ANNULAR: Post-Quantum Private Transactions from Isogenies with Recursive Batch Verification

Dustin Ray

Anchorage Digital

Abstract. ANNULAR is a shielded pool built on isogeny-based primitives. Notes are committed in a Merkle tree and spent by nullifier with hidden amounts, following the structure of Zcash's Sapling, but every primitive is post-quantum: spends are authorised by the prime-degree PRISM signature of Basso et al., note ciphertexts are keyed by the MIKE non-interactive key exchange, and each transaction carries a STARK proved natively over the isogeny field. The transaction proofs of a block are then folded into a single recursive proof. A ledger need record and verify only this one proof, 426 KB in 15 ms, whatever the batch holds, and check each transaction's two signatures alongside it. A transaction's body is the size of Sapling's at the same memo length (2,685 bytes against 2,756 with a 512-byte memo); its share of the batch proof adds under 500 bytes.

A move to post-quantum primitives is generally expected to raise a pool's operating cost. We argue that the post-quantum change and the scaling change are complementary: with a STARK as the proof system, proofs fold, and with isogeny keys, the per-transaction data remains small. The cost moves to the sender, whose proof takes 1.4 s on sixteen cores, and to one aggregator per batch, which we measure: 1,248 transactions in 99 s on one GPU. Each proof in the tree carries 100 bits of round-by-round soundness and the batch about 90 under a union bound; under Fiat–Shamir that is the work an adversary needs. The constraints of the leaf circuit are proven in Lean 4 to encode the spend statement, while the soundness of the STARK itself rests on a paper argument. The scheme's claims rest on the signature's existential unforgeability; its strong unforgeability, argued by the construction's authors and not yet proved, is not relied on.

Keywords: post-quantum · private transactions · isogenies · STARK · recursive aggregation · formal verification

1 Introduction

A private transaction scheme hides a payment's sender, recipient and amount behind a commitment, a nullifier and a proof. The design is Zerocash's [BCG+14]: notes committed in a Merkle tree, each spent once by publishing its nullifier with a proof of membership and balance. The instances of that design in use rest on the discrete logarithm three times over, in the proof system, in the key agreement that delivers a note's plaintext and in the spend signature [HBHW26]. A quantum computer of the size the literature discusses breaks all three at once. A post-quantum instance needs three primitives: a signature whose keys re-randomise, a non-interactive key exchange or a KEM for the note ciphertexts, and a proof system. This paper's are the prime-degree PRISM signature (Section 5 of [BBC+26a] with the salt of its Section 6, implemented in [Ray26]), the MIKE key exchange [BDD+26] and a STARK over the signature's own field. The re-randomisation is what makes a note's spend key unlinkable to its owner's address. The replacements on offer are expensive where a verifier counts. A transparent post-quantum proof is larger than a pairing-based one by

more than an order of magnitude, and with one proof per transaction the verifier's cost is linear in the batch.

Tables 1 and 2 are the point of this paper. ANNULAR is the first private transaction scheme on isogeny primitives with an implemented prover. Of the schemes compared, Neptune Cash verifies a batch with one proof, at the price of no signatures and no reusable addresses; the rest verify per transaction, and a batch's verification grows with it. ANNULAR moves the proofs into one prover, and that folding is what makes the scheme practical. The proof the verifier checks is one per batch and constant in size and time, with post-quantum primitives and a transaction body the size of the elliptic-curve instance's at the same memo. The proof's share is its post-quantum premium. The verifier checks the batch proof, 15 ms, and each transaction's two signatures, 0.6 s per batch across a host's threads. The price is one GPU prover per batch, and the rest of the paper is what it takes to get that row.

Table 1: Verifier cost per transaction and per batch of 1,248, by scheme: the bytes per transaction, the verification of a batch of 1,248 transactions, and who proves at what cost. Every row is the 2-in-2-out shape; n/p is not published.

scheme	bytes per transaction	batch verification, 1,248 transactions	prover cost per transaction
Sapling	2,756 all in (1,796 at a 32-byte memo)	35 to 50 s	n/p (the sender)
Orchard	9,160	about 37 s	n/p (the sender)
MatRiCT+	47 KB (proof only)	2.5 to 10 s	100 to 261 ms (the sender)
Soul of Satoshi	n/p	over 12 s	0.70 s on a laptop GPU, 6.3 s on a CPU (the sender)
Neptune Cash	n/p (one merged transaction)	n/p	about 185 s per batch (the composer)
ANNULAR	body 1,725 (2,685 at a 512-byte memo); all in 3,180 at 860 transactions, 3,027 at 1,248	14 ms per transaction for its two signatures on one thread (0.5 ms across 32) plus 15 ms per batch for the proof: 17.6 s and 15 ms for the batch on one thread (0.60 s on 32)	0.079 s on one RTX PRO 6000 (the aggregator)

The figures are as each source publishes them, unbatched on both sides; the arithmetic behind each cell, the batched figures that exist and the sources are in Appendix A, the notes to this table.

Table 2: The properties, scheme by scheme. The signature column decides whether the verification column of Table 1 is constant: a signature outside the proof is checked by the verifier per transaction.

scheme	post-quantum	privacy	signature in the proof
Sapling	no	yes	no
Orchard	no	yes	no
MatRiCT+	yes	yes	yes
Soul of Satoshi	yes	yes	yes
Neptune Cash	yes	yes	yes
ANNULAR	yes	yes	no

MatRiCT+'s ring is 10 to 50; Neptune Cash hides amounts and origins through mutator sets; Soul of Satoshi's private transactions are opt-in. Sources as in Table 1.

Three facts undo the expected cost. Proofs fold, so a batch's proofs are verified inside one recursive proof whose size and verification time do not depend on the batch. An isogeny key is tens of bytes and also signs, so the data a transaction carries stays where it is today. What remains is one prover per batch, on one GPU. Section 2 states the three in symbols and Section 3 gives the primitives. Sections 4 to 6 are the scheme, the

transaction's proof and the tree, Section 7 the security arithmetic, Sections 8 and 9 the verification method and the measurements. Sections 10 to 12 are the comparison, the surrounding work and what is left, and Appendix D the method of the work.

Contributions.

1. The scheme: the first private transaction scheme on isogeny primitives with an implemented prover, and the cost argument that its post-quantum upgrade and its scaling upgrade are complementary. With a folding proof and a NIKE the verifier pays the proof once per batch and the data floor it pays today (Sections 2 and 9).
2. The specification: Zerocash's notes and nullifiers with viewing keys, outgoing records and disclosures, specified to the byte, with the two isogeny primitives in the roles an elliptic curve has today (Section 4).
3. The proofs: a native-field STARK with a subfield check that keeps every hash input in $\mathbb{F}_p$ (Section 5), and recursive batch aggregation on one card with a constant-size top proof (Sections 6 and 9).
4. The verification method: every proof at a stated number of proven bits with a union bound at the batch (Section 7). The leaf's constraints are proven in Lean 4 to encode the spend statement, the provers are gated by byte identity, and forged-proof tables stand at every level (Section 8).
5. The comparison and a design table over authorisation and delivery (Section 10; the table in Appendix A).

What this paper is not. It is not a system design. The scheme is specified against the verifier of Section 4, which supplies two predicates, a valid anchor and an unspent nullifier, and nothing is said about how that verifier is replicated or ordered.

2 Verification cost of a batch

Take a batch of n transactions. Under per-transaction proofs the verifier checks each proof, b_p bytes in time t_v, and checks each transaction's two signatures at t_s each. Every transaction also carries data the proof does not replace: an ephemeral key and a ciphertext per output and the outgoing record, d bytes in all. The verifier's cost and the bytes for the batch are

$$C_1(n) = n\,(t_v + 2t_s), \qquad D_1(n) = n\,(b_p + d). \tag{1}$$

Under a folded batch proof the transaction proofs are verified inside one proof of B bytes that the verifier checks in time T, neither depending on n. The signatures are checked per transaction in both models. The batch then costs

$$C_2(n) = T + 2n\,t_s, \qquad D_2(n) = B + n\,d, \tag{2}$$

so a transaction's share of the proof's bytes is B/n and of its verification T/n. Both fall with the batch; d does not. The claim is then that the data floor d is unchanged by this paper's primitives and the post-quantum premium is the proof share B/n, which falls with the batch. The data floor d is the term the primitives decide. With an elliptic curve it is keys of tens of bytes and ciphertexts of the plaintext's size plus a tag. With a lattice key encapsulation each output adds a ciphertext of hundreds of bytes and an 800-byte key [Nat24b], and d grows by more than the proof amortisation saves. With an isogeny NIKE a key is a curve coefficient, tens of bytes again, so d stays at today's floor. Because the same key re-randomises and signs, authorisation adds a signature of two hundred bytes

and no second key. The prover's cost moves rather than vanishes. The transaction's own proof costs p_ℓ at the sender, and the batch's aggregation

$$P(n) = \frac{n\,p_\ell}{c} + A(n) \tag{3}$$

runs the n leaf proofs across c cores and the aggregation $A(n)$, linear in n at a measured rate, on one GPU. Table 6 sets the terms side by side; their measured values are in Section 9 and Table 4.

3 Preliminaries

Fields and parameters. Everything lives in one prime field $\mathbb{F}_p$ or its quadratic extension $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$, $i^2 = -1$. Two parameter sets are implemented and measured (Table 7). The first is SQIsign's level I prime $p = 3 \cdot 2^{324} - 1$, the signature's published prime [Ray26], on which every measured row stands unless it says otherwise; MIKE runs on it under a parameter set this work adds (below). The second is MIKE's rigorous level I prime $p = 117 \cdot 2^{374} - 1$. Curves are supersingular Montgomery curves $y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{p^2}$, carried as the coefficient A of a canonical model.

The prime-degree PRISM signature. The signature is the prime-degree variant of PRISM [BBC+25], Section 5 of [BBC+26a] with the salt of its Section 6. It is hash-and-sign: the message is hashed to a prime degree $2^{2e-1} - s^2$, the signature is one isogeny of that degree from the public curve, and verification goes through a two-dimensional embedding with a scalar check. Its authors describe the construction without implementing or naming it, and argue its strong unforgeability under stronger assumptions without a proof. The prime degree removes the auxiliary isogeny whose factors a forger could swap [Ray26]. The variant is used because its authors argue a property the two-dimensional scheme provably lacks, at no extra cost here; the claims of Section 7 rest on its existential unforgeability alone. [Ray26] is its first implementation, with a canonical wire format and the measurements this paper uses. The keys re-randomise, following the randomizable keys of [BBC+26b], as [Ray26] implements them. RandPK(E, ρ) takes a public curve and 32 bytes to a public curve, and RandSK takes the matching secret along. The pair is distributed as a fresh key pair, and only the holder of the original secret can sign under the new key. Mode A is two hops of 2^{f-2}-isogenies, f the two-adicity. Mode B, the one a spend uses, is a radical walk of $2f - 4$ radical 2-isogenies. The root at each step is chosen by a bit of a SHAKE256 stream over the curve and the 32 bytes. Each step is one of the two non-backtracking 2-isogenies from the previous one, so the composite is a cyclic 2^{2f-4}-isogeny whichever roots are taken. That lets the circuit prove a walk exists without fixing which (Section 5).

MIKE. MIKE is a non-interactive key exchange on a prime of the same shape: a public key is a curve, a secret a scalar, and two parties' shared secret 32 bytes [MIK26c, MIK26b]. A recipient derives its scanning key from its seed through the crate's seeded key generation. Its paper states its assumption as the CDH problem for the rank-2 Hermitian module action [BDD+26]; its announcement says its passive security reduces to the supersingular isogeny problem in the algebraic isogeny model [Rob26a]. It ships two level I sets, both sized against Wesolowski's $p^{1/3+o(1)}$ algorithm for that problem [Wes26]. The fast set is on $633 \cdot 2^{308} - 1$ with walk length $e = 228$ and 80-byte keys, the rigorous set on $117 \cdot 2^{374} - 1$ with $e = 372$ and 96-byte keys [Rob26a, MIK26a]. Its announcement says the sets grew after [Wes26] from those of the team's July 2026 message [Rob26a]. The rigorous set takes a prime of 3λ bits; the fast set counts the algorithm's logarithmic factors under a memory bound, as SQIsign's round 3 did.

