen Chang, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash, and ParallelHash. Special Publication 800-185, NIST, 2016. doi:10.6028/NIST.SP.800-185.

[KK26] Anders Kallesøe and Hamidreza Khoshakhlagh. Grand Danois: Succinct multilinear polynomial commitments over lattices. Cryptology ePrint Archive, Paper 2026/1196, 2026. URL: https://eprint.iacr.org/2026/1196.

[KKA+25] Maciej Kostrzewa, Marek Klein, Alex Adkins, Grzegorz Swirski, and Wojciech Zmuda. Keccacheck: Towards a SNARK-friendly Keccak. Cryptology ePrint Archive, Paper 2025/1764, 2025. URL: https://eprint.iacr.org/2025/1764.

[KLOT25] Shuto Kuriyama, Russell W. F. Lai, Michał Osadnik, and Lorenzo Tucci. SALSAA: Sumcheck-aided lattice-based succinct arguments and applications. Cryptology ePrint Archive, Paper 2025/2124, 2025. URL: https://eprint.iacr.org/2025/2124.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In *Advances in Cryptology (EUROCRYPT)*, pages 552–586, 2018. doi:10.1007/978-3-319-78372-7_18.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.

[LSS24] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The LaZer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In *ACM Conference on Computer and Communications Security*, pages 3125–3137, 2024. doi:10.1145/3658644.3690330.

[NA25] Andrija Novakovic and Guillermo Angeris. Ligerito: A small and concretely fast polynomial commitment scheme. Cryptology ePrint Archive, Paper 2025/1187, 2025. URL: https://eprint.iacr.org/2025/1187.

[Nat15] National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions. FIPS 202, NIST, 2015. doi:10.6028/NIST.FIPS.202.

[Nat24] National Institute of Standards and Technology. Module-lattice-based digital signature standard. FIPS 204, NIST, August 2024. doi:10.6028/NIST.FIPS.204.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In *Advances in Cryptology (CRYPTO)*, pages 243–275, 2024. doi:10.1007/978-3-031-68403-6_8.

[OS26] Michał Osadnik and Gregor Seiler. LaBinius – fast lattice-based binary polynomial commitment scheme. Cryptology ePrint Archive, Paper 2026/2103, 2026. URL: https://eprint.iacr.org/2026/2103.

[Osa26] Michał Osadnik. PikkuFold: Efficient folding in a few kilobytes. Cryptology ePrint Archive, Paper 2026/1809, 2026. URL: https://eprint.iacr.org/2026/1809.

[Oum26] Oumuamua Labs. Hekate, version 0.38.0. https://github.com/oumuamua-labs/hekate/releases/tag/v0.38.0, 2026.

[PH23] Shahar Papini and Ulrich Haböck. Improving logarithmic derivative lookups using GKR. Cryptology ePrint Archive, Paper 2023/1284, 2023. URL: https://eprint.iacr.org/2023/1284.

[PWFW23] Guru-Vamsi Policharla, Bas Westerbaan, Armando Faz-Hernández, and Christopher A. Wood. Post-quantum Privacy Pass via post-quantum anonymous credentials. Cryptology ePrint Archive, Paper 2023/414, 2023. URL: https://eprint.iacr.org/2023/414.

[Qua22] Yunjia Quan. Improving Bitcoin's post-quantum transaction efficiency with a novel lattice-based aggregate signature scheme based on CRYSTALS-Dilithium and a STARK protocol. *IEEE Access*, 10:132472–132482, 2022. doi:10.1109/ACCESS.2022.3227394.

[Red26] Red Hat. Deprecation of x86-64-v1 and x86-64-v2 x86_64 Microarchitecture CPUs in RHEL9 / x86-64-v3 Is Required by RHEL10. https://access.redhat.com/solutions/7066628, 2026. Accessed: 2026-11-09.

[RW19] Dragos Rotaru and Tim Wood. MArBled circuits: Mixing arithmetic and Boolean circuits with active security. In *Progress in Cryptology (INDOCRYPT)*, pages 227–249, 2019. doi:10.1007/978-3-030-35423-7_12.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Advances in Cryptology (CRYPTO)*, pages 704–737, 2020. doi:10.1007/978-3-030-56877-1_25.

