

Aerie: Efficient Aggregation of Falcon Signatures

Alberto Centelles

Imperial College London
a16z crypto research

Quang Dao*

Carnegie Mellon University

Mohammadtaghi Badakhshan

LayerZero Labs

Justin Thaler

a16z crypto research

Abstract

The post-quantum signatures selected by NIST are ten to over a hundred times larger than the elliptic-curve signatures they replace. Moreover, unlike the BLS signatures that blockchains use to aggregate validator votes, they do not aggregate natively. Both problems can be addressed with a SNARK: a short proof that a batch of signatures is valid, replacing the signatures themselves.

The prevailing approach, central to Ethereum’s roadmap, pairs hash-based signatures with hash-based SNARKs. This is slow because the prover must prove many hash evaluations per signature. We instead aggregate lattice-based signatures with a lattice-based SNARK.

We present Aerie, a proof system for aggregating Falcon-512 and Falcon-1024 signatures. By retaining each signature’s short random salt in the aggregate, Aerie lets the verifier compute the hashes directly. The proof then covers only algebraic and norm checks. Aerie combines protocols tailored to these checks with the Akita polynomial commitment scheme. Unlike prior Falcon aggregation schemes, it also proves that the signatures satisfy Falcon’s byte-format requirements.

On an Apple M4 Max laptop without GPU acceleration, Aerie aggregates 32,768 Falcon-512 signatures, roughly the number of validator votes in one Ethereum slot, in 0.65 seconds. A single proof covers the whole batch, with no recursion. The aggregate is 42.1 bytes per signature including the salts, or about 66 KiB in total if the signatures share one salt. It verifies in 54 milliseconds on one core, 10× faster than checking the signatures individually. On one core, proving takes only about 8× as long as individual verification. We prove knowledge soundness of Aerie in the classical random-oracle model and derive aggregate unforgeability from Falcon’s security.

1 Introduction

Proof-of-stake blockchains depend on signature aggregation: each block must certify the votes of many validators, while bandwidth and block space are at a premium. Ethereum uses the BLS signature scheme to combine roughly 32,000 validator signatures per block into a single short signature [Eth26b, BBG⁺26, Eth26a]. This aggregation saves nodes from transmitting and checking every validator’s signature separately. Solana is also adopting BLS aggregation as part of its Alpenglow consensus upgrade [Sol26].

Post-quantum signatures. BLS is not secure against quantum computers, and standard post-quantum signature schemes do not offer its native aggregation. The post-quantum signatures selected by NIST are

*Part of this work was done while the author was at LayerZero Labs.

Emails: centelles.alberto@gmail.com, qvd@andrew.cmu.edu, mtbadakhshan@gmail.com, justin.r.thaler@gmail.com.

This work is licensed under [CC BY 4.0](#).

also much larger than elliptic-curve-based ones. Falcon-512 signatures occupy 666 bytes, and ML-DSA and SLH-DSA signatures occupy thousands of bytes, compared with about 64 bytes for the elliptic-curve signatures used in today’s transactions [FHK⁺20, Nat24a, Nat24b]. Without aggregation, post-quantum signatures would consume much of the space in a block, limiting throughput and increasing the cost of running a node.

Aggregation with SNARKs. SNARKs address both problems by replacing a batch of signatures with a short proof that they are valid. The prevailing approach, central to Ethereum’s post-quantum roadmap [DAH25], pairs hash-based signatures with hash-based SNARKs. Ethereum’s leanVM is being developed for this purpose. The main cost is hashing: verifying a hash-based signature requires many hash evaluations, and the prover must prove that each was computed correctly.

leanVM reports aggregating 900 signatures from the hash-based scheme XMSS in 563 milliseconds on an Apple M4 Max [lea26b], or about 1,600 signatures per second. At this rate, one laptop cannot aggregate a block’s signatures within Ethereum’s twelve-second block time. Ethereum’s proposed design distributes the work across provers, then combines their proofs recursively: each new proof certifies that several preceding proofs verify [lea26c]. These rounds add latency and require coordination among provers.

Aerie. We present Aerie,¹ a lattice-based proof system for aggregating Falcon signatures. Falcon is a lattice-based signature scheme selected by NIST for standardization [Nat26]. Aerie supports its two parameter sets, Falcon-512 and Falcon-1024, which use polynomials with 512 and 1024 coefficients and target NIST security categories I and V, respectively [FHK⁺20].² Falcon-1024 offers higher security at the cost of larger signatures and more computation. Aerie follows the *public-salt* approach of Aardal et al. [AAB⁺24]: the aggregate retains each signature’s 40-byte random salt, so the verifier computes Falcon’s message hashes itself and the proof needs to cover only algebraic and norm checks.

On an Apple M4 Max laptop without GPU acceleration, Aerie aggregates 32,768 Falcon-512 signatures in 0.65 s, or about 50,800 signatures per second. One proof covers the whole batch, with no recursion. This is 29 times leanVM’s XMSS throughput on the same laptop (Section 6.3), and well above the 1,000 signatures per second targeted by Ethereum’s lean consensus effort [lea26a].

1.1 Our Results

Performance. For 32,768 Falcon-512 signatures, Aerie’s aggregate is 42.1 bytes per signature, 16 times smaller than the signatures it replaces. Checking the aggregate takes 54 ms on one core, 10 times faster than checking the signatures individually. On one core, producing the proof takes 8 times as long as checking the signatures individually and 1.4 times as long as signing them.

For 16,384 Falcon-1024 signatures, proving takes 0.63 s and verification takes 49 ms on one core, a 10-fold verification speedup. The aggregate is 44.1 bytes per signature, a 29-fold reduction. Both batches use 2.1 GiB peak prover memory.

Aerie also substantially outperforms Orthus [BBL⁺26], the previous state of the art for Falcon aggregation. With both systems on one core of the same AVX-512 server, Aerie proves 32,768 Falcon-512 signatures 1.5 times faster, verifies them 38 times faster, and produces proofs 1.7 times smaller (Table 6). Unlike prior schemes, Aerie also proves that each signature fits Falcon’s prescribed padded format, enforcing verification conditions that those schemes omit.

Aggregate size and latency. The public salts account for most of the aggregate’s size: for the Falcon-512 batch above, the proof is about 66 KiB of the 1,346 KiB aggregate. The signatures can have different keys, messages, and salts. If they share a salt, the aggregate shrinks to about 66 KiB in total (Section C.7).

¹An aerie is the nest of a bird of prey, such as a falcon. The name evokes bringing many Falcons together in one aggregate.

²In category I (resp. V), any attack must require computational resources comparable to or greater than those of key search on AES-128 (resp. AES-256) [Nat16, Section 4.A.5].

At Aerie’s proving speed, one node can aggregate a block’s signatures directly, leaving more time for collecting votes and propagating the aggregate. No intermediate proofs need to be generated or merged, and another node holding the signatures can produce a replacement if the aggregator fails. Our leanVM measurements instead imply about 2.0 seconds for a similar batch even with a separate M4 Max for every proof in its recursion tree. Minimizing total proving work gives about 29 machine-seconds, the sum of all provers’ running times (Section 6.3).

Security. We prove knowledge soundness of Aerie in the classical random-oracle model: from a prover that produces accepting proofs with sufficient probability, one can extract signatures satisfying Falcon’s verification conditions. The result covers both variants at every supported batch size. We then reduce aggregate forgery to Falcon forgery without a security loss that grows with the batch size. The error contains a term of at most 2^{-128} per prover hash query, together with terms for breaking Falcon and Module-SIS, the lattice problem underlying the Akita polynomial commitment scheme we use [DBK⁺26].³

Avoiding recursion also avoids a gap in the security analysis. In recursive aggregation, each proof shows that the SNARK verifier accepts the proofs it combines, so the proven computation runs that verifier, including its hash function. The random-oracle model cannot describe this computation, and known analyses of recursion rest instead on non-falsifiable assumptions without concrete bounds (Section 1.2). Aerie’s relation evaluates no hash function, and its security reduces in the random-oracle model, with explicit bounds, to Falcon and Module-SIS. Section 1.3 describes the construction and its main technical contributions.

1.2 Related Work

Falcon aggregation. Aardal et al. [AAB⁺24] introduced public-salt Falcon aggregation using LaBRADOR [BS23]. They prove knowledge soundness in the random-oracle model and derive aggregate unforgeability from Falcon security. Their proof enforces Falcon’s equation and exact norm bound, with additional bounds to prevent modular arithmetic from accepting invalid integer solutions. It does not enforce Falcon’s byte-format requirements. The implementation by Nevado, Kim, and Stopar using Dachshund proves a batch of 10,000 Falcon-512 signatures in 5.96 seconds and verifies it in 2.65 seconds on a single Intel i5-11400H core [NKS25].

Orthus [BBL⁺26] gives a batch-verification protocol for lattice relations whose verifier does less work than checking each relation separately. Its Falcon benchmarks use distinct public keys with a common hash target, omitting per-signature hashing and byte-format checks. The reported timings for its composition with LaBRADOR use a single Intel i7-11850H core and preprocessed public keys [BBL⁺26]. Our comparison uses the same host for both systems (Section 6.3).

Norm bounds by random projection. LaBRADOR [BS23] bounds the norm of a witness by sending a short random projection of it, using a modular Johnson–Lindenstrauss lemma of Gentry, Halevi, and Lyubashevsky [GHL22]. Its projection matrix has a column for every witness coordinate, so the verifier’s work to check the projection is linear in the witness size. Aardal et al. combine this projection with the exact norm proof of [GHL22] for Falcon signatures [AAB⁺24]. Orthus [BBL⁺26] projects each vector of its batch with the same matrix and commits to the projections instead of sending them, which shrinks the matrix the verifier samples.

Kloof et al. [KLNO25] reuse one small matrix on every block of the input, which gives the matrix a short description, but the output is still too long to send and is committed. PikkuFold [Osa26] applies such layers repeatedly until the output is short enough to send. Its analysis bounds the norm of each block separately and takes a union bound over the blocks. Geng and Plançon [GP26] generate each output coordinate from an independent tensor train. Aerie’s norm projection uses layers in the style of PikkuFold, but its analysis

³For a prover making Q hash queries, counting those of verification, the reductions’ expected number of prover runs is below $(Q + 2) 2^{40}$; Section 5 states the full bounds and assumptions.

bounds the total squared norm of all blocks of a layer at once (Section A.2). Section A.5 compares it with TTRP at matched error bounds.

Other lattice-based signatures. Other constructions change the signature scheme or the signing model. Boneh and Kim [BK20] obtain logarithmic-size lattice aggregates, but their publicly aggregatable construction allows each key to sign only once. Supporting multiple signatures per key requires interaction with the signers. Lemur [ELS⁺26] lets signers independently sign the same message in the same time step and combines their signatures. These constructions do not aggregate already generated Falcon signatures with independent keys, salts, and messages.

Tomita and Shikata [TS23] use LaBRADOR to aggregate a different lattice-based signature scheme, adapted from CLMQ. It supports multiple signatures per key, requires no interaction with signers, and keeps message hashing outside the proof. They give estimates of aggregate sizes.

Hash-based signatures. Flock [BRW26] targets hash-based signatures such as XMSS by proving the repeated hashing operations and Merkle-tree authentication paths used in their verification [BDH11, Hül13, HBG⁺18], and leaves complete signature verification as an extension.

The recursive aggregation used by leanVM introduces a further cost: combining two proofs into one is reported to take 287 milliseconds on an Apple M4 Max for input proofs of 900 signatures each [lea26b]; Section 6.3 measures leanVM and Aerie on one machine.

Security of recursive composition. In recursive composition, a SNARK proves that its own verifier accepts earlier proofs, so the computation being proven includes the verifier’s hash evaluations. When the SNARK is proven secure in the random-oracle model, that model cannot analyze this step: no succinct argument in the random-oracle model can prove computations that themselves query the oracle [BCG24]. Known analyses therefore replace the oracle by a concrete hash function and assume that the resulting SNARK has a non-black-box knowledge extractor [COS19, BCMS20, CGSY23]. This assumption is not falsifiable and gives no concrete bound, and the security loss of the composed proof is exponential in the recursion depth [BCCT12, CGSY23].

In practice, recursive SNARKs are parameterized as if composition lost no security [CGSY23]. Proofs in oracle models designed for recursion avoid the extractor assumption [CCG⁺23, CGSY23], but it is a conjecture that a concrete hash function instantiates these models. Circuits that evaluate their own Fiat–Shamir hash, as recursive verifiers do, are also the setting of the practical Fiat–Shamir attack of Khovratovich et al. [KRS25].

1.3 Technical Overview

Falcon verification conditions. For each signature i , the verifier knows the public-key polynomial h_i and, since the salt is public, computes the polynomial c_i itself by hashing the message and salt, as in Aardal et al. [AAB⁺24]. Falcon requires two short integer polynomials $s_{1,i}, s_{2,i}$ satisfying an equation involving h_i and c_i . Here “short” means that the sum of their squared coefficients is at most Falcon’s bound $\lfloor \beta^2 \rfloor$. We turn this inequality into an equality by adding four squares, represented by a vector $e_i \in \mathbb{Z}^4$. The proof checks

$$\begin{aligned} s_{1,i} + h_i \otimes s_{2,i} &\equiv c_i \pmod{q}, \\ \|s_{1,i}\|_2^2 + \|s_{2,i}\|_2^2 + \|e_i\|_2^2 &= \lfloor \beta^2 \rfloor, \\ |s_{2,i,j}| &\leq 2047, \quad \sum_j \lfloor |s_{2,i,j}|/128 \rfloor \leq L. \end{aligned}$$

Here $q = 12,289$, n is the number of coefficients, and $\otimes$ is polynomial multiplication modulo $X^n + 1$. The last two conditions ensure that $s_{2,i}$ fits Falcon’s fixed-length signature format, which encodes each coefficient by a sign, seven low magnitude bits, and $\lfloor |s_{2,i,j}|/128 \rfloor$ in unary, and leaves room for L unary zero-bits in total [FHK⁺20, Section 3.11.2]. Section 3.1 gives the parameters for both Falcon variants.

Integer conditions over a prime field. These conditions concern integers, but the proof system works over $\mathbb{F}_p$, with $p = 2^{64} - 59$. Over the integers, the ring equation becomes

$$s_{1,i} + h_i \cdot s_{2,i} = c_i + (X^n + 1) \cdot t_i + q \cdot k_i,$$

where t_i and k_i account for the reductions modulo $X^n + 1$ and modulo q . A direct approach commits to both quotients for every signature and proves bounds on their coefficients. The norm equation also needs an integer bound, since a squared norm of $\lfloor \beta^2 \rfloor + p$ satisfies it over $\mathbb{F}_p$. Aerie commits only to s_1 , the norm slack e , the signs and magnitude digits of s_2 in Falcon’s encoding, and digits of each signature’s unused encoding length. Every other witness-dependent polynomial is *virtual*: its values on Boolean inputs are determined by the committed and public polynomials, and the prover never commits to it. We check the conditions that would otherwise need quotients with two short random projections and the sum-check protocol.

One set of digits for s_2 . We commit to the digits and signs of Falcon’s encoding as multilinear polynomials a_0, a_1, a_2, σ over the coefficient and signature indices, with $a_0 \in [-4, 3]$, $a_1, a_2 \in [-8, 7]$, and $\sigma \in \{-1, 1\}$. The magnitude $M = (a_0 + 4) + 8(a_1 + 8) + 128(a_2 + 8)$ and the coefficient $S_2 = \sigma M$ are virtual. Since $0 \leq M \leq 2047$ and $a_2 + 8$ is the number of unary zero-bits, the digit ranges bound both the coefficients and their encoded lengths. The total length becomes an equality $\sum_j (a_2(j, i) + 8) + u(i) = L$, where committed digits u_0, u_1, u_2 encode the unused length $u(i) \geq 0$. Both sides are small, so this field identity holds over the integers. The magnitude digits also enter the ring and norm checks. The six digit polynomials are the only ones that need range checks, which we prove with Akita’s range argument [DBK⁺26, Section 6.1], and an identity $\sigma^2 = 1$ fixes the signs.

A norm bound from one structured projection. The entries of s_1 and e reach 6144 and 8191, so we commit to the batch’s polynomials S_1 and e directly and bound their combined squared norm by a random projection. After the commitments, the verifier samples a seed for a structured projection Π_{norm} [KLNO25, Osa26, GP26]: a composition of small matrices with entries in $\{-1, 0, 1\}$, each applied with the same coefficients to every block of its input. The prover sends $y_{\text{norm}} = \Pi_{\text{norm}}(S_1 \| e) \in \mathbb{F}_p^{156}$, and the verifier checks that its squared norm, computed from the representatives closest to zero, is at most a public threshold Θ .

Our analysis bounds the total squared norm of all blocks through each matrix, rather than the norm of each block separately, and accounts for reduction modulo p . An input fixed before sampling whose squared norm is at least a limit $E_\star$ then passes with probability at most 2^{-136} (Theorem 5.1). Below $E_\star$, and with $|S_2| \leq 2047$, every signature’s squared norm is less than p , so the norm identity holds over the integers. With the same normalized threshold and error bounds, our instantiation of the tensor-train projection of Geng and Plançon [GP26] sends 526 values instead of 156 at power-of-two batch sizes from 1K to 64K (Section A.5).

Divisibility by q without quotients. Given the norm bound, every coefficient of the virtual residual $\Delta = S_1 + h \otimes S_2 - C$ is below $p/2$ in absolute value, where h and C encode the batch’s h_i and c_i . Its representative closest to zero is the integer residual, which must be divisible by q . A second projection Π_{res} of three matrices with entries uniform in $\{-12, \dots, 12\}$ tests this: the prover sends $y_{\text{res}} = \Pi_{\text{res}}(\Delta) \in \mathbb{F}_p^{31}$, and q must divide the centered representative of every output. A fresh matrix maps an input that is nonzero modulo q to an output that is zero modulo q with probability at most 25^{-31} . An invalid residual fixed before sampling therefore passes with probability less than 2^{-142} , including wraparound modulo p (Lemma A.8); Section 5 accounts for a prover that tries many seeds. This test replaces the quotients k_i .

Following Akita and Grand Danois [DBK⁺26, KK26], we verify the product $h \otimes S_2$ with the sum-check protocol, using public weights that include the reduction modulo $X^n + 1$, so the quotients t_i are never committed either. Figure 1 illustrates both projections.

Three sum-check stages, one opening. The remaining checks are identities among polynomials on the Boolean hypercube: the ring and norm equations, the encoding-length equality, $\sigma^2 = 1$, the digit ranges, and the 156 + 31 projected values the prover sent. The verifier combines these identities with random weights and

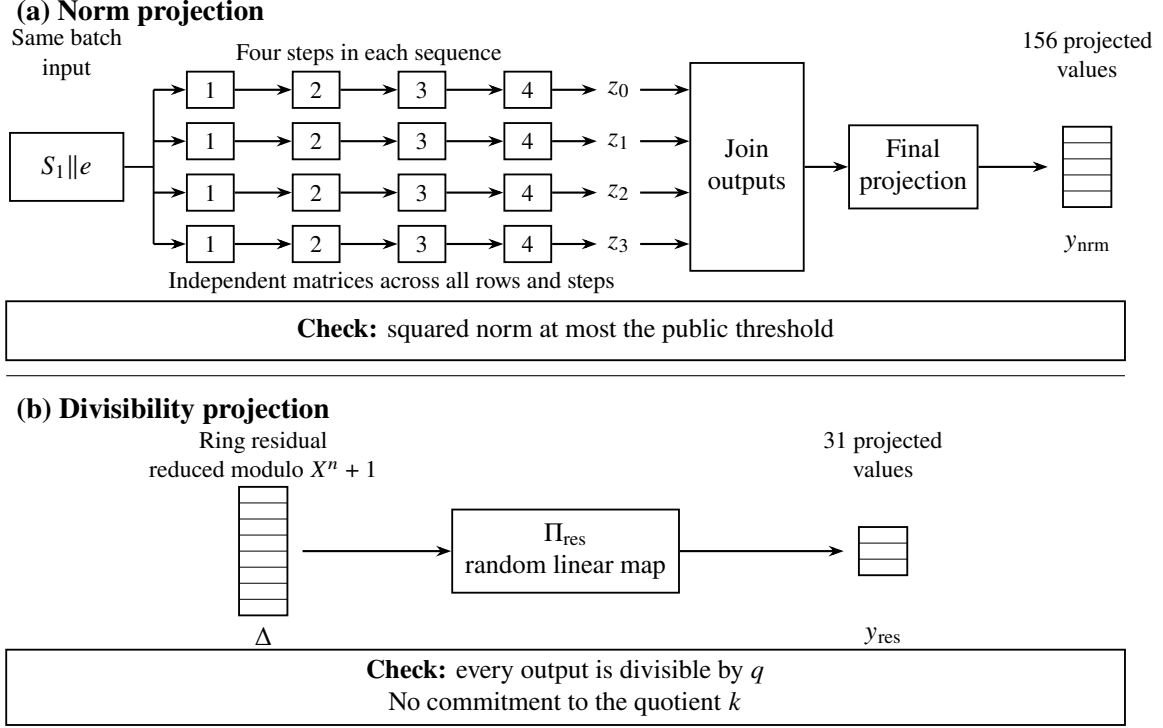


Figure 1: The two projection checks. Each numbered box applies a random matrix. Each row starts from the same batch and performs four steps. The four outputs are joined before one shared final projection. Within each step, its matrix is reused across input blocks (Section 2.3). Vector lengths are schematic, and both checks use centered integer representatives of the field outputs. We use the sum-check protocol to verify that both projected vectors were computed from the committed data.

applies the sum-check protocol, which reduces them to claims about the values of the polynomials involved at a random point. For a committed polynomial such as S_1 , such a claim is settled by opening the commitment.

Many of these polynomials are virtual, however, such as $S_2 = \sigma M$ and the residual Δ , and there is no commitment to open. The verifier knows only how each virtual polynomial is defined from other polynomials on Boolean inputs, and this definition need not hold at a random point. A claim about a virtual polynomial therefore needs another sum-check, which reduces it to claims about the polynomials it is defined from. Affine definitions such as that of M hold at every point, so the verifier evaluates them directly.

Each reduction can start only after the previous one has fixed its random point, so the proof proceeds in stages, and the number of stages is the length of the longest chain of reductions. In Aerie this chain comes from the range checks, where Akita’s range argument places two layers of virtual polynomials between the checked identity and the committed digits. The proof therefore has three sum-check stages, and every other check is batched into one of them. At the end, every claim is about a committed polynomial, and one batched Akita opening proves the nine resulting evaluations (Figure 3 and Lemma C.1).

2 Preliminaries

2.1 Fields, Rings, and Integer Representatives

We write $[v] = \{0, \dots, v - 1\}$; all logarithms are to base two. Falcon uses the ring $R_q = \mathbb{F}_q[X]/(X^n + 1)$, where $q = 12,289$ and $n \in \{512, 1024\}$ [FHK⁺20]. We identify an integer polynomial of degree less than n

with its coefficient vector in $\mathbb{Z}^n$, with $\|a\|_2^2 = \sum_j a_j^2$ and $\|a\|_\infty = \max_j |a_j|$. For $a \in \mathbb{F}_q$, $\text{ctr}_q(a)$ denotes the unique integer in $(-q/2, q/2)$ congruent to a ; ctr_p is defined in the same way for the field $\mathbb{F}_p$ below, and both apply coefficientwise to polynomials. Multiplication in $\mathbb{Z}[X]/(X^n + 1)$ is negacyclic convolution:

$$(a \otimes b)_k = \sum_{j=0}^k a_j b_{k-j} - \sum_{j=k+1}^{n-1} a_j b_{n+k-j}, \quad k \in [n].$$

Its reduction modulo q is multiplication in R_q .

The proof uses the base field $\mathbb{F}_p$, where $p = 2^{64} - 59$, and the challenge field $K = \mathbb{F}_{p^2}$. We embed integers into $\mathbb{F}_p$ by reduction modulo p . An integer identity implies the corresponding field identity. Conversely, integers $a \equiv b \pmod{p}$ with $|a - b| < p$ are equal. We lift field identities in this way, for nonnegative integers below p or for integers in $(-p/2, p/2)$, which equal ctr_p of their residues. All algebraic challenges in the interactive protocol are independent and uniform in K , sampled after the preceding prover messages.

2.2 Multilinear and Virtual Polynomials

For $r, x \in K^v$, define

$$\text{eq}(r, x) = \prod_{t \in [v]} (r_t x_t + (1 - r_t)(1 - x_t)).$$

Every function $T : \{0, 1\}^v \rightarrow K$ has a unique multilinear extension, which we denote by the same letter:

$$T(r) = \sum_{x \in \{0, 1\}^v} \text{eq}(r, x) T(x), \quad r \in K^v.$$

In particular, a function with values in $\mathbb{F}_p$ defines a polynomial over $\mathbb{F}_p$ that may be evaluated at points in K^v .

A *virtual polynomial* is a multilinear polynomial specified by its Boolean values as a function of committed polynomials and public data, without a separate commitment. Formally, if these values are $f(x)$, the virtual polynomial is

$$P_f(r) = \sum_{x \in \{0, 1\}^v} \text{eq}(r, x) f(x).$$

For example, $P_Z(x) = Z(x)(Z(x) + 1)$ on Boolean inputs, but generally $P_Z(r) \neq Z(r)(Z(r) + 1)$ for $r \in K^v$. Products within a sum-check summand retain their degrees. They are not implicitly multilinearized. We use the sum-check protocol to reduce a claim on P_f to evaluations of the committed polynomials defining f .

2.3 Structured Projections

For positive integers c, c', J, m , each projection step applies a matrix $\Psi \in \mathbb{F}_p^{c' \times (cJ)}$ to an array $T \in \mathbb{F}_p^{c \times J \times m}$ by

$$T'(b, t) = \sum_{a \in [c]} \sum_{j \in [J]} \Psi_{b, (a, j)} T(a, j, t), \quad (b, t) \in [c'] \times [m].$$

For each fixed t , we arrange the cJ entries $T(a, j, t)$ as a vector and multiply it by the same matrix Ψ , producing c' entries. Here a indexes the values from the preceding step, j indexes the next block of J coordinates, and t indexes the remaining coordinates. We call each subarray $T(\cdot, \cdot, t)$ an input block. Repeating this operation, with the remaining coordinates reshaped between steps, defines the structured projection [KLNO25, Osa26]. The matrix width c' counts the projected values per input block.

Claim. $\sum_{x \in \{0,1\}^v} F(x) = t_0$, where F has degree at most $D_j \geq 1$ in its j th variable.

Round $j = 0, \dots, v-1$. The prover forms $g_j(Y) = \sum_{x \in \{0,1\}^{v-j-1}} F(r_0, \dots, r_{j-1}, Y, x)$ with coefficients $g_{j,k}$ and sends all except $g_{j,1}$. The verifier recovers $g_{j,1} = t_j - 2g_{j,0} - \sum_{k=2}^{D_j} g_{j,k}$, then samples and sends a fresh uniform $r_j \in K$; both parties set $t_{j+1} = g_j(r_j)$.

Final check. Authenticate $F(r_0, \dots, r_{v-1}) = t_v$.