MIKE at the signature's prime. This paper runs MIKE's Rust implementation, at the commit the bibliography pins [MIK26c], on SQIsign's level I prime $3 \cdot 2^{324} - 1$ with $e = 228$. The set is this work's addition, made so that one key serves both roles: a MIKE key pair on it is a prime-degree PRISM key pair, the signing key derived from the MIKE secret, a property of the implementation [Ray26]. It is this paper's instantiation, not a MIKE parameter set. The crate's key validation and field arithmetic run unchanged, the arithmetic being the fp2 construction upstream uses for its own moduli. A known-answer test against the SageMath implementation [MIK26b] holds byte for byte. MIKE's authors state that it runs at SQIsign's primes at the cost of one more word per field element [Rob26a]. The prime has 326 bits against the fast set's 318, and $e = 228$ is the fast set's walk length, so the set inherits the fast set's assumption that its public keys are indistinguishable from random [MIK26a, Rob26a]. The walk covers a fraction of the curves set by e against $\log_2 p$: 228 against 326 bits is 0.70, close to the fast set's 228 against 318, 0.72. The set therefore sits where the fast set sits, not between the fast and the rigorous one. Its public keys are then 82 bytes, two 41-byte field elements, against 80 on the fast set and 96 on the rigorous one (Table 7). On this prime its key generation is 4.4 megacycles and a shared secret 38.2, against 2.9 and 23.0 on the fast set and 7.7 and 64.3 on the rigorous one. The three rows are one implementation, the Rust crate with its generic field arithmetic, on one machine, a Xeon at 2.8 GHz on one thread [Ray26].

The rows compare the primes on one implementation and machine, not implementations. The site's C implementation, with assembly field arithmetic, reports 0.651 and 4.86 ms on the fast set and 1.49 and 10.5 ms on the rigorous one on a Ryzen 7 PRO 7840U at 3.3 GHz [MIK26a].

A STARK over $\mathbb{F}_{p^2}$. The leaf prover is a native-field STARK: an AIR over $\mathbb{F}_{p^2}$ with a trace of 2^{11} or 2^{12} rows committed in Merkle trees, the DEEP opening at ζ [BGKS19], and FRI over $\mathbb{F}_{p^2}$ [BCI+20]. FRI needs a multiplicative subgroup of two-power order. $\mathbb{F}_p^*$ has none of useful size, but $p + 1$ has two-adicity 324, so the domains lie in the norm-one torus of $\mathbb{F}_{p^2}$, where $\overline{x} = x^{-1}$. The subfield check of Section 5 uses that fact. FRI folds by eight and queries are drawn with replacement. Fiat–Shamir uses Keccak-256 for a proof verified directly and a Poseidon2 duplex sponge for a proof a recursive verifier consumes; the round-by-round argument is [BGK+23]. Zero knowledge is Haböck and Kindi's construction [HK24]: every witness column is masked with h random coefficients, and h is counted against every opening the verifier makes. Post-quantum soundness of this kind of argument, in the quantum random oracle model, is [CMS19].

The three FRI regimes. A FRI query at blowup 2^b buys b bits under the conjecture that the proximity gap holds up to list-decoding capacity. It buys $b/2$ under the proven Johnson-radius bound, less $\log_2(1 + \eta/\sqrt{\rho})$ at the $\eta > 0$ the bound needs (Section 7), and $-\log_2((1 + \rho)/2)$ under unique decoding, 0.83 at blowup 8. The capacity conjecture is refuted as stated [CS25, KKH26]; the conjectured figures are reported once, in Appendix A.

Plonky3 over Baby Bear. The recursion's prover is Plonky3's batch STARK [Plo26]: several AIRs proved together with LogUp buses between them. Poseidon2 over Baby Bear serves at width 16 for the Merkle paths and the transcript and at width 24 in the verifier's rows. The tree's extensions, blowups and query counts are in Table 11 and the field terms of their bounds in Table 3. The leaf's 326-bit arithmetic is emulated in 31 limbs with six range-check groups.

Poseidon2. The leaf's hash is Poseidon2 [GKR+21, GKS23] over $\mathbb{F}_p$ at width $t = 3$ with S-box x^3, a permutation of $\mathbb{F}_p$ on both primes, and 12 full and 92 partial rounds. A second instance at $t = 5$ with 12 full and 93 partial rounds serves the four-ary commitment tree.

The designers' script gives $(8, 83)$ and $(8, 84)$ at 128 bits. The extra rounds answer the round-skipping line and the finding that the Gröbner bounds overestimate solving cost [MG26, BTA26, GKR25]; [ABM23] is the analysis both instances are read against. The margin, in one place: the univariate bound of [GKR25], its Eq. (11), needs $r_F + r_P \geq 70$ at 128 bits and $\alpha = 3$ on both primes, so $r_P \geq 62$ at $r_F = 8$. The script's 83 is 1.3 times that and the 92 and 93 in use are 1.5 times it. The subspace trails of [GKR25] linearise $t - 2$ partial rounds, one at $t = 3$ and three at $t = 5$, against the nine extra partial rounds. The S-box-skipping gadget of [BTA26] absorbs $2t - 2k$ S-boxes, four at $t = 3$ and $k = 1$, against the twelve S-boxes of the four extra full rounds. The round skipping of [MG26] uses the tensor structure of Poseidon2's linear layers at $t \in \{12, 16, 20, 24\}$, absent at $t = 3$ and $t = 5$, whose layers are MDS. The sponge has rate 2 and capacity 1 with a domain tag in the capacity and fixed input lengths, so there is no padding. In the circuit an S-box is two degree-2 constraints through an intermediate column and one permutation is 23 trace rows.

4 The scheme

The scheme follows Zerocash [BCG+14]: notes committed in a tree, each spent once by its nullifier, and a proof per transaction. It adds the viewing keys and outgoing records of the Zcash specification [HBHW26] and a per-output disclosure [Gri26], and every primitive is post-quantum. This section fixes the objects; the specification in the artifact, available to reviewers through the editor, fixes every byte and every check.

The verifier. The verifier is a party that holds a set of accepted anchors and a set of spent nullifiers. It applies the verification predicate of Section 4.6 to each transaction, or to a batch proof over many (Section 6). It then appends the new commitments to its tree, whose root becomes an anchor, and the nullifiers to its set, and thereby maintains the ledger, the record of anchors, nullifiers and accepted transactions. The aggregator of Section 6 can withhold a transaction from a batch and cannot forge or alter one: the verifier checks each transaction's signatures itself, and the batch proof binds the bodies it covers. Everything below is stated against this verifier. A blockchain is one instance of it; so is a server or a notary.

4.1 Syntax and security notions

The scheme is five algorithms over a verifier state $st = (\mathcal{A}, \mathcal{N}, T)$: the accepted anchors, the spent nullifiers and the commitment tree.

- KeyGen$(seed) \to (sk, fvk, ivk, ovk)$: the spending key and the three viewing keys of Table 8, each a domain-separated derivation of the 32-byte seed.
- Address$(sk) \to addr = (E_{\text{scan}}, E_{\text{sign}}, nkc)$.
- Output$(addr, v, memo) \to (out, open)$: an output (cm, eph_pk, ct, out_ct) for the recipient $addr$ and the note's opening, which the sender keeps.
- Spend$(sk, notes, outs, anchor) \to tx$: a transaction spending $notes$ held under sk and creating the outputs $outs$, with a signature per spend, of the form ($anchor$, $spends$, $outputs$, $value_balance$, $proof$).
- Verify$(st, tx) \to \{0, 1\}$, and on 1 the state st': the outputs' commitments appended to T, the new root added to $\mathcal{A}$ and the spends' nullifiers to $\mathcal{N}$. VerifyBatch(st, $(tx_1, \ldots, tx_n)$, π) is the same under one batch proof π (Section 6).

Scanning is the recipient's side of Output under ivk; disclosure (Appendix C) is a sixth algorithm the notions below do not cover.

The notions follow Zerocash [BCG+14]. In each, a polynomial-time adversary faces a challenger that holds the verifier state and answers four oracles. Create makes an honest key pair and returns its address. Pay and Spend run Output and Spend for honest parties on the adversary's inputs and return the published transaction. Insert(tx) runs Verify on an adversarial transaction and applies it on acceptance. Corrupt($addr$) returns an honest party's sk. The adversary sees every published transaction and the state.

Definition 1 (Ledger indistinguishability)**.** The challenger draws $b \in \{0, 1\}$. The adversary submits pairs of honest queries (q_0, q_1) of one kind whose public consequences agree: the same shape, value balance and anchor, the same corrupted parties involved, and spends of notes that paired queries created. The challenger answers q_b. The adversary wins if its output b' equals b, with advantage $|\Pr[b' = b] - 1/2|$. The notion covers the anonymity of outputs, which hide the recipient and the value, and of spends, which hide the note spent. With dummies the paired queries may differ in their number of real spends and outputs.

Definition 2 (Balance)**.** The adversary interacts and ends by listing the notes it can spend, with their openings and spending keys. It wins if the value of those notes plus the value it paid to honest addresses exceeds the value honest spends paid to it plus the net value balance its own accepted transactions declared. Winning means an accepted transaction created value.

Definition 3 (Non-malleability)**.** A transaction's identity is its hash txh, which excludes the signatures and the proof as the identifier of [NHG21] does. The adversary outputs a transaction tx^* whose identity differs from every published transaction's and that shares a nullifier with one of them, tx. It wins if the verifier accepts tx^* in the state in which it accepted tx. A malleated transaction is one whose signatures or proof were re-formed around another's spends under a new identity.

Definition 4 (Spend authority)**.** The adversary names an honest address it has not corrupted. It wins if the verifier accepts a transaction the oracles did not publish whose spend nullifies a note that a Pay query sent to that address.

Definition 5 (Nullifier integrity)**.** The adversary wins if the verifier accepts two spends, in one transaction or two, that nullify two leaves of the tree under one nullifier, or one leaf under two nullifiers. A leaf's nullifier is then unique to the leaf and computable by the holder of nk alone.

4.2 Keys

The spending key is a 32-byte seed. From it come a MIKE key pair $(E_{\text{scan}}, s_{\text{scan}})$, a signature key pair $(E_{\text{sign}}, s_{\text{sign}})$, the nullifier key $nk \in \mathbb{F}_p$ with its commitment $nkc = H_4(nk)$, and the 32-byte outgoing viewing key ovk (Table 8). Each is derived under its own domain string, so none is computable from the others. There is no hierarchy below the seed and no diversifier. An address is two curves and one field element.

4.3 Notes, commitments and nullifiers

A note is $(E_{\text{out}}, nkc, v, \rho, r)$: its own spend key, a curve, the recipient's nullifier key commitment, a 64-bit value, and the nullifier and commitment randomness in $\mathbb{F}_p$. Its commitment and nullifier are

$$cm = H_2(A_{\text{out}}.\text{re},\ A_{\text{out}}.\text{im},\ nkc,\ v,\ \rho,\ r), \qquad nf = H_3(nk,\ \rho,\ \text{rev}_{32}(pos)),$$

with *pos* the note's leaf position and rev_{32} the 32-bit reversal, the order in which the circuit accumulates the path. The position is in the nullifier because ρ is the sender's choice: with the position, two notes under one ρ have two nullifiers, the defence against the Faerie Gold attack [HBHW26]. The tree has 2^{32} leaves and is four-ary under the $t = 5$ instance, 16 levels with empty subtrees the iterated hash of zero. The root after every append is an anchor, and the verifier keeps the set of anchors it accepts.

4.4 The two-curve identity

A note's spend key is a re-randomisation of the recipient's signing curve under the output's shared secret:

$$\rho_{\mathrm{bytes}} = \mathrm{SHA3\text{-}256}(\ \texttt{"eclipse-sp"} \parallel \mathit{shared} \parallel \mathrm{LE}(k, 4)),$$

$$E_{\mathrm{out}} = \mathrm{RandPK}(E_{\mathrm{sign}}, \rho_{\mathrm{bytes}}).$$

Here RandPK is the one Section 9 of [Ray26] defines, ρ_{bytes} is derived from the shared secret, the implementation's silent-payment profile, and k is the output's index. ρ is the same 32 bytes read as a field element. The sender holds the recipient's public keys and a shared secret, and the recipient recomputes ρ_{bytes}. No one else can link E_{out} to E_{sign}. At spend time the key is re-randomised once more with fresh 32 bytes r_{rk} by the mode B walk. $rk = \mathrm{RandPK}(E_{\mathrm{out}}, r_{rk})$ is published, and the signature over the transaction hash is under the matching secret. The proof binds rk's curve to a walk from the E_{out} inside the committed note (Section 5), and only the holder of s_{sign} can sign.