[SG26] Chad Sharp and Paul Grubbs. Faster and smaller anonymous credentials from standardized signatures. Cryptology ePrint Archive, Paper 2026/2615, 2026. URL: https://eprint.iacr.org/2026/2615.

[STW24] Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In *Advances in Cryptology (EUROCRYPT)*, pages 180–209, 2024. doi:10.1007/978-3-031-58751-1_7.

[STZ26] Srinath T. V. Setty, Justin Thaler, and Michael Zhu. Twist and Shout: Faster memory checking arguments via one-hot addressing and increments. In *Advances in Cryptology (CRYPTO)*, pages 267–296, 2026. doi:10.1007/978-3-032-35424-2_10.

[Tha25] Justin Thaler. Sum-check is all you need: An opinionated survey on fast provers in SNARK design. Cryptology ePrint Archive, Paper 2025/2041, 2025. URL: https://eprint.iacr.org/2025/2041.

[The26] The Binius Authors. Binius64: Binary proofs, blazing fast. https://github.com/binius-zk/binius64, 2026.

[Val26] Valve Corporation. Steam Hardware & Software Survey: October 2026. https://store.steampowered.com/hwsurvey/, 2026. Accessed: 2026-11-09.

[WYX+21] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In *USENIX Security Symposium*, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/weng.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *Advances in Cryptology (CRYPTO)*, pages 733–764, 2019. doi:10.1007/978-3-030-26954-8_24.

[ZMLH25] Mingmei Zheng, Masahiro Mambo, Junzuo Lai, and Xinyi Huang. Towards a lattice-based non-interactive aggregate signature scheme following the Fiat-Shamir with aborts paradigm. In *Information Security Practice and Experience (ISPEC)*, 2025. doi:10.1007/978-981-95-9284-5_11.

[ZWW25] Tianyu Zhaolu, Zhiguo Wan, and Huaqun Wang. Post-quantum rollup: Falcon signature aggregation based on SNARG with enhanced gates. *IEEE Transactions on Information Forensics and Security*, 20:2899–2914, 2025. doi:10.1109/TIFS.2025.3544490.

A Proof of Lemma 1

Proof. We complete the sketch of Lemma 1. Decompose [Nat24, Algorithm 36] returns $(\bar{r}_1, \bar{r}_0)$ unless $\bar{r}_1 = m$, where $\bar{r}_0 \leq 0$ and it returns $(0, \bar{r}_0 - 1)$. Since $0 \equiv m \pmod{m}$ and $\bar{r}_0 - 1 < 0$, UseHint [Nat24, Algorithm 40] satisfies (5).

(a) Completeness. Given w' and hint bit h, choose (θ, L) with $r_0 = \bar{r}_0$, so $\theta = \mathrm{sgn}(\bar{r}_0)$. Set $\delta = h\theta$, $w_1' = \mathrm{UseHint}(h, r) = (\bar{r}_1 + \delta) \bmod m$ and $e = (\bar{r}_1 + \delta - w_1')/m$. As $\bar{r}_1 = m$ forces $\theta = -1$, $\bar{r}_1 + \delta \in [-1, m]$. Hence $e = -1$ implies $\bar{r}_1 + \delta = -1$, so $\delta = -1$ and $w_1' = m - 1$, while $e = 1$ implies $\bar{r}_1 + \delta = m$ and $w_1' = 0$. The constraints of (3) thus hold, with $\delta^2 = h$ and $u_h = \omega - \sum \delta^2 \geq 0$. As $\tilde{r}_1 = \bar{r}_1$, we get $2\gamma_2 w_1' + u = r - eq \equiv w'$, and $|u| \leq B_u < q/2$ gives $u = \mathrm{ctr}_q(w' - 2\gamma_2 w_1')$. Since $T \equiv c\,\mathbf{t}_1 2^d \pmod q$, the quotient $\mathbf{v}$ of (2) exists, and $|u| \leq B_u$ gives $\|\mathbf{v}\|_\infty \leq B_v$.