Equality-factored form. For fixed $\tau \in K^v$, let $F = \text{eq}(\tau, \cdot)G$, where G has degree at most $D_j - 1$ in its j th variable. Then $g_j(Y) = \text{eq}(\tau_j, Y) \bar{g}_j(Y)$, where $\text{eq}(\tau_j, Y) = (1 - \tau_j)(1 - Y) + \tau_j Y$, $\deg \bar{g}_j \leq D_j - 1$, and $\bar{g}_j$ includes the fixed-prefix equality weights. The prover instead sends $\bar{g}_{j,1}, \dots, \bar{g}_{j,D_j-1}$. Since $g_j(0) + g_j(1) = \bar{g}_{j,0} + \tau_j \sum_{k \geq 1} \bar{g}_{j,k}$, the verifier recovers $\bar{g}_{j,0} = t_j - \tau_j \sum_{k=1}^{D_j-1} \bar{g}_{j,k}$ and uses the same challenge, update, and final check.

Figure 2: The prover sends D_j field elements per round in the sum-check protocol [Tha23, Tha25] and $D_j - 1$ in its equality-factored form [DDB⁺26].

2.4 Sum-check and Polynomial Commitments

Figure 2 specifies sum-check and its equality-factored variant. For individual degree bounds $D_0, \dots, D_{v-1}$, a false initial claim is accepted with probability at most $\sum_j D_j / |K|$, provided that the terminal evaluation is correct.

Batching Sums on Different Domains. We batch claims $\sum_{x \in \{0,1\}^{v_\ell}} F_\ell(x) = a_\ell$ with $v_\ell \leq d$ into one d -round sum-check stage. The summand F_ℓ is scaled by $2^{v_\ell - d}$ and made constant in the first $d - v_\ell$ variables, so its terminal evaluation uses the last v_ℓ challenges, possibly under a fixed permutation of its variables. Independent claims receive independent uniform coefficients $\gamma_\ell \in K$, sampled after the claims are fixed; a false vector of claims passes the combined check with probability at most $1/|K|$. Terms of one identity on different domains keep their relative coefficients and share one γ_ℓ , so no selector variables are needed.

In the equality-factored form of Figure 2, recovering $\bar{g}_{j,0}$ needs no division, so it is valid even when $\tau_j = 0$. A summand without the common equality factor contributes a round polynomial g_j^{pub} known to the verifier; for example, a summand that is inactive in the current round, with running claim b including its batching coefficient, contributes the constant $b/2$. Both parties subtract $g_j^{\text{pub}}(0) + g_j^{\text{pub}}(1)$ from t_j before the recovery and add $g_j^{\text{pub}}(r_j)$ to t_{j+1} .

Polynomial Commitments. Akita is a polynomial commitment scheme for multilinear polynomials specified by their base-field Boolean evaluations [DBK⁺26]. It can commit to a group of polynomials of the same dimension and prove their evaluations at a common point, with openings for several groups batched into one proof. Evaluation binding rules out inconsistent evaluation claims; knowledge soundness permits extraction of the committed polynomials.

2.5 Sums of Squares

Lagrange’s four-square theorem states that every nonnegative integer is a sum of four integer squares. Thus a squared-norm inequality $\|s\|_2^2 \leq B$, for $s \in \mathbb{Z}^m$ and $B \in \mathbb{Z}_{\geq 0}$, is equivalent to the existence of $e \in \mathbb{Z}^4$ such that $\|s\|_2^2 + \|e\|_2^2 = B$. We use the following bounded version for Falcon’s norm thresholds.

Lemma 2.1. Every integer $0 \leq \delta \leq 70,265,242$ is a sum of four squares of integers of magnitude at most 8191.

The proof appears in Appendix B.

3 The Statement

3.1 The Batch Relation

Falcon verifies a signature, consisting of a salt r and a decoded polynomial s_2 , on a message m under a public key h by computing a target $c \in R_q$ from r and m , reconstructing $s_1 = \text{ctr}_q(c - h \otimes s_2)$, and checking $\|s_1\|_2^2 + \|s_2\|_2^2 \leq \lfloor \beta^2 \rfloor$ [FHK⁺20]. Here β is Falcon’s norm bound and $\lfloor \beta^2 \rfloor = \lfloor \beta^2 \rfloor$. We use Falcon’s fixed-length padded format of ℓ_{sig} bytes, including a one-byte header and the 40-byte salt. It encodes each coefficient of s_2 , which must satisfy $|s_{2,j}| \leq 2047$, by its sign, seven low magnitude bits, and a unary encoding of $\lfloor |s_{2,j}|/128 \rfloor$ with a terminator, so the coefficient stream has $9n + \sum_j \lfloor |s_{2,j}|/128 \rfloor$ bits and must fit in the remaining $8\ell_{\text{sig}} - 328$ bits. Thus $L = 8\ell_{\text{sig}} - 328 - 9n$ unary bits are available. The parameters $(n, \lfloor \beta^2 \rfloor, \ell_{\text{sig}}, L)$ are (512, 34,034,726, 666, 392) for Falcon-512 and (1024, 70,265,242, 1280, 696) for Falcon-1024. Aerie proves the following relation for a batch of $N_{\text{sig}} \geq 1$ signatures.

The Relation $\mathcal{R}_{\text{batch}}$.

Statement. A parameter set and public records $(h_i, m_i, r_i)_{i < N_{\text{sig}}}$, with verifier-computed targets c_i (Section 3.2).

Witness. Integer polynomials $s_{1,i}, s_{2,i} \in \mathbb{Z}[X]/(X^n + 1)$ and vectors $e_i \in \mathbb{Z}^4$ satisfying, for every $i < N_{\text{sig}}$:

1. **Ring equation.** $s_{1,i} + h_i \otimes s_{2,i} - c_i \equiv 0 \pmod{q}$.
2. **Exact norm.** $\|s_{1,i}\|_2^2 + \|s_{2,i}\|_2^2 + \|e_i\|_2^2 = \lfloor \beta^2 \rfloor$.
3. **Encodability.** $|s_{2,i,j}| \leq 2047$ for every j , and $\sum_j \lfloor |s_{2,i,j}|/128 \rfloor \leq L$.

In the relation, h_i and c_i denote the integer polynomials $\text{ctr}_q(h_i)$ and $\text{ctr}_q(c_i)$. The slack vector e_i turns Falcon’s norm inequality into an equality (Lemma 2.1). An honest prover takes $s_{1,i} = \text{ctr}_q(c_i - h_i \otimes s_{2,i})$, so $|s_{1,i,j}| \leq 6144$, and chooses $|e_{i,k}| \leq 8191$.

3.2 Public Records and Targets

The verifier supplies the canonical public-key bytes pk_i and the messages m_i ; the aggregate consists of the salts r_i and a proof. The verifier itself computes the 64-byte key hash $\text{hpk}_i = \text{SHAKE256}(\text{pk}_i, 64)$, which the prover never supplies, and then $h_i = \text{Decode}(\text{pk}_i)$ and

$$c_i = \text{HashToPoint}(r_i \| \text{hpk}_i \| \text{0x00} \| \text{0x00} \| m_i) \in R_q.$$

We call this the *key-prefixed* framing. The Falcon v1.2 specification [FHK⁺20] hashes $r_i \| m_i$, which is the special case obtained by dropping $\text{hpk}_i \| \text{0x00} \| \text{0x00}$. The relation takes each c_i as a public input, so the proof system and its knowledge-soundness theorem are the same for either framing; only the verifier’s hashing differs. Our implementation uses the key-prefixed framing (pure mode, empty context string), which was proposed for the forthcoming standard [Per25]; the standard is not final, and a different final framing would change only the verifier’s hashing. Public keys and targets are represented by their centered coefficients in $[-6144, 6144]$. Each salt has 40 bytes [FHK⁺20, Section 3.10.1].

The *salt mode* records whether the aggregate carries one salt per signature or one common salt for the batch. With one salt per signature, signatures come from FN-DSA signing as implemented in `fn-dsa` [Por25]: every sampling attempt draws a fresh salt, and an attempt that fails the norm or encoding check is discarded with its salt. Fouque et al. reduce the unforgeability of this signer to a lattice assumption in the random-oracle model [FGdG⁺26], and our main security theorem uses it.

A common salt is shared by the whole batch, so a failed attempt cannot draw a new one. A common-salt signer therefore uses the *fixed-target signer*: it retries the sampler at the common salt’s target, as Falcon v1.2 does [FHK⁺20, Algorithm 10], and derives its sampler randomness deterministically, so a repeated request returns the same signature [LP26]. Its signatures are ordinary signatures under the key-prefixed framing. The common-salt theorem of Section C.7 assumes that this signer is unforgeable (Assumption C.8). We report common-salt sizes only as estimates. Section 4 binds the ordered records (hpk_i, m_i, r_i) , their count, and the salt mode into the transcript.

3.3 Polynomial Representation of a Batch

Let $d_{\text{coef}} = \log_2 n$, $d_{\text{sig}} = \max(\lceil \log_2 N_{\text{sig}} \rceil, 12 - d_{\text{coef}})$, and $d = d_{\text{coef}} + d_{\text{sig}}$. We pad the batch to $N^* = 2^{d_{\text{sig}}} = \max(2^{\lceil \log_2 N_{\text{sig}} \rceil}, 4096/n)$ rows, so $d \geq 12$, and write $\text{live}(i) = 1$ if $i < N_{\text{sig}}$ and 0 otherwise. The relation is defined for every positive batch size. The projection parameters analyzed here support $12 - d_{\text{coef}} \leq d_{\text{sig}} \leq 16$. On padded rows, set $h_i = c_i = 0$ and extend the witness by $s_{1,i} = s_{2,i} = 0$ and $e_i = 0$. These rows have no public message, salt, or target.

The coefficient polynomials are specified by the values in $\mathbb{F}_p$ $S_1(j, i) = s_{1,i,j} \bmod p$, $S_2(j, i) = s_{2,i,j} \bmod p$, $h(j, i) = \text{ctr}_q(h_i)_j \bmod p$, and $C(j, i) = \text{ctr}_q(c_i)_j \bmod p$. Their common domain is $\{0, 1\}^{d_{\text{coef}}} \times \{0, 1\}^{d_{\text{sig}}}$, with coefficient bits preceding signature bits, each least significant bit first. The polynomial e is specified by $e(k, i) = e_{i,k} \bmod p$ on $\{0, 1\}^2 \times \{0, 1\}^{d_{\text{sig}}}$; a polynomial with one value per signature is specified on $\{0, 1\}^{d_{\text{sig}}}$. These Boolean values define their multilinear extensions as in Section 2.

A point $r \in K^d$ splits as $(r_{\text{coef}}, r_{\text{sig}})$. Its suffix r_e consists of the last two coefficient coordinates and all signature coordinates. The salt r_i belongs to the public record; challenge points carry a superscript identifying the sum-check stage or a descriptive subscript.

The ring equation has no quotient witness. Instead, the virtual residual Δ is defined by its Boolean values $\Delta(\cdot, i) = S_1(\cdot, i) + h(\cdot, i) \otimes S_2(\cdot, i) - C(\cdot, i)$ over $\mathbb{F}_p$. The protocol proves that the corresponding integer residual is divisible by q ; multiplication by q^{-1} in $\mathbb{F}_p$ would not establish the ring equation.

3.4 From a Witness to Signature Bytes

Let $\text{EncodePadded}(r, s_2)$ be the deterministic encoding of length ℓ_{sig} with the parameter-set header, positive sign for zero, and zero-filled unused bits.

Lemma 3.1 (Canonical Padded Encoding). If (s_1, s_2, e) satisfies $\mathcal{R}_{\text{batch}}$ for a record (h, m, r) , then $\text{EncodePadded}(r, s_2)$ is defined and accepted by Falcon verification with the same parameter set and hash-to-point framing.

Proof. The coefficient and unary-encoding length bounds make the encoding defined. The verifier decodes s_2 , recomputes c , and reconstructs $s'_1 = \text{ctr}_q(c - h \otimes s_2)$. The ring equation gives $s'_1 \equiv s_1 \pmod{q}$. Centered reduction minimizes the absolute value in each residue class, so $\|s'_1\|_2^2 \leq \|s_1\|_2^2$. Consequently $\|s'_1\|_2^2 + \|s_2\|_2^2 \leq \lfloor \beta^2 \rfloor - \|e\|_2^2 \leq \lfloor \beta^2 \rfloor$, and the verifier accepts. $\square$

The lemma does not need the extracted s_1 to be centered modulo q , which the norm identity cannot force at Falcon-1024 (it permits a coefficient of $8382 > 6144$); recentering only decreases the norm.

4 The Three-Stage Protocol

Figure 3 specifies the interactive protocol. Stage s follows Figure 2. Its d independent uniform challenges form $r^{(s)} \in K^d$. We use the dimensions and Boolean index $x = (j, i)$ of Section 3. The verifier rejects if any check fails.

Public Inputs: Falcon degree $n \in \{512, 1024\}$ and ordered records (pk_i, m_i, r_i) for $i < N_{\text{sig}}$ (public-key bytes, message bytes, 40-byte salt). The verifier computes hpk_i, h_i and c_i (Section 3.2) and derives $h, C : \{0, 1\}^d \rightarrow \mathbb{F}_p$ as in Section 3.

Witness: Multilinear extensions of

$$S_1, a_0, a_1, a_2, \sigma : \{0, 1\}^d \rightarrow \mathbb{F}_p, \quad e : \{0, 1\}^{2+d_{\text{sig}}} \rightarrow \mathbb{F}_p, \\ u_0, u_1, u_2 : \{0, 1\}^{d_{\text{sig}}} \rightarrow \mathbb{F}_p.$$

Sum-checks: Stage s uses batching coefficients $\gamma^{(s)}$ and terminal point $r^{(s)}$ (Section 2.4). Claims for u_0, u_1, u_2 use signature coordinates. $r_{\Pi}, r_{\Pi, e}$ follow Section A.1.

1. Commit and project

$P \rightarrow V :$ $[S_1], [e], [a_0, a_1, a_2, \sigma], [u_0, u_1, u_2]$ (Table 1).

$V \rightarrow P :$ uniform 256-bit projection seed (Section 4.2).

$P \rightarrow V :$ $y_{\text{nrm}} = \Pi_{\text{nrm}}(S_1 \| e) \in \mathbb{F}_p^{156}, \quad y_{\text{res}} = \Pi_{\text{res}}(\Delta) \in \mathbb{F}_p^{31}.$

$$V \text{ checks : } \sum_o \text{ctr}_p(y_{\text{nrm}, o})^2 \leq \Theta, \quad q \mid \text{ctr}_p(y_{\text{res}, o}) \quad (o \in [31]).$$

$V \rightarrow P :$ $\mu \leftarrow K^8, \quad \tau \leftarrow K^d, \quad \gamma^{(1)}.$

Public weights: $\lambda_o = \prod_{t=0}^7 \mu_t^{o_t}$ for $o = \sum_t o_t 2^t < 187, y = (y_{\text{nrm}}, y_{\text{res}}), (w_{S_1}, w_e) = \Pi_{\text{nrm}}^\top \lambda_{\text{nrm}}, w_\Delta = \Pi_{\text{res}}^\top \lambda_{\text{res}}.$

2. Stage one: range factors and projected values

Degree 3

$$\text{Range factors: } \sum_x \text{eq}(\tau, x) A_Z^{\text{lo}}(x) A_Z^{\text{hi}}(x) = 0,$$

$$\text{Projected values: } \sum_x (w_{S_1} S_1 + w_\Delta \Delta)(x) + \sum_{k,i} (w_e e)(k, i) = \langle \lambda, y \rangle.$$

Base-16 digits: $Z = a_1, a_2$, and also u_2 for Falcon-1024, with $b \in \{\text{lo}, \text{hi}\}.$

$P \rightarrow V :$ $v_Z^b = A_Z^b(r^{(1)}), \quad v_{S_1} = S_1(r_{\Pi}), \quad v_e = e(r_{\Pi, e}), \quad v_\Delta = \Delta(r_{\Pi}). \quad V \rightarrow P : \gamma^{(2)}.$

3. Stage two: quartic range checks

Degree 5, equality-factored

$$\text{Base-16 digits: } \sum_x \text{eq}(r^{(1)}, x) Q_{16}^b(P_Z(x)) = v_Z^b,$$

$$\text{Base-8 digits: } \sum_x \text{eq}(r^{(1)}, x) Q_8(P_Z(x)) = 0.$$

Base-8 digits: a_0, u_0, u_1 , and also u_2 for Falcon-512.

$P \rightarrow V :$ $v_Z^P = P_Z(r^{(2)}) \quad (Z = a_0, a_1, a_2, u_0, u_1, u_2). \quad V \rightarrow P : \gamma^{(3)}.$

4. Stage three: checks on the committed polynomials

Degree 3

$$\text{Six digit checks: } \sum_x \text{eq}(r^{(2)}, x) Z(x)(Z(x) + 1) = v_Z^P,$$

$$\text{Signs: } \sum_x \text{eq}(r^{(2)}, x) (\sigma(x)^2 - 1) = 0.$$

Norm and encoding length:

$$\sum_i \text{eq}(r_{\text{sig}}^{(2)}, i) \left(\sum_j (S_1^2 + M^2)(j, i) + \sum_k e(k, i)^2 \right) = \lfloor \beta^2 \rfloor \sum_{i < N_{\text{sig}}} \text{eq}(r_{\text{sig}}^{(2)}, i),$$

$$\sum_i \text{eq}(r_{\text{sig}}^{(2)}, i) \left(\sum_j (a_2(j, i) + 8) + (u_0 + 8u_1 + 64u_2)(i) \right) = L - \kappa.$$

Ring identity and evaluation claims from stage one:

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

$$\sum_x \text{eq}(r_{\Pi}, x) S_1(x) = v_{S_1}, \quad \sum_{k,i} \text{eq}(r_{\Pi, e}, (k, i)) e(k, i) = v_e.$$

Public $C(r_{\Pi}), w_{\text{ring}}$: Section 4.5.

5. Open and accept

$P \rightarrow V :$ evaluations of $S_1, a_0, a_1, a_2, \sigma$ at $r^{(3)}$, e at $r_e^{(3)}$, and u_0, u_1, u_2 at $r_{\text{sig}}^{(3)}$, with one batched Akita opening.

Accept iff both projection predicates, all three sum-checks, and the batched opening verify.

Figure 3: The interactive Aerie protocol: three sum-check stages and one batched opening.

Table 1: The four Akita groups, with $(5n + 7)2^{d_{\text{sig}}}$ field elements in total before PCS padding. The point $r_e^{(3)}$ consists of the last two coefficient coordinates and all signature coordinates of $r^{(3)}$.

Group	Variables	Width	Opening point
$[S_1]$	d	14	$r^{(3)}$
$[e]$	$2 + d_{\text{sig}}$	14	$r_e^{(3)}$
$[a_0, a_1, a_2, \sigma]$	d	4	$r^{(3)}$
$[u_0, u_1, u_2]$	d_{sig}	4	$r_{\text{sig}}^{(3)}$

Fiat–Shamir derives each challenge from the preceding prover messages and a digest binding the ordered records (hpk_i, m_i, r_i) , their count, and salt mode. Before selected challenges, the prover sends a work nonce whose hash must have a prescribed number of leading zero bits, increasing the cost of challenge search. [Section C.1](#) specifies the transcript, sampling, and work difficulties.

4.1 Commitments and Polynomial Domains

[Table 1](#) gives the commitment groups on their original domains and the signed bit widths used to choose commitment parameters. Width w accommodates honest integers in $[-2^{w-1}, 2^{w-1} - 1]$ but does not constrain a malicious prover.

The polynomial S_1 represents the first Falcon component and σ the coefficient signs. The polynomial e represents four auxiliary integers per signature whose squares sum to the gap in Falcon’s norm bound. The polynomials a_0, a_1, a_2 encode the magnitude of the second component, while u_0, u_1, u_2 encode the number of unused unary bits. Their affine combinations are

$$M = a_0 + 8a_1 + 128a_2 + 1092, \quad u = u_0 + 8u_1 + 64u_2 + \kappa.$$

Here $a_0, u_0, u_1 \in [-4, 3]$ and $a_1, a_2 \in [-8, 7]$. At Falcon-512, $u_2 \in [-4, 3]$ and $(L, \kappa) = (392, 292)$. At Falcon-1024, $u_2 \in [-8, 7]$ and $(L, \kappa) = (696, 548)$. Thus $0 \leq M \leq 2047$ and $u \geq 0$. The virtual signature polynomial is $S_2 = \sigma M$ on the Boolean domain, and the virtual residual is $\Delta = S_1 + h \otimes S_2 - C$, where $\otimes$ denotes negacyclic multiplication within each signature. Honest padded rows use $S_1 = e = 0$, $(a_0, a_1, a_2) = (-4, -8, -8)$, and $\sigma = 1$, giving $M = S_2 = 0$. Their digits u_0, u_1, u_2 encode $u = L$ in the same balanced digit ranges. Together with the zero public polynomials, this gives $\Delta = 0$ and satisfies every padded-row identity.

The width-14 groups accommodate the honest S_1 coefficients in $[-6144, 6144]$ and the slack bound 8191 of [Lemma 2.1](#). We use the mixed-domain batching convention of [Section 2.4](#): each stage has d rounds, and an identity on v variables uses the last v challenges. The independent identities of stage s are combined with coefficients $\gamma^{(s)}$.

4.2 Commit and Project

The prover sends the four commitments, and the verifier replies with an independent uniform 256-bit projection seed. The seed determines the twenty independent matrices: four norm projection sequences, each with four matrices of width 43, one shared final matrix of width 156, and three residual projection matrices of width 31. The norm projection matrices have entries that take the values $-1, 0, 1$ with probabilities $1/4, 1/2, 1/4$, and residual entries are uniform on $\{-12, \dots, 12\}$. Each matrix combines the values from the preceding step over the next index block, using the same coefficients for every remaining input block ([Section 2.3](#)). [Section A.1](#) fixes the number of possible indices in each block. The norm map acts on a virtual input that places $S_1(\cdot, i)$ in

the first n positions of a $2n$ -entry row, the four entries of $e(\cdot, i)$ in the next four, and zeros elsewhere. The slack commitment keeps its four-entry domain.

The prover sends the 156 norm outputs $y_{\text{norm}} = \Pi_{\text{norm}}(S_1 \| e)$ and the 31 residual outputs $y_{\text{res}} = \Pi_{\text{res}}(\Delta)$. The verifier checks $\sum_o \text{ctr}_p(y_{\text{norm}, o})^2 \leq \Theta$ and $q \mid \text{ctr}_p(y_{\text{res}, o})$ for every $o < 31$, where ctr_p takes centered integer representatives modulo p and the public norm threshold is

$$\Theta = \begin{cases} 5,092,300,183,991,442,303,232, & n = 512, \\ 10,513,136,047,130,310,946,696, & n = 1024. \end{cases} \quad (1)$$

The first test bounds the input's squared norm. The second tests residual divisibility. Their error probabilities are given in [Section 5.1](#).

After checking both predicates, the verifier sends independent uniform $\mu \in K^8$, $\tau \in K^d$, and batching coefficients for stage one $\gamma^{(1)}$. For $o = \sum_{t=0}^7 o_t 2^t < 187$, put $\lambda_o = \prod_{t=0}^7 \mu_t^{o_t}$, and split λ and $y = (y_{\text{norm}}, y_{\text{res}})$ into their first 156 and last 31 coordinates.

4.3 Stage One: Factor Products and Projection Evaluations

We use the balanced-digit range construction of Akita [[DBK⁺26](#), Section 6.1]. For each digit polynomial Z , let P_Z be the multilinear extension of $x \mapsto Z(x)(Z(x) + 1)$ on its Boolean domain, and define

$$\begin{aligned} Q_8(P) &= P(P - 2)(P - 6)(P - 12), \\ Q_{16}^{\text{lo}} &= Q_8, \\ Q_{16}^{\text{hi}}(P) &= (P - 20)(P - 30)(P - 42)(P - 56). \end{aligned}$$

For $P = z(z + 1)$, $Q_8(P) = 0$ exactly when $z \in [-4, 3]$, and $Q_{16}^{\text{lo}}(P)Q_{16}^{\text{hi}}(P) = 0$ exactly when $z \in [-8, 7]$. Define A_Z^b as the multilinear extension of $Q_{16}^b(P_Z(x))$ on Boolean inputs. Stages two and three check the resulting virtual evaluation claims.

Splitting the degree-eight base-16 test into two degree-four factors keeps stage one at degree three. For $Z \in \{a_1, a_2\}$, and for u_2 on the signature domain at Falcon-1024 using the signature coordinates τ_{sig} of τ , stage one proves the factor-product zero-check of [Figure 3](#). It produces four virtual evaluations v_Z^b at $r^{(1)}$, or six including $A_{u_2}^{\text{lo}}, A_{u_2}^{\text{hi}}$ at $r_{\text{sig}}^{(1)}$.

We also check that the prover's reported values are the projections of $S_1 \| e$ and of the virtual polynomial Δ . With the public weight polynomials $(w_{S_1}, w_e) = \Pi_{\text{norm}}^\top \lambda_{\text{norm}}$ and $w_\Delta = \Pi_{\text{res}}^\top \lambda_{\text{res}}$, the prover establishes the single degree-two identity of [Figure 3](#), whose right side is $\langle \lambda, y \rangle$. Distinct output indices give distinct monomials in μ , so a false output vector passes the batched check with probability at most $8/|K|$. The verifier evaluates the public weights by tensor contraction through the matrices. At the end of stage one the prover sends the factor evaluations and the evaluation claims $v_{S_1} = S_1(r_\Pi)$, $v_e = e(r_{\Pi, e})$, and $v_\Delta = \Delta(r_\Pi)$. The points r_Π and $r_{\Pi, e}$ apply the coordinate maps for these domains of [Section A.1](#) to the challenges of stage one. These maps let the prover contract the later projection matrices over the high signature variables before expanding the weights of the first block.

4.4 Stage Two: Quartic Range Checks

After receiving the evaluation claims at the end of stage one, the verifier sends independent uniform batching coefficients $\gamma^{(2)}$. Stage two reduces the factor claims to claims on P_Z . For each base-16 digit and $b \in \{\text{lo}, \text{hi}\}$, it proves that the equality-weighted sum of $Q_{16}^b(P_Z)$ at $r^{(1)}$ equals v_Z^b . For each base-8 digit, namely a_0, u_0, u_1 and also u_2 at Falcon-512, it proves the zero-check of $Q_8(P_Z)$ at $r^{(1)}$ or its signature coordinates. There are

eight or nine instances, all of degree at most five. Their equality factors permit the shared-equality round format of [DDB⁺26], with four elements of K per round.

During the initial coefficient rounds, the signature-domain terms are constant in the current variable. We subtract their known contributions before applying the equality-factored protocol of Section 2.4. In the signature rounds, all terms share the same equality factor. At the end, the prover sends the six values v_Z^P for $Z \in \{a_0, a_1, a_2, u_0, u_1, u_2\}$, evaluated at $r^{(2)}$ or its signature coordinates, and the verifier then sends independent uniform batching coefficients $\gamma^{(3)}$.

4.5 Stage Three: Checks on the Committed Polynomials

Stage three checks all remaining relations against the committed polynomials. Its zero-check point is $r^{(2)}$, sampled after the commitments, and its terminal point is $r^{(3)}$. We state identities on their original domains. The embedding rule of Section 2.4 normalizes them within the common d -round sum-check.