4.5 Creating an output, and scanning

The sender draws an ephemeral MIKE key pair and computes the shared secret with the recipient's E_{scan}. From it the sender derives ρ_{bytes} and E_{out}, draws r and forms cm. The note's plaintext is the value, ρ_{bytes}, r and a 32-byte memo. It is sealed under AES-256-GCM [Dwo07] with a key and nonce from SHAKE256 over the shared secret, the ephemeral key and cm. The outgoing record seals the recipient's address and the ephemeral secret under a key derived from ovk; a sender who wants no record uses random bytes in its place. An output in the transaction is (cm, eph_pk, ct, out_ct). Scanning with the incoming viewing key is one MIKE shared secret per candidate output, one AEAD open and the check $cm = H_2(\ldots)$. The check binds the ciphertext to the commitment. The full viewing key also computes nf, and the holder of ovk opens the record and re-runs the recipient's steps. A sender can also disclose one output to a third party from the published transaction alone, following [Gri26]; Appendix C has the form.

4.6 Transactions, shapes and dummies

A transaction is $(anchor, spends, outputs, value_balance, proof)$ with a spend (nf, rk, sig) and an output as above (Table 9). The transaction hash is SHA3-256 over a domain string, a 32-bit instance identifier and every field but the signatures and the proof. The identifier keeps a transaction from replaying against another instance of the verifier, the role of the branch identifier in [NHG21]. The signatures are over this hash and the proof's transcript absorbs it, so a changed transaction invalidates both.

The scheme has two shapes, 2-in-2-out and 4-in-4-out, one verifying key each. The sender pads a smaller payment to the nearer one with dummy notes [HBHW26, HNG26]. A dummy spend is a fresh note in no tree spent with its enable bit off. Its value is forced to zero by the circuit, and its nullifier, re-randomised key and signature are formed as a real spend's. A dummy output is a zero-value output to a fresh address. What the verifier learns of a transaction is its shape and its value balance. Verification is a pure function

of the transaction and two predicates the verifier supplies. Every field decodes strictly, the anchor is accepted, and the nullifiers are distinct and unspent. Each signature verifies under its rk, and the proof verifies under the shape's key on public inputs assembled from the transaction. The verifier then records the nullifiers, appends the commitments and applies the value balance.

4.7 Delivery profiles

Note delivery as specified is MIKE: a recipient tries every output of a batch against its scanning key. Two more profiles are defined for Section 10 and not built, a mailbox keyed by ML-KEM-512 [Nat24b] and out-of-band delivery with no ciphertext in the transaction. Neither changes a soundness claim unless the circuit binds the ciphertext to the output.

4.8 Key separation

Scanning, building the witness and proving need the viewing keys and the notes. Signing needs s_{sign} and the 32 bytes of each spend's re-randomisation, and nothing else. A signer therefore needs no note data, and a prover no signing key.

5 The leaf

5.1 The statement

The proof of a transaction is a STARK for the relation $R(\mathit{public}; \mathit{witness})$ with

$$\begin{aligned}\mathit{public} = (&\mathit{anchor},\ \mathit{value_balance} \bmod p,\ \mathit{txh},\\ &nf_1 \dots nf_s,\ J(rk_1) \dots J(rk_s),\ cm'_1 \dots cm'_o),\end{aligned}$$

where $J(E) = (A^2-3)^3/(A^2-4)$ is the j-invariant over 256 of a curve of coefficient A, computed from rk's bytes. The witness is

$$\begin{aligned}\mathit{witness} = \big(&(A_{\text{out}}, nkc, v, \rho, r, nk, \mathit{path}, \mathit{walk}, en) \text{ per spend},\\ &(A'_{\text{out}}, nkc', v', \rho', r') \text{ per output}\big),\end{aligned}$$

and the shape (s, o) fixes the circuit. For each spend h:

1. $cm_h = H_2(A_{\text{out}}.\text{re}, A_{\text{out}}.\text{im}, nkc, v, \rho, r)$;
2. cm_h is in the tree at anchor unless the spend is a dummy: 16 four-ary levels, the position at each a two-bit digit of the path, both bits boolean, and $en\,(x_{16} - \mathit{anchor}) = 0$ with $en\,(en-1) = 0$ and $(1-en)\,v = 0$;
3. $nf_h = H_3(nk, \rho, P)$ with P accumulated from the path bits as $P_{j+1} = 4P_j + 2b_{2j} + b_{2j+1}$, and $nkc = H_4(nk)$;
4. rk_h's curve is 2^{2f-4}-isogenous to E_{out} by a radical walk. From $(A_0, B_0) = (A_{\text{out}}, 1)$, each step has a witness α_k with $\alpha_k^2 = B_k$, $A_{k+1} = A_k + 6\alpha_k$ and $B_{k+1} = 4\alpha_k A_k + 8B_k$. At the end $(A^2 - 3B)^3 = J(rk_h)\,B^2(A^2 - 4B)$. The relation is existential, any such walk and not the one r_{rk} prescribes, since either root continues the walk. Signing under any curve on a walk from E_{out} requires E_{out}'s secret. RandSK carries that secret along the walk and only the holder of s_{sign} has it, so the existential relation gives an adversary no curve it can sign under;
5. $v < 2^{64}$ by 64 boolean bits.

For each output, $cm'_j = H_2(\ldots)$ with $v' < 2^{64}$. The circuit does not check how an output's key or ρ' were formed; a sender who misforms an output loses its value. The balance is $\sum_h v_h - \sum_j v'_j = value_balance$ in $\mathbb{F}_p$. With every value below 2^{64} and at most 2^{16} of them, that is the integer identity. The signature is verified outside the proof, over txh, which the proof carries as a public input.

5.2 The subfield check

Every witness value the hash consumes is an $\mathbb{F}_p$ element, and the STARK runs over $\mathbb{F}_{p^2}$, where "in $\mathbb{F}_p$" is not a low-degree condition. Without a check the circuit would prove Poseidon2 over $\mathbb{F}_{p^2}$, whose S-box is 3-to-1. The circuit designates the columns that carry free $\mathbb{F}_p$ witnesses. For each such column polynomial f the prover proves that $f(x) \in \mathbb{F}_p$ on every x of the trace domain H. Since H lies in the norm-one torus, $f(x) \in \mathbb{F}_p$ exactly when $f(x) = f^\sigma(x^{-1})$, f^σ the polynomial of conjugated coefficients. The prover commits $Q_f = X^d(f(X) - f^\sigma(X^{-1}))/(X^n - 1)$. The verifier opens f at $\zeta' = 1/\overline{\zeta}$ as well as at ζ and checks $\zeta^d(f(\zeta) - \overline{f(\zeta')}) = Q_f(\zeta)(\zeta^n - 1)$. If some value of f on H leaves $\mathbb{F}_p$, no Q makes the difference the zero polynomial. The check then passes at a uniform ζ with probability below 2^{-630}. Every other $\mathbb{F}_p$ value in the circuit is an exact polynomial function of the checked ones. The check costs one more opening and one committed polynomial per designated column, counted in the hiding budget h. Its polynomial core is a Lean theorem (Section 8); the binding of the openings to the commitments is the DEEP claim.

5.3 The circuit

The layout is a slot layout of 49 columns. A spend is 764 rows and an output 74, so a padded 2-in-2-out transaction uses 1,676 rows of 2^{11} and a 4-in-4-out 3,352 of 2^{12}. Every constraint has degree 2, an S-box being $y = x \cdot x$, $z = y \cdot x$. The four proof systems, 2-in-2-out and 4-in-4-out on each prime, have 233, 239, 226 and 232 constraints. In the aggregation profile the preprocessed columns are committed once per shape and the commitment is the verifying key, so the recursive verifier never evaluates a selector.

5.4 Parameters and the leaf row

Table 10 has the leaf's configurations. A proof verified directly takes the standalone configuration, the table's first rows, with Keccak-256 commitments. A proof the recursion consumes takes the aggregation profile, setting B: blowup 32, 57 queries, $h = 512$, Poseidon2 commitments and a Poseidon2 duplex transcript. The recursive verifier then hashes with the permutation it has rows for. Its preprocessed trace is committed, so the proof ships without its preprocessed openings: 609,515 bytes against 847,034 with them, and 611,240 with the transaction's 1,725-byte body. The 4-in-4-out shape and the rigorous prime are measured as leaves only, in the standalone configuration; the aggregation path of Sections 6 and 9 is measured on the 2-in-2-out leaf at the shared prime. Its proven error, read under the accumulation lemma of Arc [BMNW24] at the Johnson radius, is in Table 3. The leaf row is the one every per-transaction figure of this paper starts from; its times and bytes are in Tables 10 and 9. The prover scales with the cores the sender has, not with the card: the prove alone is 11.3 s on one thread, 3.2 on four, 1.8 on eight and 1.05 on sixteen.

6 Aggregation

6.1 The tree

A batch's leaves are verified in a tree of STARKs over Baby Bear (Table 11). A node proof verifies up to $K = 8$ leaf proofs in eight slots. Nodes are joined pairwise by a join-of-nodes

and joins by a join-of-joins, an odd child promoted a level. The root join is wrapped into the one proof the verifier reads; a batch's proofs are counted in Table 16. The public values are hash chains. A node's are its child key's commitment and a hash chain over its leaves' public inputs. A join's are its children's key commitments and the hash chain $H(c_0 \| c_1)$ over their chains, which binds the children's order. The wrap's 16 values are the root key's commitment and the hash chain. The verifier recomputes that hash chain from the transactions' data and verifies the wrapped proof against it. It applies each transaction's anchor, nullifiers and signatures itself.

The recursion has no hiding of its own and needs none. The leaves hide, and the aggregator holds per transaction exactly what the verifier would hold, the body and the leaf proof, and learns nothing an observer of the transactions does not.

6.2 The fill is a verifier program

A node's trace replays every step of the leaf verifier in the child's transcript order. The steps are the Poseidon2 duplex, the constraint identity at ζ, the DEEP combination at every query and the FRI folds. Every Merkle path is hashed on a Poseidon2 AIR against its root. The leaf's symbolic constraints are compiled once per child key into rows of multiply-accumulate units over the quartic extension. The AIRs exchange values over LogUp buses. The key fixes the program and its preprocessed trace, so a node over another leaf shape is another circuit. The joins run the same program twice, over two node children or over two join children. The wrap runs it once, in a narrow layout over the degree-8 extension. The filler is run in two ways during development. Natively it runs as a mirror. The mirror records every transcript event and every piece of prover advice, and the record is compared event by event with the real verifier. On the device the filler is gated cell for cell against the CPU filler's trace.

6.3 Keys from shapes

The node's key is the leaf's committed preprocessed root together with a Keccak-256 hash of the leaf AIR's extracted constraints and parameters. The key occupies 16 words on row 0 of the arithmetic AIR's preprocessed trace. The leaf proof travels without its preprocessed openings, and the node restores them from the key's own tree at the queried index. A join's key and the wrap's key are planned from the children's shapes alone. The verifier therefore derives every key of a batch from the leaf key and the tree's shape. The aggregator's proofs are data, never keys. The extraction the node key hashes is the one the Lean development proves over (Section 8).

6.4 The wrap between fields

The nodes and the joins are proved at blowup 8 over the quartic extension and verified in-circuit by the level above, so their bytes never reach the verifier. The wrap is the one proof that does. It is proved over the degree-8 extension at blowup 32 with FRI arity 8 and no grinding, where the Johnson bound gives its bits from the queries alone (Table 3). Table 14 (Appendix A) is the sweep it was chosen from.

7 Security

The model is the random oracle model for the transcript hash, Keccak-256 for a proof verified directly and the Poseidon2 duplex otherwise. A prover making Q oracle queries has advantage at most the Fiat–Shamir term of Theorem 1, Corollary 4.3 of [BGK+23], over the per-proof figure, which is the interactive proof's round-by-round bound. The union

over the batch's proofs bounds the probability that any accepted proof is false, which is the event an adversary needs.