(b) Soundness. Now $r \equiv x = 2\gamma_2 \tilde{r}_1 + r_0$. For $e = 0$, $\tilde{r}_1 = w_1' - \delta \in [-1, m]$. For $e = 1$, $w_1' = 0$ and $\tilde{r}_1 = m - \delta$. For $e = -1$, $\delta = -1$, $w_1' = m - 1$ and $\tilde{r}_1 = 0$. So $x \in (-q, 2q)$, and $r = x - \zeta q = 2\gamma_2(\tilde{r}_1 - \zeta m) + (r_0 - \zeta)$ for some $\zeta \in \{0, \pm 1\}$. If $r_0 - \zeta \in (-\gamma_2, \gamma_2]$ and $\mathrm{sgn}(r_0 - \zeta) = \theta$, this is the decomposition of r. Then $\bar{r}_1 \equiv \tilde{r}_1 \equiv w_1' - \delta \pmod m$ and $\delta \in \{0, \mathrm{sgn}(\bar{r}_0)\}$, so $w_1' = \mathrm{UseHint}(\delta^2, r)$ by (5). If $\delta = 0$, then $w_1' = \mathrm{HighBits}(r)$. Hence $h^\star \leq \delta^2$ and $\mathrm{UseHint}(h^\star, r) = w_1'$. Otherwise (ζ, r_0) is $(-1, 0)$, $(-1, \gamma_2)$, $(1, 1)$ or $(1, 1 - \gamma_2)$. The first needs $x < 0$ with $r_0 = 0$, hence $\tilde{r}_1 = -1$ and $\theta = \delta = 1$, whereas $r_0 = 0$ means $\theta = -1$. In the other three, $x < 0$ with $r_0 = \gamma_2$ forces $\tilde{r}_1 = -1$, and $x \geq q$ forces $\tilde{r}_1 = m$ with $r_0 > 0$ or $\tilde{r}_1 = m + 1$. The cases above allow these only when $w_1' = 0$, giving $r \in \{q - \gamma_2, 0, \gamma_2\}$, $\mathrm{HighBits}(r) = 0 = w_1'$, $h^\star = 0$ and $\mathrm{UseHint}(0, r) = w_1'$.

Algorithm 1 accepts $\mathrm{sigEncode}(\tilde{c}, \mathbf{z}, \mathbf{h}^\star)$: $\mathbf{h}^\star$ has weight at most $\sum \delta^2 \leq \omega$, line 5 computes $\mathbf{w}'$ by (2), line 6 returns $\mathbf{w}_1'$, and line 7 accepts by (4) and $\|\mathbf{z}\|_\infty \leq B_z < \gamma_1 - \beta$. □

B Proof of Knowledge Soundness

Plan. We prove Theorem 2 via Akita's checkpoint extraction [DBK+26, Theorem 10.17] with two checkpoints: one at sd testing $\mathcal{R}_{\mathrm{FIPS}}$ on the round-1 tables, and one at $X = (X_1, X_2)$ testing the PIOP identities on all committed tables. After setting the oracle model, we prove Lemmas 5 to 7. Step 1 expands digests in challenge inputs, and Steps 2 to 4 check Lemma 5.

Model. The oracles $\mathcal{H}_P$ (the PIOP sponge), $\mathcal{H}_F$ (the Flock [BRW26] challengers), $\mathcal{H}_A$ (Akita's streams), $\mathcal{H}_X$ (the seed-expansion oracle), $\mathcal{H}_M$ (Ligerito's [NA25] Merkle hash), Akita's setup oracle and the hashed entries use disjoint input formats. Each squeeze of $\mathcal{H}_P$, $\mathcal{H}_F$ or $\mathcal{H}_A$ answers an injective encoding of its stream's history [DBK+26, Section 10.1]. A *stage* is a verifier-read squeeze, or consecutive squeezes with no intervening prover message. The extractor reprograms $\mathcal{H}_P$, $\mathcal{H}_F$ and $\mathcal{H}_A$, while the rest are auxiliary [DBK+26, Definition 10.15].

Akita's extraction. Akita classifies each stage as a fold (extraction) stage, a checkpoint or a plain stage [DBK+26, Definition 10.16]. A checkpoint reads a candidate w from a *stage output* (from accepting runs), tests it and passes valid outputs backward, while two stage outputs on one prefix with distinct candidates give a binding witness. A *checkpoint segment* runs to the next fold stage or checkpoint and requires, per candidate w, a predicate $\mathrm{Fail}(\xi; w)$ on prefixes ξ and bad sets $\mathrm{Bad}_j(\xi; w)$ of mass at most ε_j with: (a) Fail = 1 at the checkpoint when the test fails, (b) Fail = 1 persisting to the next stage outside Bad_j for any prover message, and (c) Fail = 0 at the segment's end on every stage output. Rerunning a failed checkpoint with a fresh table below its prefix fails on an accepting run with probability at most $\Pr[\mathrm{aux} \notin \mathrm{Good}] + \mathbb{E}[\sum_a \varepsilon(a)]$, where (a) to (c) hold on Good and each queried input a adds $\varepsilon(a)$.