Digit Expressions and Signs. Six checks reduce the claimed values v_Z^P of the virtual polynomials P_Z to evaluations of the committed digits Z , using $\sum_x \text{eq}(r^{(2)}, x) Z(x)(Z(x) + 1)$, on the signature domain for u_0, u_1, u_2 . The sign zero-check of $\sigma^2 - 1$ forces $\sigma = \pm 1$. Together with stages one and two, these identities establish the exact digit ranges and $S_2 = \sigma M$ with $|S_2| \leq 2047$ on Boolean inputs.

Norm Equality. Since $\sigma^2 = 1$, we have $S_2^2 = M^2$ on the Boolean domain, so the norm instance uses the committed digits instead of S_2 . With weights $\text{eq}(r_{\text{sig}}^{(2)}, i)$ on the rows, it proves that $\sum_j (S_1^2 + M^2)(j, i) + \sum_k e(k, i)^2$ equals $\lfloor \beta^2 \rfloor$ on rows corresponding to actual signatures and zero on padded rows. The verifier computes the right-hand side from the binary expansion of N_{sig} in $O(d_{\text{sig}})$ field operations. The coefficient and slack terms have the same identity coefficient. The slack term ends at $r_e^{(3)}$. Once the norm projection bounds the squared norm of the input, this field identity lifts to the integer equality of $\mathcal{R}_{\text{batch}}$ and forces zero norms on padded rows.

Encoding Length. Since $a_2 + 8 = \lfloor M/128 \rfloor$ and $u \geq 0$, the rowwise identity $\sum_j (a_2(j, i) + 8) + u(i) = L$ bounds each signature's unary encoding length by L . Moving the constant κ to the right gives $L - \kappa$ in Figure 3. Padded rows satisfy the same identity with $a_2 = -8$ and $u = L$, so no signature selector is needed.

Ring Identity. After stage one the verifier computes $C(r_\Pi)$ from the public targets and sets $v_{\text{ring}} = v_\Delta - v_{S_1} + C(r_\Pi)$. Both parties form the public polynomial

$$w_{\text{ring}}(j, i) = \text{eq}(r_{\Pi, \text{sig}}, i) \sum_{j'} \text{eq}(r_{\Pi, \text{coef}}, j') \text{coef}_{j'}(h_i X^j \bmod (X^n + 1)).$$

Here $\text{coef}_{j'}(f)$ denotes the coefficient of $X^{j'}$ in f . This is the transpose of the public negacyclic multiplication map, so the ring instance proves $\sum_x w_{\text{ring}}(x) \sigma(x) M(x) = v_{\text{ring}}$. This degree-three check ends at committed digit and sign evaluations: $M(r^{(3)})$ is their affine combination. No evaluation of S_2 or quotient commitment is needed [DBK⁺26, KK26].

Stage-One Evaluation Claims. We use two degree-two evaluation reductions to move the stage-one claims v_{S_1} and v_e from r_Π and $r_{\Pi, e}$ to the points used by the norm instance. In all, stage three has twelve independent identities: six checks of the digit expressions, the sign, norm, encoding-length, and ring identities, and these two reductions. Its maximum degree is three.

4.6 Final Check and Opening

The prover sends nine evaluations: $S_1, a_0, a_1, a_2, \sigma$ at $r^{(3)}$, e at $r_e^{(3)}$, and u_0, u_1, u_2 at $r_{\text{sig}}^{(3)}$. The verifier computes the public equality weights, $w_{\text{ring}}(r^{(3)})$, and the affine value $M(r^{(3)})$, then checks the final identity

of stage three with its batching and domain-normalization factors. One batched Akita opening proves these nine evaluations for the four groups [DBK⁺26]. At the last level of the Akita opening, the prover replaces one binding block and one partial-evaluation block by two 256-bit digests. The verifier reconstructs these blocks from the explicitly sent folded response and checks their digests. Their positions in the transcript and the additional security error are specified in [Section C.1](#).

The prover sends three elements of K per round in stages one and three, and four in stage two, totaling $10d$ elements.

5 Security

Akita extraction recovers the nine committed polynomials or a Module-SIS collision. We bound the probability that these polynomials pass verification while failing $\mathcal{R}_{\text{batch}}$. The range and sign checks fix the allowed digit sets and give encodability. The norm projection bounds the squared norm of the committed input, so the field norm identities hold over the integers, and the residual projection shows that the integer Falcon residual is divisible by q . The extracted (s_1, s_2, e) then satisfies $\mathcal{R}_{\text{batch}}$, so `EncodePadded` produces signatures that Falcon verification accepts ([Lemma 3.1](#)). [Sections A](#) and [C](#) give the full projection and extraction proofs.

5.1 Structured Projection Guarantees

Use the projections of [Section 4.2](#) and supported parameters of [Section 3](#). Write $S = S_1 \| e$ and $E(S) = \|\text{ctr}_p S\|_2^2$. Seed expansion uses independent, domain-separated random oracles with unbiased sampling. [Lemma C.2](#) handles inputs chosen after seeing the seed.

Theorem 5.1 (Norm Projection Bound). For the norm projection of [Section 4.2](#), let Θ be its acceptance threshold from [Equation \(1\)](#), and define

$$\zeta = 1024^{-5} (33/32)^{1470}, \quad \varepsilon_{\text{nrm}} = 2^{-136},$$

$$E_\star = \frac{\Theta + 1}{\zeta}.$$

For every supported parameter choice and any input S fixed before the matrices are sampled,

$$\Pr[E(S) \geq E_\star \wedge \|\text{ctr}_p \Pi_{\text{nrm}}(S)\|_2^2 \leq \Theta] \leq \varepsilon_{\text{nrm}}.$$

Proof. See [Section A.2](#). □

The norm bound lets us interpret the field norm identities over the integers. Combined with the residual divisibility analysis, it gives the following guarantee for both projections.

Theorem 5.2 (Projection Guarantees for Fixed Polynomials). Fix a supported parameter choice, the public polynomials h, C derived from its statement, and the nine committed polynomials before sampling the independent matrices. Suppose their Boolean values satisfy the digit ranges, sign constraints, and rowwise encoding-length and norm identities of [Section 4](#) over $\mathbb{F}_p$. If their centered values fail $\mathcal{R}_{\text{batch}}$, then

$$\Pr[\text{both true projections pass their predicates}]$$

$$\leq \max(\varepsilon_{\text{nrm}}, \varepsilon_{\text{res}}) = 2^{-136},$$

where $\varepsilon_{\text{nrm}} = 2^{-136}$ and $\varepsilon_{\text{res}} = 3 \cdot 25^{-31} + \varepsilon_{\text{wrap}} < 2^{-142}$, with $\varepsilon_{\text{wrap}} < 2^{-437}$ for Falcon-512 and $\varepsilon_{\text{wrap}} < 2^{-153}$ for Falcon-1024. For an honest batch witness, the probability that either projection predicate rejects is less than 2^{-128} .

No width bound on S_1 or e is assumed: the norm projection supplies the bound needed to establish the norm equations over the integers. Exact rational certificates verify the numerical bounds.

Our Lean 4 formalization [dMU21] checks the 2^{-136} soundness bound of [Theorem 5.2](#) and the valid-input rejection bound $2^{-129} + 2^{-142}$ for independent matrices at every supported parameter choice. Of the hypotheses of [Theorem 5.2](#), the Lean statement assumes the two consequences that the proof uses: $|S_2| \leq 2047$ and the norm identities over $\mathbb{F}_p$. We will make the formalization publicly available.

Comparison with Tensor-Train Projections. We compare norm projections that accept an oversized input with probability at most 2^{-136} and reject a valid one with probability at most 2^{-129} . Both use the same modulus, valid-input bound and cutoff $E_\star$. Both also use the same normalized threshold: the squared-norm threshold divided by the map’s expected squared-norm multiplier. Aerie’s norm projection then outputs 156 field elements for 32K Falcon-512 signatures, against 526 for the tensor-train construction of Geng and Plançon [GP26]. [Section A.5](#) gives the setup, the comparison table and the exact rational checks.

5.2 PIOP Analysis

We show that the PIOP enforces the Falcon batch relation. The following theorem bounds two errors: rejecting a proof generated as specified (completeness) and accepting invalid witness data (soundness).

The prover fixes the nine multilinear polynomials f over $\mathbb{F}_p$ of [Table 1](#) before the verifier’s random choices. Field challenges are independent and uniform, and the matrices are sampled independently as in [Section 4.2](#).

At the end of the PIOP, the prover sends a claimed evaluation of each of the nine witness polynomials. The verifier queries each polynomial oracle at the point of the claimed evaluation and rejects if the returned value differs from the prover’s claim.

Let $w(f)$ be the integer witness obtained by representing the entries of S_1 , $S_2 = \sigma M$, and e on the actual signature rows as integers in $(-p/2, p/2)$.

Theorem 5.3 (PIOP Completeness and Soundness). Fix a supported parameter choice and a public statement x . For every valid padded signature batch, the verifier rejects a proof generated according to the protocol with probability at most $2^{-129} + \varepsilon_{\text{wrap}} < 2^{-128}$. For every prover fixing f before the projection matrices are sampled,

$$\Pr[V^f \text{ accepts} \wedge (x, w(f)) \notin \mathcal{R}_{\text{batch}}] \leq 2^{-136} + \frac{12d + 11}{|K|}. \quad (2)$$

The prover’s later messages may depend arbitrarily on all preceding challenges. The probability is over the prover’s and verifier’s randomness.

Proof. Completeness. The witness tables constructed from valid signatures satisfy every Boolean identity. The verifier therefore accepts each sum-check stage and the evaluations checked by oracle queries. Only the norm and divisibility checks can reject, with the bound proved in [Section A.4](#).

Projection checks. Fix f with an invalid witness. If its digit, sign, norm or encoding-length identities fail, an algebraic claim is false. Otherwise, [Theorem 5.2](#) shows that, except with probability at most 2^{-136} , the prover can pass the projection checks only by sending an incorrect projected value.

Algebraic checks. By [Lemma C.1](#), a false claim either causes rejection or leaves a false claim after each challenge, except with probability at most $8/|K|$ for μ , $d/|K|$ for τ , $1/|K|$ for each of the three independent batching vectors, and $3/|K|$, $5/|K|$, $3/|K|$ per round in stages one, two and three. These bounds include reuse of $r^{(1)}$ and $r^{(2)}$. A union bound gives

$$\varepsilon_{\text{alg}} \leq \frac{8 + d + 3 + (3 + 5 + 3)d}{|K|} = \frac{12d + 11}{|K|}.$$

Thus, except with probability at most ε_{alg} , a prover starting with a false claim can pass all three stages only by sending a false evaluation of a witness polynomial. The verifier detects it by querying that polynomial. $\square$

Reusing challenge points. An identity tested at $r^{(1)}$ or $r^{(2)}$ is determined by f before the preceding stage draws that point. If the identity is false, the weighted sum used to test it is a nonzero multilinear polynomial. As each coordinate is sampled, at most one value can make the remaining polynomial identically zero. The proof of [Lemma C.1](#) includes this probability in the bound for that round.

Seed expansion. The theorem assumes independently sampled matrices. For the 256-bit seed and independent expansion oracle used by Aerie, [Lemma C.2](#) instead gives $2^{-135} + \delta_X + (12d + 11)/|K|$, even when the statement and polynomials depend on earlier expansion queries. The Fiat–Shamir analysis in [Section 5.6](#) also accounts for a prover that searches for favorable challenges, for the work nonces, and for extracting the committed polynomials.

Proposition 5.4 (PIOP Costs). The tables defining the nine witness polynomials contain $(5n + 7)N^*$ base-field elements. The protocol has $3d$ sum-check rounds, followed by one query to each polynomial. Excluding these tables, the prover sends 187 elements of $\mathbb{F}_p$ and $10d + 22$ elements of K for Falcon-512, or $10d + 24$ elements of K for Falcon-1024.

Given the public coefficient tables h, C and the witness tables, the prover uses $O(N^*M(n) + nN^*)$ field operations and the verifier uses $O(M(n) + nN^*)$, in addition to its oracle queries. Here $M(n)$ bounds the cost of a length- n negacyclic product and its transpose, and projection widths are fixed as in [Section A.1](#). Public-key decoding and hashing the records to obtain h, C are additional costs.

Proof. The projection message contains $156 + 31$ base-field elements. The round messages contain $(3 + 4 + 3)d$ elements of K . The messages at the ends of the three stages contain $7 + 6 + 9$ elements for Falcon-512 and $9 + 6 + 9$ for Falcon-1024. The operation counts are derived in [Section C.2.1](#). $\square$

These counts exclude Akita commitments and openings. Excluding the oracle tables, the prover’s messages grow logarithmically with the padded batch size, while the verifier must still read the public batch.

5.3 Completeness of Aerie

Theorem 5.5 (Completeness). Every batch of valid padded Falcon signatures with supported parameters admits an honest Aerie prover. Its rejection probability satisfies

$$\begin{aligned} & \Pr[\text{honest prover rejects}] \\ & \leq 2^{-129} + \varepsilon_{\text{wrap}} + \varepsilon_{\text{Akita}} + N_{\text{nonce}} 2^{-184}. \end{aligned}$$

Here $\varepsilon_{\text{Akita}}$ is the completeness error of the selected Akita schedule, and N_{nonce} counts the challenge sites that require a nonce search. Each such search fails with probability below 2^{-184} ([Section C.1](#)). The prover uses a single projection seed without retries.

See [Section A.4](#).

5.4 Extraction and Virtual Evaluations

Aerie runs its three sum-check stages between the Akita commitments and the Akita opening. Akita’s composition theorem [[DBK⁺26](#), Theorem 10.45] covers exactly this shape: its extractor recovers the committed polynomials, which determine the virtual polynomials, or produces a short Module-SIS collision, provided the protocol between commitment and opening is round-by-round sound for fixed committed polynomials. [Lemma C.1](#) establishes this soundness for the three stages, including reuse of one stage’s final point in the next, and [Lemma C.3](#) verifies the remaining hypotheses.

Original execution	$\text{Exec}_P(\text{pp}, \xi)$
$(x, \pi, a) \leftarrow P^{\mathcal{H}}(\text{pp}, \xi)$ $b \leftarrow \text{Verify}^{\mathcal{H}}(\text{pp}, x, \pi)$ return $\omega := (x, \pi, a, b)$	
Knowledge-extraction game	$\text{KS}_{P, \mathcal{E}}$
$\mathcal{H} \leftarrow \text{Definition 5.6}; \text{pp} \leftarrow \text{Setup}^{H_{\text{setup}}}$ $(\omega, w) \leftarrow \mathcal{E}^{P, \mathcal{H}}(\text{pp}, \xi) \quad \text{! as in } \text{Exec}_P$ return $\text{KErr}_{P, \mathcal{E}}$	

Figure 4: The knowledge-extraction game (Definition 5.7).

5.5 Digit Ranges and Encodability

The range and sign checks enforce the allowed digit sets in Section 4.1, so $M \in [0, 2047]$, $\lfloor M/128 \rfloor = a_2 + 8$, and $|S_2| = M$ on Boolean inputs. The number of unused unary bits satisfies $0 \leq u \leq 511$ for Falcon-512 and $0 \leq u \leq 1023$ for Falcon-1024. Thus the field identity $\sum_j (a_2(j, i) + 8) + u(i) = L$ lifts to the integers: both sides are nonnegative and the left side is at most $15n + 1023 < p$. This establishes encodability independently of either projection. Establishing the norm equations over the integers still requires the norm projection.

5.6 Knowledge Soundness in the Classical ROM

Knowledge soundness asks that an extractor recover a valid witness from any prover whose proof verifies (Figure 4). Unforgeability asks that no aggregate proof cover a fresh message under an honest key (Figure 5). Both games use the following oracles.

Definition 5.6 (Oracle Model). We work in the classical ROM with the independent oracle families $\mathcal{H} = (H_{\text{tr}}, H_{\text{exp}}, G, H_{\text{cmp}}, H_{\text{pk}}, H_{\text{dig}}, H_{\text{setup}})$ for transcript challenges, projection expansion, signature hashing, prefix compression, public-key hashing, the terminal-block digests and setup. The setup oracle H_{setup} generates the shared matrix coefficients by exact rejection sampling (Section C.1). Section C.1 specifies what each oracle takes as input, how its answers are sampled, and which hash outputs must begin with zero bits.

Definition 5.7 (Knowledge-Extraction Game). Fix a supported parameter choice or the supported family, a prover P , auxiliary input ξ and an extractor $\mathcal{E}$ with rewinding access to P . In Figure 4, $\mathcal{E}$ returns $w \in \mathcal{W} \cup \{\perp\}$ and the original execution $\omega = (x, \pi, a, b)$, where $\mathcal{W}$ is the witness space and a is arbitrary auxiliary output. $\mathcal{E}$ must preserve Exec_P on the same setup, oracles and prover random choices, including the verifier’s decision b . During rewinding, $\mathcal{E}$ resamples transcript challenges from H_{tr} , keeps setup and projection expansion fixed and forwards signature-hash queries to G . It records the queries to $H_{\text{cmp}}, H_{\text{pk}}, H_{\text{dig}}$ and their first preimages. The game returns

$$\text{KErr}_{P, \mathcal{E}} := \mathbf{1}[b = 1 \wedge (w = \perp \vee (x, w) \notin \mathcal{R}_{\text{batch}})]. \quad (3)$$

The knowledge error is $\Pr[\text{KErr}_{P, \mathcal{E}} = 1]$ over setup, oracle, prover and extractor randomness. The statement x is chosen by P , including the parameter choice when the family is fixed. The setup accommodates that family (Section C.1). The auxiliary input ξ may be sampled if it is independent of pp and $\mathcal{H}$.

Query Bounds. Let Q bound all hash evaluations by the original prover, including abandoned attempts, nonce searches, rejected candidates and compression. Q also counts every sparse-coordinate stream the prover queries. Let R_{f} and R_{coord} bound the field challenge families (including the projection seed) and sparse-coordinate streams queried by the verifier but not the prover.

All bounds hold separately for every execution, including reruns. Unless restricted to the prover or verifier, counts below include both. As in Akita, the analysis lets the verifier read every challenge address of the output transcript before it decides; an implementation may stop at the first failed check. Queries made only during rewinding contribute to running time, not to the original execution's error bound (Equation (18)).

Definition 5.8 (Hash-Binding Errors). Let Q_{pk} bound distinct public-key-hash queries and V_{pk} bound the number of key-hash fields in canonical public-records digest queries for supported parameters. Define

$$\varepsilon_{\text{pk}} := \frac{\binom{Q_{\text{pk}}}{2} + Q_{\text{pk}} V_{\text{pk}}}{2^{512}}. \quad (4)$$

This term covers a key-hash collision and a key hash used in a record before its key is queried.

Let Q_c bound compression queries, ℓ_{cmp} the compression output length, and V_c the number of compressed values in the inputs of all queries. Define

$$\varepsilon_{\text{bind}} := \frac{\binom{Q_c}{2} + Q_c V_c}{2^{\ell_{\text{cmp}}}}, \quad (5)$$

which covers a compression collision and a compressed value used in an input before its preimage is queried. Let Q_{dig} and Q_{FS} bound terminal-digest and transcript-oracle queries, respectively. Transcript queries include work-nonce and sparse-coordinate stream inputs. Define

$$\varepsilon_{\text{dig}} := \frac{\binom{Q_{\text{dig}}}{2} + 2(Q_{\text{FS}} + Q_c)Q_{\text{dig}}}{2^{256}}. \quad (6)$$

This term covers a collision between terminal digests and a digest used in a challenge input or compressed context before its preimage is queried. The three terms are the per-type errors of Akita's lemma on hashed transcript entries [DBK⁺26, Lemma 10.47], which counts the queries of the verifier on its output.

For the extraction cost, let V_{ver} bound the additional hash evaluations by the expansion algorithms and verifier. The simulated prover makes at most $Q_{\text{wrap}} := Q + V_{\text{ver}}$ queries per run.

Theorem 5.9 (Classical-ROM Knowledge Soundness). For Falcon-512 and Falcon-1024 at every padded batch $N^* = 2^{d_{\text{sig}}}$ in the supported range of Section 3, that is, $N^* = 8, \dots, 2^{16}$ and $N^* = 4, \dots, 2^{16}$ respectively, use the selected four-group Akita schedules, the transcript of Section C.1, and the work difficulties of Equation (17). For every prover P in Definition 5.7, there is an extractor of expected cost T_E that preserves the original execution and has $\Pr[\text{KErr}_{P,\mathcal{E}} = 1]$ at most

$$\begin{aligned} \varepsilon_{\text{ext}}^{\text{ROM}} &:= (Q + R_f + R_{\text{coord}})2^{-128} + \delta_X + \delta_{\text{setup}} \\ &\quad + \varepsilon_{\text{bind}} + \varepsilon_{\text{pk}} + \varepsilon_{\text{dig}} + \delta_{\text{samp}} \\ &\quad + \sum_I \text{Adv}_{\text{MSIS},I}^{\text{exp}}(T_I), \end{aligned} \quad (7)$$

where $\delta_{\text{setup}} = 0$ for the exact setup sampling and consistent oracle simulation of Section C.1, and δ_{samp} bounds the statistical distance of the complete challenge-sampling experiment from its ideal law. Here $\varepsilon_{\text{bind}}$, ε_{pk} , and ε_{dig} are the terms of Definition 5.8, δ_X bounds the failure of uniform seed density (Lemma C.2), and $\text{Adv}_{\text{MSIS},I}^{\text{exp}}(T)$ is the largest collision probability at scheduled occurrence I for an algorithm of expected cost at most T [DBK⁺26, Lemma 10.20].

The expected reduction costs T_I exceed T_E by the cost of embedding the instance and outputting the collision. The extractor runs the prover at most $(Q_{\text{wrap}} + 2)\widehat{K}$ times in expectation, where $\widehat{K}$ is the tree size of Equation (16). It satisfies $2^{21.4} < \widehat{K} < 2^{39.7}$ and $\widehat{K} \leq (nN^*)^2$ at every supported parameter choice.

If P chooses the supported parameters after seeing the setup, the Module-SIS sum ranges over all instances in the supported schedules (Section C.3). Use the largest $\widehat{K}$, R_f and R_{coord} over those schedules.

Aggregate-forgery game	AggForge_{Aerie}($\mathcal{A}$)
$(h^*, \text{sk}^*) \leftarrow \text{Falcon.KeyGen}$ $\mathcal{H} \leftarrow \text{Definition 5.6}; \text{pp} \leftarrow \text{Setup}^{H_{\text{setup}}}; Q_S \leftarrow \emptyset$ $(\mathcal{L}, \pi) \leftarrow \mathcal{A}^{\text{Sign}, \mathcal{H}}(h^*, \text{pp})$ $\text{Fresh} \leftarrow \{j : h_j = h^* \wedge m_j \notin Q_S\}$ return $\text{Verify}^{\mathcal{H}}(\text{pp}, \mathcal{L}, \pi) = 1 \wedge \text{Fresh} \neq \emptyset$	
Signing oracle	Sign(m)
$\sigma \leftarrow \text{Falcon.Sign}^G(\text{sk}^*, m)$ / FN-DSA, fresh salt per attempt $Q_S \leftarrow Q_S \cup \{m\}$ return σ	

Figure 5: The chosen-key aggregate-forgery game (Definition 5.10). $\mathcal{L}$ is an ordered list of records (pk_j, m_j, r_j) (Section 3.2), with $h_j = \text{Decode}(\text{pk}_j)$.

Proof. Section C.5 gives the details. Ideal setup and samplers cost $\delta_{\text{setup}} + \delta_{\text{samp}}$. Akita’s lemma on hashed transcript entries [DBK⁺26, Lemma 10.47] replaces every compressed prefix, key hash, and terminal digest by its preimage, at cost $\varepsilon_{\text{bind}} + \varepsilon_{\text{pk}} + \varepsilon_{\text{dig}}$. Akita’s composition theorem [DBK⁺26, Theorem 10.45], with the hypotheses of Lemma C.3 and the seed-density event of Lemma C.2, bounds the remaining failure by δ_X , the query term, and the Module-SIS term, and bounds the number of prover runs. $\square$

The query term includes the addresses read by the final verifier. Over all supported parameter choices $R_f + R_{\text{coord}} \leq 1207$, so at $Q = 0$ this term is at most $1207 \cdot 2^{-128} \approx 2^{-117.8}$. Concrete security also depends on the remaining error terms and the Module-SIS advantages at costs T_I . The factor $Q_{\text{wrap}} + 2$ is the expected run count of the Fiat–Shamir extractor [AFK21], and $\tilde{K}$ bounds the number of transcripts that the Akita extractor combines. The lattice estimates used to choose Akita’s parameters exclude this factor $(Q_{\text{wrap}} + 2)\tilde{K}$.

5.7 Aggregate Unforgeability

Definition 5.10 (Chosen-Key Aggregate Forgery). The game $\text{AggForge}_{\text{Aerie}}(\mathcal{A})$ is that of Figure 5. The signer is FN-DSA (Section 3): each attempt draws a fresh uniform salt and fresh sampler randomness, and only the accepted attempt is returned. All public keys in $\mathcal{L}$ other than h^* may be chosen adversarially, without proofs of possession. The parameter set and hash-to-point interface are those of the signing game, and the transcript binds the complete ordered record list. This is chosen-key aggregate unforgeability [BGLS03].

Let Q_S bound signing queries and Q_G distinct signature-hash streams, including partial requests, the streams needed to verify the final list, and the targets that the simulator assigns before it returns a records digest (Section C.6). The reduction obtains preimage–target pairs from the Falcon challenger before the game starts and reuses them throughout extraction, without further signing queries [AAB⁺24, Lemma C.2 and Remark I.1]. The challenger runs the same FN-DSA signer, so a presampled pair has the honest signer’s joint law of target and preimage. Its target law is not uniform in general (Lemma C.4), and the reduction does not need it to be.

Let $\bar{a}_{\text{sk}}$ be the probability that one attempt at a uniform target passes the norm and encoding checks, conditional on the signing key sk . Counting every attempt, rejected ones included, Lemma C.6 bounds the error in the original execution by

$$\delta_{\text{salt}} \leq 2^{-320} \mathbb{E}_{\text{sk}} \left[\frac{2Q_S^2}{\bar{a}_{\text{sk}}^2} + \frac{2Q_S Q_G}{\bar{a}_{\text{sk}}} \right]. \quad (8)$$

The expectation is over key generation. This accounts for colliding attempt salts, an attempt on an input hashed before it is signed, and a later hash of a rejected attempt’s input.

Theorem 5.11 (Chosen-Key Aggregate Unforgeability). Work in the model of [Theorem 5.9](#), with the hash-then-sign interface of [Section C.6](#), FN-DSA signing for both the honest signer and the Falcon challenger, and the public-records digest a random oracle whose queries the reduction observes. Key generation and signing have expected polynomial cost. For every aggregate adversary $\mathcal{A}$, there is an expected-polynomial-time Falcon adversary $\mathcal{B}$ such that

$$\begin{aligned} \Pr[\text{AggForge}_{\text{Aerie}}(\mathcal{A}) = 1] \\ \leq \Pr[\text{EUF-CMA}_{\text{FN-DSA}}^+(\mathcal{B}) = 1] + \varepsilon_{\text{ext}}^{\text{ROM}} + \delta_{\text{salt}}. \end{aligned}$$

Here EUF-CMA^+ allows expected-polynomial-time adversaries.