7.1 Per-proof bounds

A false proof passes with probability at most the bound of Table 3. The per-proof figures are the round-by-round soundness errors of the interactive proofs. Under Fiat–Shamir a prover making Q oracle queries has advantage at most Q times the figure (Theorem 1), so a figure of b bits is about 2^b queries for constant advantage. The configuration uses no grinding, so no further work term applies. The paper's bit figures are read this way throughout. The nodes and the joins are read at unique decoding, the wrap at the Johnson bound, and the leaf under Arc's accumulation lemma at the Johnson radius, each with its field term. Nothing is proved about Plonky3's implementation beyond the byte gate of Section 8.

The wrap is read at $\delta = 1 - \sqrt{\rho} - \eta$ with $\rho = 1/32$ and $\eta = \sqrt{\rho}/20 = 0.0088$, so $\delta = 0.814$. A query then buys $-\log_2(\sqrt{\rho} + \eta) = 2.43$ bits, $2^{-102.0}$ at 42 queries. The field term of Theorem 1.2 of [BCI+20], $(k+1)^2/((2\eta)^7|F|)$, is 2^{-164} at $k = 2^{21}$ and $|F| = 2^{247.2}$. The record's 105 is the limit $\eta \to 0$ at $b/2 = 2.5$ bits a query; the batch's union is $2^{-89.6}$ under either reading. The leaf is read under Arc's accumulation theorem, Theorem 6.2 of [BMNW24], with its FRI taken as one step of Arc's Construction 6.3 at $m = 1$. The step's shape is FRI's: the DEEP batch is the one oracle and the batching challenge its γ. The opening at ζ is its $s = 1$ out-of-domain point and the 57 query points its t in-domain openings, so Lemma 6.5's error applies with the leaf's parameters. An accumulator holding leaves of both shapes needs one degree bound $d_{\max}$ and one domain. The reading therefore takes the larger shape's $n+2h-1+(1+t) = 5{,}177$ at $n = 2^{12}$ on the 2^{16}-point domain, a rate $d_{\max}/|L| = 0.079$ rather than the leaf's $1/32$. Theorem 6.2 holds for δ below $1 - 1.05\sqrt{\rho} - t/|L|$, taken at that end, 0.704, and its field-size condition holds at 652 bits with 477 bits to spare. Lemma 6.5's error is the largest of a proximity term, an out-of-domain term and $(1-\delta)^t$. At 652 bits the first two are below 2^{-577}, and $(1-\delta)^t$ at $t = 57$ is $2^{-100.1}$, 1.756 bits a point, the term that decides. The wrap, a FRI proof the verifier checks itself, is read at the Johnson radius of its own rate under Theorem 8.3 of [BCI+20].

Table 3: The per-proof bounds under the reference configuration.

proof	field	blowup	queries	regime	bits a query	bound
leaf (setting B)	$\mathbb{F}_{p^2}$, 652 bits	32	57	Arc's accumulation, $\delta = 0.704$	1.756	$2^{-100.1}$
node	quartic, $2^{123.6}$	8	120	unique decoding	0.83	2^{-100}
join (both kinds)	quartic, $2^{123.6}$	8	120	unique decoding	0.83	2^{-100}
wrap	degree 8, $2^{247.2}$	32	42	Johnson, $\delta = 0.814$	2.43	$2^{-102.0}$

7.2 The union at the batch

Theorem 1 (The union at the batch)**.** *Let a batch's tree carry N_n node proofs, N_j join proofs, one wrap and L leaf proofs, and let a false proof pass its verifier with probability at most ϵ_{node}, ϵ_{join}, ϵ_{wrap} and ϵ_{leaf} under the regimes of Table 3, each the table's figure with its field term, read as the round-by-round bound of the interactive proof. Let the transcript hash be a random oracle of κ bits and the prover make at most Q queries to it. Then the*

verifier accepts a batch holding a transaction no honest prover could prove with probability at most

$$Q\left(N_n\epsilon_{\text{node}} + N_j\epsilon_{\text{join}} + \epsilon_{\text{wrap}} + L\epsilon_{\text{leaf}}\right) + \frac{3(Q^2+1)}{2^\kappa},$$

by Corollary 4.3 of [BGK+23].

Proof. Accepting such a batch needs a false proof accepted somewhere in the tree: a false leaf by its node, a false node or join by its parent, or a false wrap by the verifier. The union over those $N_n + N_j + 1 + L$ events bounds the interactive error, and Corollary 4.3 of [BGK+23] turns a round-by-round bound ϵ into $Q\epsilon + 3(Q^2+1)/2^\kappa$ under Fiat–Shamir. □

The term uses $\kappa = 247$, the smallest capacity among the batch's transcripts, the Baby Bear duplex's eight lanes of 30.9 bits. The leaf's own duplex has one lane of 326 or 381 bits, and Keccak-256 serves only a proof checked directly (Appendix A). At $Q = 2^{64}$ the collision term is 2^{-117} and the bound is Q times the union; Table 16 gives it at $Q = 2^{64}$ and at $Q = 2^{80}$. A tree of 2^k nodes has $N_n = 2^k$, $N_j = 2^k - 1$ and $L \le K2^k$. Table 16 (Appendix A) instantiates the theorem for the measured batch. The configuration that puts the batch's union under 2^{-105}, its measured cost and the conjectured readings are in Appendix A; the paper claims the proven figures only.

7.3 The protocol's properties

Table 20 (Appendix A) restates each property as a claim under its hypotheses, pointing at its notion in Section 4.1.

The STARK's zero knowledge is Haböck and Kindi's construction [HK24], Protocol 3 at reduced degree $d = 1$, since every constraint has degree 2. Each witness column is masked as $w + v_H r$ with r of h random coefficients, and its Theorem 6 gives perfect honest-verifier zero knowledge when $2d(e\,n_F + n_D) + n_D \le h \le |H|$. The openings are counted as follows. $e = 1$: the challenge lies in the trace field $\mathbb{F}_{p^2}$. n_F counts a column's out-of-domain openings, the DEEP point ζ and, for a designated column, the subfield check's $\zeta' = 1/\overline{\zeta}$, so $n_F = 2$. n_D counts the in-domain openings, one per FRI query and, for a designated column, a second through its Q_f, so $n_D = 2 \cdot 57 = 114$ at the reference configuration; the budget is read with these counts for every column. A uniform mask R of the batch's degree, below $|H| + 2h - 1$ here, is committed with the trace and added raw to the DEEP batch before FRI (Protocol 2 of [HK24]). The FRI layers' coset openings are therefore of committed oracles, independent of the witness given the openings at the index (Lemma 2 and Remark 3 of [HK24]), and a query reveals one in-domain point per column. The inequality is $2(2 + 114) + 114 = 346 \le h = 512 \le |H| = 2{,}048$, 166 coefficients of margin, and 512 is the smallest power of two that closes it. The exact count, one point per column and query, is $2(2 + 57) + 57 = 175$, the 114 being the prover's doubling for the designated columns' Q_f. A query index of the recursion's duplex is the low bits of a lane, a bias of $2^{\log m}/p$.

The model. The sketches below use the assumptions this section already lists and add none. The hash: Poseidon2 is modelled as a random oracle for H_2, H_3 and H_4, its sponge being indifferentiable from one up to $2^{c/2}$ calls under a random permutation [GKR+21, GKS23], and SHAKE256 likewise for the key derivations. A commitment is then hiding by the uniform r and binding up to a collision, and the nullifier is a PRF in nk. The proof system: zero knowledge from [HK24] at the budget above, and knowledge soundness from the BCS transform of the underlying IOP in the random oracle model [BCS16]. The Fiat–Shamir term of [BGK+23] enters as Theorem 1 carries it, and [CMS19] is the post-quantum statement. The signature: existential unforgeability under randomised keys, Theorem 2 of [BBC+26a] with its Remark 11 for the prime-degree variant, under

randomised keys as [BBC+26b] defines them, with the key-randomisation unlinkability of [BBC+26b]. Its strong unforgeability is argued by the construction's authors [BBC+26a, Section 5.1], not proved, and not relied on. MIKE: CDH on the rank-2 Hermitian module action [BDD+26], at the paper's own instantiation of Section 3 rather than a published MIKE set. AES-256-GCM: authenticated encryption [Dwo07]. The sketches obtain a leaf's witness through the tree by composing the BCS extractor level by level. The wrap yields the root join, each join its children, each node its leaf proofs and each leaf its witness. That is the proof-carrying-data argument of [BCCT13], whose results cover constant depth. For the measured batch the composition runs through ten levels, the wrap, eight levels of joins and the node, before the leaf. Its loss at that depth is not bounded here and is taken as an assumption alongside the others. Theorem 1 is a soundness statement and does not depend on extraction.

Proof sketches. The structure is Zerocash's [BCG+14]. Its Appendix C states the experiments and its Appendix D proves its Theorem 4.1 by hybrids and extractions. Each sketch below names the assumption every step spends.

Sketch 1, ledger indistinguishability (Definition 1). A sequence of hybrids runs from the $b = 0$ experiment to $b = 1$. First, every honest transaction's proof is replaced by a simulated one. The distance is the STARK's zero knowledge: honest-verifier zero knowledge of the interactive proof at the hiding budget of this section, carried to the non-interactive proof by Fiat–Shamir in the random oracle model [HK24]. Second, every honest output's *ct* and *out_ct* are replaced by encryptions of zero. To a party without the scanning secret the key SHAKE256, a random oracle here, derives from the MIKE shared secret is uniform unless that party computes the secret, which is CDH on the module action [BDD+26]. The AEAD then hides the plaintext [Dwo07]. Third, every rk is replaced by a fresh random key, the key-randomisation unlinkability of [BBC+26b] as [Ray26] implements it. Fourth, every commitment and nullifier is replaced by a random oracle output under fresh randomness, which the adversary distinguishes only by querying the oracle on the hidden input. After the hybrids the two experiments are identically distributed, since the paired queries agree on everything else the verifier sees. The advantage is at most the sum of the four distances: the zero-knowledge term, the CDH and AEAD terms, the unlinkability term and the random-oracle query term (Table 20, the first three rows).

Sketch 2, balance (Definition 2). Run the knowledge extractor of the BCS transform [BCS16] on every accepted proof; an extraction fails with the knowledge-soundness error of Theorem 1's per-proof figure. The extracted witnesses satisfy the relation of Section 5, so every value is below 2^{64} and a transaction's values sum to its declared balance. A note spent twice yields two witnesses with the same nk, ρ and position, hence the nullifier the set refuses, or two nullifiers from one input, a collision in H_3. Value created from nothing needs a spend whose commitment opening is not in the tree at the anchor, a collision in H_2 or in the tree hash. The bound is the knowledge-soundness error times the number of accepted proofs plus the two collision terms (Table 20, the balance row).

Sketch 3, non-malleability (Definition 3). Let tx^* share a nullifier with a published tx, carry another txh and be accepted in its place. Its signatures are over its own txh, so tx's do not verify on it, and its proof's transcript absorbs txh, so tx's proof does not either. It therefore needs a fresh signature under tx's published rk over the new hash, a forgery against existential unforgeability under randomised keys (Theorem 2 and Remark 11 of [BBC+26a]), or a fresh accepted proof. The re-formed statement may be true, so the fresh proof need not be false. By extraction it yields a witness holding the note's nk, ρ and path, which only the note's holder has, and the branch reduces to spend authority (Sketch 4).

The same holds with a fresh rk, whose walk from E_{out} the extracted witness also carries. The term is the extraction error, not the soundness term. A transaction with tx's hash and other signature or proof bytes has tx's identity, nullifiers and commitments; strong unforgeability would exclude it and existential unforgeability does not. Two such wire forms of one transaction can be in flight. The verifier accepts whichever arrives first and refuses the other on its nullifiers, so the only effect is that a transaction's accepted bytes can differ from the sender's. The bound is the EUF-CMA advantage plus the extraction error (Table 20, the non-malleability row).

Sketch 4, spend authority (Definition 4). An accepted transaction nullifying an honest note has, by extraction, a witness whose rk lies on a walk from the note's E_{out}, item 4 of the relation, and the transaction carries a signature under rk over txh. RandSK is what carries the secret along the walk, and only the holder of s_{sign} has it. A signature under any curve on the walk without it is a forgery against existential unforgeability under randomised keys (Theorem 2 and Remark 11 of [BBC+26a]). The bound is the knowledge-soundness error plus the EUF-CMA advantage (Table 20, the spend-authority row).