Lemma 5 (Projected challenge inputs, full statement). Let stage r with prefix ξ_r query input $\mathrm{pr}_r(\xi_r)$, and suppose:

(i) for every checkpoint or fold stage ck and $j \geq \mathsf{ck}$, the input at j determines that at ck;

(ii) a fold stage's input holds all that Akita's combiner Comb_r [DBK+26, Proposition 10.44] and every check and checkpoint from stage r on read, except plain stages

of earlier checkpoint segments, and Comb_r returns the prefix of its reference child;
(iii) at a checkpoint, the test, Fail and the binding map depend on ξ_{ck} only through $\mathrm{pr}_{\mathsf{ck}}(\xi_{\mathsf{ck}})$;
(iv) each $\mathrm{Bad}_j(\xi; w)$ depends on ξ only via $\mathrm{pr}_j(\xi)$; and
(v) each fold stage allows resampling [DBK+26, Lemma 10.14].

Then Akita's Theorem 10.17 holds when a failed checkpoint is rerun with a fresh table on the inputs below its own, and each queried stage-r input weighs ε_r.

Proof. Akita's proof [DBK+26, Appendix E.2] uses transcript prefixes in three places: the rerun rule, Claim 1 and Claim 2. We replace each prefix by its challenge input. For input a at ck, let $\mathrm{Sub}(a)$ be the later inputs mapping to a.

1. *Partition.* By (i) the sets $\mathrm{Sub}(a)$ partition later inputs.
2. *Claim 2.* A prover querying no input in $\mathrm{Sub}(a)$ runs unchanged when $\mathrm{Sub}(a)$ is redrawn, so a is selected after the redraw iff before. The partition turns Akita's sum into $\sum_a N_{a,j} = N_j \le Q_j + V_j$ across Q_j prover and V_j verifier queries ($N_{a,j}$ in $\mathrm{Sub}(a)$, $\sum_j V_j \le R$).
3. *Claim 1.* Fix a, a candidate w with $\mathrm{Fail}(a; w) = 1$ and the table outside $\mathrm{Sub}(a)$. A rerun returning a stage output with candidate w and checkpoint input a starts with $\mathrm{Fail} = 1$ by (iii) and ends with $\mathrm{Fail} = 0$ by (c). By (b), some stage j draws an answer in $\mathrm{Bad}_j(\xi_j; w)$ at an input in $\mathrm{Sub}(a)$. Under lazy sampling, (iv) fixes this bad set before the answer is drawn, so each query hits it with probability at most ε_j.
4. *Fold stages.* Akita's game redraws one entry and keeps the others. The children share the input and differ only in stages unread from r on. By (ii) their combined opening is valid, and Comb_r attaches it to the reference child's passing prefix, while nested sampling sets give (v).
5. *Counting.* The telescoped recurrence charges each queried input once, at most $Q + R$ inputs, hence $(Q + R)2^{-128}$ after the work. The cost recurrence $C_{\mathsf{ck}} \le 2C_{\mathsf{ck}+1} + (Q + 1)$ gives at most $(Q + 2)\hat{K}$ runs of a prover with Q queries, for Akita's tree size $\hat{K}$. □