Proof. [Section C.6](#) uses three games ([Figure 7](#)). F_0 is $\text{AggForge}_{\text{Aerie}}$. F_1 answers signing and signature-hash queries from the presampled pairs, at cost δ_{salt} ([Lemma C.6](#)). F_2 also requires the extracted witness to hold for the original record list, at cost $\varepsilon_{\text{ext}}^{\text{ROM}}$. [Lemmas C.5](#) and [C.7](#) ensure that extraction uses the original record list’s assigned hash targets. The exceptional event of [Lemma C.7](#) is included in $\varepsilon_{\text{bind}}$. `Convert` turns a successful F_2 execution into a Falcon forgery without guessing a record index. $\square$

The expected cost of $\mathcal{B}$ is T_E plus key generation and preparing the signing data. Both costs are also added to each T_I . Thus the Falcon advantage includes the same extraction factor $(Q_{\text{wrap}} + 2)\widehat{K}$ as the Module-SIS advantages.

Common Salts. The verification relation also permits equal salts. Security then needs a different simulation, because a caller may know the salt and query its hash before asking for a signature. [Section C.7](#) gives a conditional classical-ROM reduction for caller-chosen salts under key-prefixed framing. Different messages may share a salt under one key.

FN-DSA would replace the common salt after a failed attempt, so the common-salt signer retries the sampler at the common salt’s target, as Falcon v1.2 does [[FHK⁺20](#), Algorithm 10]. It must never sample twice for one complete key–salt–message input, so it either caches its output or derives its sampler randomness from a secret key and that input, which needs no state ([Corollary C.11](#)). The reduction assumes that this fixed-target signer is unforgeable ([Assumption C.8](#)). Fouque et al. prove unforgeability only for FN-DSA’s fresh salts [[FGdG⁺26](#)], and we know of no attack on fixed-target retries. The partition argument of [[AAB⁺24](#), Lemma C.3] incurs a factor $\rho_S \leq e(Q_S + 1)$ on the Falcon advantage, simulation distance and extraction error ([Remark C.10](#)).

6 Implementation and Measurement

6.1 Implementation

Our Rust implementation [[CDZT26](#)] uses Akita for commitments and the batched opening [[Lay26](#)], in the variant whose final step replaces its last block by two digests [[DBK⁺26](#), Section 8.2]. The verifier computes Falcon’s hash-to-point function from the public salts and messages [[FHK⁺20](#)]. Fiat–Shamir binds the public statement and canonically encoded prover messages in the order of [Figure 3](#). Projection expansion, record-leaf hashing, and instance hashing use distinct cSHAKE256 customizations [[KjCP16](#)], separate from Falcon’s SHAKE256 calls. Falcon’s key hash and signature hash keep their prescribed framing ([Section C.1](#)). Rejection sampling gives uniform challenges in K , as required by [Section C.1](#).

We compress the projection outputs with Rice coding, encoding centered norm outputs and centered residual outputs divided by q .

Table 2: Falcon-512 prover time, peak prover memory, verifier time, and complete proof size from 1K to 64K signatures. Times are medians of seven runs. Common-salt sizes are estimated from the measured one-salt sizes by removing $40(N_{\text{sig}} - 1)$ salt bytes.

Batch	Prover		Verifier Time (ms)		Proof Size (KiB)	
	Time (s)	Peak Memory (MiB)	Single Thread	Multithread	Common Salt	One Salt per Signature
1K	0.13	198	5.5	4.6	61.8	101.8
2K	0.17	280	7.5	5.2	62.8	142.8
4K	0.21	418	11.1	6.0	63.0	222.9
8K	0.26	658	16.9	6.7	64.7	384.7
16K	0.39	1,126	29.0	8.2	64.9	704.8
32K	0.65	2,113	53.6	11.6	66.4	1,346.4
64K	1.07	4,127	101.2	15.8	66.7	2,626.6

Table 3: Falcon-1024 prover time, peak prover memory, verifier time, and complete proof size from 1K to 64K signatures. Times are medians of seven runs. Common-salt sizes are estimated as in Table 2.

Batch	Prover		Verifier Time (ms)		Proof Size (KiB)	
	Time (s)	Peak Memory (MiB)	Single Thread	Multithread	Common Salt	One Salt per Signature
1K	0.17	286	7.2	5.2	62.9	102.8
2K	0.20	411	10.2	5.6	63.1	143.0
4K	0.26	658	16.1	6.3	64.7	224.7
8K	0.38	1,126	26.9	7.9	64.9	384.9
16K	0.63	2,106	49.4	11.1	66.4	706.3
32K	1.07	4,108	92.1	15.3	66.7	1,346.7
64K	1.86	7,947	177.0	23.8	67.0	2,627.0

6.2 Evaluation

We measure honestly generated Falcon-512 and Falcon-1024 batches; key generation and signing are excluded. Except for the Orthus comparison below, measurements use an Apple M4 Max (12 performance and 4 efficiency cores, 64 GiB) running macOS 27.0.1, Rust 1.95.0, the code of Aerie release v0.2.0 [CDZT26], and Akita revision e023b6a4.

Tables 2 and 3 cover both variants at $N_{\text{sig}} = 2^{10}, \dots, 2^{16}$ using the projection parameters of Section 4.2. Timing and memory measurements use one salt per signature. Parallel runs use 16 worker threads. Timings are medians of seven runs. Peak memory is the benchmark process’s maximum resident set size, including the input batch and commitment setup. Sizes include salts and headers but exclude verifier-supplied keys and messages. One KiB is 1024 bytes.

Timing Breakdown. Prover time covers input verification, public-record binding, witness derivation, the four commitments, projections, all three sum-check stages, and the batched Akita opening; it excludes generating the commitment key. For 32K Falcon-512 signatures, the batched opening and the third sum-check stage take 50% of the prover time, and the prover’s own native check of its input signatures takes 7%. Verification starts from the encoded aggregate, ordered public keys, and messages and ends with acceptance or rejection. On one thread, hashing the public keys and evaluating the hash-to-point targets at the stage point take 70% of the

verification time, and checking the Akita opening takes 14%. The rest binds the public records, takes one inner product with each public key, and replays the sum-check stages and projection checks.

Proof Size. Common-salt estimates remove $40(N_{\text{sig}} - 1)$ salt bytes from the measured proofs. An actual common-salt proof has different targets, so its Rice-coded projection outputs may differ by a few bytes.

A salt known before signing requires the conditional common-salt theorem of [Section C.7](#), even if each key signs only once. That theorem uses key-prefixed framing and the fixed-target signer, assumes that this signer is unforgeable ([Assumption C.8](#)), and requires cached or deterministic responses on repeated complete signing inputs. Different messages may share a salt. The size estimate does not establish compliance with this policy.

At $d = 24$, the prover sends 240 elements of K across the three sum-check stages, or 3,840 bytes, excluding the evaluation claims sent at the end of each stage and the Akita opening. This applies to Falcon-512 at 32K signatures and Falcon-1024 at 16K signatures.

6.3 Comparison with Native Falcon, leanVM, and Orthus

Native Falcon. We benchmarked two native Falcon implementations on one M4 Max core: version 0.3.0 of the fn-dsa Rust library [[Por25](#)], whose signature format and message framing Aerie supports, and the aarch64 implementation of Nguyen and Gaj [[NG23](#)] as packaged in PQClean [[KSSW22](#)], which uses NEON vector instructions. Both use the hardware floating-point unit for signing. Each run takes the median of five rounds of 8,192 signatures under 64 keys on 32-byte messages, and verification includes decoding the public key and signature.

PQClean is faster at both operations: Falcon-512 verification takes 15.8 microseconds against 16.4 for fn-dsa, and signing 96 against 106. PQClean implements the round-3 Falcon framing, which hashes the salt and message without the public key, so it does slightly less work per signature than the framing Aerie proves. We compare against PQClean. Aerie’s one-thread figures are medians of seven runs at the revision of [Table 2](#).

[Table 4](#) collects these figures with the multithreaded ones of [Tables 2](#) and [3](#). On one core, Aerie proves 32K Falcon-512 signatures in 4.3 s, 8 times the cost of verifying them natively and 1.4 times the cost of signing them.

Aerie’s verifier is 10 times faster than native verification on one core, about 1.6 microseconds per signature, and 3.7 times faster with 16 threads. Like native verification, it still computes every hash-to-point target, and it takes one inner product with each public key, so its cost keeps a term linear in the batch size.

Memory and Size. The prover’s input, namely public keys, padded signatures, and 32-byte messages, occupies 50 MiB for 32K Falcon-512 signatures. Peak prover memory is about 42 times larger. Sum-check provers keep tables indexed by the 2^{24} points of the Boolean hypercube, and one such table of elements of K occupies 256 MiB. The aggregate with one salt per signature is 16 times smaller than the padded signatures it replaces, and salts are 40 of its 42.1 bytes per signature.

leanVM. We ran leanVM [[lea26b](#)] at revision 127510d1 on the same M4 Max. Its aggregation benchmark proves 900 XMSS signatures in 0.51 s with all cores and 4.7 s on one thread, so Aerie’s prover handles 29 times as many signatures per second with all cores and 39 times as many on one thread. leanVM’s proofs verify faster, and its verifier is single-threaded. A recursive leanVM proof with 2 children verifies in 5.9 ms and one with 16 children in 6.5 ms. Aerie verifies 32K signatures in 54 ms on one thread and 11.6 ms with 16 threads, 9 and 1.8 times longer.

A leanVM proof of more than 900 signatures merges proofs recursively, at most 16 per step. [Table 5](#) models one proof of 33,300 signatures, the smallest multiple of 900 above 32K: 37 leaf proofs merged in groups of a fixed size until one proof remains. Leaves and merges use the code rate of leanVM’s recursion benchmark, at which a leaf takes 0.64 s. The latency model gives every node of the tree its own M4 Max, so

Table 4: Aerie against native Falcon on the Apple M4 Max. Native figures are one-core timings of PQClean’s aarch64 implementation, scaled to the batch; Aerie’s multithreaded figures and sizes repeat [Tables 2 and 3](#). Input records are public keys, padded signatures, and 32-byte messages.

	Falcon-512, 32K	Falcon-1024, 16K
Native signing, one core (s)	3.2	3.1
Native verification, one core (ms)	519	505
Native verification, 16 threads (ms)	42.9	43.7
Aerie proving, one thread (s)	4.3	4.3
Aerie proving, 16 threads (s)	0.65	0.63
Aerie verification, one thread (ms)	54	49
Aerie verification, 16 threads (ms)	11.6	11.1
Input records (MiB)	50	49
Aerie peak prover memory (MiB)	2,113	2,106
Padded signatures (KiB)	21,312	20,480
Aggregate, one salt per signature (KiB)	1,346	706
Aggregate, common salt (KiB)	66	66

Table 5: One leanVM proof of 33,300 XMSS signatures (37 leaves of 900) on Apple M4 Max machines, by the number of proofs combined per merge. Latency assumes one machine per tree node and no network time. Total work sums all nodes. Aerie proves 32K Falcon-512 signatures in 0.65 s on one machine. For a merge combining fewer proofs, we use the timing for the next larger measured group size.

Proofs per merge	Merge levels	Latency (s)	Total work (M4 Max s)
2	6	2.20	34.2
3	4	2.26	32.6
4	3	2.03	29.9
5	3	2.23	30.1
8	2	2.21	28.8
16	2	2.95	28.6

each level costs its slowest merge. It ignores the time to send proofs between machines. The fastest group size, 4, gives 2.0 s, 3.1 times Aerie’s time for 32K signatures on one machine.

The total-work model adds the time of every node: about 29 M4 Max seconds, 44 times Aerie’s. In exchange, leanVM can spread this work over many provers and merge proofs made by others, which Aerie cannot. The signature schemes also differ: leanVM’s XMSS variant is stateful and uses a hash function chosen for proving.

Orthus. Orthus reports its Falcon-512 timings on one core of an Intel Core i7-11850H [BBL⁺26]; its implementation requires AVX-512 and is single-threaded. We therefore ran both systems on one pinned core of the same AVX-512 host, a dedicated AMD Ryzen 9 9900X server running Ubuntu 26.04. It is a Zen 5 part, like the Ryzen 9 9950X used to evaluate Akita [DBK⁺26]. Orthus revision c1ae0e61 [laz26] was built by Ubuntu clang 21.1.8, and Aerie at the revision of [Table 2](#) by Rust 1.95.0, both for the native instruction set. Each run started once the machine was idle.

[Table 6](#) reports medians of three runs. The two workloads differ in Orthus’s favor. Orthus proves its restricted statement with one common salt and hash target, and its reported prover time excludes public-key

Table 6: Aerie and Orthus on one core of the same AVX-512 host (AMD Ryzen 9 9900X) for Falcon-512. Times and memory are medians of three runs; Aerie uses one salt per signature. Proof sizes compare Aerie’s common-salt aggregate with Orthus, which also uses one common salt.

Batch	Prover Time (s)		Verifier Time (ms)		Peak Memory (MiB)		Proof Size (KiB)	
	Aerie	Orthus	Aerie	Orthus	Aerie	Orthus	Aerie	Orthus
8K	1.62	2.66	14.9	772	450	1,507	64.7	109.9
16K	2.94	4.54	23.5	1,049	817	2,521	64.9	112.7
32K	5.63	8.24	41.1	1,574	1,596	4,816	66.4	112.7
64K	10.79	16.10	74.0	2,410	3,013	9,401	66.7	114.7

preprocessing (0.03 to 0.18 s here). Aerie proves the full relation with one salt per signature, checks every signature natively in the prover, and computes every hash-to-point target in the verifier. On this host Aerie proves 1.5 to 1.6 times faster, verifies 33 to 52 times faster, and uses about a third of the peak memory.

A common salt changes Aerie’s times by at most 1%. Orthus requires AVX-512 and Aerie does not. Replacing Aerie’s AVX-512 kernels with AVX2 ones, with Akita’s AVX-512 number-theoretic transform left in place, changes proving time by at most 2% and slows verification by up to 82%. Orthus runs faster on this host than reported, 8.2 s against 16.8 s to prove 32K signatures, so we compare only ratios measured on one host.

7 Conclusion and Future Work

Building on public-salt Falcon aggregation [AAB⁺24], Aerie proves the complete verification relation, including padded encodability. Its tailored PIOP reuses magnitude digits for arithmetic and encoding checks, avoids quotient commitments, and ends in one batched Akita opening. These choices illustrate the efficiency of designing a proof around a specific verification algorithm.

Private Salts and Hash Proofs. Private salts would save $40N_{\text{sig}}$ transmitted bytes but require a zero-knowledge proof linking each message and hidden salt to its hash-to-point target. LaBinius, a lattice-based PCS for binary polynomials, offers a route to proving this SHAKE-based computation [OS26].

Merging Aggregates. An Aerie prover holds every signature of its batch, and two aggregates cannot be combined. Deployments that collect signatures across a network, as in leanVM’s recursive design, would need either a way to merge Aerie proofs or a recursive verifier for them, and the second reintroduces the heuristic step that public salts avoid.

Accelerated Proving. The batched Akita opening and the third sum-check stage account for about half of the prover’s time (Section 6). Both are dominated by independent field and ring operations over large tables, which suit a GPU. Apple’s unified memory would let a Metal prover share those tables with the CPU without copies.

Acknowledgments

Quang Dao thanks LayerZero Labs for providing equipment and for funding AI model subscriptions and API usage during this work. We thank LayerZero Labs, and in particular the other Akita maintainers Omid Bodaghi, Amirhossein Khajepour, and Giuseppe Vitto, for ongoing discussions about Aerie and for their support in adapting Akita to its needs. They designed, implemented, or reviewed several Akita features that

Aerie relies on, including committing polynomials of different sizes and coefficient bounds in a single batch, a prover that serves several parameter sets from one setup, and the Fiat–Shamir proof format that Aerie’s opening proof uses. We also thank Michael Zhu for contributions to the Aerie implementation.

AI Assistance

AI models from OpenAI (GPT Sol and Astra) and Anthropic (Claude Opus and Fable) are used extensively in the implementation and benchmarking. We also used these models to help brainstorm the protocol, derive the projection theorems and the security proofs, and edit the manuscript. The authors take full responsibility for the accuracy and correctness of the paper and for any issues in the implementation.

Disclosures

Justin Thaler is a Research Partner at a16z crypto and is an investor in various blockchain-based platforms, as well as in the crypto ecosystem more broadly (for general a16z disclosures, see <https://www.a16z.com/disclosures/>).

References

- [AAB⁺24] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. Aggregating falcon signatures with LaBRADOR. Cryptology ePrint Archive, Paper 2024/311, 2024.
- [AFK21] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat–shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Paper 2021/1377, 2021.
- [BBG⁺26] Dan Boneh, John Bradley, Sergey Gorbunov, Riad S. Wahby, Hoeteck Wee, Christopher A. Wood, and Zhenfei Zhang. BLS signatures. Internet-Draft draft-irtf-cfrg-bls-signature-07, 2026. Work in progress, July 2026.
- [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. Cryptology ePrint Archive, Paper 2026/398, 2026. Revision dated June 30, 2026.
- [BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. Cryptology ePrint Archive, Paper 2012/095, 2012. STOC 2013.
- [BCG24] Annalisa Barbara, Alessandro Chiesa, and Ziyi Guan. Relativized succinct arguments in the ROM do not exist. Cryptology ePrint Archive, Paper 2024/728, 2024. TCC 2025.
- [BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. Cryptology ePrint Archive, Paper 2020/499, 2020. TCC 2020.
- [BDH11] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS: A practical forward-secure signature scheme based on minimal security assumptions. In *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2011.

- [BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
- [BK20] Dan Boneh and Sam Kim. One-time and interactive aggregate signatures from lattices. Preprint, 2020.
- [BRW26] Benedikt Bünz, Ron D. Rothblum, and William Wang. Flock: Fast proving for batch boolean computations. arXiv preprint arXiv:2607.27491, 2026. Version dated July 29, 2026.
- [BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In *Advances in Cryptology – CRYPTO 2023*. Springer, 2023.
- [CCG⁺23] Megan Chen, Alessandro Chiesa, Tom Gur, Jack O’Connor, and Nicholas Spooner. Proof-carrying data from arithmetized random oracles. Cryptology ePrint Archive, Paper 2023/587, 2023. EUROCRYPT 2023.
- [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 *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1082–1090. ACM, 2019.
- [CDZT26] Alberto Centelles, Quang Dao, Michael Zhu, and Justin Thaler. Aerie implementation. GitHub repository, 2026. Release v0.2.0.
- [CGSY23] Alessandro Chiesa, Ziyi Guan, Shahar Samocha, and Eylon Yogev. Security bounds for proof-carrying data from straightline extractors. Cryptology ePrint Archive, Paper 2023/1646, 2023. TCC 2024.
- [COS19] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. Cryptology ePrint Archive, Paper 2019/1076, 2019.
- [DAH25] Justin Drake, Ryan Sean Adams, and David Hoffman. Ethereum’s quantum plan before Q-Day with justin drake. Bankless interview transcript hosted by ethereum.org, July 2025.
- [DBK⁺26] Quang Dao, Omid Bodaghi, Amirhossein Khajepour, Giuseppe Vitto, Mohammadtaghi Badakhshan, Markos Georgiades, Fengrun Liu, Jiapeng Zhang, and Justin Thaler. Akita: A high-performance lattice-based polynomial commitment scheme. Cryptology ePrint Archive, Paper 2026/1983, 2026. Revision of October 15, 2026.
- [DDB⁺26] Quang Dao, Zachary DeStefano, Suyash Bagad, Yuval Domb, and Justin Thaler. Speeding up sum-check proving (extended version). Cryptology ePrint Archive, Paper 2026/587, 2026.
- [dMU21] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021.
- [ELS⁺26] Muhammed F. Esgin, Yini Lin, Amin Sakzad, Ron Steinfeld, and Markku-Juhani O. Saarinen. Lemur: Scalable post-quantum synchronized multi-signatures. Cryptology ePrint Archive, Paper 2026/1161, 2026.
- [Eth26a] Ethereum consensus specification contributors. Phase 0 beacon chain specification, 2026. Revision caa9c8e804efb6ab43cddc7d536e1172f5e1ed87; committee assignment and mainnet epoch preset.

- [Eth26b] Ethereum Foundation. Attestations. ethereum.org documentation, 2026. Accessed August 26, 2026.
- [FGdG⁺26] Pierre-Alain Fouque, Phillip Gajland, Hubert de Groote, Jonas Janneck, and Eike Kiltz. A closer look at Falcon. In *Advances in Cryptology – EUROCRYPT 2026*, 2026. Full version: Cryptology ePrint Archive, Paper 2024/1769.
- [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, 2020. Specification, version 1.2.
- [GHL22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, 2022.
- [GP26] Zhiyuan Geng and Maxime Plançon. Succinct shortness check under a few kilobytes via tensor train random projections. Cryptology ePrint Archive, Paper 2026/2146, 2026. September 22, 2026.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 197–206, 2008.
- [HBG⁺18] Andreas Hülsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: extended merkle signature scheme. Technical Report RFC 8391, RFC Editor, 2018.
- [Hül13] Andreas Hülsing. W-OTS+: Shorter signatures for hash-based signature schemes. In *Progress in Cryptology—AFRICACRYPT 2013*, volume 7918 of *Lecture Notes in Computer Science*, pages 173–188. Springer, 2013.
- [KjCP16] John Kelsey, Shu jen Chang, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Technical Report NIST Special Publication 800-185, National Institute of Standards and Technology, December 2016.
- [KK26] Anders Kallesøe and Hamidreza Khoshakhlagh. Grand danois: Succinct multilinear polynomial commitments over lattices. Cryptology ePrint Archive, Paper 2026/1196, 2026. Revision of August 21, 2026.
- [KLNO25] Michael Klooß, Russell W. F. Lai, Ngoc Khanh Nguyen, and Michał Osadnik. RoK and roll: Verifier-efficient random projection for $\tilde{O}(\lambda)$ -size lattice arguments (extended abstract). In *Advances in Cryptology – ASIACRYPT 2025, Part III*, volume 16247 of *LNCS*, pages 297–329. Springer, 2025.
- [KRS25] Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. How to prove false statements: Practical attacks on Fiat-Shamir. Cryptology ePrint Archive, Paper 2025/118, 2025.
- [KSSW22] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. Improving software quality in cryptography standardization projects. In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2022. PQClean, <https://github.com/PQClean/PQClean>, revision 0586a82.
- [Lay26] LayerZero Labs. Akita implementation. GitHub repository, 2026. Aerie dependency pinned at commit e023b6a46143f7b23199cceb2684192baf7b640.

- [laz26] lazer-crypto. LaBRADOR and Orthus implementation. GitHub repository, 2026. Measured at commit c1ae0e612d5accd9e7d9d1c264951c1afe949c83.
- [lea26a] Lean Consensus: 2026 planning. Research and engineering plan, 2026. Post-Quantum Cryptography and Aggregation: target of 1,000 XMSS signatures per second.
- [lea26b] leanEthereum contributors. leanVM: Benchmarks. GitHub repository, 2026. Revision 2e6fe21d605a30be50770be9c0222fe737c5a47b, September 25, 2026.
- [lea26c] leanEthereum contributors. PQ devnet 4, 2026. Revision cec07aa71fdc, accessed September 26, 2026.
- [LP26] David Lazar and Chris Peikert. Deterministic Falcon signatures, 2026. Algorand specification, version of September 24, 2026; first version 2021.
- [Nat16] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, December 2016. Call for proposals.
- [Nat24a] National Institute of Standards and Technology. FIPS 204: Module-lattice-based digital signature standard. Technical Report Federal Information Processing Standards Publication 204, U.S. Department of Commerce, 2024.
- [Nat24b] National Institute of Standards and Technology. FIPS 205: Stateless hash-based digital signature standard. Technical Report Federal Information Processing Standards Publication 205, U.S. Department of Commerce, 2024.
- [Nat26] National Institute of Standards and Technology. Post-quantum cryptography, 2026. Accessed September 26, 2026; Falcon standardization ongoing.
- [NG23] Duc Tri Nguyen and Kris Gaj. Fast Falcon signature generation and verification using ARMv8 NEON instructions. In *Progress in Cryptology – AFRICACRYPT 2023*, 2023.
- [NKS25] David Nevado, Dohoon Kim, and Miha Stopar. Lattice-based signature aggregation. Privacy Scaling Explorations and Ethereum Research, 2025.
- [OS26] Michał Osadnik and Gregor Seiler. LaBinius: Fast lattice-based binary polynomial commitment scheme. Cryptology ePrint Archive, Paper 2026/2103, 2026. September 18 manuscript.
- [Osa26] Michał Osadnik. PikkuFold: Efficient folding in a few kilobytes. Cryptology ePrint Archive, Paper 2026/1809, 2026. Revision of September 15, 2026.
- [Per25] Ray Perlner. FIPS 206 status update. 6th NIST PQC Standardization Conference, 2025. FN-DSA remains under standardization; requirements cited in this paper are provisional until FIPS 206 is final.
- [Por25] Thomas Pornin. fn-dsa: FN-DSA signature scheme in Rust. Rust crate, version 0.3.0, 2025.
- [Sol26] Solana Foundation. Alpenglow, 2026. Upgrade overview, accessed September 26, 2026.
- [Tha23] Justin Thaler. Proofs, arguments, and zero-knowledge, 2023. Lecture notes, July 2023.
- [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.
- [TS23] Toi Tomita and Junji Shikata. Compact aggregate signature from module-lattices. Cryptology ePrint Archive, Paper 2023/471, 2023.

A Structured Projection Proofs

We prove the norm bound of [Theorem 5.1](#), the guarantees of [Theorem 5.2](#), and the completeness bound of [Theorem 5.5](#). The committed polynomials are fixed before the independent matrices are sampled. Extraction and Fiat–Shamir composition are treated in [Sections C.2](#) and [5.4](#).

A.1 Projection Parameters

The norm input has $2n$ entries per row: the first n contain S_1 , the next four contain e , and the rest are zero. Its first index block contains the coefficient bits, one bit selecting the S_1 or e half of the row, and $12 - (d_{\text{coef}} + 1)$ low signature bits, giving $J_1 = 4096$ possible indices. Divide the remaining signature bits into four consecutive blocks: the first three take at most four bits each, and the last takes the remainder. These blocks have $J_2, \dots, J_5$ possible indices, respectively. A block with no bits has one possible index. Each of the first four steps combines the values from the preceding step over the next index block. The final matrix combines all 172 values from the four sequences over the last block.

The residual first block omits the bit selecting the half-row and has 2048 possible indices. Split its remaining bits equally between its last two blocks, assigning an odd extra bit to the last block. All matrices remain present even when a block has only one possible index. At the minimum padded batch size for either Falcon variant, $d_{\text{sig}} = 12 - d_{\text{coef}}$, exactly one signature bit remains after the first block. Thus the numbers of possible indices in the blocks are $(4096, 2, 1, 1, 1)$ for the norm projection and $(2048, 1, 2)$ for the residual projection, including at Falcon-1024 with $d_{\text{sig}} = 2$. The matrix widths and independence are unchanged. Increasing the batch size only increases these index counts, so the maximum numbers of matrix columns below still occur at $N^* = 2^{16}$.