Sketch 5, nullifier integrity (Definition 5). Two accepted spends of two leaves under one nullifier give, by extraction, two witnesses whose $H_3(nk, \rho, P)$ agree at different (nk, ρ, P), a collision in H_3, since the position P is bound to the leaf by the path and the anchor. One leaf under two nullifiers gives two witnesses with one cm and different H_3 outputs, so they differ in nk or ρ under one commitment, a collision in H_4 or H_2. The bound is the collision terms plus the knowledge-soundness error (Table 20, the nullifier-integrity row).

Open items.

1. The signature's existential unforgeability under randomised keys is the assumption, Theorem 2 of [BBC+26a] with its Remark 11 and [BBC+26b]. Its strong unforgeability is argued by its authors under stronger assumptions [BBC+26a, Section 5.1], is not proved and is not relied on. MIKE's NIKE security is its paper's claim.
2. The joint security of one seed behind three keys rests on the random-oracle argument alone. The three derivations are SHAKE256 outputs under distinct domain strings, independent uniform strings in that model, so the joint security reduces to each primitive's with independent keys.
3. The Poseidon2 instances over these primes are fresh. The subfield check, this work's one arithmetisation device, has not been reviewed outside this work.
4. There is no independent proof of the hiding of the composed circuit beyond [HK24]. The Fiat–Shamir argument is the classical round-by-round one [BGK+23]; the quantum random-oracle argument [CMS19] is cited, not instantiated.
5. The sketches above are in the random oracle model for the hash and the proof system and rest on the signature's existential unforgeability; no claim is made in the standard model.

8 Verification method

Three kinds of evidence stand behind the claims, each with a stated reach: machine-checked theorems about the leaf circuit, gates between two provers, and forged-proof tables at every level of the tree. The Orchard circuit's soundness bug of May 2026, a missing copy

constraint in a scalar-multiplication gadget that could have allowed balance violation [Hop26a], is why circuit-level verification matters.

8.1 What Lean proves

The Lean 4 development [Lea26, mat26] proves that every trace satisfying the leaf circuit's constraints encodes witnesses of the spend relation of Section 5, for each of the four leaf proof systems. The witnesses are the note openings, the Merkle membership of every enabled spend, the nullifiers, the radical walks, the ranges, the dummy rule, and the balance in the field and, lifted, as an integer. The hypotheses are stated. The primes are prime, a hypothesis since a kernel primality check at this size is impractical. The selector and preprocessed columns are the layout's, and the designated columns lie in $\mathbb{F}_p$, which is what the subfield check establishes with its own error. The $\mathbb{F}_p$ public values, the anchor, the value balance, the nullifiers and the commitments, are hypothesised in $\mathbb{F}_p$ as well. The verifier supplies that by decoding each strictly as an $\mathbb{F}_p$ element and lifting it with imaginary part zero when it assembles the public inputs. The relation is written from the specification, not from the circuit. The agents wrote the relation from the specification's spend statement; the authors checked it clause by clause against that statement, the table being `LEAN-RELATION.md` in the artifact. The three encodings that are not literal are listed in Table 18. The lift of the balance from the field identity to the integers is stated in Appendix A. Table 18 (Appendix A) is the coverage table.

The constraints are the prover's. A hook evaluates the Rust AIR's constraint function, the one function the prover and the verifier run, on Plonky3's symbolic builder and prints every constraint as a Lean term. The committed modules are what the hook emits, and a test fails if they are not, by the counts of Section 5, the degree and the bytes. The node's key carries a Keccak-256 hash of the same extraction, so the AIR the recursion verifies is the AIR Lean proved over. There is no Lean model of the verifier. Its correspondence to the written protocol is by inspection, and a second verifier in Python agrees with the Rust one on every exported proof.

Build facts. The development's 36 headline theorems stand on the standard axioms, with no `sorry`, no `axiom` and no `native_decide`, by a script that fails on anything beyond them; its toolchain, size and build time are in Appendix A. The pre-quantum counterpart is Ironwood's Lean development, published at the pool's activation [BD26]. It proves knowledge soundness of the Halo 2 verifier in the algebraic group and random oracle models, the verifier taken as the linear combination it outputs and matched against the Rust verifier on captured runs. The Action circuit is transcribed in Clean from the Rust source, its verifying key proven equal to the captured one, and balance integrity holds over a ledger model, over 2,700 theorems down to named assumptions [BD26]. The Clean port of Orchard that preceded it, of July 2026, had no mechanical link to the Rust source [zkS26]. This development proves less: the leaf's constraints, extracted from the prover, encode the spend statement; the verifier has no Lean model and the STARK's bound is a paper argument.

8.2 The gates

The GPU prover is held to the CPU prover, Plonky3's at the pinned revision, by byte identity on the same leaf files, sequentially and with four proofs in flight. The device fill is held to the CPU filler cell for cell, and the batch's digest over every proof is the same at any worker count (Table 19, Appendix A). This is prover correctness relative to the CPU prover, not a soundness claim, and it is what lets every number of Section 9 stand for the same proofs.

8.3 The forged-proof tables

Every level has a table of forgeries (Table 15, Appendix A). Each row is a mutated input proved by the release prover with the circuit's own checks let through, and the verifier's rejection is named. At the leaf and in the recursion it is the constraint identity; where the children are valid proofs of another tree or in the wrong order it is the public-value equality. A table is evidence that the circuit refuses each listed forgery, not an equivalence proof. The one gap the leaf suite could not see, an output's nkc off $\mathbb{F}_p$, the Lean theorem found. The output theorem closes only with that column among the subfield-checked ones, and the constant now carries it.

9 Implementation and measurements

The implementation is in Rust. It comprises the protocol, the leaf prover, the recursion over Plonky3 and a GPU build of the recursion's prover. The device runs the trace fill, the low-degree extensions, the Merkle trees, the quotient, the LogUp columns, the DEEP reduction and the FRI folds. Several provers share one card through per-worker stream sets under a resident budget. The CPU build is untouched by the GPU work and is the reference the gates hold the GPU build to. The leaf prover is not ported.

Procedure. Every circuit row is one untimed warm-up and three runs. The median is reported and all three runs are in the record. The batch's configuration rows are five runs of a 512-transaction slice and one run of the full batch. The build of record is a6fb606, run with four provers and two builders, $K = 8$ and no flags. Every measurement after it was kept only if the batch's digest and the slice's median held, and none was. First-use compilation of the generated kernels takes 28.5 s and 38.2 s for the two largest on a 32-thread host. It is cached and kept out of every median. The workstation is 16 Cascade Lake cores, 32 threads at 2.8 GHz with AVX-512 and 125 GB. The cards are rented instances, each with its host in Table 22 (Appendix A), which also records how the cards are read.

9.1 The four circuits

Table 12 is the build of record on the RTX PRO 6000: each circuit's fill and prove with the device off and on, the proof's bytes and its verification on the host. A transaction at $K = 8$ is one node and its join over eight, the table's last two rows.

9.2 The batch

Table 4 is the full batch of 1,248 transactions on the RTX PRO 6000 with four provers and two builders: 99.2 s on the wall and 0.079 s per transaction on the build of record. The leaves reach the aggregator as 1,248 proofs of 609,515 bytes, about 760 MB per batch. The verifier's cost per transaction, signatures included, is 14 ms on one thread and 0.5 ms across 32, plus 15 ms per batch for the proof. The two signatures per transaction are verified alongside it, 2,496 for the batch in 17.6 s on one thread and 0.60 s across a host's threads. That is the batched rate of [Ray26], 149 verifications a second on one thread, where a single verification is 8.0 ms, or 6.5 with a prepared key [Ray26]. The published batched figures for Sapling and Orchard are block-level, zcashd's and Zebra's (Appendix A, the notes to Table 1); no per-proof batched figure is published, so that table's column is unbatched on both sides.

The capacity line is a 75 s batch budget. At the batch's rate of 0.079 s per transaction it holds 949, and the record carries 860, which take 67.9 s, as the line with a margin

Table 4: A batch of 1,248 transactions on one RTX PRO 6000.

the 1,248-transaction batch on one RTX PRO 6000	the closeout (7fceeec)	the build of record (a6fb606)
aggregation wall	105.3 s	99.2 s
per transaction on the wall	0.084 s	0.079 s
the node phase, per transaction over it	96.5 s, 0.077 s	91.8 s, 0.074 s
the wrapped proof, with its 16 public values	426,478 bytes	426,478 bytes
the wrapped proof verified	0.015 s	0.015 s
the 2,496 signatures, one thread / across the threads	17.6 s / 0.60 s	

A blank cell is a row that build's record does not carry. The digest over every proof's bytes is the same on both builds.

of 89 transactions. At 860 the wrapped proof's share per transaction is 495 bytes, the comparison's figure, and the batch is about 1.91 MB with the bodies.

9.3 The scan

Table 21 (Appendix A) is the cost of trying a batch's outputs against an incoming viewing key, a MIKE shared secret, a key derivation and an AEAD open per output. Beside it is an ML-KEM-512 decapsulation alone, the floor a KEM-keyed mailbox would sit on. The MIKE trial costs about 300 times the decapsulation, the 652-bit field arithmetic against the lattice's polynomial arithmetic.

At throughput the scan is a budget. One batch per 75 s is 1,152 batches a day, 990,720 transactions at the capacity line of 860 and 1,981,440 outputs. At 11.2 ms an output, a recipient's trial decryption of a day's outputs is 22,200 s, 6.2 hours of one core, against 73 s at the decapsulation's 37 microseconds. That is why the mailbox profile of Section 4 exists: a recipient who cannot spend six core-hours a day scanning takes delivery by the KEM-keyed mailbox at 768 bytes more per output.

10 Comparison

Table 5 places ANNULAR beside the schemes that hide a payment's parties and amount. Tables 1 and 2 carry their numbers and properties. Three readings follow.

The transaction. A 2-in-2-out ANNULAR transaction body is the size of Sapling's at the same memo, 2,685 bytes against 2,756 at a 512-byte memo, with every key and ciphertext post-quantum. All in, with the proof's share at capacity, it is 3,180 bytes at 860 transactions and 3,027 at 1,248, against Sapling's 2,756 all in. The proof's share at the measured batch capacity is under Sapling's four Groth16 proofs and about a fifteenth of Orchard's two actions (Table 1).

The batch. Neptune Cash is the closest prior scheme, post-quantum, each of its batches one merged transaction under one STARK (Table 5) [Sze22, Sch25]. ANNULAR's security figure is a proven per-proof bound with a union at the batch. Neptune states 160 bits, and a contributor reads its FRI query phase as 160 conjectured or 80 provable [Sze23b, rob26b]; its verification time is not published.

The assumption. Neptune rests on a hash function alone. ANNULAR adds the endomorphism-ring problem of supersingular curves for the signature and the key exchange, in return for the note-and-nullifier shape: a reusable address, notes sent without interaction, keys that re-randomise and a per-output disclosure. The lattice RingCT line has the other

shape, decoys in place of a tree and no aggregation [EZS+19, ESZ22]. MatRiCT and MatRiCT+ are Monero's shape under module lattices, a balance proof without a wide range proof and the anonymity set the ring; their proof sizes and verification times are in Table 1.

Table 5: Private transaction schemes in the terms each publishes.

scheme	assumption	hidden	batch aggregation
ANNULAR (this work)	isogenies and Poseidon2; FRI	sender, recipient, amount	one proof per batch
Neptune Cash	hash functions (Triton VM)	amounts, origins (mutator sets)	one merged transaction per batch, about 1 MB of proof
Sapling	discrete logarithm (Groth16)	sender, recipient, amount	none; aggregates proposed
Orchard	discrete logarithm (Halo 2)	sender, recipient, amount	none
Compact Shielded CSV	hash commitments; the proofs with the parties	payments	none; the recipients validate
Shielded CSV	Schnorr, not post-quantum	payments	none; a nullifier per transaction published
MatRiCT, MatRiCT+	module lattices (RingCT)	amounts; the spender within a ring	none; verification linear in transactions

Sources: ANNULAR, Sections 7 and 9; Neptune Cash [Sze22, Sch25, Sze23b, rob26b]; Sapling [HBHW26, Hop21]; Orchard [HBHW26, Hop21, Hop26a, Ele26]; Compact Shielded CSV [ILS+26]; Shielded CSV [NEL25]; MatRiCT and MatRiCT+ [EZS+19, ESZ22].