Proof of Lemma 6. We induct over the Keccak rounds. *Columns and rows.* The committed columns are the input bits ($256kn_w$ of w1Encode and 512 of μ), the constant cst, the 2λ output bits out, and the χ bits $a_{n,\varrho}$ of round $\varrho < 24$ of permutation $n < n_{\mathrm{perm}}$. The last round of the last permutation keeps only the $2\lambda/64$ output lanes. Every other value is GF(2)-linear in these columns. Let $S_{n,\varrho}$ be the state entering round ϱ, $B_{n,\varrho}$ the state after its linear steps, and $S^+_{n,\varrho} = B_{n,\varrho} + a_{n,\varrho} + \mathrm{RC}_\varrho\mathsf{cst}$ the state leaving it, with round constant RC_ϱ [Nat15]. The absorbing map XORs block n of the padded message $\mu\|\mathrm{w1Encode}$ into the first 1,088 state bits, with nonzero padding bits equal to cst. The R1CS rows are $\mathsf{cst} \cdot \mathsf{cst} = \mathsf{cst}$, $b \cdot \mathsf{cst} = b$ per input bit, $S^+_{n_{\mathrm{perm}}-1,23} \cdot \mathsf{cst} = \mathsf{out}$ per output bit, $0 \cdot 0 = \mathsf{e}$ per unused column e, and the χ rows $(B_{n,\varrho}[x{+}1] + \mathsf{cst}) \cdot B_{n,\varrho}[x{+}2] = a_{n,\varrho}[x]$ on lanes x mod 5 of each row.

Induction. Let $\hat{S}, \hat{B}, \hat{a}$ be the honest values of SHAKE256 on the message read from the input columns. Suppose $\mathsf{cst} = 1$ and every row constraint holds. We show $S_{n,\varrho} = \hat{S}_{n,\varrho}$ by induction over (n, ϱ) in lexicographic order, starting from the first absorbed block. If $S_{n,\varrho} = \hat{S}_{n,\varrho}$, then $B_{n,\varrho} = \hat{B}_{n,\varrho}$, every χ row forces $a_{n,\varrho} = \hat{a}_{n,\varrho}$, and $S^+_{n,\varrho} = \hat{S}^+_{n,\varrho}$ on committed lanes. Before the last round every lane is committed, keeping the next state honest, and the output rows give $\mathsf{out} = \hat{S}^+_{n_{\mathrm{perm}}-1,23}$ on the output bits.

Uniqueness. Every committed column is an input, the constant, a column constrained to zero or a value fixed by the induction. The zero vector satisfies every row, so Flock's lincheck and the tie point enforce $\mathsf{cst} = 1$. □

Proof of Lemma 7. We prove the items in turn. *(i) First challenge input.* A listed codeword C' agrees with the committed word on more than $\rho_0 n_0$ columns, for rate ρ_0 and length n_0 at level 0. So it is $\mathbb{F}_{2^{128}}$-valued and decodes to a Boolean assignment. Until the level-0 folds end, C' fails iff this assignment violates an R1CS row, $\mathsf{cst} = 1$ or

a pending claim. Afterwards, if every C' failed, every listed codeword fails the running claim [BRW26, Appendix C.3].

(ii.a) Bad sets before the folds. On a ν-variable sponge table, 6 coordinates are skipped and 7 are fixed, whose 128 equality weights are GF(2)-independent. A nonzero constraint residual of a listed codeword thus survives the $\nu - 13$ drawn coordinates with probability at most $(\nu - 13)/2^{128}$. A drawn coordinate equal to one adds $(\nu - 13)/2^{128}$, giving mass $(\nu - 13)(L_{\mathrm{J}} + 1)/2^{128}$, with $D = 10$ at $N = 2^{15}$ and covered by Flock's schedule for $\nu \leq 1{,}313$. Violating $\mathsf{cst} = 1$ leaves a nonzero multilinear discrepancy, kept by the ν_{s} outer zero-check rounds outside mass $3L_{\mathrm{J}}/2^{128}$ ($D = 7$). The remaining sum-check, lincheck and skip sites have degree at most 127 ($D \leq 13$), and ring switching reduces a false claim to degree 7 [BRW26, Appendix B.2]. Each mass carries the factor L_{J}, except the multipoint rounds ($2/2^{128}$, $D = 1$) and batch (degree 384, $D = 9$).