Coordinate Order. The factor-product instances use the standard coefficient-then-signature order, so the challenges of stage one are $r_0^{(1)}, \dots, r_{d-1}^{(1)}$ in that order. The projection terms use a different order. Let $d_{\text{lo}} = 12 - (d_{\text{coef}} + 1)$ be the number of low signature bits in the first index block, and let $d_{\text{hi}} = d_{\text{sig}} - d_{\text{lo}}$ be the number of remaining signature bits. The projection coefficient terms bind the d_{hi} high signature bits first, then the coefficient bits, then the d_{lo} low signature bits, each group least significant bit first. In the polynomial’s canonical variable order the point is therefore

$$\begin{aligned} r_{\Pi, \text{coef}} &= (r_{d_{\text{hi}}}^{(1)}, \dots, r_{d_{\text{hi}}+d_{\text{coef}}-1}^{(1)}), \\ r_{\Pi, \text{sig}} &= (r_{d_{\text{hi}}+d_{\text{coef}}}^{(1)}, \dots, r_{d-1}^{(1)}, r_0^{(1)}, \dots, r_{d_{\text{hi}}-1}^{(1)}). \end{aligned}$$

The projection term for e is inactive during the first $a = d_{\text{coef}} - 2$ rounds. It then binds the remaining challenges in the same pattern: the d_{hi} high signature bits, then the two bits indexing the four entries of e , then the d_{lo} low signature bits. Thus

$$\begin{aligned} r_{\Pi, e} &= (r_{a+d_{\text{hi}}}^{(1)}, r_{a+d_{\text{hi}}+1}^{(1)}, \\ &\quad r_{a+d_{\text{hi}}+2}^{(1)}, \dots, r_{d-1}^{(1)}, r_a^{(1)}, \dots, r_{a+d_{\text{hi}}-1}^{(1)}). \end{aligned}$$

This point uses the domain of e and is not, in general, a suffix of r_{Π} . The order lets the prover contract the matrices of the later index blocks over the high signature bits before it expands the weights of the first block. It does not change the sum being proved.

A.2 The Norm Projection

The acceptance threshold Θ is given in [Equation \(1\)](#). The five matrices have column counts $J_1, 43J_2, 43J_3, 43J_4$, and $172J_5$. Their maxima over the supported parameter choices are

$$(4096, 688, 688, 688, 688) \text{ and } (4096, 688, 688, 688, 1376)$$

at Falcon-512 and Falcon-1024, respectively. Write these maxima as $f_0, \dots, f_4$ and set

$$m = \lfloor \sqrt{\Theta} \rfloor, \quad t_5 = 2m + 1, \\ t_i = 2^{5-i} t_5 \quad (0 \leq i < 5).$$

These constants satisfy

$$2(t_{i+1} - 1) < t_i, \quad f_i(t_i - 1) \leq (p - 1)/2 \quad (0 \leq i < 5). \quad (9)$$

These conditions separate modular reduction from estimates of squared norms over the reals.

Lemma A.1 (Large Coordinates and Exact Reduction). A fresh ternary matrix with c rows maps an array with a centered coordinate of magnitude at least t_i to an array whose every centered coordinate is less than t_{i+1} in magnitude with probability at most 2^{-c} . If every input coordinate has magnitude less than t_i , its output is exactly the unreduced integer output under Equation (9).

Proof. Select a large coordinate and its input block without inspecting the fresh matrix. In one output row, condition on all coefficients except the one multiplying this coordinate. Two consecutive choices in $\{-1, 0, 1\}$ cannot both yield centered outputs of magnitude at most $t_{i+1} - 1$: their difference would be a representative of the selected coordinate with magnitude at most $2(t_{i+1} - 1) < t_i$. The allowed choices therefore have total probability at most $1/2$. Independence of the rows gives 2^{-c} , even though the matrix is shared across input blocks. For a small input, every integer output has magnitude at most $f_i(t_i - 1) \leq (p - 1)/2$, so reduction modulo p preserves it. $\square$

A.2.1 Scalar Laplace Bound

For $G \sim N(0, 1)$ define

$$h_t(w) = \mathbb{E} |\cos(\sqrt{tw/2} G)|^{2/w}, \quad 0 < w \leq 1.$$

For a real nonzero matrix X , write X_k for its k th row and $\|X\|_F^2 = \sum_k \|X_k\|_2^2$ for its squared Frobenius norm.

Lemma A.2 (Scalar Laplace Comparison). For every $t > 0$ and nonzero real matrix X , a row a of independent ternary coefficients with probabilities $(1/4, 1/2, 1/4)$ satisfies

$$\mathbb{E} \exp \left(-t \frac{\|a^\top X\|_2^2}{\|X\|_F^2} \right) \leq \sup_{0 < w \leq 1} h_t(w). \quad (10)$$

Proof. Introduce an independent standard Gaussian vector g . The Gaussian characteristic function rewrites the left side as $\mathbb{E}_g \prod_k \cos^2(\sqrt{t/2} \langle g, X_k \rangle / \|X\|_F)$. Set $w_k = \|X_k\|_2^2 / \|X\|_F^2$ and discard zero rows. Hölder's inequality with exponents $1/w_k$ bounds this expression by $\prod_k h_t(w_k)^{w_k}$, which proves Equation (10). Correlation across the input blocks is included in X and requires no independence assumption. $\square$

Lemma A.3 (Scalar Certificate). The scalar function above satisfies

$$\sup_{0 < w \leq 1} h_3(w) \leq \alpha, \quad \alpha = 21/40. \quad (11)$$

Proof with an exact-rational certificate. We verify this scalar bound as follows. Put $x = 1/w$ and $H(x) = h_3(1/x)$. Fourier expansion of $|\cos u|^{2x}$ gives

$$H(x) = a_0(x) \left(1 + 2 \sum_{k \geq 1} \prod_{j=1}^k \frac{x - j + 1}{x + j} e^{-3k^2/x} \right), \\ a_0(x) = \frac{1}{\pi} \int_0^\pi |\cos u|^{2x} du.$$

To derive the coefficients, set

$$I_k = \int_0^{\pi/2} \cos^{2x} u \cos(2ku) du.$$

The function $\cos^{2x+1} u \sin((2k-1)u)$ vanishes at both endpoints. Integrating its derivative gives $(2x+2k)I_k = (2x-2k+2)I_{k-1}$, which proves the displayed ratios. The second derivative of $|\cos u|^{2x}$ is integrable for $x \geq 1$, so its Fourier coefficients are absolutely summable; the Gaussian expectation may therefore pass through the series.

The Wallis recurrence $a_0(x+1) = a_0(x)(2x+1)/(2x+2)$, with $a_0(k) = \binom{2k}{k}/4^k$ at integers, and the log-convexity of a_0 (by Hölder's inequality) give rational upper bounds on a_0 : shift the argument upward by 2048 steps of the recurrence and interpolate linearly between integers. Each Fourier ratio has absolute value at most one, so truncating at k_0 leaves a tail bounded by $2 \sum_{k \geq k_0} e^{-3k_0 k/x}$. For $1 \leq a \leq x \leq b$, the change of variables

$$H(x) = \sqrt{\frac{x}{3\pi}} \int_{\mathbb{R}} |\cos u|^{2x} e^{-xu^2/3} du$$

shows $H(x) \leq \sqrt{b/a} H(a)$: the integrand is nonincreasing in x , and the prefactor grows by at most $\sqrt{b/a}$.

The exact-rational certificate covers $[1, 4]$ by 40 consecutive subintervals with dyadic endpoints and verifies $(b/a)\overline{H}(a)^2 \leq (262447/500000)^2$ in each subinterval, where $\overline{H}$ is the preceding Fourier upper bound. Every exponential is enclosed by truncated Taylor series with an explicit remainder, using outward rounding to multiples of 2^{-320} , so the bound on each coefficient has the correct sign. The final subinterval reaches $b \geq 4$.

For $x \geq 4$, put $U = \sqrt{3/(2x)}G$. On $|U| < \pi/2$, the inequality $\log \cos U \leq -U^2/2$ bounds $|\cos U|^{2x}$ by $e^{-3G^2/2}$, whose expectation is $1/2$. Off this interval use the bound one and $\Pr[|G| > a] \leq \sqrt{2/\pi} e^{-a^2/2}/a$, with $a = \pi\sqrt{x/6}$. Consequently

$$H(x) \leq \frac{1}{2} + \sqrt{\frac{12}{\pi^3 x}} e^{-\pi^2 x/12} < \frac{21}{40}.$$

The last strict inequality already follows from $\pi > 3$ and $e^3 > 20$, which bound the tail at $x = 4$ by $1/60$. Thus the finite calculation and the tail cover every $w \in (0, 1]$. $\square$

For a fresh c -row matrix A , row independence and Markov's inequality then give

$$\Pr \left[\frac{\|AX\|_F^2}{\|X\|_F^2} \leq u \right] \leq F_c(u) := \min(1, \alpha^c e^{3u}), \quad u \geq 0.$$

Let Z_c have this distribution function, including its atom α^c at zero, and put $F_c(u) = 0$ for $u < 0$.

A.2.2 Conditional Comparison and Reduction Modulo p

Lemma A.4 (Conditional Norm Coupling). Suppose a fresh ternary matrix with c rows acts on a nonzero real matrix X measurable from the previous history. The ratio of squared norms $V = \|AX\|_F^2 / \|X\|_F^2$ can be coupled to Z_c so that $V \geq Z_c$, while Z_c is independent of that history. Successive fresh matrices admit independent comparison variables.

Proof. Condition on the history. The Laplace bound and Markov's inequality give the conditional CDF $G(u) = \Pr[V \leq u \mid \text{history}] \leq F_c(u)$. With a fresh uniform $\xi \in [0, 1]$, the randomized distributional transform

$$U = G(V^-) + \xi(G(V) - G(V^-))$$

is conditionally uniform on $[0, 1]$ and satisfies $G^{-1}(U) = V$ almost surely. Thus $Z_c = F_c^{-1}(U) \leq G^{-1}(U) = V$. The conditional law of U does not depend on the history, which proves independence and permits iteration. If an input is zero, choose a fresh independent comparison variable: in a chain that started nonzero, its first zero output already forced the preceding comparison product to zero. $\square$

Apply this lemma along each branch's unreduced integer chain, starting from $\text{ctr}_p S$, without conditioning on any size threshold. At the final matrix, apply it to the actual centered modular input. Its comparison variable is independent of the preceding choices in all four sequences.

For a projection sequence g , let F_g be the event that one of its four matrices turns a large input into a small output at the successive thresholds. Lemma A.1 gives $\Pr[F_g] \leq \nu := 1 - (1 - 2^{-43})^4$. Indeed, conditional on avoiding the first j events, the next matrix is fresh and the probability of its large-to-small event is at most 2^{-43} . Multiplying the complementary conditional bounds gives ν . The analogous terminal event has probability at most 2^{-156} .

If the terminal input is small and F_g does not occur, working backward shows that all inputs of branch g are small. Its modular chain therefore equals its integer chain. Acceptance makes the terminal output small, since each output has magnitude at most $m < t_5$. Outside the terminal large-to-small event, its input is consequently small as well. Dropping the nonnegative squared norms of the other branches gives, on norm acceptance outside the terminal event,

$$\Theta \geq E(S) Z_{156} \sum_{g < 4} \mathbf{1}[\neg F_g] \prod_{i=1}^4 Z_{43}^{(g,i)}.$$

The four branch contributions are independent, since the input is fixed and each branch uses disjoint matrices and auxiliary uniform random variables. Each has CDF at most $\min(1, F_P + \nu)$, where F_P is the CDF of a product of four independent Z_{43} variables. If the terminal input is zero, every retained branch has zero final integer output and hence zero comparison product, so the same inequality holds.

Let $Y_0, \dots, Y_3$ be independent with CDF $\min(1, F_P(u) + \nu)$ for $u \geq 0$, zero for $u < 0$, and independent of Z_{156} . Use ζ and ε_{nm} from Theorem 5.1. Quantile coupling separately in the four branches places each Y_g below the corresponding branch contribution and preserves independence from the terminal variable.

Lemma A.5 (Finite Comparison Certificate). For the comparison laws just defined,

$$2^{-156} + \Pr \left[Z_{156} \sum_{g < 4} Y_g < \zeta \right] \leq \varepsilon_{\text{nm}}. \quad (12)$$

Proof with an exact-rational certificate. Put $a = 1/1024$, $R = 33/32$, $M = 2^{192}$ and $U = 2^{160}$. All masses below are rational, and every convolution is exact.

Rounding the comparison laws. For $x \in [0, 130)$, define

$$u_0 = U, \quad u_k = \left\lceil \frac{x u_{k-1}}{k} \right\rceil \quad (1 \leq k \leq 129),$$

$$E(x) = \frac{1}{U} \left(\sum_{k=0}^{128} u_k + \left\lceil \frac{u_{129}}{1 - x/130} \right\rceil \right).$$

The Taylor series and its geometric tail give $E(x) \geq e^x$. For $c \in \{43, 156\}$ and integers $j \geq 0$, set

$$\bar{F}_c(j) = \min \left\{ 1, \frac{\lceil M(21/40)^c E(3aR^j) \rceil}{M} \right\},$$

$$p_{c,\perp} = \bar{F}_c(0), \quad p_{c,j} = \bar{F}_c(j+1) - \bar{F}_c(j).$$

Stop at the first j with $\bar{F}_c(j+1) = 1$. Every evaluated argument is below 130. Assign mass $p_{c,\perp}$ to zero and $p_{c,j}$ to aR^j . Its CDF dominates F_c , so this grid variable can be coupled below Z_c .

Branch products and failure events. Four independent grid variables have positive masses and zero mass

$$p_j = \sum_{j_1+\dots+j_4=j} \prod_{i=1}^4 p_{43,j_i}, \quad p_\perp = 1 - \sum_j p_j,$$

on the grid $a^4 R^j$ and at zero. Move mass $\hat{v} = \lceil M^4 v \rceil / M^4$ from the largest positive values to zero, obtaining a law $\tilde{p}$. This increases its CDF to $\min(1, F_p + \hat{v})$, which dominates the CDF of Y_g .

Rounded sums. For two independent laws v, w on this same grid, let $S(v, w)$ have zero mass $v_\perp w_\perp$ and positive masses

$$S(v, w)_j = v_\perp w_j + w_\perp v_j + \sum_{\substack{r,s \geq 0 \\ \max(r,s)+k(|r-s|)=j}} v_r w_s,$$

$$k(\delta) = \max\{k \in \mathbb{Z}_{\geq 0} : R^k \leq 1 + R^{-\delta}\}.$$

Thus each sum is rounded down, since $R^{\max(r,s)+k(|r-s|)} \leq R^r + R^s$. The law $q = S(S(\tilde{p}, \tilde{p}), S(\tilde{p}, \tilde{p}))$ is a lower comparison for $\sum_{g < 4} Y_g$.

Final comparison. Write $t_j = p_{156,j}$. The final product has positive values $a^5 R^{i+j}$, and $\zeta = a^5 R^{1470}$. The exact calculation gives

$$2^{-156} + 1 - \left(\sum_i q_i \right) \left(\sum_j t_j \right) + \sum_{i+j < 1470} q_i t_j$$

$$\leq \frac{7}{8} 2^{-136} < 2^{-136}.$$

Here the middle term is the product's zero mass. All probabilities have denominator dividing M^{17} , so multiplying by $8 \cdot 2^{156} M^{17}$ makes the comparison an integer inequality. Every replacement decreases the random variable, proving the bound. $\square$

Proof of Theorem 5.1. The preceding comparison shows that acceptance for $E(S) \geq E_\star$ implies the terminal event or $Z_{156} \sum_g Y_g < \zeta$ under a monotone coupling, since $\Theta/E_\star < \zeta$. Apply Equation (12). $\square$

The bound concerns acceptance jointly with a bad fixed input, not the probability of a bad input conditioned on acceptance. There is no union bound over signatures, coordinates, or input blocks.

A.3 Integer Lifting and the Residual Projection

Lemma A.6 (Integer Norm and Residual Bounds). Suppose the fixed polynomials satisfy the algebraic hypotheses of Theorem 5.2 and $E(S) < E_\star$. Then every rowwise norm identity holds over the integers, padded rows have zero norm, and

$$\text{ctr}_p \Delta = \Delta^{\mathbb{Z}} := \text{ctr}_p S_1 + h \otimes S_2 - C.$$

The residual satisfies $|\Delta_{j,i}^{\mathbb{Z}}| \leq L_\Delta$ and $\|\Delta^{\mathbb{Z}}\|_2^2 \leq H_\Delta := 2^{d_{\text{sig}}} n L_\Delta^2$, where $L_\Delta = 811,061,395$ for Falcon-512 and $L_\Delta = 1,648,068,606$ for Falcon-1024. Here h, C, S_2 use their bounded integer representatives.

Proof. The exact parameter inequalities give

$$E_\star + n \cdot 2047^2 < p. \tag{13}$$

Each row's nonnegative squared norm is at most $E(S) + n \cdot 2047^2 < p$, since the digit and sign constraints give $|S_2| \leq 2047$. Its residue equals $\lfloor \beta^2 \rfloor \text{live}(i)$, an integer in $[0, p)$, by the assumed field identity. The two integers therefore coincide. On padded rows their sum is zero, forcing every coefficient of S_1, S_2, e to vanish. On an unpadded row,

$$|S_{1,j}|, |e_k| \leq \lfloor \sqrt{\lfloor \beta^2 \rfloor} \rfloor, \quad \|S_2\|_2 \leq \sqrt{\lfloor \beta^2 \rfloor}.$$

A coefficient of $h \otimes S_2$ is a signed inner product of the n coefficients of h with a permutation of those of S_2 . Since $|h_j|, |C_j| \leq 6144$, Cauchy–Schwarz gives

$$|\Delta_{j,i}^{\mathbb{Z}}| \leq \lfloor \sqrt{\lfloor \beta^2 \rfloor} \rfloor + 6144\sqrt{n\lfloor \beta^2 \rfloor} + 6144 \leq L_{\Delta} < p/2.$$

The residual reduces to Δ modulo p , so the last strict bound identifies it with the centered representative. Summing its squared coefficient bound proves the bound on its squared norm. This argument uses no ring validity assumption. $\square$

Lemma A.7 (Expected Powers of the Projected Squared Norm). Let B have c rows of independent, symmetric real coefficients satisfying $\mathbb{E}B_{jk}^{2r} \leq (2r-1)!! \nu^r$ for every integer $r \geq 1$. For any fixed real matrix X of input blocks,

$$\mathbb{E}\|BX\|_F^{2r} \leq (2\nu)^r (c/2)_r \|X\|_F^{2r}.$$

Here $(a)_r = a(a+1) \cdots (a+r-1)$ and $(a)_0 = 1$. The hypothesis holds with $\nu = 1/2$ for a ternary norm matrix and with $\nu = 52$ for a residual matrix.

Proof. Let G be a matrix of independent standard Gaussians with the shape of BX , independent of B . Conditional on B ,

$$\mathbb{E}_G \langle G, BX \rangle^{2r} = (2r-1)!! \|BX\|_F^{2r}.$$

Conditional on G , the inner product is a real linear combination of the independent entries of B . In its even-power expansion, symmetry makes every term with an odd exponent have zero expectation. Every remaining coefficient is nonnegative, since the corresponding linear coefficients have even powers. Replacing the entries of B by independent $N(0, \nu)$ variables can only increase this conditional expectation. Averaging over G therefore bounds $\mathbb{E}\|BX\|_F^{2r}$ by its Gaussian counterpart, even when the input blocks overlap.

For a Gaussian matrix, diagonalize $X^\top X$, with nonnegative eigenvalues λ_j and $\sum_j \lambda_j = \|X\|_F^2$. Its projected squared norm has law $\nu \sum_j \lambda_j U_j$, where the U_j are independent χ_c^2 variables. If $X \neq 0$, convexity of $t \mapsto t^r$ gives

$$\mathbb{E} \left(\sum_j \frac{\lambda_j}{\|X\|_F^2} U_j \right)^r \leq \sum_j \frac{\lambda_j}{\|X\|_F^2} \mathbb{E} U_j^r = 2^r (c/2)_r.$$

The zero matrix is immediate. For a ternary coefficient b , $\mathbb{E}b^{2r} = 1/2 \leq (2r-1)!!/2^r$ for every integer $r \geq 1$. Equality holds at $r = 1$, and the right-hand side increases thereafter. A residual coefficient a has $\mathbb{E}a^2 = 52$ and $|a|^2 \leq 144 < 3 \cdot 52$. Consequently $\mathbb{E}a^{2r} \leq 52^r 3^{r-1} \leq 52^r (2r-1)!!$. $\square$

Lemma A.8 (Residual Divisibility). Let W be a fixed integer array with $\|W\|_2^2 \leq H$ and $W \not\equiv 0 \pmod{q}$. For three fresh residual matrices of width 31 with entries uniform on $\{-12, \dots, 12\}$, let A be their integer composition. For every integer $r \geq 1$,

$$\begin{aligned} & \Pr[q \mid \text{ctr}_p(AW \bmod p)_o \text{ for all } o] \\ & \leq \left(\frac{4H}{p^2} \right)^r 104^{3r} (31/2)_r^3 + 3 \cdot 25^{-31}, \end{aligned}$$

where $(x)_r = x(x+1) \cdots (x+r-1)$.

Proof. By [Lemma A.7](#), conditionally on the input X to any fresh residual matrix B ,

$$\mathbb{E}[\|BX\|_F^{2r} \mid X] \leq 104^r (31/2)_r \|X\|_F^{2r}.$$

Iterating over three fresh matrices and applying Markov at $p^2/4$ gives the first term for the event $\|AW\|_2^2 \geq p^2/4$. Outside this event, every output is its centered representative modulo p . Acceptance therefore implies $AW = 0$ modulo q .

At a fresh matrix whose input is nonzero over $\mathbb{F}_q$, select a nonzero coordinate in one input block. In each row, condition on all other coefficients. The remaining coefficient has 25 distinct choices modulo q , and at most one makes the selected output zero. Independence of the 31 rows bounds the probability of mapping the whole input to zero modulo q by 25^{-31} . If the composition maps a nonzero input to zero, one of its three matrices must be the first to do so. A union bound gives the second term. $\square$

Apply the lemma with $H = H_\Delta$, using 2^{16} for the maximum number of padded rows. Orders $r = 136$ and $r = 60$ give wrap probabilities $\varepsilon_{\text{wrap}} < 2^{-437}$ and 2^{-153} at Falcon-512 and Falcon-1024, respectively. Put $\varepsilon_{\text{res}} = \varepsilon_{\text{wrap}} + 3 \cdot 25^{-31}$. In both cases $\varepsilon_{\text{res}} < 2^{-142}$, so $\max(\varepsilon_{\text{nm}}, \varepsilon_{\text{res}}) = 2^{-136}$.

Proof of the soundness assertion of [Theorem 5.2](#). The range and encoding-length hypotheses imply encodability by [Section 5.5](#). If $E(S) \geq E_\star$, norm acceptance has probability at most ε_{nm} by [Theorem 5.1](#). Otherwise, [Lemma A.6](#) gives the integer norm identities and the fixed residual bound. If the polynomials fail the batch relation, their residual is nonzero modulo q . [Lemma A.8](#) then bounds residual acceptance by ε_{res} , regardless of the norm predicate. These are exhaustive cases determined by the fixed polynomials before the matrices are sampled, so the larger of the two bounds applies. In particular, the distribution of each newly sampled matrix is not conditioned on subsequent acceptance. $\square$

A.4 Acceptance of Valid Inputs

Proof of [Theorem 5.5](#). The verifier's centered first component, the padded encoding digits, and [Lemma 2.1](#) give the required committed polynomials in their declared widths. Their Boolean identities hold over the integers and hence over $\mathbb{F}_p$. Sum-check is perfectly complete, and the opening succeeds except for the PCS completeness error. It remains to bound rejection by the two checks on the projected values.

For an honest input, $E(S) \leq H_S := 2^{16} \lfloor \beta^2 \rfloor$. Apply the matrices over the integers without intermediate reduction. Applying [Lemma A.7](#) with $v = 1/2$ gives $\mathbb{E}\|AX\|_F^{2k} \leq (c/2)_k \|X\|_F^{2k}$. Set $\mu_k = ((43/2)_k)^4$. Independence of the four branches and the terminal gives

$$\Pr[\|\text{ctr}_p y_{\text{nm}}\|_2^2 > \Theta] \leq \frac{H_S^{46} \mathcal{M}_{46}}{\Theta^{46}} \leq 2^{-129},$$

$$\mathcal{M}_r = (78)_r r! [z^r] \left(\sum_{k=0}^r \frac{\mu_k}{k!} z^k \right)^4.$$

Expanding the fourth power of the generating polynomial bounds the expected r th power of the sum of the four branch squared norms. The final projection contributes $(78)_r$. Centered reduction can only decrease an integer output's absolute value, so intermediate reductions modulo p do not affect this bound.

The two displayed thresholds are the least integers satisfying this order-46 inequality. An honest residual is divisible by q before projection, so its public divisibility test can reject only when reduction modulo p changes an integer output. The total probability of rejecting valid projected values is consequently at most $2^{-129} + \varepsilon_{\text{wrap}} < 2^{-128}$. These bounds are uniform over the supported parameter choices. This proves the completeness claim and the honest-witness assertion of [Theorem 5.2](#). $\square$

Table 7: Norm projections for batches of 32,768 Falcon-512 signatures.

Projection	Matrix dimensions (rows $\times$ columns)	Projected values	Sampled entries
Aerie	43×4096 , then three 43×688 ; finally 156×344	156	1,113,184
TTRP [GP26]	14×8192 , then 1×114688	526	120,651,776

A.5 Matched TTRP Comparison

Table 7 compares the norm projections at 32K Falcon-512 signatures and 16K Falcon-1024 signatures. Each input vector has 2^{25} coordinates, including zero padding. We use the same modulus p , squared-norm bound for valid inputs, and squared-norm cutoff $E_\star$ for oversized inputs. We require probabilities at most 2^{-136} for accepting an oversized input and 2^{-129} for rejecting a valid input. To compare maps with different scales, we divide each squared-norm threshold by the map’s expected squared-norm multiplier before modular reduction. This normalized threshold is also fixed.

Aerie meets both error bounds with 156 projected field elements, one per row of its final matrix. The TTRP construction in Table 7 meets them with 526 values, one per independently sampled pair of matrices.

Aerie’s bounds follow from Theorem 5.1 and Section A.4. For TTRP, we apply Geng and Plançon’s bound on accepting an oversized input [GP26, Theorem 3]. To bound rejection of valid inputs, we apply Lemma A.7 through both matrices in each pair and combine the independent pairs. Exact rational evaluation verifies both error targets and the condition on p required by their theorem (below). Both Aerie and TTRP therefore reject valid inputs with probability at most 2^{-129} in this comparison. For TTRP, we establish this bound using our analysis at the chosen parameters and acceptance threshold. Geng and Plançon’s theorem gives a bound of $1/2$ at a different acceptance threshold [GP26, Theorem 4].