11 Related and concurrent work

Deployed schemes. Sapling and Orchard are the instances of Zerocash's design in use, and Ironwood is Orchard's successor on the same proof system. They run Groth16 over a pairing-friendly curve and Halo 2 over Pallas, elliptic-curve Diffie–Hellman for note encryption and a Schnorr-style signature for spend authorisation [HBHW26, Ele26]. Ironwood, Zcash's NU6.3 upgrade, activated on 28 July 2026 at block 3,428,143 with quantum-recoverable notes under ZIP 2005 [Hop26b, Mal26b, HG25]. The plan after it is post-quantum signatures for transparent payments, with the private side's own safeguards later [Mal26c, Mal26a]. Aggregates of their proofs were proposed in 2021 and are not built [Hop21]; Tachyon takes note delivery out of the published transaction against the harvest-now-decrypt-later risk [Bow25b].

Folded batches. Proof-carrying data and private rollups fold proofs into one, over elliptic curves in Zexe, Aztec Connect and Mina and under a STARK in Miden [BCG+20, Azt24, BMRS20, Mid26]. Zexe validates transactions in constant time by recursive composition, Aztec Connect's rollup circuit aggregated users' proofs into one, Mina's blocks are recursively composed proofs, and Miden proves transactions into batches and batches into blocks. Tachyon proposes the same for Zcash, a proof-carrying-data aggregation of a block's Orchard proofs with a wallet's state carried as proof-carrying data [Bow25a]. This paper's contribution is not the folding but a post-quantum instantiation over isogeny primitives, with the signature's field as the leaf's field.

Neptune Cash. Neptune Cash is a scheme whose every primitive is a hash function [Sze22]. Transactions are proved in Triton VM [Tri26], and a verifier for its own proofs runs inside it [Vær24]. Every batch is one merged transaction with one proof [Sch25],

proved on a 768 GB Threadripper workstation at 2.65 s for 2^{16} padded rows and 418 s for 2^{23} [swo25]. Spends are removal records against a mutator set with a sliding-window Bloom filter in place of a nullifier set [Sze23a].

Client-side validation. Shielded CSV publishes a 64-byte nullifier per transaction, and its users verify one Schnorr signature per nullifier [NEL25]. Compact Shielded CSV keeps every large artifact with the parties and publishes three hashes, 96 bytes, with a degriefer in place of a signature, for the post-quantum setting [ILS+26]. Both move validation to the recipient, where this scheme's verifier checks every transaction and pays the body.

Soul of Satoshi. Soul of Satoshi is a draft scheme whose every signature is ML-DSA-65 [Sou26]. Its opt-in private transactions are a hash-only commitment tree with one RISC Zero STARK receipt per action, each checked by every verifier, the spend's ML-DSA check inside the zkVM. No spend-proof time, proof size or aggregation is published, soundness is assumed at the STARK's conjectured parameters, and it has had no external audit.

Signature aggregation. Hash-based multi-signatures aggregated through hash-based succinct arguments [DKKW25] are the same shape of argument as this paper's tree, applied to validators' votes rather than a scheme's spends. Aerie aggregates Falcon-512 signatures under a lattice proof over the signatures' own linear algebra, for validators' votes and with no private statement [CDBT26]; its figures are in the notes under Table 1.

Proof systems and their verification. The native-field accumulation route, Arc and WARP [BMNW24, BCFW25], was costed for this leaf and not taken because its hashing is over the big field. SP1 Hypercube proved 93 percent of Ethereum blocks under 12 s on about 160 RTX 4090s in May 2025 and 99.7 percent on 16 RTX 5090s by late 2025, without proximity-gap conjectures [Suc25b, Suc26]. Its formal verification covers the 62 RISC-V opcodes' constraints against the Sail specification in Lean and assumes the proof system and the lookup bus [Suc25a]. Hirai's Lean formalisation of the round-by-round soundness proof of FRI is complete but for one result cited from WHIR [Hir26]. Neither verifies a prover to its verifier end to end, and this paper's development stops short of that too.

Isogenies. Signatures with randomizable keys from SQIsign, two-dimensional PRISM and the Deuring VUF are [BBC+26b], with code [IC26]; the prime-degree PRISM implementation's own randomisation keeps the key in the hash [Ray26]. Proofs of isogeny knowledge by dedicated protocols are [CLL23, LP25, dHLM+26]. This paper's relation, a walk of fixed length between two curves, is proved inside a general-purpose STARK, and the comparison with a dedicated proof is open. Spender anonymity by ring rather than by tree is the isogeny ring signature line, Erebor at 0.71 KB for four users and Durian at 9.87 KB for 1024 [BLL25].

12 Conclusion, and what is left

The post-quantum upgrade of a private transaction scheme and its scaling upgrade are complementary. A proof that folds and a key exchange that stays small make the proof one per batch and constant, with post-quantum primitives. The figures are one proof of 426,478 bytes verified in 15 ms per batch, a 1,725-byte transaction body, and 1,248 transactions aggregated in 99.2 s on one GPU. Each proof carries 100 proven bits and the batch about 90 (Theorem 1).

Limitations. A consumer card cannot produce the reference wrap nor run the tree with a builder in flight, so the batch has been measured on the RTX PRO 6000 alone (Section 9). The signature's strong unforgeability is argued and not proved, and the claims rest on its existential unforgeability (Section 7). The FRI bound is a paper argument outside the Lean development (Section 8). Eight further items, each with its reason, are in Appendix B.

References

[ABM23] Tomer Ashur, Thomas Buschman, and Mohammad Mahzoun. Algebraic cryptanalysis of HADES design strategy: Application to POSEIDON and Poseidon2. Cryptology ePrint Archive, Paper 2023/537, 2023. https://eprint.iacr.org/2023/537.

[Azt24] Aztec. Aztec Connect specifications: the rollup circuit and the root rollup circuit. `markdown/specs/aztec-connect/src/rollup_circuit.md` and `root_rollup_circuit.md` in the AztecProtocol/aztec-connect repository at commit 9374aae687ec5ea01adeb651e7b9ab0d69a1b33b (2024-05-29), https://github.com/AztecProtocol/aztec-connect; the repository's README records the product as deprecated, 2024. Read 2026-10-22.

[BBC+25] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: Simple and compact identification and signatures from large prime degree isogenies. PKC 2025; Cryptology ePrint Archive, Paper 2025/135, 2025. https://eprint.iacr.org/2025/135.

[BBC+26a] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM with a pinch of salt: Simple, efficient and strongly unforgeable signatures from isogenies. Journal of Cryptology (2026); Cryptology ePrint Archive, Paper 2026/443, 2026. https://eprint.iacr.org/2026/443; the citations are to the ePrint revision of 2026-08-03, in which Theorem 2 is on pages 34 to 35 and Remark 11 on page 37.

[BBC+26b] Andrea Basso, Giacomo Borin, Maria Corte-Real Santos, Pierrick Dartois, Riccardo Invernizzi, Luciano Maino, Robi Pedersen, and Michel Seck. Isogeny-based signatures with randomizable keys. Cryptology ePrint Archive, Paper 2026/1169, 2026. https://eprint.iacr.org/2026/1169.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. STOC 2013; Cryptology ePrint Archive, Paper 2012/095, 2013. https://eprint.iacr.org/2012/095, the revision of 2012-12-28.

[BCFW25] Benedikt Bünz, Alessandro Chiesa, Giacomo Fenzi, and William Wang. Linear-time accumulation schemes. TCC 2025; Cryptology ePrint Archive, Paper 2025/753, 2025. https://eprint.iacr.org/2025/753.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. IEEE Symposium on Security and Privacy 2014; the extended version of 18 May 2014, http://zerocash-project.org/paper, 2014. Read 2026-10-21.

[BCG+20] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. Zexe: Enabling decentralized private computation. IEEE S&P 2020; Cryptology ePrint Archive, Paper 2018/962, 2020. https://eprint.iacr.org/2018/962, last revised 2025-11-06; read 2026-10-22.

[BCI+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for Reed-Solomon codes. FOCS 2020; Cryptology ePrint Archive, Paper 2020/654, 2020. https://eprint.iacr.org/2020/654, the version of 3 July 2021.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. TCC 2016; Cryptology ePrint Archive, Paper 2016/116, 2016. https://eprint.iacr.org/2016/116, the revision of 2016-04-29.

[BD26] Sean Bowe and Tal Derei. Formal verification of Zcash Ironwood completed. Project Tachyon, 28 July 2026, https://tachyon.z.cash/blog/ironwood-verification-complete/; the development at https://github.com/zcash/ironwood, 2026. Read 2026-10-22.

[BDD+26] Andrea Basso, Pierrick Dartois, Max Duparc, Jonathan Komada Eriksen, Sabrina Kunzweiler, Michael Meyer, Giacomo Pope, Krijn Reijnders, Damien Robert, Ryan Rueger, and Sina Schaeffler. MIKE: a fast and compact post-quantum NIKE. Cryptology ePrint Archive, Paper 2026/2377, 2026. https://eprint.iacr.org/2026/2377, received 2026-10-06; the project site is https://mike.isogeni.es/.

[BGK+23] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michał Zając. Fiat-Shamir security of FRI and related SNARKs. ASIACRYPT 2023; Cryptology ePrint Archive, Paper 2023/1071, 2023. https://eprint.iacr.org/2023/1071, the version of 15 February 2024.

[BGKS19] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. Cryptology ePrint Archive, Paper 2019/336, 2019. https://eprint.iacr.org/2019/336.

[BLB+21] Eli Ben-Sasson, David Levit, Dan Brownstein, Dan Carmon, Dori Medini, Alon Shtaierman, Lior Goldberg, and Eylon Yogev. ethSTARK documentation, version 1.2. Cryptology ePrint Archive, Paper 2021/582; the archive lists Eli Ben-Sasson as the author, the document's cover the eight, 2021. https://eprint.iacr.org/2021/582, the version of July 2023.

[BLL25] Giacomo Borin, Yi-Fu Lai, and Antonin Leroux. Erebor and Durian: Full anonymous ring signatures from quaternions and isogenies. IACR Communications in Cryptology, vol. 1, no. 4, doi 10.62056/ava3zivrzn; Cryptology ePrint Archive, Paper 2024/1185, 2025. https://cic.iacr.org/p/1/4/4, https://eprint.iacr.org/2024/1185.

[BMNW24] Benedikt Bünz, Pratyush Mishra, Wilson Nguyen, and William Wang. Arc: Accumulation for Reed–Solomon codes. CRYPTO 2025; Cryptology ePrint Archive, Paper 2024/1731, 2024. https://eprint.iacr.org/2024/1731, the version of 25 October 2024.

[BMRS20] Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Mina: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Paper 2020/352, 2020. https://eprint.iacr.org/2020/352, received 2020-03-26, last revised 2026-01-05; read 2026-10-22.

[Bow25a] Sean Bowe. Tachyon: Scaling Zcash with oblivious synchronization. 2 April 2025, https://seanbowe.com/blog/tachyon-scaling-zcash-oblivious-synchronization/, 2025. Read 2026-10-22.

[Bow25b] Sean Bowe. Zcash and quantum computers. 16 October 2025, https://seanbowe.com/blog/zcash-and-quantum-computers/, 2025. Read 2026-10-21.

[BTA26] Amit Singh Bhati, Sundas Tariq, and Tomer Ashur. How to attack Poseidon via smart subspace restriction. Cryptology ePrint Archive, Paper 2026/1692, 2026. https://eprint.iacr.org/2026/1692.

[CDBT26] Alberto Centelles, Quang Dao, Mohammadtaghi Badakhshan, and Justin Thaler. Aerie: Efficient aggregation of Falcon signatures. Cryptology ePrint Archive, Paper 2026/2501, 2026. https://eprint.iacr.org/2026/2501, received 2026-10-15.

[CLL23] Kelong Cong, Yi-Fu Lai, and Shai Levin. Efficient isogeny proofs using generic techniques. ACNS 2023; Cryptology ePrint Archive, Paper 2023/037, 2023. https://eprint.iacr.org/2023/037.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. TCC 2019; Cryptology ePrint Archive, Paper 2019/834, 2019. https://eprint.iacr.org/2019/834.