(ii.b) Bad sets in the folds. At level $h < n_{\mathrm{lev}}$ of length n_h and rate ρ_h^-, decoding 0.02 above the Johnson threshold gives fewer than $8n_h t_h^3/(3\rho_h^-)$ exceptional fold challenges per row pair [DKT26, Theorem 4.4], below $2^{37.1}$ for $N \leq 2^{15}$. At the threshold itself, binary additive domains inside a proper subfield of the challenge field admit quadratically many exceptional challenges at rate 1/16 [DKTZ26, Theorem 4.1]. Across a round's row pairs, each folded word's list consists of the folded listed codewords [BRW26, Lemma 9], [ACFY25, Lemma 4.13], except with probability at most $2^{-213.9}$ at level 0 and $2^{-216.9}$ deeper for $N \leq 2^{15}$. For $h < n_{\mathrm{lev}} - 1$, both out-of-domain points of level $h+1$ are drawn in one verifier message before the prover replies. Two listed codewords agree at both with probability at most $\binom{L_{\mathrm{J},h+1}}{2}(\nu_{h+1}/2^{128})^2 \leq 2^{-229.1}$ for $N \leq 2^{15}$, leaving at most one passing codeword. Level h, of rate ρ_h, has relative decoding radius $1 - \sqrt{\rho_h} - 0.02$, using 244 queries at level 0 down to 23 at level 6 for $N = 2^{15}$. Queries miss a codeword beyond this radius with probability at most $2^{-112.1}$ ($D = 16$) up to $N = 2^{16}$, and a sound configuration keeps these fold, out-of-domain and query terms within 2^{-128} after work.

(iii) and (iv). The prover sends the final level in clear. Each site hashes its prefix and nonce once [DBK+26, Proposition 10.3]. □

Site counts. For any N with $|\mathcal{F}| \leq p/2$, the difficulty rule (1), Lemmas 4 and 7 and Corollary 1 and the sound Akita schedule keep every site within 2^{-128}. For ML-DSA-44 at $N = 2^{15}$ ($\mathcal{R}_{\mathrm{FIPS}}$, rate 1/2), a transcript has $R = R_f + R_{\mathrm{coord}} + R_F = 949 + 920 + 244$ sites. Here R_f comprises 457 PIOP sites, 31 in sd, the tie point and parity chain, and 461 in Akita's opening. Before work, $\log_2 \varepsilon_\iota$ (rounded up) is -138.4 at sd and -203.4 at the lookup challenges (no work), -128 to -122.1 at the remaining PIOP sites ($D_\iota \leq 6$), -121.8 to -118.6 in the tie point and parity chain with $L_{\mathrm{J}} = 35.36$ ($D_\iota \leq 10$), and -128 to -116 in the Akita opening ($D_\iota \leq 12$). After work, the heaviest PIOP site in any set is a batching vector at $2^{-128.19}$. Each sparse fold coordinate across Akita's 13 fold draws is one site [DBK+26, Definition 10.16, Theorem 10.46] of weight at most $2^{-128.08}$ without work. Drawing one coordinate per source block and claim gives R_{coord}: 690 root, 225 across five recursive folds and 5 terminal. Flock attaches a work nonce to each squeeze, making $R_F = 5\nu + 11n_{\mathrm{lev}} - 3$ sites for $\nu = 34$ sponge variables and $n_{\mathrm{lev}} = \nu/2 - 10$ Ligerito levels, weighing $2^{-235.8}$ to $2^{-112.1}$ before work ($D_\iota \leq 16$). ML-DSA-65 and -87 have 451 PIOP sites, $R_f = 943$, $R_F = 244$, and $R_{\mathrm{coord}} = 980$ and 928.

Step 1: an expanded protocol. Let P make Q queries and set $\bar{Q} = Q + R$. Three hops turn P into a prover for Π whose challenge inputs list the data they hash, running P once on the same statement with at most $\bar{Q}$ challenge queries and preserving acceptance except with $\varepsilon_{\mathrm{hash}} - \varepsilon_{\mathrm{seed}}$.

1. Send every hashed entry, including the event-6 image digest, in full [DBK+26, Lemma 10.47].
2. Replace every Flock challenger digest and branch digest in an input by its preimage, failing only if two answers collide, a digest is queried before its answer, or a branch digest starts with the label absorbed at the fork.

3. Replace each Merkle root by its committed word and each path by a read [BSCS16], failing only on an $\mathcal{H}_M$-collision or a pre-answer query.

In Π, a Flock input encodes the history through event 4, the event-6 images and the messages of the joint GF(2) argument. A stage-branch input encodes the history through event 6, its label and the branch's messages. Stage 3 and Akita's stream encode both branches while omitting Flock messages. Since event 2 sends the Flock commitment C, its Johnson list $\mathcal{J}(C)$ [BRW26, Equation (10)] is fixed at every later input. The hops cost $\varepsilon_{\text{hash}}$ of Section 6.