We instantiate the TTRP norm projection of [GP26, Definition 4 and Theorem 3]. Its modulus is our p . Its k output coordinates use independent tensor trains of order μ , internal rank c , and equal mode size d . Write $m_{\text{TT}} = d^\mu$ for the dimension after zero-padding. The expected squared-norm multipliers, before modular reduction, are

$$G_A = 4(43/2)^4 \cdot 78, \quad G_T = k\eta_2, \quad \eta_2 = c^{\mu-1}/2^\mu.$$

Fix $E_\star$ from Theorem 5.1 and set $\Theta_T = G_T\Theta/G_A$, so the normalized acceptance thresholds agree. The TTRP test compares an integer squared norm with this rational threshold, equivalently with its floor. With $\theta = \Theta_T/E_\star$, $\delta = \theta/\eta_2$, and $R = 3(1 + 2/c)\mu^{-1}$, the imported soundness bound is

$$\max \left\{ \left(\frac{R-1}{R-2\delta+\delta^2} \right)^k, \left(\frac{1}{2} + \frac{\mu-1}{2^{c+1}} \right)^k \right\},$$

provided $0 < \delta < 1$ and

$$(2^{\mu+1}c^{\mu-1}m_{\text{TT}})^2\Theta_T < p^2.$$

This last inequality is the squared form of the condition on the modulus in Theorem 3 of [GP26]. It must hold as well as the probability bound.

For completeness, put $H = N^*\lfloor\beta^2\rfloor$. Applying Lemma A.7 through one train and summing the k independent squared outputs bounds the honest-rejection probability by

$$\frac{H^r r!}{\Theta_T^r} [z^r] \left(\sum_{j=0}^r \frac{(1/2)_j (c/2)_j^{\mu-1}}{j!} z^j \right)^k.$$

Centered reduction only decreases the final integer squared norm. Our rejection bound uses the comparison threshold Θ_T . Geng and Plançon prove a bound of $1/2$ at a different threshold [GP26, Theorem 4]. The fixed choice $(\mu, c, k) = (2, 14, 526)$, with the smallest power-of-two d satisfying $d^2 \geq 2nN^*$, meets both error targets and the modulus condition at every power-of-two batch size from 1K to 64K for both Falcon variants. The completeness calculation uses $r = 46$.

For the two parameter choices in the comparison, $d = 8192$. The matrix-entry counts in Table 7 are

$$4(43 \cdot 4096 + 3 \cdot 43^2 \cdot 16) + 156 \cdot 172 \cdot 2, \\ 2kcd,$$

respectively. The following parameter exclusions use the equal mode size and common rank specified above, with the normalized threshold and imported soundness formula fixed. Indeed, $\delta = k\Theta/(G_A E_\star)$ and $1699 < G_A E_\star/\Theta < 1700$. Even using $R = 3$, the smallest possible value, requires $k \geq 340$ in that formula. For $\mu \geq 2$, ranks $c \leq 13$ cannot meet the target for any admissible k . With $k \geq 340$, $c \geq 14$ and $m_{TT} \geq 2^{25}$, every $\mu \geq 3$ violates the condition on the modulus. At $\mu = 2$, the remaining choices with fewer matrix entries reduce to $14 \leq c \leq 21$ and $340 \leq k < 7364/c$. None meets the probability target. The dense case $\mu = 1$ needs more entries as well.

B Bounded Sums of Squares

We prove Lemma 2.1 using Legendre’s three-square theorem: a nonnegative integer is a sum of three integer squares exactly when it is not of the form $4^v(8b + 7)$ for nonnegative integers v, b .

Proof of Lemma 2.1. For $\delta \leq 8191^2$, Lagrange’s theorem suffices. Otherwise, choose $a \leq 8191$ so that $R = \delta - a^2$ is not of the form $4^v(8b + 7)$. Legendre’s three-square theorem then writes R as three squares. If either $a = 8191$ or $a = 8190$ works, use it. If both fail, reduction modulo 8 and 32 gives $\delta \equiv 0 \pmod{32}$: the obstruction for odd a either forces $\delta \equiv 0 \pmod{8}$, or has positive 4-adic valuation and makes $\delta - 8190^2 \equiv 1 \pmod{4}$, which cannot be an obstruction. In the first case, $(\delta - 8190^2)/4 \equiv \delta/4 - 1 \pmod{8}$, so the second obstruction forces $\delta \equiv 0 \pmod{32}$.

Let $t = v_2(\delta)$, where v_2 denotes the exponent of 2 in an integer, and let $k = \lfloor (t - 1)/2 \rfloor$. Then $5 \leq t \leq 26$, $2 \leq k \leq 12$, and $a = 8192 - 2^k \in [4096, 8188]$ has $v_2(a) = k$. The odd integer $a/2^k$ has square 1 modulo 8, whereas $\delta/4^k$ has 2-adic valuation 1 or 2. Consequently $R/4^k$ is 1, 3, or 5 modulo 8, and R is not an obstruction. In every case $0 \leq R \leq 70,265,242 - 4096^2 < 8191^2$, so the three squares representing R also have components at most 8191. $\square$

C Extraction and Security Proofs

We specify the Fiat–Shamir transcript and prove the claims of Section 5. Extraction uses Akita’s composition theorem for an application protocol followed by the Akita opening [DBK⁺26, Theorem 10.45]; this appendix verifies its hypotheses for Aerie and evaluates its error terms. We cite the long version of Akita [DBK⁺26] in its revision of October 15, 2026. The application’s polynomials and restriction to the original domain are those of Section 5.4.

C.1 Fiat–Shamir Transcript and Work Nonces

Public Setup. We use Akita’s shared-prefix setup [DBK⁺26, Section 4.3]. Fix the public seed and the supported schedule family independently of oracle answers. Let N_{setup} accommodate the longest matrix view

in that family. Every matrix is a prescribed reshaping of a prefix of one vector $S \in \mathbb{F}_p^{N_{\text{setup}}}$. The independent oracle H_{setup} supplies streams indexed by the seed, field and page. Each page produces 4096 coefficients by reading independent 64-bit words and rejecting words at least p . Concatenating the pages gives one consistent coefficient stream for every matrix view. For $p = 2^{64} - 59$, each accepted word is exactly uniform in $[0, p)$, independently of the other accepted words, so

$$\delta_{\text{setup}} = 0, \quad \mathbb{E}[\text{words per coefficient}] = 2^{64}/p < 2. \quad (14)$$

Reducing uniform w -bit words modulo p instead would give Akita's bound $N_{\text{setup}}p2^{-w}$ on the setup distance Δ_{setup} of its Theorem 10.45.

The setup step in the proof of that theorem draws S uniformly and answers setup queries conditionally on S . Here this conditional law is exact: before each assigned coefficient, draw a geometric number of rejected words with rejection probability $59/2^{64}$, each uniform in $[p, 2^{64})$, and leave the unused suffix uniform. Streams are extended lazily and answer partial or repeated requests consistently. Other seeds, and coefficients beyond the provisioned prefix, use independent ordinary streams. This preserves the joint law of the setup and all setup-oracle answers, including queries that recompute the matrices, and rewinding keeps these streams fixed. Setup queries count in Q and in running time but create no extraction site.

Transcript and Challenges. The transcript, the projection expansion, and the signature hash use separate oracle domains. Before the first challenge, the transcript absorbs the public-records digest, which binds the ordered records (hpk_i, m_i, r_i) of Section 3.2, their count, and the salt mode, together with the setup and catalog digest and the selected Akita schedule. Each challenge input injectively encodes the complete preceding transcript. Labels included in the hash input identify fixed-width message types and distinguish challenge families and positions in the protocol. The PCS session binds the complete PIOP prefix through a domain-separated digest and also binds its own opening statement.

A challenge site has one random stream, addressed by the full preceding transcript, the statement through its public-records digest, setup and catalog identity, the site identifier, and a work nonce of $g_t + 7$ bits. The stream's first g_t bits must be zero. Disjoint later bits generate the challenge. An element of K uses two base-field components, each drawn by rejecting unsigned candidates outside $[0, p)$. A vector challenge draws all its base-field components from the same stream. This gives the exact uniform law, not the reduction of a wider integer modulo p , and we treat each vector as one field challenge whose conditional answer distribution is sampled exactly.

Conceptually the whole stream is sampled at the first request to its address. Revealing it all at once only helps the adversary. The bits tested for leading zeros and those generating the challenge must come from this one stream rather than from two separately queryable addresses, and a chained implementation must realize this joint experiment or account for its distinguishing error. If no nonce passes, the prover fails, which adds the completeness term $(1 - 2^{-g_t})^{2^{g_t+7}} < e^{-128} < 2^{-184}$ for each site t with $g_t > 0$. A site with $g_t = 0$ has no nonce. The difficulties g_t are given in Equation (17).

Sparse folds of the Akita opening use a separately addressed XOF stream for each challenge coordinate. Its input contains the transcript and public data fixed before the fold, the group, and the coordinate index, but no other coordinate's answer (Section C.3). Sparse folds carry no work nonce.

Hashed Transcript Entries. Some oracle inputs contain a digest in place of data that the verifier reads or recomputes. There are three types.

1. *Compressed prefixes*, hashed by H_{cmp} . The public-records digest stands for the record list and the PCS session digest for the PIOP prefix. If an implementation hashes the transcript and public data fixed before a sparse fold, that digest replaces those data in the coordinate-stream inputs.
2. *Key hashes*, hashed by H_{pk} with 512-bit outputs. The records contain $\text{hpk}_i = \text{SHAKE256}(\text{pk}_i, 64)$, whereas the ring equation uses $h_i = \text{Decode}(\text{pk}_i)$. Key hashes occur inside the public-records digest.

3. *Terminal digests*, hashed by H_{dig} with 256-bit outputs. We use Akita’s terminal variant [DBK⁺26, Section 8.2], which replaces the last binding block t_j^{term} and the last partial-evaluation block e_j^{term} by digests. The prover sends $\text{dig}_t = H_{\text{dig}}(\text{enc}(t_j^{\text{term}}))$ before the predecessor’s ring-switch and sum-check challenges, and $\text{dig}_e = H_{\text{dig}}(\text{enc}(e_j^{\text{term}}))$ after the terminal tensor reduction but before its response nonce and fold challenges. These are the positions of the full blocks, and enc is the injective block encoding fixed by the schedule. Every accepted terminal fold challenge is a unit, so the verifier recomputes both blocks from the folded response; it checks both digests and keeps the linear, trace, and response checks of the full terminal.

We model each type as a random oracle whose queries the extractor observes, independent of the other types and of the remaining oracle families. For the compression and the key hash, this is an explicit modeling hypothesis on the specified SHAKE256 operations. The extractor resamples the oracle on every input that extends a given prefix, and it must recognize such an input when the prefix appears only through its digest. The typed transcript encoding is injective and records the type and position of every digest; a challenge input, including a coordinate-stream input, has at most two terminal-digest positions. Digest queries count in Q .

Write Π_H for Aerie and Π for the same protocol with every hashed entry sent in full. Akita’s lemma on hashed transcript entries [DBK⁺26, Lemma 10.47] converts a prover for Π_H into a prover for Π that runs it once, outputs the same statement, and makes no more challenge queries, including those of its final verification. The converted prover forwards a challenge query with every digest replaced by its first observed preimage, and answers a query containing a digest without a recorded preimage by a fresh local answer.

When the original prover halts, the converted prover runs the verifier of Π_H on its output under the same rules. If one of the two bad events in Akita’s proof occurred, a collision of one hash type or a fresh hash output already present at a digest position, or if the expansion of the output is undefined, it outputs a fixed malformed proof that the verifier of Π rejects before any query. Otherwise it outputs the expanded proof. Without a bad event, a proof that the verifier of Π_H accepts has a defined expansion, so this rule does not change the lemma’s bound.

The verifier of Π_H accepts while that of Π rejects with probability at most $\varepsilon_{\text{bind}} + \varepsilon_{\text{pk}} + \varepsilon_{\text{dig}}$, the three terms of Definition 5.8, which are not multiplied by $Q + 1$. The hash oracles belong to the auxiliary environment, so every rerun uses the same ones. Every challenge input of Π contains the canonical keys, so transcripts combined at one input use the same decoded keys. The protocol still transmits and hashes the same records.

C.2 Soundness of the Three Sum-check Stages

The virtual polynomials P_Z , A_Z^{lo} , A_Z^{hi} , and Δ have Boolean definitions determined by the nine committed polynomials. We test claims about them in three sum-check stages. In stage two, we reduce the factor claims from stage one to claims on P_Z ; in stage three, we reduce these to the committed Z . For the residual, the ring identity in stage three tests $v_{\text{ring}} = v_\Delta - v_{S_1} + C(r_\Pi)$ against σM , while two evaluation reductions test v_{S_1} and v_e against the final openings.

The extractor of Section C.5 obtains the nine polynomials from the Akita opening. It creates no separate branches at individual sum-check rounds. Retries at the projection-seed checkpoint can rerun the three stages and the opening. What Akita’s composition theorem [DBK⁺26, Theorem 10.45] needs from them is soundness as a proof about nine *fixed* polynomials: if the polynomials fail $\mathcal{R}_{\text{batch}}$, then after every challenge some claim about them must still be false, except when the challenge falls in a small set that is determined before it is drawn. This is round-by-round soundness [CCH⁺19] relative to the polynomials, and Lemma C.1 proves it.

One step needs care. The terminal point $r^{(1)}$ of stage one is also the zero-check point of stage two, and $r^{(2)}$ has the same role for stage three. If the polynomials violate an identity that the next stage tests at such a point, the identity is a false claim throughout the rounds that draw the point, and each of those rounds needs its own bad set; the proof supplies it.

Fix a supported parameter choice and a statement, and hence h and C . We write $f = (f_1, \dots, f_9)$ for nine multilinear polynomials whose values on the original Boolean domains of [Table 1](#) lie in $\mathbb{F}_p$. We write z_k for the opening point in the original domain of f_k , and we name the polynomials $S_1, e, a_0, \dots$ as in [Section 4.1](#). On the Boolean domain, f determines the values M^f , $S_2^f = \sigma M^f$, and $\Delta^f = S_1 + h \otimes S_2^f - C$ of that section, the multilinear extensions P_Z^f and $A_Z^{b,f}$ of [Section 4.3](#), and, for a given projection seed, the projections $y^f = (\Pi_{\text{nm}}(S_1||e), \Pi_{\text{res}}(\Delta^f))$. As in [Theorem 5.2](#), f is *algebraically consistent* if its Boolean values satisfy the digit ranges, the sign constraint, and the rowwise norm and encoding-length identities of [Section 4](#) over $\mathbb{F}_p$, and it is *valid* if its centered values satisfy $\mathcal{R}_{\text{batch}}$.

A *challenge input* is the oracle input of one challenge site, that is, the address of its stream in [Section C.1](#). It encodes the site and the transcript through the prover's last message before that challenge, including the work nonce. From the projection seed to the opening, the sites are, in order: the seed; μ ; τ ; $\gamma^{(1)}$; the d rounds of stage one; $\gamma^{(2)}$; the d rounds of stage two; $\gamma^{(3)}$; the d rounds of stage three; and the vector $\eta \in K^9$ of independent coefficients with which the opening batches the nine evaluation claims $v_1, \dots, v_9$. The first fold challenge of the opening follows η . Let $(\ell_1, \ell_2, \ell_3) = (3, 8, 12)$ for Falcon-512 and $(4, 9, 12)$ for Falcon-1024 be the numbers of batching coefficients in the three stages.

The lemma associates two objects with polynomials f . For every challenge input u , a bit $D(u; f)$ records whether the transcript prefix in u fails a verifier check or makes a claim that is false for f . For every site, a set of bad oracle answers contains the answers after which this may cease to hold. Items (i) and (ii) determine the bit at the seed and at the first fold challenge, item (iii) says that the bit remains 1 outside the bad set, and item (iv) bounds the bad sets.

Lemma C.1 (Checks of Fixed Polynomials). Fix an expansion oracle in the event of [Lemma C.2](#). For all polynomials f and every challenge input u from the seed through the first fold challenge there is a bit $D(u; f)$, and for every site t from the seed through η and every input u of that site there is a set $\text{Bad}_t(u; f)$ of oracle answers, such that:

- (i) at the input of the seed, $D(u; f) = 1$ if f is not valid for the statement in u ;
- (ii) at the input of the first fold challenge, $D(u; f) = 0$ exactly when no verifier check determined by u fails and $\sum_k \eta_k v_k = \sum_k \eta_k f_k(z_k)$;
- (iii) if $D(u; f) = 1$ at an input of site t and the answer ρ is outside $\text{Bad}_t(u; f)$, then $D(u'; f) = 1$ at every input u' of the next site that extends u by ρ ; and
- (iv) $\text{Bad}_{\text{seed}}(u; f)$ contains at most a fraction $\vartheta = 2^{-135}$ of the seeds, and every other $\text{Bad}_t(u; f)$ has probability at most $2^{-g_t} v_t / |K|$, where g_t is the work difficulty of the site and v_t is 8 for μ , d for τ , 1 for each of $\gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)}, \eta$, and 3, 5, 3 for a round of stage one, two, three.

The equation in (ii) states that the nine claimed values, combined with the coefficients η , equal the same combination of the evaluations of f . It is the only equation through which the opening uses the nine claims. We call it the batched evaluation equation below.

Proof. Invariant and quantifiers. Fix the polynomials f and a challenge input u . Let $\mathcal{P}(u; f)$ be the ordered list of pending claims, namely the equalities and predicates on f specified below. Let $\text{Reject}(u)$ mean that a verifier check determined by u fails. Define

$$D(u; f) = \mathbf{1}[\text{Reject}(u) \vee \exists C \in \mathcal{P}(u; f) : C \text{ is false}].$$

At the seed, $\mathcal{P}$ consists of validity of f . At the first fold, it consists of the scalar-row equality in item (ii). A failed verifier check persists on every extension. Otherwise choose the first false claim and define $\text{Bad}_t(u; f)$

as below. The set is empty if the current site does not act on that claim. We prove, for every prover response and every resulting next input u' , that

$$D(u; f) = 1 \wedge \rho \notin \text{Bad}_t(u; f) \implies D(u'; f) = 1. \quad (15)$$

For a field site, an answer can be bad only when its work bits are zero. The work bits and challenge are disjoint parts of one sampled stream. If the challenge condition has probability at most $v_t/|K|$, then

$$\Pr[\rho \in \text{Bad}_t(u; f)] \leq 2^{-g_t} v_t/|K|.$$

Each condition below is a zero set of a nonzero polynomial of total degree at most v_t . The seed is handled separately using [Lemma C.2](#).

The seed. After the seed and the prover's outputs y , the pending claims are (C1) $y = y^f$ and (C2) f is algebraically consistent. If f is not algebraically consistent, (C2) is false after every seed, and $\text{Bad}_{\text{seed}}(u; f)$ is empty. Otherwise let it be the set of seeds for which both projections in y^f pass their predicates. Since f is not valid, this set contains at most a fraction ϑ of the seeds by [Lemma C.2](#). For any other seed, y^f fails a predicate, so either $y = y^f$ and the verifier's predicate check fails, or (C1) is false.

The point μ . Suppose (C1) is false. The weights λ_o are distinct multilinear monomials in μ , so $\sum_o \lambda_o(y_o - y_o^f)$ is a nonzero polynomial in μ of total degree at most 8, and the bad set is its zero set. Outside it, $\langle \lambda, y \rangle \neq \langle \lambda, y^f \rangle$. Since (w_{S_1}, w_e) and w_Δ are the transposed projections of λ , the value $\langle \lambda, y^f \rangle$ is the left side of the projected-values check of [Figure 3](#) computed from f . The input claim of the projected-values check is therefore false, and it replaces (C1) as a pending claim.

The point τ . Suppose a base-16 digit Z of f takes a value outside its allowed set at some Boolean input. Then $A_Z^{\text{lo},f} A_Z^{\text{hi},f}$ is nonzero there, and $\sum_x \text{eq}(\tau, x) A_Z^{\text{lo},f}(x) A_Z^{\text{hi},f}(x)$ is a nonzero multilinear polynomial in at most d coordinates of τ . The bad set is its zero set. Outside it, the factor-product claim for Z is false. The factor-product claims replace the base-16 allowed digit sets as pending claims. The base-8 allowed digit sets, the sign constraint, and the norm and encoding-length identities of (C2) stay pending.

The next three paragraphs treat the steps that the three stages share: the batching coefficients, the sum-check rounds, and the identities whose zero-check point is drawn by the preceding stage. The three paragraphs after them list the instances of each stage and its pending claims.

Batching coefficients. Stage s has ℓ_s instances, each with an input claim c_ℓ , the value that the verifier uses for its sum. Let c_ℓ^f be the same sum computed from f , and call the instance *false* if $c_\ell \neq c_\ell^f$. At $\gamma^{(s)}$ the pending claims include the equations $c_\ell = c_\ell^f$. If some instance is false, then $\sum_\ell \gamma_\ell (c_\ell - c_\ell^f)$ is a nonzero linear form in $\gamma^{(s)}$, and the bad set is its kernel, of probability $1/|K|$. Outside it, the batched claim differs from the sum of the batched summand computed from f , under the domain rule of [Section 2.4](#). The pending claim at the first round is the equality of these two values, and it is false.

Sum-check rounds. Let F^f be the batched summand of the stage, with every virtual and committed polynomial replaced by the corresponding multilinear extension computed from f . Call the running claim of a round true if it equals the sum of F^f over the remaining variables at the challenges drawn so far. The verifier recovers a round polynomial whose values at 0 and 1 sum to the running claim ([Figure 2](#)). Write g_j for this polynomial and g_j^f for the true round polynomial of F^f . If the running claim is false, the consistency check implies

$$g_j \neq g_j^f, \quad \deg(g_j - g_j^f) \leq v_t, \\ \Pr_{r_j \leftarrow K} [g_j(r_j) = g_j^f(r_j)] \leq v_t/|K|.$$

Here $v_t = 3, 5, 3$ in the three stages. These roots, together with zero work bits, form the bad set. At any other challenge the next running claim is false.

After the last round the prover sends evaluations, and the verifier's terminal check equates the last running claim with a fixed expression in them and in public values. The value of F^f at the terminal point is the same

expression in the evaluations computed from f . If the last running claim is false and the check passes, some evaluation therefore differs from the one computed from f , and the equality of those two is a pending claim.

Identities tested at a point of the preceding stage. The base-8 allowed digit sets are tested in stage two at the point $r^{(1)}$, and the sign constraint and the norm and encoding-length identities are tested in stage three at $r^{(2)}$. Each of these identities states that a function ϕ computed from f vanishes on a Boolean domain, and its test is the claim $\sum_x \text{eq}(r, x) \phi(x) = 0$ at the coordinates of r that belong to that domain.

While the earlier stage draws r , the identity is pending as the claim that $\sum_x \text{eq}(r, x) \phi(x)$, with the drawn coordinates substituted, is the zero polynomial in the coordinates not yet drawn. This polynomial is multilinear, and it is nonzero before the first round if $\phi \neq 0$. A nonzero multilinear polynomial becomes the zero polynomial for at most one value of its next coordinate. In a round where the first false pending claim is such an identity, the bad set is this one value, so its probability is within the bound $v_t/|K|$ of that round. After the last round the pending claim is the equation at the complete point.

Stage one. Its instances are the factor products, with input claim zero, and the projected-values check, with input claim $\langle \lambda, y \rangle$. After its terminal message the pending claims are (E1) $v_Z^b = A_Z^{b,f}(r^{(1)})$ for every base-16 digit and $b \in \{\text{lo}, \text{hi}\}$, (E2) $v_{S_1} = S_1(r_\Pi)$, (E3) $v_e = e(r_\Pi, e)$, and (E4) $v_\Delta = \Delta^f(r_\Pi)$, together with the base-8 allowed digit sets at $r^{(1)}$ and the sign, norm, and encoding-length identities.

Stage two. For a base-16 digit the input claim is v_Z^b and the sum computed from f is $A_Z^{b,f}(r^{(1)})$, so the instance is false exactly when (E1) is. For a base-8 digit the input claim is zero and the sum computed from f is the left side of its range test at $r^{(1)}$. After the terminal message the pending claims are (L) $v_Z^P = P_Z^f(r^{(2)})$ for the six digits, together with (E2)–(E4) and the sign, norm, and encoding-length identities at $r^{(2)}$.

Stage three. The six digit checks have input claims v_Z^P and sums $P_Z^f(r^{(2)})$, so a check is false exactly when its claim in (L) is. The sign, norm, and encoding-length instances are false exactly when their identities at $r^{(2)}$ are. The two evaluation reductions have input claims v_{S_1}, v_e and sums $S_1(r_\Pi), e(r_\Pi, e)$, so they are false exactly when (E2) and (E3) are. For the ring instance, w_{ring} is the transpose of negacyclic multiplication by h applied to the equality weights of r_Π (Section 4.5), so the sum computed from f is the multilinear extension of $h \otimes S_2^f$ at r_Π , which equals $\Delta^f(r_\Pi) - S_1(r_\Pi) + C(r_\Pi)$. The input claim is $v_\Delta - v_{S_1} + C(r_\Pi)$. Hence, if (E4) is false, then (E2) is false or the ring instance is. Every false pending claim thus makes an instance of stage three false. The summand of stage three is a polynomial in the nine polynomials, so after its terminal message the pending claims are (V) $v_k = f_k(z_k)$ for $k = 1, \dots, 9$.

The vector η . If (V) is false, the bad set is the kernel of the nonzero linear form $\eta \mapsto \sum_k \eta_k (v_k - f_k(z_k))$, of probability $1/|K|$. Outside it, the equation in (ii) is false. $\square$

For Theorem 5.3, use the same pending claims with fresh interactive challenges and $g_t = 0$. At the projection step, Theorem 5.2 gives the bound 2^{-136} for freshly sampled matrices. Stop at (V), where the verifier checks all nine claims by oracle queries. Thus the last $1/|K|$ contribution for η belongs only to the batched opening, not to Equation (2).

C.2.1 PIOP Arithmetic Costs

We justify the arithmetic bounds of Proposition 5.4 for the supported projection profiles. A projection step with input shape $c \times J \times m$ and output width c' uses $O(c'cJm)$ operations. The widths and number of steps are fixed, so all forward and transposed projections use $O(nN^*)$ operations. The number of sampled matrix entries is bounded by the same quantity. Forming the virtual digit and range tables also takes $O(nN^*)$ operations. Forming Δ and w_{ring} requires N^* negacyclic products or transposed products, costing $O(N^*M(n))$.

Each sum-check summand is a fixed-degree expression in a constant number of multilinear tables. At a round with 2^v entries, forming the round polynomial and folding these tables takes $O(2^v)$ operations. The three stages therefore cost $O(\sum_{v=1}^d 2^v) = O(nN^*)$. The smaller domains, their normalization factors and the

fixed coordinate permutations do not increase this bound. Evaluating the terminal claims also takes $O(nN^*)$ operations.

The verifier evaluates the projection weights by contracting each matrix against the equality weights of its index block. This costs at most $O(nN^*)$ operations. It evaluates $C(r_\Pi)$ by a scan of the public target table. To evaluate the ring weight without constructing its full table, define

$$\begin{aligned}\alpha_j &= \text{eq}(r_{\Pi, \text{coef}}, j), & \beta_j &= \text{eq}(r_{\text{coef}}^{(3)}, j), \\ \omega_i &= \text{eq}(r_{\Pi, \text{sig}}, i) \text{eq}(r_{\text{sig}}^{(3)}, i), & \bar{h} &= \sum_i \omega_i h_i.\end{aligned}$$

Linearity of negacyclic multiplication gives

$$w_{\text{ring}}(r^{(3)}) = \sum_{j'} \alpha_{j'} (\bar{h} \otimes \beta)_{j'}.$$

Forming $\bar{h}$ takes $O(nN^*)$ operations and the single product takes $O(M(n))$. The remaining equality weights, batching and round checks fit within these bounds. For example, ordinary quadratic multiplication gives $M(n) = O(n^2)$ without any transform assumption. The costs here concern the polynomial-oracle protocol and do not include witness construction, public-record hashing or the PCS.