[CS25] Elizabeth Crites and Alistair Stewart. On Reed–Solomon proximity gaps conjectures. Cryptology ePrint Archive, Paper 2025/2046, 2025. https://eprint.iacr.org/2025/2046.

[dHLM+26] Thomas den Hollander, Shai Levin, Marzio Mula, Robi Pedersen, Daniel Slamanig, and Sebastian A. Spindler. SPRINT: New isogeny proofs of knowledge and isogeny-based signatures. ASIACRYPT 2026; Cryptology ePrint Archive, Paper 2026/364, 2026. https://eprint.iacr.org/2026/364.

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum Ethereum. IACR Communications in Cryptology (2025), doi 10.62056/aey7qjp10; Cryptology ePrint Archive, Paper 2025/055, 2025. https://eprint.iacr.org/2025/055.

[Dwo07] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/Counter mode (GCM) and GMAC. NIST SP 800-38D, November 2007, doi 10.6028/NIST.SP.800-38D, https://csrc.nist.gov/pubs/sp/800/38/d/final, 2007. Read 2026-10-21.

[Ele21] Electric Coin Company. Technical explainer: Halo on Zcash. 8 January 2021, https://electriccoin.co/blog/technical-explainer-halo-on-zcash/, 2021. Read 2026-10-21.

[Ele22] Electric Coin Company. zcashd 5.1.0 release notes. `doc/release-notes/release-notes-5.1.0.md` in the zcash/zcash repository, release v5.1.0 of 2022-07-08, https://github.com/zcash/zcash/releases/tag/v5.1.0, 2022. Read 2026-10-22.

[Ele26] Electric Coin Company and contributors. halo2. https://github.com/zcash/halo2, commit 2d54aaf07c8135d0ce54313048999b4002b4cfbf (2026-10-09), 2026. Read 2026-10-21.

[ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. MatRiCT+: More efficient post-quantum private blockchain payments. IEEE S&P 2022; Cryptology ePrint Archive, Paper 2021/545, 2022. https://eprint.iacr.org/2021/545.

[EZS+19] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. CCS 2019, doi 10.1145/3319535.3354200; the full version, Cryptology ePrint Archive, Paper 2019/1287, 2019. https://eprint.iacr.org/2019/1287.

[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. Specification v1.2, 1 October 2020, https://falcon-sign.info/, 2020. Read 2026-10-21.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. USENIX Security 2021; Cryptology ePrint Archive, Paper 2019/458, 2021. https://eprint.iacr.org/2019/458.

[GKR25] Lorenzo Grassi, Katharina Koschatko, and Christian Rechberger. Poseidon and Neptune: Gröbner basis cryptanalysis exploiting subspace trails. IACR Transactions on Symmetric Cryptology (2025); Cryptology ePrint Archive, Paper 2025/954, 2025. https://eprint.iacr.org/2025/954.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the Poseidon hash function. AFRICACRYPT 2023; Cryptology ePrint Archive, Paper 2023/323, 2023. https://eprint.iacr.org/2023/323.

[GMN21] Nicolas Gailly, Mary Maller, and Anca Nitulescu. SnarkPack: Practical SNARK aggregation. Cryptology ePrint Archive, Paper 2021/529, 2021. https://eprint.iacr.org/2021/529.

[GNHS24] Jack Grigg, Kris Nuttycombe, Daira-Emma Hopwood, and Arya Solhi. ZIP 231: Memo bundles. Zcash Improvement Proposal, status Draft, created 2024-04-26; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-0231.md, 2024. Read 2026-10-22.

[Gri26] Jack Grigg. ZIP 311: Zcash payment disclosures. Zcash Improvement Proposal, status Draft, undated; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-0311.rst, 2026. Read 2026-10-21.

[HBHW26] Daira-Emma Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. Source `protocol/protocol.tex` in the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), 329 commits after the release tag v2026.7.0, https://github.com/zcash/zips; the section numbers are those of the released build, Version v2026.7.0-191-g0fae78 [NU6.2] of 15 September 2026, https://zips.z.cash/protocol/protocol.pdf, 2026. Read 2026-10-21.

[HG25] Daira-Emma Hopwood and Jack Grigg. ZIP 2005: Ironwood quantum recoverability. Zcash Improvement Proposal, status Proposed, created 2025-03-31; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-2005.md, 2025. Read 2026-10-21.

[Hir26] Yoichi Hirai. Lean4 formalization of "A Simplified Round-by-round Soundness Proof of FRI". ZK/SEC Quarterly, 26 January 2026, https://blog.zksecurity.xyz/posts/simple-rbr-fri/, 2026. Read 2026-10-21.

[HK24] Ulrich Haböck and Al Kindi. A note on adding zero-knowledge to STARKs. Cryptology ePrint Archive, Paper 2024/1037, 2024. https://eprint.iacr.org/2024/1037, the revision of 22 February 2025.

[HNG26] Daira-Emma Hopwood, Kris Nuttycombe, and Jack Grigg. ZIP 317: Proportional transfer fee mechanism. Zcash Improvement Proposal, original author Aditya Bharadwaj, created 2022-08-15, revision 0 Active; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-0317.rst, 2026. Read 2026-10-21.

[Hop21] Daira-Emma Hopwood. Shielded transaction aggregates. zcash/zcash issue 4946, opened 2021-01-05, open, https://github.com/zcash/zcash/issues/4946, 2021. Read 2026-10-21.

[Hop26a] Daira-Emma Hopwood. ZIP 257: Deployment of the Orchard temporary vulnerability mitigation and NU6.2 network upgrade. Zcash Improvement Proposal, status Final, created 2026-06-09; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-0257.md, 2026. Read 2026-10-21.

[Hop26b] Daira-Emma Hopwood. ZIP 258: Deployment of the NU6.3 network upgrade. Zcash Improvement Proposal, status Final, created 2026-06-19; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-0258.md, 2026. Read 2026-10-22.

[IC26] Riccardo Invernizzi and Maria Corte-Real Santos. isogeny_swrk: Isogeny signatures with randomizable keys in Sage. https://github.com/giacomoborin/isogeny_swrk, commit 37f504a52ca34b1a12c8cb6e9cabe76ad78d0e6e (2026-06-04), 2026. Read 2026-10-21.

[ILS+26] Dragos I. Ilie, Uri Lee, Iain D. Stewart, Jonathan Zhu, Elliot Jones, and William J. Knottenbelt. Compact Shielded CSV: Post-quantum, private, lightweight client-side validation blockchain. arXiv 2609.32015, version 1 of 25 September 2026, 2026. https://arxiv.org/abs/2609.32015.

[KKH26] Dmitry Krachun, Stepan Kazanin, and Ulrich Haböck. Failure of proximity gaps close to capacity. Cryptology ePrint Archive, Paper 2026/782, 2026. https://eprint.iacr.org/2026/782.

[Lea26] Lean developers. Lean 4. https://github.com/leanprover/lean4, release v4.34.1, commit 5045d0056413266e57c625dcd7c365b10e377c52, 2026. Read 2026-10-21.

[LP25] Shai Levin and Robi Pedersen. Faster proofs and VRFs from isogenies. ASIACRYPT 2025, doi 10.1007/978-981-95-5113-2_10; Cryptology ePrint Archive, Paper 2024/1626, 2025. https://eprint.iacr.org/2024/1626.

[Mal26a] Shaurya Malwa. Zcash developers set January target for quantum-resistant payments after 'bunker mode' scare. CoinDesk, 9 October 2026, https://www.coindesk.com/tech/2026/10/09/zcash-developers-set-january-target-for-quantum-resistant-payments-after-bunker-mode-scare, 2026. Read 2026-10-21.

[Mal26b] Shaurya Malwa. Zcash seals $1.7 billion shielded pool as Ironwood upgrade activates. CoinDesk, 28 July 2026, https://www.coindesk.com/tech/2026/07/28/zcash-seals-usd1-7-billion-shielded-pool-as-ironwood-upgrade-activates, 2026. Read 2026-10-22.

[Mal26c] Shaurya Malwa. Zcash to roll out quantum-recoverable wallets within a month, go quantum-proof by 2027. CoinDesk, 8 May 2026, https://www.coindesk.com/tech/2026/05/08/zcash-to-roll-out-quantum-recoverable-wallets-within-a-month-go-quantum-proof-by-2027, 2026. Read 2026-10-21.

[mat26] mathlib community. mathlib4. https://github.com/leanprover-community/mathlib4, tag v4.34.1, commit d13f23b723b8a846827a245b89c10fc7d3f11612, 2026. Read 2026-10-21.

[MG26] Simon-Philipp Merz and Àlex Rodríguez García. Skipping class: Algebraic attacks exploiting weak matrices and operation modes of Poseidon2(b). Cryptology ePrint Archive, Paper 2026/306, 2026. https://eprint.iacr.org/2026/306.

[Mid26] Miden. Miden: a zero-knowledge rollup for high-throughput and private applications. The README of the 0xMiden/miden-base repository at commit 2cfe2a05d0d2cea45c492218a34e922a5389d3c4 (2026-10-21) and the block producer's README in 0xMiden/miden-node at commit 076f4dbdee1e150682534de378d13905eef0462e (2026-10-21), https://github.com/0xMiden, 2026. Read 2026-10-22.

[MIK26a] MIKE team. MIKE: PQ NIKE. The project site, https://mike.isogeni.es/: the performance table and its note on the fast parameters, 2026. Read 2026-10-22.

[MIK26b] MIKE team. mike_py: MIKE in SageMath. https://github.com/tensor-MIKE/mike_py, commit 321a686f5832f2d54731f25b820b180b58606a5a (2026-10-19), 2026. Read 2026-10-21.

[MIK26c] MIKE team. mike_rs: MIKE in Rust. https://github.com/tensor-MIKE/mike_rs, commit 0d79121dad71d1211c68f1f20e4d843fd44faf91 (2026-10-06), the commit the measurements ran, vendored in the signature implementation's artifact with the parameter set of this paper's Section 3 added, 2026. Read 2026-10-22.

[Nat24a] National Institute of Standards and Technology. Module-lattice-based digital signature standard. FIPS 204, 13 August 2024, doi 10.6028/NIST.FIPS.204, https://csrc.nist.gov/pubs/fips/204/final, 2024. Read 2026-10-21.

[Nat24b] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. FIPS 203, 13 August 2024, doi 10.6028/NIST.FIPS.203, https://csrc.nist.gov/pubs/fips/203/final, 2024. Read 2026-10-21.

[NEL25] Jonas Nick, Liam Eagen, and Robin Linus. Shielded CSV: Private and efficient client-side validation. Cryptology ePrint Archive, Paper 2025/068, 2025. https://eprint.iacr.org/2025/068.

[NHG21] Kris Nuttycombe, Daira-Emma Hopwood, and Jack Grigg. ZIP 244: Transaction identifier non-malleability. Zcash Improvement Proposal, status Final, created 2021-01-06; read from the zcash/zips repository at commit d167dd3994e40790971f8f00b0b330885d2c9fe5 (2026-10-20), https://raw.githubusercontent.com/zcash/zips/d167dd3994e40790971f8f00b0b330885d2c9fe5/zips/zip-0244.rst, 2021. Read 2026-10-21.

[Plo26] Plonky3 contributors. Plonky3: a toolkit for polynomial IOPs. https://github.com/Plonky3/Plonky3, commit eab7f0e3662500cde47cb0b6dad61afc99e3deac (2026-10-08), the one this paper's provers build against, 2026. Read 2026-10-21.

[Ray26] Dustin Ray. ECLIPSE: Strongly unforgeable isogeny signatures from the prime-degree variant of PRISM. Cryptology ePrint Archive, Paper 2026/2312, 2026. https://eprint.iacr.org/2026/2312, received 2026-10-02, revised 2026-10-08.

[Rob26a] Damien Robert. MIKE (Module Isogeny Key Exchange): a fast and compact post-quantum NIKE. Announcement for the MIKE team on pqc-forum, 6 October 2026, and the thread's replies to 17 October 2026, among them Andrea Basso's of 6 October (the two sets' sizing), Damien Robert's reply of 6 October (the fifth message: SQIsign's primes can be reused), Michael Scott's of 15 October (the fast primes) and Sina Schaeffler's of 15 October (SQIsign's primes at one more word, https://groups.google.com/a/list.nist.gov/d/msgid/pqc-forum/CAEseHRofLBbGd%3Ds%2B%3D320j-A6DeGNMeidJHC7UOmMxFLzptWyeQ%40mail.gmail.com), https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/PUdioCNdEug, 2026. Read 2026-10-22.