Step 2: checkpoints, tests and order. Π starts with events 0 to 2 and checkpoint ck_1 at sd, followed by the plain stages of Ψ, event 4, the Flock challenger digests, event 6 and the joint GF(2) argument: tie point, chain point, chain rounds, bit value and Flock's sites. Next come checkpoint ck_2 at X, the plain stages of both branches, stage 3, Akita's evaluation batch and Akita's opening suffix [DBK+26, Proposition 10.44], whose fold coordinates are the fold stages. Every stage output before the root fold pairs a passing prefix with a root weak opening Ω (satisfying $\mathcal{R}_{\text{PIOP}}$ before X).

- At ck_1 the candidate is the round-1 part Ω_1, which decodes to the tables f_1 of every record. The test is $\mathcal{R}_{\text{FIPS}}(f_1)$ on every live record.
- At ck_2 the candidate is Akita's source decommitment $\text{Src}(\Omega)$, which decodes to all tables f. The test $\mathcal{R}_{\text{PIOP}}(f; a_X)$ is the conjunction of the identities proved after X: lookup membership and leaves, digit reconstructions, Booleanity, the constraints of (3), $\Delta = 0$ on the hypercube, the projection identity (8) and the level identities (10) with the images Y_σ of event 6.

Equal challenge inputs at a checkpoint share the commitments, so distinct candidates solve Module-SIS [DBK+26, Corollary 10.25] with $\hat{K} = 4\,\text{Tree}^0_{\text{ck}}(sch)$. Inputs after sd encode the history through event 2, inputs after X encode that through event 6 (X's input a_X), and suffix inputs encode a_X and both branches while omitting Flock messages, unread by suffix maps. This gives (i) and (ii), and [DBK+26, Lemma 10.2] gives (v). Placing the joint GF(2) argument before X preserves all queries and checks, as Flock challengers hash only data sent before X, and choosing Flock messages after X leaves Fail at X unchanged.

Step 3: the checkpoint segment of X. Fix the candidate and its tables f, and set Fail $= 1$ iff a pending claim is false or a verifier check in the segment fails. At a_X, the segment's checks lie ahead and the pending claims are $\mathcal{R}_{\text{PIOP}}$, so Fail reads only a_X and f, giving (iii) and (a). At the root fold only the opening-row claim remains, which Ω satisfies, giving (c). For (b), each site acts on the first false claim determined by its input [CDBT26, Lemma C.1]. The lookup identity at X has the bound of Lemma 4. Each layer or stage round has its degree over $|\mathbb{K}|$, including a later zero-check's degree-1 equality factor, and the zero-check point has $d_{\text{main}}/|\mathbb{K}|$. Merges, batching vectors and Akita's evaluation batch each fail on one hyperplane of mass $1/|\mathbb{K}|$, since a false claim falsifies the batched row outside one hyperplane. Stage-branch inputs leave lookup-branch claims undetermined, so a false claim in either branch persists across the other until stage 3 reads both. Every bad set reads only the input and f, giving (iv).

Step 4: the checkpoint segment of sd. Fix Ω_1, its tables f_1 with bits b, and $\mathcal{J}(C)$. For a listed codeword C' with bits $b'(C')$ we use three predicates. $\text{Spg}(C')$: C' satisfies the R1CS rows and $\mathsf{cst} = 1$. $\text{Pub}(C')$: its public region holds $(\mu_i, \tilde{c}_i)$ on live rows. $\text{Chn}(C')$: $\Gamma b'_\sigma(C') = Y_\sigma$ in every segment σ, with $\Gamma = \Gamma_3\Gamma_2\Gamma_1$. By Lemma 6, Spg $\wedge$ Pub implies $\tilde{c}_i = \mathsf{H}(\mu_i \| b'_i(C'), \lambda/4)$ on live records. The initial claim $\text{Cl}_1 = \mathcal{R}_{\text{FIPS}}(f_1)$ at sd gives (a) and (iii). Subsequent sites pass a false claim down the chain below, with bad sets reading only the input and Ω_1, giving (iv).