C.3 Composition with the Akita Opening

Aerie as a Composite Protocol. Akita’s Theorem 10.45 applies to a protocol in three parts: a first message with the statement, the schedule, and the root commitments; an *application segment* of challenge sites that ends at the input ξ_f of the first fold challenge; and the Akita opening from ξ_f . In Aerie the first message contains the statement through its public-records digest, the selected schedule, and the four root commitments of Table 1. The application segment runs from the projection seed through η , with the sites listed in Section C.2. The input ξ_f fixes the four opening points, the claimed values, and the coefficients η , and Akita’s root row [DBK⁺26, Equation (202)] becomes the batched evaluation equation

$$\sum_k \eta_k v_k = \sum_k \eta_k f_k(z_k).$$

In Akita, a stage denotes a single challenge site; an Aerie sum-check stage contains several such sites.

The root row has one term for each group and polynomial, and every group keeps its own packing map, decomposition basis, commitment matrix, point, and evaluation functional. The four groups, with polynomial counts (1, 1, 4, 3) and signed source widths (14, 14, 4, 4), therefore need not share parameters. If a PCS domain has extra coordinates, the opening point is zero in them, and the evaluation equals that of the polynomial obtained by fixing those coordinates to zero, since every other Boolean value has zero interpolation weight. These restrictions are the polynomials f on the original domains. The application relation of the theorem is $\mathcal{R}_{\text{batch}}$ for the centered values of f ; the composed relation adds that the extracted source blocks decommit the four root commitments. Akita’s own root batching by powers of one scalar is not used: the coefficients η lie in the application segment, and the opening reads them only as public data [DBK⁺26, Proposition 10.44].

Schedule Conditions. The theorem requires an extraction-admissible schedule, meaning items (i)–(iii) of Akita’s Definition 9.2. Item (iv), which adds the standalone Fiat–Shamir error condition, is not needed; the error of the composed protocol is bounded in Section C.4.

The selected schedules satisfy items (i)–(iii). The root has the four groups above, and each later fold receives one recursive witness. There are no setup-opening groups or split responses. The first two nonterminal

folds use coefficient packing and coefficient-norm bounds, and later folds use evaluation trace. Compression and quotient lifting each form a prefix, and every Euclidean norm check applies to one complete recursive response.

Reconstructing each fold's outgoing witness gives exactly its successor's input length and representation, with zero extension to the successor's opening domain. With b_j^* the certified digit base of fold j , and $b_{1,j,g}$ and $b_{z,j,g}$ its opening and response digit bases for group g , the parameters satisfy $b_{1,j,g}, b_{z,j,g} \leq b_j^*$, and quotient and compression digits lie in this certified digit set. For coefficient packing, the commitment-ring dimension is divisible by twice the challenge-ring dimension, as the quadratic extension requires. The challenge families and samplers are certified below, and the tree size is bounded in [Section C.4](#). Collision bounds use the certified digit sets and response bounds, not the honest input widths.

Sparse Challenge Distribution. A sparse fold draws each challenge coordinate from its own stream. The stream's input contains the transcript and public data fixed before the fold, the group, and the coordinate index, but no other coordinate's answer. Each coordinate makes at most $L = 4096$ independent uniform draws from a sparse family and returns the first that a deterministic operator-norm predicate accepts, or $\perp$. Conditioned on success, its output is uniform on the accepted set [[DBK⁺26](#), Lemma 10.1].

Every nonzero difference of two challenges in a base family is a unit at the selected modulus, and the filter only removes elements, so the accepted families keep this property [[DBK⁺26](#), Section 3.2 and Appendix H.3]. Akita's Lemma 10.2 programs one stream conditionally on an accepted challenge with every other stream unchanged. Each sparse fold is therefore an indexed challenge site of Akita's Definition 10.16, with one reference certificate and one alternative per coordinate, obtained by changing only that coordinate. Sampler exhaustion is a completeness event. Rejection that depends on the prover's response is a separate search, and its hash queries count in Q . Sparse folds carry no work nonce.

A sparse family in dimension D with ω_1 entries of magnitude one and ω_2 entries of magnitude two has cardinality

$$C(D, \omega_1, \omega_2) = \binom{D}{\omega_1 + \omega_2} \binom{\omega_1 + \omega_2}{\omega_1} 2^{\omega_1 + \omega_2}.$$

The factors select support, magnitudes, and signs.

The selected L_2 levels use $D = 128$, $\omega_1 = 31$, $\omega_2 = 0$, with operator threshold 13. The certified subset has true operator norm at most $13 - 351/2^{48}$, sufficient for the strict fixed-point test. Let $U(X)$ be the certificate's rational polynomial dominating the indicator above $(13 - 351/2^{48})^2$ on $[0, 31^2]$. Its expectation, evaluated exactly, is q_1 . The certificate verifies nonnegativity and the indicator bound using Bernstein coefficients. There are 64 distinct frequencies. The union bound therefore proves accepted fraction at least $p_0 = 1 - 64q_1$, and hence

$$\begin{aligned} |A| &\geq C_{\text{acc}} := \lceil C(128, 31, 0)p_0 \rceil \\ &= 502822610611069237129947393810277364817, \\ \log_2 C_{\text{acc}} &> 128.5633. \end{aligned}$$

We evaluate this lower bound exactly from the rational certificate. The number of challenges before rejection sampling is not a valid denominator at these L_2 sites. The accepted fraction is at least $\alpha := C_{\text{acc}} / \left(\binom{128}{31} 2^{31} \right) > 2/5$, so a coordinate succeeds with probability $a = 1 - (1 - \alpha)^L \geq \alpha > 2/5$, and Lemma 10.2 samples a conditional stream with expected work $O(L/a)$. Unfiltered families accept every draw.

Write $W_{j,g}$ for the number of polynomial-block coordinates of group g at fold j , $W_j = \sum_g W_{j,g}$, and $C_{j,g}$ for a lower bound on the number of challenges accepted by the sampler. Every $C_{j,g}$ exceeds 2^{128} . At fold j , evaluating V_j fresh complete response candidates queries $W_j V_j$ coordinate streams. Partial evaluations count only the coordinates queried.

Module-SIS Occurrences. For a schedule s , let $\mathcal{I}(s)$ be its multiset of Module-SIS instances, each with its matrix role, ring dimension, ranks, and collision bound. Depending on the parameter choice it has between 33

and 51 inner, outer, opening, and compression entries. The collision bounds are those of Akita’s Lemma 10.23, Remark 10.24, and Corollary 10.35. They bound extracted responses, not bit widths of valid inputs. For fixed parameters set $\mathcal{I} = \mathcal{I}(s)$. If the prover may choose s after seeing the setup, fix the supported family $\mathcal{S}$ before setup and use

$$\mathcal{I} = \bigsqcup_{s \in \mathcal{S}} \mathcal{I}(s).$$

An instance in this disjoint union retains its schedule identifier, and occurrences that share a parameter row are not merged. The common setup accommodates every matrix view in the family.

Akita’s Lemma 10.20 reduces a collision at any occurrence to ordinary Module-SIS, also for an algorithm that chooses the schedule after seeing the setup. Part (i) gives, for each $I \in \mathcal{I}$, an algorithm of expected cost

$$T_I = T_E + T_{\text{embed}, I} + T_{\text{collision}, I},$$

and bounds the collision probability by $\sum_{I \in \mathcal{I}} \text{Adv}_{\text{MSIS}, I}^{\text{exp}}(T_I)$. Truncating the complete reduction for occurrence I at time t costs at most T_I/t in success probability. For the Falcon reduction, we use its complete expected running time in the same bound. The lattice cost estimates used to choose the Akita parameters model particular attacks on these instances. They do not bound the advantages above, and they do not include the factor of about $(Q_{\text{wrap}} + 2)\tilde{K}$, with $\tilde{K}$ the tree size of Equation (16), between T_I and the cost of one prover run.

C.4 The Projection Seed and Work Difficulties

Uniform Density of Bad Projection Seeds. For fixed polynomials satisfying the algebraic identities but not $\mathcal{R}_{\text{batch}}$, Theorem 5.2 bounds the probability that their projections pass by $\varepsilon_P \leq 2^{-136}$ over the expansion oracle. The extractor resamples the projection seed but not the expansion oracle, and the polynomials that it tests may depend on that oracle. We therefore need a density bound after the expansion oracle has been fixed, uniformly over all polynomials.

Lemma C.2 (Uniform Seed Density). In the independent expansion-stream model, except with probability $\delta_X < 2^{-2^{117}}$ over the expansion oracle, every supported parameter choice and every invalid algebraically consistent tuple of polynomials has at most a $\vartheta := 2^{-135}$ fraction of accepting 256-bit seeds.

Proof. The expansion addresses contain only the public parameter choice, seed, projection matrix, and row, with injective framing. Different seeds therefore define independent complete projection matrix families. For fixed parameters and polynomials, the number of accepting seeds is stochastically dominated by $\text{Bin}(2^{256}, 2^{-136})$, whose mean is 2^{120} . The multiplicative Chernoff bound gives

$$\Pr[\text{more than } 2^{121} \text{ accepting seeds}] \leq \exp(-2^{120}/3).$$

The polynomials f and the public polynomials h, C use at most $(7n + 7)2^{d_{\text{sig}}} < 2^{29}$ elements of $\mathbb{F}_p$: five committed coefficient polynomials, h, C , four slack entries and three digits for unused unary bits per row. There are at most $2 \cdot 2^{16}$ supported parameter-set and batch-size pairs. Since $p < 2^{64}$, the logarithm base two of the total number of parameter-polynomial tuples is less than 2^{36} . The union bound is consequently at most

$$\exp((\log 2)2^{36} - 2^{120}/3) < 2^{-2^{117}}.$$

Public messages and commitment bytes are not additional expansion contexts. Their influence is through the seed and the finite public data already counted. The resulting event is simultaneous over all polynomials, including polynomials chosen after inspecting oracle answers. $\square$

The Hypotheses of the Composition Theorem. Let F be the number of folds of the selected schedule, counting the terminal fold, and let W_j be the number of challenge coordinates of fold j , as defined in [Section C.3](#). The checkpoint at the start of the application segment is the projection seed, its candidate is the nine polynomials, and its test is $\mathcal{R}_{\text{batch}}$. The argument requires neither polynomials extracted before the seed nor independence of the extracted polynomials from their own seed.

Lemma C.3 (Checkpoint Structure of Aerie). Fix a supported parameter choice and consider Π , which sends every hashed entry in full. The hypotheses of Akita’s Theorem 10.45 hold with the projection seed as the first challenge of the application segment, with the event Good_X of [Lemma C.2](#) on the expansion oracle, and with the bits and bad sets of [Lemma C.1](#). The resulting checkpoint structure, for $\mathcal{R}_{\text{batch}}$ and the binding relation whose witnesses are the collisions at the schedule’s Module-SIS occurrences, has the F fold challenges as extraction sites, with $1 + W_j$ children at fold j , and the projection seed and the ring-switch challenge of each of the $F - 1$ nonterminal levels as checkpoints. Its bad sets have probability at most ϑ at the seed and at most $2^{-g_t} \nu_t / |K|$ at any other field site t , where ν_t is the numerator in [Lemma C.1](#) or in [\[DBK⁺26, Lemma 10.42\]](#). Its tree size is

$$\widehat{K} = 2^F \prod_{j=1}^F (1 + W_j) = \prod_{j=1}^F 2(1 + W_j). \quad (16)$$

Proof. A tuple σ of source projections determines polynomials f , the restrictions of its decoded vectors to the original domains ([Section C.3](#)). Put $D(u; \sigma) = D(u; f)$ and $\text{Bad}_t(u; \sigma) = \text{Bad}_t(u; f)$ for the bits and sets of [Lemma C.1](#), which that lemma provides for every expansion oracle in Good_X . Condition (a) of the theorem is item (i) of the lemma, since the application relation fails exactly when f is not valid. Condition (b) is items (iii) and (iv). Condition (c) is item (ii), because Akita’s root row for σ is the batched evaluation equation for f .

The theorem places one checkpoint at the seed and takes the structure of the opening from Akita’s Proposition 10.44, whose checkpoints are the ring-switch challenges of the $F - 1$ nonterminal levels. Each fold is an indexed site with one reference certificate and one alternative per coordinate, so fold j has arity $1 + W_j$. The bad sets of the opening are those of [\[DBK⁺26, Lemma 10.42\]](#); with the work bits of [Equation \(17\)](#), a bad answer must also have zero work bits, which multiplies the bound by 2^{-g_t} [\[DBK⁺26, Remark 10.18\]](#). The tree size of the theorem is twice that of the opening alone, that is, $2 \cdot 2^{F-1} \prod_j (1 + W_j)$, which is [Equation \(16\)](#). $\square$

Error Bounds and Proof-of-Work Difficulties. Let $\mathcal{F}$ be the set of field sites other than the seed. The protocol assigns every site $t \in \mathcal{F}$ a rational upper bound e_t on its bad-set probability before proof of work. This bound determines the work difficulty. A uniform point with w coordinates has bound $w/|K|$, a sum-check round has bound equal to the degree of its round polynomial divided by $|K|$, a batch of k claims by the powers of one scalar has bound $(k - 1)/|K|$, and a vector of ℓ independent batching coefficients has bound $\ell/|K|$. The PIOP bounds are therefore $8/|K|$ for μ , $d/|K|$ for τ , $\ell_s/|K|$ for $\gamma^{(s)}$, and $3/|K|$, $5/|K|$, $3/|K|$ per round of the first, second, and third stages, respectively, and the root vector η has bound $9/|K|$.

A ring-switch challenge has bound $(2d' - 1)/|K|$, where d' is the largest source ring dimension among the level’s ring-valued rows [\[DBK⁺26, Theorem 10.36\]](#). This is the largest ring dimension among the level’s commitment matrices, and it can exceed the dimension of the ring from which the level draws its fold challenges. In the selected schedules d' lies between 128 and 1024. The difficulties are

$$g_t = \min\{g \in \mathbb{Z}_{\geq 0} : 2^{128} e_t \leq 2^g\} \quad (t \in \mathcal{F}), \quad (17)$$

and sparse folds and the projection seed require no work: a queried coordinate stream of group g at fold j has weight $1/C_{j,g}$, and the seed has error bound ϑ . Since $|K| = p^2$ with $p = 2^{64} - 59$, even a bound of $1/|K|$ needs one bit.

The analysis requires only $e_t \geq \nu_t/|K|$, where ν_t is the numerator in [Lemma C.1](#) or in [\[DBK⁺26, Lemma 10.42\]](#). The assigned bound is strictly larger at a vector of $\ell > 1$ independent coefficients, where the bad set is the kernel of one linear form, of probability $1/|K|$, and the assigned bound is $\ell/|K|$. A larger assigned bound gives a larger difficulty g_t and hence a smaller weight $2^{-g_t} \nu_t/|K|$, so [Equation \(19\)](#) is unaffected.

Address Weights. Akita’s Theorem 10.45 bounds the probability that the verifier accepts while the extractor returns neither a witness nor a collision by $\Pr[\neg \text{Good}_X]$ plus the expected total weight of the challenge addresses read by the prover and by the verifier on its output [\[DBK⁺26, Equation \(208\)\]](#).

We use disjoint site labels for the field challenge families and the projection seed. Following Akita’s Definition 10.16 and Remark 10.18, a field-family address at site t has weight $2^{-g_t} \nu_t/|K|$, a coordinate stream of group g at fold j has weight $1/C_{j,g}$, and the seed address has weight ϑ . The output transcript contributes one address per field site and per fold coordinate, at most $R_f + R_{\text{coord}}$ addresses read only by the verifier; every other address is a query of the prover.

The converted prover of [Section C.1](#) adds verification queries. If it outputs the malformed proof, the verifier reads no address. Otherwise no bad event occurred, so the expansion of each query was fixed when the query was made, and each forwarded verification query is the address that the verifier of Π reads at the same position of the expanded proof. Verification therefore adds no address outside the output transcript. Hence

$$\mathbb{E}\left[\sum_{a \in \text{Addr}(J)} \epsilon(a)\right] \leq \rho(Q + R_f + R_{\text{coord}}), \quad \rho = \max_a \epsilon(a). \quad (18)$$

Queries made only by a rerun of the prover enter the running time, not this bound. An answer that fails the leading-zero check is not a bad answer, and the extractor resamples complete answers, work bits included, without conditioning on the predicate.

Using the field size and the number of sparse challenges accepted by the sampler,

$$\rho \leq \max \left\{ \vartheta, \max_{t \in \mathcal{F}} 2^{-g_t} e_t, \max_{j,g} C_{j,g}^{-1} \right\} \leq 2^{-128}. \quad (19)$$

No factor counting all sites is needed in the coefficient of Q . The sum over sites remains in the weighted sum for queries made only by the verifier, which [Equation \(18\)](#) bounds using the explicitly counted verifier queries.

For the highlighted Falcon-512 batch of 2^{15} signatures and Falcon-1024 batch of 2^{14} signatures, $d = 24$ and the PIOP difficulties are

	Falcon-512	Falcon-1024
μ, τ	(4, 5)	(4, 5)
$(\gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)})$	(2, 4, 4)	(3, 4, 4)
rounds in stages 1, 2, 3	(2, 3, 2)	(2, 3, 2)

The PCS field sites, which are its sum-check rounds, ring-switch and interstage batches, anchor points, and the root vector η , receive difficulties from the same formula: the assigned bound $9/|K|$ of η needs four bits, and a ring-switch challenge needs between eight and eleven. For every supported parameter choice, we compute each site’s assigned bound, the bound ν_t , and the least difficulty, and verify the accepted-support norm certificate at every L_2 fold.

Tree Size and Verifier Query Bounds. There are 29 supported parameter choices: $N^* \leq 2^{16}$ with $nN^* \geq 2^{12}$. Let $N_{\text{exp}} \geq nN^* = 2^d$ be the total explicit instance-and-witness length in base-field coordinates. At every parameter choice the tree size of [Equation \(16\)](#) satisfies $\widehat{K} \leq 2^{2d} \leq N_{\text{exp}}^2$. This is the tree-size condition of Akita’s Definition 9.2. The other schedule conditions are checked in [Section C.3](#), and the transcript error is bounded by [Equations \(18\) and \(19\)](#). Numerically $2^{21.4} < \widehat{K} < 2^{39.7}$. The smallest value occurs, for example,

at Falcon-512 with $N^* = 8$, where $F = 5$ and $(W_j) = (18, 9, 8, 8, 5)$. The largest occurs at Falcon-1024 with $N^* = 2^{16}$, where $F = 7$ and $(W_j) = (526, 86, 20, 13, 8, 8, 5)$. The bounds on queries made only by the verifier are $R_f \leq 1 + |\mathcal{F}|$ and $R_{\text{coord}} \leq \sum_j W_j$. Over the supported parameter choices R_f ranges from 350 to 541, R_{coord} from 48 to 666, and $R_f + R_{\text{coord}} \leq 1207$.

C.5 Proof of Knowledge Soundness

We prove [Theorem 5.9](#). Let P be a prover in [Definition 5.7](#).

Ideal samplers. Replacing the setup and the challenge samplers by their ideal laws changes the knowledge error by at most $\delta_{\text{setup}} + \delta_{\text{samp}}$, with $\delta_{\text{setup}} = 0$ by [Equation \(14\)](#). The sampler distance is zero for the exact samplers of [Section C.1](#).

Hashed entries. Let P' be the prover for Π obtained from P in [Section C.1](#). It outputs the statement of P , and a proof that Aerie accepts and Π rejects has probability at most $\varepsilon_{\text{bind}} + \varepsilon_{\text{pk}} + \varepsilon_{\text{dig}}$. Since P' runs P followed by verification, it makes at most $Q_{\text{wrap}} = Q + V_{\text{ver}}$ queries per run.

Composition. Apply Akita's Theorem 10.45 to P' , with the hypotheses of [Lemma C.3](#) and $\Delta_{\text{setup}} = \delta_{\text{setup}} = 0$. The extractor $\mathcal{E}$ runs the theorem's extractor on P' , restricts the polynomials of the certificate at the seed to their original domains, and returns them with the original execution, which its handle fixes [[DBK⁺26](#), Definition 10.15]. By [Lemma C.2](#), [Equations \(18\)](#) and [\(19\)](#), and the Module-SIS step of [Section C.3](#), the probability that Π accepts and $\mathcal{E}$ returns no witness of $\mathcal{R}_{\text{batch}}$ is at most

$$\delta_X + (Q + R_f + R_{\text{coord}})2^{-128} + \sum_I \text{Adv}_{\text{MSIS}, I}^{\text{exp}}(T_I).$$

The Module-SIS term needs no conditioning on Good_X . The tests and bad sets are functions of challenge inputs that encode the statement, so P may choose it. Summing the three steps gives [Equation \(7\)](#).

Running time. The theorem's extractor runs P' at most $\widehat{K} + (Q_{\text{wrap}} + 1)(\widehat{K} - 1) \leq (Q_{\text{wrap}} + 2)\widehat{K}$ times in expectation [[DBK⁺26](#), Theorem 10.17]. Let T_0 bound one execution of P followed by verification, and T_{step} the additional work per run, as conditional expected bounds uniform over the history at which a run starts. The additional work maintains the tables of observed digest queries, recomputes the terminal blocks, samples oracle answers, including sparse-coordinate streams conditional on their accepted challenges [[DBK⁺26](#), Lemma 10.2], and evaluates the combiners, tests, and source comparisons. Each takes expected polynomial work, so

$$T_E \leq (Q_{\text{wrap}} + 2)\widehat{K}(T_0 + T_{\text{step}}).$$

No prover run is truncated.

Adaptive parameters. If P also selects the supported parameter choice, its first message selects the checkpoint structure [[DBK⁺26](#), Remark 10.19]. Every parameter choice has $\rho \leq 2^{-128}$, so the query term holds with R_f , R_{coord} , and $\widehat{K}$ maximized over the supported choices, and the Module-SIS sum ranges over the union of [Section C.3](#).

C.6 Signing Simulation and Unforgeability

Write $X = \text{Frame}(\text{pk}, r, m)$ for the complete signature-hash input of [Section 3](#). Before the game starts, the reduction obtains two *presampled lists* from its Falcon challenger: signatures $\text{Sig}[i]$, each a preimage of a known target, which answer signing queries, and targets $\text{Tgt}[j]$ for a message it never asks to sign, which answer hash queries. A preimage of such a target gives a Falcon forgery. The internal H_{pk} is independent of the presampled lists. Internal framing uses its answer, while forgery conversion uses the recorded external input for the same target.

We use the hash-then-sign interface of Aardal et al. [[AAB⁺24](#), Lemma C.2 and Remark I.1] with FN-DSA's retry policy. Distinct fresh inputs give independent uniform targets. Every signing attempt draws a fresh

uniform salt and fresh sampler randomness, and its output law depends only on the key and the decoded target. Only the accepted attempt is returned. The honest signer and the Falcon challenger run this same algorithm, so their accepted target–preimage pairs have the same law, given by [Lemma C.4](#). The simulation needs only this equality, not uniform accepted targets.

Lemma C.4 (Signing Retry Laws). Fix a key and sampler. Let $K_y(u)$ be the probability that one attempt at target y returns coefficients u and passes both the norm and encoding checks, and put $a(y) = \sum_u K_y(u)$. Independent retries at a fixed target y with $a(y) > 0$ return u with probability

$$D_y(u) = \frac{K_y(u)}{a(y)}. \quad (20)$$

If instead every attempt has an independent uniform target $Y \sim \mu$, and $\bar{a} = \mathbb{E}_{Y \sim \mu} a(Y) > 0$, then attempts are accepted independently with probability $\bar{a}$, and the accepted pair has law

$$J(y, u) = \frac{\mu(y)K_y(u)}{\bar{a}}. \quad (21)$$

Its target has law $\nu(y) = \mu(y)a(y)/\bar{a}$, with $\Delta(\nu, \mu) = \mathbb{E}_{Y \sim \mu} |a(Y) - \bar{a}|/(2\bar{a})$, where Δ is total variation distance. Given the target y , the coefficients have law D_y .

Proof. At a fixed target the probability of returning u on attempt $j + 1$ is $(1 - a(y))^j K_y(u)$. Summing the geometric series proves [Equation \(20\)](#). With fresh targets, an attempt is accepted with probability $\sum_y \mu(y)a(y) = \bar{a}$, independently of earlier attempts, and returns (y, u) on attempt $j + 1$ with probability $(1 - \bar{a})^j \mu(y)K_y(u)$. Summing gives [Equation \(21\)](#). Marginalizing gives ν and conditioning gives D_y . Substitution in $\Delta(P, Q) = \frac{1}{2} \sum_y |P(y) - Q(y)|$ gives the distance. $\square$

Accepted FN-DSA targets are therefore uniform only when a is constant. The reduction below never needs them to be uniform, because its presampled pairs come from the same signer. A fixed-target signer at a fresh input instead returns a uniform target with coefficient law D_y . The common-salt reduction of [Section C.7](#) needs this property: it must answer hash queries for inputs it can sign and for inputs reserved for the forgery with the same target law.

Lemma C.5 (Extraction with Assigned Targets). Replace G in [Definition 5.7](#) by an algorithm that assigns hash targets and retains state, with the following properties.

- (i) Its initial auxiliary data and random choices are independent of the proof setup and proof oracles. It does not query those oracles itself.
- (ii) Each framed input receives one target per execution. Before returning a canonical records digest, the simulator assigns all targets for that list and retains them in its table. Any exposed hash stream decodes to its assigned target.
- (iii) Rewinding restores the algorithm’s state with the prover’s state. The output includes the original table snapshot, and verification and the extracted relation use the targets in that snapshot.

Under the remaining hypotheses of [Theorem 5.9](#), extraction has the same error bound. More precisely, write $\mathcal{R}_{\text{batch}}^C$ for the batch relation with public targets supplied as C , and C_ω for the original list’s targets. Use $\mathcal{R}_{\text{batch}}^{C_\omega}$ in the knowledge-error event of [Equation \(3\)](#). All work by this algorithm is included in the bounds on query counts and running time. Targets need not be uniform or independent of one another. In particular, the algorithm may use fixed presampled lists.

Proof. Expose the targets. In the converted prover of [Section C.1](#), augment each forwarded challenge input and the analysis verifier’s statement with the vector C assigned to its record list. Cache answers by the original input. If the record preimage is unavailable, answer locally as that prover does. This modification is used

only in the proof. It changes no transmitted message. Deleting C reverses the added encoding, so distinct original inputs cannot merge. The algorithm fixes C before the fresh challenge answer. Thus the simulation algorithm preserves the original view, and verification agrees outside the compression and key-hash events already included in the error bound.

Extract at a fixed input. All certificates combined at an augmented input use the same keys and C . The algebraic checks and checkpoint combiners use these public polynomials as fixed data. None requires a distribution for C . The seed-density event in [Lemma C.2](#) is simultaneous over h, C and the committed polynomials. Its bound therefore also applies when the algorithm's targets depend on the earlier execution history.

Preserve the original statement. Apply the proof of [Section C.5](#) to this fixed simulated prover. As for any stateful auxiliary algorithm in Akita's extraction experiment [[DBK⁺26](#), Definition 10.15], its tables and random choices are restored with its state at every rewind. Reruns preceding a target assignment may assign a different C , but then have a different augmented challenge input. Their certificates are not combined. The certificate for the original accepted input uses its original C . Retaining the original table therefore gives a witness for the original targets, with the same error and run bounds. No uniform-target or random-oracle assumption on the algorithm was used. $\square$

Signing State and Hash Streams. The state consists of a table $T_G[X] = (y, \text{tag}, j, S_X)$, indices of the next unused entries, a signature cache $\Sigma[X]$, and the signed-message set Q_S . The tag and index identify the presampled entry. Only y or the requested bytes of S_X are returned on a hash query. To assign target y to a new input X , construct S_X conditional on $\text{HashToPoint}(S_X) = y$. For each coefficient y_k , sample a geometric number of rejected 16-bit words with rejection probability $4091/65536$, uniform in $[61445, 65535]$, followed by a uniform word in $\{y_k + tq : 0 \leq t < 5\}$. Sample the unused tail independently. This gives the exact conditional stream in expected polynomial time. A first partial stream request also assigns the target. Later requests use the same stream.

[Figure 6](#) gives the uniform-salt simulator. Its `RecDigest` assigns every target of a canonical record list before returning the list's digest. The simulator of [Section C.7](#) uses the same digest procedure and the following rewind rule. Fix the presampled lists and simulator randomness. Restore T_G , indices of the next unused entries, Σ and Q_S with the prover state on every rewind. Make no new external signing query. Preserve the original state through final verification and save it before any rerun. An abort returns a rejecting output, and its original state is also retained. The request bounds ensure that neither list is exhausted.

The salt r chosen by Sign_{sim} takes the place of the accepted attempt's salt. The simulator does not reproduce the honest signer's rejected attempts. Their salts stay hidden, and [Lemma C.6](#) charges the events in which they would matter.

Lemma C.6 (Resettable Signing Simulation). Let F_0 be [Definition 5.10](#) and F_1 the same game using the uniform-salt simulator. With δ_{salt} bounded by the key-averaged expression in [Equation \(8\)](#),

$$\Pr[F_0 = 1] \leq \Pr[F_1 = 1] + \delta_{\text{salt}}.$$

The presampled lists are independent of proof randomness, and every rerun uses no new external signing query.

Proof. Condition on the signing key and write $\bar{a} = \bar{a}_{\text{sk}}$. Compare both games with a hybrid in which every signing attempt, internal or external, receives a fresh uniform target. Let N_I and N_E count the internal and external attempts, rejected ones included. Pad unused signing requests up to Q_S for this bound. In the hybrid each count is a sum of Q_S independent geometric variables with success probability $\bar{a}$ ([Lemma C.4](#)), so $\mathbb{E}N = Q_S/\bar{a}$ and $\mathbb{E}N^2 \leq 2Q_S^2/\bar{a}^2$.

Preparation	Prepare
for $i = 1, \dots, Q_S$ do $(u_i, y_i) \leftarrow$ external signature on m_0 , salt r_i $\text{Sig}[i] \leftarrow (u_i, y_i)$ if $r_i = r_{i'}$ for some $i \neq i'$ then $\text{bad}_{\text{sig}} \leftarrow \text{true}$; abort for $j = 1, \dots, Q_G$ do $\bar{y}_j \leftarrow$ external target of $m_1 \neq m_0$ at a fresh salt $\bar{r}_j$ $\text{Tgt}[j] \leftarrow (\bar{r}_j, \bar{y}_j)$	
Signing queries	$\text{Sign}_{\text{sim}}(m)$
$r \xleftarrow{\$} \{0, 1\}^{320}$; $X \leftarrow \text{Frame}(\text{pk}^*, r, m)$ if $X \in T_G$ then $\text{bad}_{\text{prog}} \leftarrow \text{true}$; abort $(u_i, y_i) \leftarrow$ next entry of Sig $S_X \leftarrow$ stream conditioned on y_i $T_G[X] \leftarrow (y_i, \text{sig}, i, S_X)$ $Q_S \leftarrow Q_S \cup \{m\}$ return $\text{EncodePadded}(r, u_{i,2})$	
Hash queries	$\text{Hash}(X)$
if $X \notin T_G$ then $(\bar{r}_j, \bar{y}_j) \leftarrow$ next entry of Tgt $S_X \leftarrow$ stream conditioned on $\bar{y}_j$ $T_G[X] \leftarrow (\bar{y}_j, \text{tgt}, j, S_X)$ return target or requested bytes of $T_G[X]$	
Record-list digest	$\text{RecDigest}(\mathcal{L})$
for each record i of $\mathcal{L}$, in order do $\text{Hash}(X_i)$ I counted in Q_G return $H_{\text{cmp}}(\mathcal{L})$	

Figure 6: Signing simulator for F_1, F_2 in Figure 7. Prepare uses the external Falcon interfaces before proof setup. Rewinding restores the state of Section C.6, including T_G and Q_S . Lemma C.6 proves $\Pr[F_0 = 1] \leq \Pr[F_1 = 1] + \delta_{\text{salt}}$.

Let bad be the union of three events. Two internal or two external attempts share a salt, which includes bad_{sig} . An internal attempt falls on an already exposed input, which includes bad_{prog} . The input of a rejected internal attempt is exposed later.

There are at most Q_G exposures, and the target-only entries use a message m_1 that the challenger never signs. Each salt is uniform and hidden until its attempt is accepted, so

$$\begin{aligned}
\Pr[\text{bad}] &\leq \frac{\mathbb{E}N_I^2 + \mathbb{E}N_E^2}{2^{321}} + \frac{2Q_G \mathbb{E}N_I}{2^{320}} \\
&\leq \frac{2Q_S}{\bar{a}} \left(\frac{Q_S}{\bar{a}} + Q_G \right) 2^{-320}.
\end{aligned}$$

The genuine game, the hybrid and the simulator agree until bad . Outside it, each accepted internal pair and each presampled pair has the law J of Lemma C.4, independently of everything else. The accepted salt is uniform and independent of its pair, rejected attempts never reach the adversary's view, and conditional streams preserve the raw hash view. Partial hash requests and the targets assigned by RecDigest are included in Q_G . Later aborts belong to the fixed simulated prover's behavior. These error probabilities are included only for the original execution. Averaging this conditional bound over key generation gives Equation (8). If

Unforgeability games	F_0, F_1, F_2
$(h^*, \text{sk}^*) \leftarrow \text{Falcon.KeyGen}$ Prepare $/ F_1, F_2$: simulated oracles $\mathcal{H} \leftarrow \text{Definition 5.6}; \text{pp} \leftarrow \text{Setup}^{H_{\text{setup}}}$ $(\omega, w) \leftarrow \mathcal{E}(\mathcal{A}, \text{pp}) \quad / \omega \text{ as in AggForge}$ $(\mathcal{L}, \pi, a, b) \leftarrow \omega$ $\text{Win} \leftarrow \mathbf{1}[b = 1 \wedge \text{Fresh}(\mathcal{L}, Q_S) \neq \emptyset]$ Ext $\leftarrow \mathbf{1}[(\mathcal{L}, w) \in \mathcal{R}_{\text{batch}}]; \text{return Win} \cdot \text{Ext} \quad / F_2$	
return Win	
Conversion to a Falcon forgery	$\text{Convert}(\omega, w, T_G, Q_S)$
$(\mathcal{L}, \pi, a, b) \leftarrow \omega$ if $b = 0 \vee (\mathcal{L}, w) \notin \mathcal{R}_{\text{batch}}$ then return $\perp$ if $\text{Fresh}(\mathcal{L}, Q_S) = \emptyset$ then return $\perp$ $j \leftarrow \min \text{Fresh}(\mathcal{L}, Q_S)$ $(\bar{y}_k, \text{tgt}, k, \mathcal{S}_{X_j}) \leftarrow T_G[X_j]$ return $(m_1, \text{EncodePadded}(\bar{r}_k, s_{2,j}))$	
Transition bounds. Write $p_i^F = \Pr[F_i = 1]$. Then	
$p_0^F \leq p_1^F + \delta_{\text{salt}} \quad (\text{signing simulation, Lemma C.6}),$ $p_1^F \leq p_2^F + \varepsilon_{\text{ext}}^{\text{ROM}} \quad (\text{extraction, Theorem 5.9 and Lemma C.5}),$ $p_2^F \leq \Pr[\text{EUF-CMA}_{\text{FN-DSA}}^+(\mathcal{B}) = 1] \quad (\text{Convert, Section C.6}).$	

Figure 7: Unforgeability proof. In F_0 , $\mathcal{A}$ uses the FN-DSA signer of Section 3 and the genuine signature hash. F_1, F_2 use Figure 6. The extractor $\mathcal{E}$ of Section C.5 preserves the original execution, and Theorem 5.9 bounds its error. Reduction $\mathcal{B}$ applies Convert to its outputs and the original snapshots of T_G, Q_S . Here $s_{2,j}$ is the second signature component of record j in w .

$\bar{a}_{\text{sk}} \geq a_{\min} > 0$ for every generated key, one may instead use $\delta_{\text{salt}} \leq (2Q_S/a_{\min})(Q_S/a_{\min} + Q_G)2^{-320}$. $\square$

Lemma C.7 (Targets Preserved under Rewinding). Under the observed records-digest model, with targets assigned before returning the digest, outside the record-compression event in $\varepsilon_{\text{bind}}$, the targets of the original accepted list are fixed before its first challenge query. All transcripts combined at that input use those targets.

Proof. The record-list digest procedure assigns the targets before returning the digest. Using the digest earlier, or using a different list with the same digest, is included in the compression event of Section C.1. Outside that event, the simulation algorithm of Lemma C.5 includes these targets in the expanded challenge input. Rewinding at that input preserves the earlier assignments. Thus, for its combined transcript collection $\mathcal{T}$,

$$\forall \tau, \tau' \in \mathcal{T}, \quad C^\tau = C^{\tau'}.$$

The extracted preimage therefore has the original target. This exception, and the key-hash event of ε_{pk} , are already included in $\varepsilon_{\text{ext}}^{\text{ROM}}$. $\square$

Forgery Games and Conversion. The games F_0, F_1, F_2 and the conversion Convert are those of Figure 7. The extractor retains the original ω and state snapshot. Lemma C.5 applies to the fixed presampled lists, random choices for the conditional streams and restored tables. Together with Lemma C.6, it gives

$$\begin{aligned} \Pr[F_0 = 1] &\leq \Pr[F_1 = 1] + \delta_{\text{salt}}, \\ \Pr[F_1 = 1] &\leq \Pr[F_2 = 1] + \varepsilon_{\text{ext}}^{\text{ROM}}. \end{aligned}$$

Freshness ensures that X_j was not assigned by signing, so its entry is in Tgt. By [Lemma C.7](#), w supplies a preimage of $\bar{y}_k$. [Lemma 3.1](#) gives a valid signature. Only m_0 was externally signed. Consequently

$$\Pr[F_2 = 1] \leq \Pr[\text{EUF-CMA}_{\text{FN-DSA}}^+(\mathcal{B}) = 1].$$

No index guess is needed. The reduction costs extraction, key generation and preparing the signing data.

C.7 Common Salts with Fixed-Target Signing

This variant changes the signing oracle, not the aggregate relation. It uses the partition argument of Aardal et al. [[AAB⁺24](#), Lemma C.3], with an explicit salt argument and the assigned-target extraction lemma above. All claims below are in the classical ROM.

Signing Model. Fix the parameter set, message context and canonical key encoding of [Section 3](#). Write $X = \text{Frame}(pk, r, m)$ for the complete signature-hash input under that interface, including its key binding where applicable. The caller supplies any 320-bit salt r , possibly the same public value for every request. FN-DSA’s retries cannot keep this salt, because each draws a new one. The oracle therefore uses the *fixed-target signer*: on the first request (r, m) under the challenge key, it retries the sampler at the target $G(X)$ until an attempt passes the norm and encoding checks, as Falcon v1.2 does [[FHK⁺20](#), Algorithm 10], and caches the resulting padded signature. Every repetition returns the same bytes.

The cache key includes the complete framed input, and its state persists whenever that input can recur. A restricted interface may instead reject repetitions. The salt is distinct from the secret randomness used to sample a preimage. [Corollary C.11](#) replaces the cache by sampler randomness derived from a signing-key secret and X .

We state the variant for the key-prefixed framing of [Section 3](#) and the independent oracle families of [Theorem 5.9](#). Key-prefixed framing alone does not establish this independence: signature-hash inputs can also be canonical key encodings, so key hashing and signature hashing can query the same SHAKE256 input. The theorem below assumes separate oracle families; the cSHAKE domains used for Aerie’s other hashes do not resolve this remaining overlap.

We use the hash-then-sign model already needed in [Section C.6](#): distinct hash inputs have independent uniform targets, and, conditional on a target y , an honest signature has the fixed-target law D_y of [Equation \(20\)](#), which depends only on the key and target. Independent first signing requests use independent sampler randomness. The external challenger is the fixed-target signer with uniform salts, so its signatures also have uniform targets and conditional law D_y . Signature-hash streams are independent of the proof oracles. Any approximation to these modeled distributions must be bounded separately, since the argument below does not certify it. The freshness condition remains that of [Section 5.7](#): the selected message was never submitted to the challenge-key signing oracle at any salt.

Fixed-Target Assumption. Write $\text{EUF-CMA}_{\text{FT}}^+$ for the game EUF-CMA^+ of [Theorem 5.11](#) with the fixed-target signer as challenger. Each of its signatures draws one uniform salt and retries the sampler at that salt’s target.

Assumption C.8 (Fixed-Target Falcon). For the parameter sets and key-prefixed framing of [Section 3](#), every expected-polynomial-time adversary $\mathcal{B}$ has negligible advantage $\Pr[\text{EUF-CMA}_{\text{FT}}^+(\mathcal{B}) = 1]$ in the random-oracle model.

FN-DSA draws a fresh salt on every attempt, and Fouque et al. reduce its unforgeability to a lattice assumption in the random-oracle model [[FGdG⁺26](#)]. Their proof does not cover the fixed-target signer. Their games program each attempt’s target as the image of an unconditioned Gaussian vector and apply the norm check afterwards. If a reduction keeps only accepted vectors at a fixed target, the targets it programs have the law ν of [Lemma C.4](#) rather than the uniform law. We know of no bound on the variation of $a(y)$ across

targets that would make this difference negligible, and we know of no attack on the fixed-target signer. Its retry loop is Algorithm 10 of the Falcon specification [FHK⁺20]. The deterministic Falcon signer of Lazar and Peikert, which Algorand deploys, uses the same loop with one fixed salt for every message [LP26].

Theorem C.9 (Conditional Cached Shared-Salt Reduction). Let an aggregate adversary make at most Q_S signing requests and expose at most Q_G distinct signature-hash streams, including streams needed for final verification and the targets assigned by RecDigest (Section C.6). Q_S and Q_G are worst-case bounds that hold on every execution, including rewind ones. The digest is modeled as in that appendix. Under the signing model above and the assigned-target extraction guarantee of Lemma C.5, there is an EUF-CMA_{FT}⁺ adversary $\mathcal{B}$ such that

$$\begin{aligned} \Pr[\text{AggForge}_{\text{shared}}(\mathcal{A}) = 1] \\ \leq \rho_S (\Pr[\text{EUF-CMA}_{\text{FT}}^+(\mathcal{B}) = 1] \\ + \varepsilon_{\text{ext}}^{\text{ROM}}(P) + \delta_{\text{sim}}), \end{aligned} \quad (22)$$

where P is the fixed simulated prover defined below, $\varepsilon_{\text{ext}}^{\text{ROM}}(P)$ is the bound of Equation (7) for P , $\rho_S = 1$ if $Q_S = 0$, and, for $Q_S \geq 1$,

$$\rho_S = \frac{(Q_S + 1)Q_S^{Q_S+1}}{Q_S^{Q_S}} \leq e(Q_S + 1).$$

Here $\delta_{\text{sim}} = 0$ for the cached fixed-target signer, which meets the signing model exactly. It differs from the challenger only in taking the caller's salt. For any other signer, δ_{sim} is the statistical distance of the complete experiment from this model, covering both the internal signer and the presampled signatures.

The reduction makes at most $Q_S + Q_G$ external signing queries before extraction. Its expected cost includes $T_E(P)$, key generation and preparing the signing data. All computational terms, including the Module-SIS reduction costs T_I , are evaluated at these actual costs. The conditional-stream samplers have expected rather than strict running time, so these are expected costs.

Proof. Put $\alpha = \Pr[S_0 = 1]$, where S_0 is the genuine cached-signing game, and let $\lambda_S = Q_S/(Q_S + 1)$, including $\lambda_S = 0$ when $Q_S = 0$. Use the state, conditional streams, record-list digest procedure and rewind rule of Section C.6.

The simulator P consists of Prepare_c , Hash_c and Sign_c in Figure 8.

Distinct external messages give distinct hash inputs, even if signing salts coincide. Hash-output collisions are retained. The presampled lists are independent of the proof setup and proof oracles, including internal H_{pk} .

Games. The games $S_0, \dots, S_3$ are those of Figure 8, with the original cached-signing log. The index j^* is selected without consulting labels. S_3 applies same-execution extraction to the simulated prover and retains the original state and j^* .

Partition transition. In the exact model, couple S_1 to the simulator by sampling an unused conditional preimage even for its target-only entries. Both labels then have the same target and conditional-preimage laws. Fix a winning S_1 transcript. Let $d \leq Q_S$ count distinct inputs first exposed by hashing and subsequently signed, including exposures through the record-list digest procedure. The required labels have conditional probability

$$\lambda_S^d (1 - \lambda_S) \geq \lambda_S^{Q_S} (1 - \lambda_S) =: a_S.$$

The selected fresh input was never signed, so its label is independent of the d signing labels. First exposures by signing need no label guess. Hidden labels make this argument valid for adaptive requests. Set $a_S = 1$ when $Q_S = 0$. Charging the complete original-experiment distance once, and using $a_S \leq 1$, yields

$$\Pr[S_2 = 1] \geq a_S \alpha - \delta_{\text{sim}}.$$

Preparation	Prepare_c
choose distinct messages $M_1, \dots, M_{Q_S+Q_G}$, $U_1, \dots, U_{Q_G}$ for $i = 1, \dots, Q_S + Q_G$ do $\text{Sig}[i] \leftarrow (u_i, y_i)$, external signature on M_i for $k = 1, \dots, Q_G$ do $\text{Tgt}[k] \leftarrow \bar{y}_k$, external target of U_k at salt 0^{320}	
Hash queries	$\text{Hash}_c(X)$
if $X \notin T_G$ then $\text{tag} \leftarrow \text{sig}$ with probability λ_S , else tgt take the next entry of Sig or Tgt , by tag $T_G[X] \leftarrow (\text{target}, \text{tag}, \text{index}, S_X)$ return target or requested bytes of $T_G[X]$	
Cached signing	$\text{Sign}_c(r, m)$
$X \leftarrow \text{Frame}(\text{pk}^*, r, m)$; $Q_S \leftarrow Q_S \cup \{m\}$ if $\Sigma[X] \neq \perp$ then return $\Sigma[X]$ if $X \notin T_G$ then $(u_i, y_i) \leftarrow$ next entry of Sig $T_G[X] \leftarrow (y_i, \text{sig}, i, S_X)$ if $\text{tag}(X) = \text{tgt}$ then abort $\Sigma[X] \leftarrow \text{EncodePadded}(r, u_{i,2})$ / recorded index i return $\Sigma[X]$	
Shared-salt games	S_0, S_1, S_2, S_3
$(h^*, \text{sk}^*) \leftarrow \text{Falcon.KeyGen}$ Prepare_c / S_2, S_3 $\mathcal{H} \leftarrow \text{Definition 5.6}$; $\text{pp} \leftarrow \text{Setup}^{H_{\text{setup}}}$ $(\omega, w) \leftarrow \mathcal{E}(\mathcal{A}, \text{pp})$ oracles: cached signer and G / S_0 and hidden labels on first hash exposure / S_1 $\text{Sign}_c, \text{Hash}_c, \text{RecDigest}$ / S_2, S_3 $(\mathcal{L}, \pi, a, b) \leftarrow \omega$ $\text{Win} \leftarrow \mathbf{1}[b = 1 \wedge \text{Fresh}(\mathcal{L}, Q_S) \neq \emptyset]$ $\text{Ext} \leftarrow \mathbf{1}[(\mathcal{L}, w) \in \mathcal{R}_{\text{batch}}]$ if $\text{Win} = 0$ then return 0 $j^* \leftarrow \min \text{Fresh}(\mathcal{L}, Q_S)$; $X^* \leftarrow$ framed input of j^* $\text{Win} \leftarrow \text{Win} \cdot \mathbf{1}[\text{tag}(X^*) = \text{tgt}]$ / S_2, S_3 return $\text{Win} \cdot \text{Ext}$ / S_3 return Win	
Transition bounds. Write $p_i^S = \Pr[S_i = 1]$ and $a_S = \rho_S^{-1}$ as in Theorem C.9 . Its proof gives $ \begin{aligned} p_1^S &= p_0^S, & p_2^S &\geq a_S p_1^S - \delta_{\text{sim}}, \\ p_3^S &\geq p_2^S - \varepsilon_{\text{ext}}^{\text{ROM}}(P), & p_3^S &\leq \Pr[\text{EUF-CMA}_{\text{FT}}^+(\mathcal{B}) = 1]. \end{aligned} $	

Figure 8: Shared-salt unforgeability proof. On input an aggregate adversary $\mathcal{A}$ with caller-chosen salts and cached fixed-target signing, the games use $\mathcal{E}$ from [Section C.5](#). Assigned-target extraction ([Lemma C.5](#)) bounds $p_2^S - p_3^S$, and [Section C.7](#) gives the conversion. Here $\lambda_S = Q_S / (Q_S + 1)$, RecDigest is that of [Figure 6](#) with Hash_c . Prepare_c runs before all proof randomness. An abort, or $\text{Win} = 0$, gives value zero. A run uses at most $Q_S + Q_G$ preimage entries and Q_G target-only entries, and a first partial hash request also assigns a tag.

Extraction and conversion. The fixed presampled lists and random choices for the conditional streams are sampled before the proof randomness. The algorithm assigning hash targets makes no proof-oracle query. The targets assigned by RecDigest and the state restored on rewinding satisfy [Lemma C.5](#). Hence

$$\Pr[S_3 = 1] \geq \Pr[S_2 = 1] - \varepsilon_{\text{ext}}^{\text{ROM}}(P).$$

On $S_3 = 1$, read the target-only index k of X^* from the original T_G . Output $\text{EncodePadded}(0^{320}, s_{2,j^*})$ on U_k . By [Lemmas 3.1](#) and [C.7](#), it verifies for the external target $\bar{y}_k$. U_k was never externally signed. No new external signing query is made during extraction. Thus

$$\begin{aligned} \Pr[\text{EUF-CMA}_{\text{FT}}^+(\mathcal{B}) = 1] &\geq \Pr[S_3 = 1] \\ &\geq a_S \alpha - \delta_{\text{sim}} - \varepsilon_{\text{ext}}^{\text{ROM}}(P). \end{aligned}$$

For $Q_S \geq 1$, the chosen λ_S maximizes a_S and $1/a_S = \rho_S \leq e(Q_S + 1)$. For $Q_S = 0$, $a_S = \rho_S = 1$. Dividing by a_S gives [Equation \(22\)](#), including the multiplication of both error terms. $\square$

Remark C.10 (Size of the Partition Loss). The factor ρ_S multiplies the hash-query part of [Equation \(7\)](#). For a key that answers $Q_S = 2^{24}$ signing requests, the loss is about $2^{25.4}$, each prover query costs about $2^{-102.6}$, and the final verifier's term $1207 \cdot 2^{-128}$ becomes about $2^{-92.3}$. At $Q_S = 2^{64}$ the three figures are $2^{65.4}$, $2^{-62.6}$, and $2^{-52.3}$.

Deterministic Signing. The cache can be replaced by derandomization [[GPV08](#), [LP26](#)]. At key generation the *deterministic common-salt signer* also samples a uniform 256-bit key k , stored with the private key and used only for this purpose. On a request it runs the fixed-target signer at $G(X)$ with sampler randomness read from $F(k, X)$, where F is an extendable-output function under its own domain, such as cSHAKE256 with a dedicated customization string. It keeps no state, and a repeated input returns the same signature. Write $\text{AggForge}_{\text{det}}$ for the game of [Theorem C.9](#) with this signer, and model F as a random oracle independent of the other oracle families.

Corollary C.11 (Deterministic Common-Salt Signing). Let an aggregate adversary $\mathcal{A}$ make at most Q_F queries to F . Then

$$\begin{aligned} \Pr[\text{AggForge}_{\text{det}}(\mathcal{A}) = 1] \\ \leq \Pr[\text{AggForge}_{\text{shared}}(\mathcal{A}') = 1] + Q_F 2^{-256}, \end{aligned}$$

where $\mathcal{A}'$ runs $\mathcal{A}$ and answers its F queries by lazy sampling. $\mathcal{A}'$ has the signing-request and hash-stream bounds of $\mathcal{A}$, so [Theorem C.9](#) bounds the first term.

Proof. The signer queries F only at inputs (k, X) . Let bad_F be the event that $\mathcal{A}$ queries F at an input with key part k . Outside bad_F , the sampler randomness is uniform and independent for distinct framed inputs and equal for equal ones, which is the cached signer's, so the two games agree until bad_F . Until then the view of $\mathcal{A}$ is independent of k , so each query has key part k with probability 2^{-256} , and $\Pr[\text{bad}_F] \leq Q_F 2^{-256}$. $\square$

Lazar and Peikert key F with the Falcon private key itself [[LP26](#)]. That key is not independent of the public key, so the proof above does not apply to their keying as stated. The corollary also assumes one sampler per key. A signer that runs functionally inequivalent samplers on one input, such as two implementations, versions or floating-point configurations, can return two preimages of one target, which the model excludes.

The two salt modes therefore use different signers. [Theorem 5.11](#) uses FN-DSA for per-signature salts, which the analysis of Fouque et al. covers [[FGdG⁺26](#)]. A common salt fixes the target, so the common-salt signer retries at that target, and [Theorem C.9](#) assumes that this signer is unforgeable ([Assumption C.8](#)).

[Theorem C.9](#) permits a salt to be known before messages are chosen, and permits one key to sign different messages under that salt. It does not establish security for independently resampling a signature on every

repetition of the same framed input. The cache, or the derandomization of [Corollary C.11](#), excludes that case. The theorem does not strengthen the concrete or quantum security of the imported extractor: the runtime qualification following [Theorem 5.11](#) still applies, and the factor ρ_S also multiplies the hash-query part of $\varepsilon_{\text{ext}}^{\text{ROM}}(P)$, so that part is no longer bounded by 2^{-128} per query once $Q_S \geq 1$.