1. *Seed* sd. If Cl_1 fails other than through the hash check (4), the bad set is empty and Cl_1 stays false. Otherwise let $i^\star$, in segment $\sigma^\star$, be the first live record with a wrong hash. Every C' with Spg $\wedge$ Pub then has $\Delta_b(C') = b_{\sigma^\star} \oplus b'_{\sigma^\star}(C') \neq 0$, fixed before sd. The next claim Cl_2 says that some such C' has $\Gamma\Delta_b(C') = 0$. It fails outside L_{J} seed

sets of mass $2^{-143.64}$ each (Claim 1), totaling $2^{-138.4}$.

2. *Matrix* Ψ. Round 1 fixes v, turning Cl_1 into $\mathrm{Cl}_1^\star$: (3), $\Delta = 0$ and $|\langle \Psi_o, v_{b,i} \rangle|_c \le B_\Psi$ for all o, b, i. If $\Delta \neq 0$ or (3) fails, $\mathrm{Cl}_1^\star$ fails for all Ψ. If only (2) fails over $\mathbb{Z}$, some $|v_{b,i}(x_0)|_c > 2B_\Psi$ (Lemma 3) and bad Ψ weigh 2^{-128} (Lemma 2). A false Cl_2 persists.
3. *Event 6.* The prover sends Y_σ. A false Cl_2 falsifies Cl_3 ($\Gamma b_{\sigma^\star} = Y_{\sigma^\star}$) or Cl_4 (some C' has $\mathrm{Spg} \wedge \mathrm{Pub} \wedge \mathrm{Chn}$), and the Flock challenger digests add no bad set.
4. *Tie point.* Cl_5 replaces Pub in Cl_4 by the tie identity $C'(r_{\mathrm{tie}}) = \mathrm{pub}(r_{\mathrm{tie}})$. Here pub has $\nu_{\mathrm{pub}} = \nu_{\mathrm{s}} + 3$ variables, or $\nu_{\mathrm{s}} + 4$ for ML-DSA-87, with $\nu_{\mathrm{s}} \ge d_{\mathrm{sig}}$ row variables. One with $\neg\mathrm{Pub}$ passes at mass $L_{\mathrm{J}} \nu_{\mathrm{pub}} / 2^{128}$.
5. *Chain point and rounds.* Cl_6^0 replaces Chn by $\langle W_\Gamma, b'(C') \rangle = \langle \mathrm{eq}_Y, Y \rangle$, at mass $L_{\mathrm{J}}(12 + \log_2 S)/2^{128}$. Each of the $13 + \min(d_{\mathrm{sig}}, 13) + \log_2 S$ degree-2 rounds, 28 at $N = 2^{15}$, passes a false claim Cl_6^{n-1} to Cl_6^n outside mass $2L_{\mathrm{J}}/2^{128}$.
6. *Bit value and Flock.* The prover states $\bar{b} = b'(r)$, and the verifier checks the last chain claim against $W_\Gamma(r)\,\bar{b}$. A false last claim Cl_6^n therefore makes Cl_7 false: no C' with Spg meets both claims of the sponge opening (the tie identity and $b'(C')(r) = \bar{b}$). By Lemma 7, Flock then rejects outside its bad sets.

A stage output at the segment's end passes every check and the test at X (giving $\mathrm{Cl}_1^\star$ and Cl_3) and Flock accepts, so every claim in the chain holds, giving (c).

Conclusion. Lemma 5 applies with Good from Claim 1. Akita [DBK+26, Theorem 10.45, Lemma 10.20] turns a binding witness into a Module-SIS solution of the schedule, with no setup loss under exact rejection [CDBT26, Section C.1]. At most $Q + R$ challenge inputs are queried, each weighing at most 2^{-128} (Appendix B and Lemma 7). Hence $(\bar{Q} + 2)\hat{K}$ expected runs extract f_1 for $\mathcal{R}_{\mathrm{FIPS}}$, except with $\varepsilon_{\mathrm{ext}}$. □

C Ethics Considerations

We reviewed each category of the call for papers and found no ethics concerns. The work aggregates standard ML-DSA signatures without changing the signer or the key format, finds no vulnerability in deployed systems and accepts only valid signatures. All experiments use random keys and messages, with no human subjects, personal data or live system.

D Generative AI Usage Considerations

Generative AI was used for editorial purposes in this manuscript and to assist with programming, and all outputs were inspected by the authors to ensure accuracy and originality.