[rob26b] robinhodde. Proposal: Lean 4 formal verification of the mutator set and security arguments, with source pinning. Neptune-Crypto/neptune-core issue 982, opened 2026-10-09, https://github.com/Neptune-Crypto/neptune-core/issues/982, 2026. Read 2026-10-21.

[Sch25] Thorkil Schmidiger. Hard fork 1 and hard fork policy. Neptune Cash, 17 March 2025; read from the site's source `src/articles/blog/hard-fork-1.mdx` at commit dd6f6d02a31ee2d9c922fc90eaa7735f7181f48b (2026-08-22), https://github.com/Neptune-Crypto/neptune-website, 2025. Read 2026-10-21.

[Sou26] Soul of Satoshi Project. Soul of Satoshi (SOS): A fully post-quantum layer-1 with opt-in shielded transactions and a proof-based, non-custodial Ethereum bridge. Whitepaper v2, draft, July 2026, https://soulofsatoshi.com/wp/; its source `whitepaper/sos-whitepaper-v2.tex` in the SOS-Soul-of-Satoshi/sos-node

repository at commit f0d13384a399aa352c6dfed12f481b8ae4a5a38d (2026-08-19), 2026. Read 2026-10-21 from the repository.

[Suc25a] Succinct. Formal verification of SP1 Hypercube. 9 October 2025, https://blog.succinct.xyz/nethermind-lean/, 2025. Read 2026-10-21.

[Suc25b] Succinct. SP1 Hypercube: Proving Ethereum in real-time. 20 May 2025, https://blog.succinct.xyz/sp1-hypercube/, 2025. Read 2026-10-21.

[Suc26] Succinct. SP1 Hypercube is now live on mainnet. 19 February 2026, https://blog.succinct.xyz/sp1-hypercube-is-now-live-on-mainnet/, 2026. Read 2026-10-21.

[swo25] sword-smith. Performance numbers for Triton VM proving. talk.neptune.cash thread 69, opened 27 May 2025, https://talk.neptune.cash/t/performance-numbers-for-triton-vm-proving/69, 2025. Read 2026-10-21.

[Sze22] Alan Szepieniec. Neptune and post-quantum security. Neptune Cash, 1 July 2022; read from the site's source `src/articles/blog/post-quantum.mdx` at commit dd6f6d02a31ee2d9c922fc90eaa7735f7181f48b (2026-08-22), https://github.com/Neptune-Crypto/neptune-website, 2022. Read 2026-10-21.

[Sze23a] Alan Szepieniec. Mutator sets. Neptune Cash, 10 August 2023; read from the site's source `src/articles/blog/mutator-sets.mdx` at commit dd6f6d02a31ee2d9c922fc90eaa7735f7181f48b (2026-08-22), https://github.com/Neptune-Crypto/neptune-website, 2023. Read 2026-10-21.

[Sze23b] Alan Szepieniec. On the valuation of cryptocurrencies. Neptune Cash, 26 September 2023; read from the site's source `src/articles/blog/valuation.mdx` at commit dd6f6d02a31ee2d9c922fc90eaa7735f7181f48b (2026-08-22), https://github.com/Neptune-Crypto/neptune-website, 2023. Read 2026-10-21.

[Tri26] Triton VM contributors. Triton VM. https://github.com/TritonVM/triton-vm, commit 2f3d61789b45bd90d3aaf7a6b191f5b46894c5bd (2026-09-30), 2026. Read 2026-10-21.

[Vær24] Thorkil Værge. Habemus verifier. Neptune Cash, 30 March 2024; read from the site's source `src/articles/blog/habemus-verifier.mdx` at commit dd6f6d02a31ee2d9c922fc90eaa7735f7181f48b (2026-08-22), https://github.com/Neptune-Crypto/neptune-website, 2024. Read 2026-10-21.

[Wes26] Benjamin Wesolowski. The supersingular isogeny problem in time and memory $p^{1/3+o(1)}$. Cryptology ePrint Archive, Paper 2026/1486, 2026. https://eprint.iacr.org/2026/1486, received 2026-07-20; read 2026-10-22.

[Zca21] Zcash Foundation. Zebra: Zcash zero-knowledge proofs at scale. ZKProof blog, 3 June 2021, https://zkproof.org/2021/06/03/zebra-zcash-zero-knowledge-proofs-at-scale/, 2021. Read 2026-10-22.

[zkS26] zkSecurity. Audit of Zcash: Orchard. Report for the Tachyon Foundation, 14 July 2026, https://reports.zksecurity.xyz/reports/zcash-orchard/, 2026. Read 2026-10-20.

A Tables moved from the body

The body refers to these tables; each carries its own source comments.

Notes to Table 1. The memo is 32 bytes per output in ANNULAR and 512 in Sapling and Orchard. At a 512-byte memo ANNULAR's body is $1{,}725 + 2 \cdot 480 = 2{,}685$ bytes, 3,180 all in with the proof's 495 at 860 transactions and 3,027 with 342 at 1,248. Sapling at a 32-byte memo is $2{,}756 - 960 = 1{,}796$, its 580-byte note ciphertext at 100 as ZIP 231's memo bundles make it [GNHS24]. Of the bytes per transaction, the proofs are 768 in Sapling (four proofs of 192), 7,264 in Orchard (two actions) and 495 in ANNULAR at the measured capacity. That capacity is 860 transactions in a 75 s batch at 0.079 s each with margin (Section 9). Per transaction, Sapling's verifier checks four Groth16 proofs at 7 to 10 ms each, Orchard's about 30 ms on one thread, MatRiCT+'s 2 to 8 ms, and Soul of Satoshi's tens of milliseconds per action. Those figures are unbatched. zcashd batch-validates the Groth16 and Halo 2 proofs and the RedJubjub and RedPallas signatures with multithreading, around 80 percent off its worst-case block validation on a Ryzen 9 5950X [Ele22]. Zebra's parallel batching is about three times faster on a chain sync [Zca21]. Neither publishes a per-proof batched figure, so both sides of the column are unbatched. ANNULAR's signature check is at the batched rate of [Ray26], 149 verifications a second on one thread, against 8.0 ms for a single verification and 6.5 with a prepared key. Aerie, which aggregates signatures and has no privacy, folds 32,768 Falcon-512 signatures in 0.65 s on an M4 Max, 42.1 bytes per signature, verified in 54 ms. SnarkPack folds 8,192 Groth16 proofs in 8.7 s, verified in 163 ms, pairing-based. Sources: Sapling [HBHW26, Hop21, Ele21]; Orchard [Ele26, Hop26a, Hop21, Ele21]; MatRiCT+ [ESZ22]; Soul of Satoshi [Sou26]; Neptune Cash [Sch25, swo25]; Aerie [CDBT26]; SnarkPack [GMN21]; ANNULAR, Section 9. ANNULAR's soundness at the batch is Theorem 1, instantiated in Table 16.

Table 6: The cost terms of a batch, as expressions.

a transaction's cost to the verifier	today (an elliptic curve)	post-quantum, a lattice KEM, per-transaction proofs	ANNULAR
verification	$t_v + 2t_s$	$t'_v + 2t'_s$, $t'_v > t_v$	$T/n + 2t'_s$
proof bytes	b_p	$b'_p \gg b_p$	B/n
data floor d	d	$d + \mathrm{ct}_{\mathrm{KEM}} \cdot o$	$d' \approx d$
the prover	p, the sender	$p' > p$, the sender	p_ℓ at the sender, $A(n)/n$ on one card
bytes per transaction at capacity n^*	$b_p + d$	$b'_p + d + \mathrm{ct}_{\mathrm{KEM}} \cdot o$	$B/n^* + d'$

$\mathrm{ct}_{\mathrm{KEM}}$ is a KEM ciphertext of hundreds of bytes and o the outputs; an isogeny NIKE key is a curve coefficient of tens of bytes and also signs, so $d' \approx d$.

The design space. Table 13 sets the three authorisation choices side by side under the built delivery, MIKE trial decryption. The two other delivery profiles change the bytes per transaction and the scan alone. An ML-KEM-512 mailbox puts a 768-byte ciphertext in place of each 82-byte ephemeral key, about 3,097 bytes per transaction, with the decapsulation row of Table 21 as its scan floor. Out-of-band delivery leaves about 801 bytes per transaction, the body less the ephemeral keys, the note ciphertexts and the outgoing records, with no scan and no recovery from the published transactions. The lattice row does not change the leaf. The shielded statement is hash-shaped, a Merkle path and Poseidon2 permutations, so a lattice proof system brings to it none of the speed it has on a signature-shaped statement. The signatures would leave the transaction under one aggregate proof per batch, as Aerie's do (Table 1).

Table 7: The two parameter sets. The MIKE set on the first prime is this work's addition to the vendored crate (Section 3); the second prime carries MIKE's rigorous level I set.

	SQIsign level I, this paper's MIKE set (measured)	MIKE rigorous level I
p	$3 \cdot 2^{324} - 1$	$117 \cdot 2^{374} - 1$
bits of p	326	381
$\mathbb{F}_p$, $\mathbb{F}_{p^2}$ bytes	41, 82	48, 96
two-adicity of $p+1$	324	374
MIKE walk length e	228	372
MIKE public key, secret, bytes	82, 29	96, 47
signature public key, signature, bytes	83, 206	97, 220
signature torsion exponent	195	195
RandPK hop degree (mode A)	2^{322}	2^{372}
radical steps of a spend's walk (mode B), $2f-4$	644	744

Table 8: Key material and what each key can do.

key	contents	can
spending key	the 32-byte seed	everything below; spend
full viewing key, 185 (224) bytes	$s_{\text{scan}}, E_{\text{sign}}, nk, ovk$	detect and decrypt notes; nullifiers and spends; own outputs
incoming viewing key, 153 (192) bytes	$s_{\text{scan}}, E_{\text{sign}}, nkc$	detect and decrypt notes; not the nullifiers
outgoing viewing key, 32 bytes	ovk	own outputs; disclose one
address, 206 (241) bytes	$E_{\text{scan}}, E_{\text{sign}}, nkc$	receive

Table 9: A transaction's fields and bytes.

field	type	bytes, shared (rigorous)
anchor	$\mathbb{F}_p$	41 (48)
per spend: nf	$\mathbb{F}_p$	41 (48)
per spend: rk	signature public key	83 (97)
per spend: sig	signature	206 (220)
per output: cm	$\mathbb{F}_p$	41 (48)
per output: eph_pk	MIKE public key	82 (96)
per output: ct	AES-256-GCM over value, ρ, r and a 32-byte memo	129 (136)
per output: out_ct	AES-256-GCM over the address and the ephemeral secret	251 (304)
value balance	signed 64-bit integer	8
proof	the leaf proof, to the aggregator, not the verifier	609,515
a 2-in-2-out body without its proof, as serialised		1,725

Table 10: The leaf's configurations: the standalone configuration and its neighbours, and the aggregation profile.

shape, prime	configuration	h	prove, s	verify, s	bytes
2-in-2-out, shared	blowup 16, 25 queries, Keccak-256; 100 conjectured bits	256	0.89	0.025	263,266
2-in-2-out, shared	blowup 16, 50 queries, Keccak-256; 100 proven bits	512	0.89	0.031	515,591
2-in-2-out, shared	blowup 32, 20 queries, Keccak-256	128	1.64	0.024	217,281
4-in-4-out, shared	blowup 16, 25 queries, Keccak-256	256	1.79	0.048	278,038
2-in-2-out, rigorous	blowup 16, 25 queries, Keccak-256	256	1.08	0.032	297,181
2-in-2-out, shared	setting B: blowup 32, 57 queries, Poseidon2; $2^{-100.1}$ proven	512	11.9 (1.4)		609,515 (611,240)
2-in-2-out, shared	the cost row: 66 queries, Poseidon2; $2^{-115.6}$ proven	512	n/m		704,840

Rows 1 to 5 on the results record's machine, 7 threads of a 2.8 GHz Xeon; rows 6 and 7 one leaf alone on the 16-core workstation, the prove on one thread with sixteen threads in parentheses. The 66-query row was not re-measured (n/m) after the prover's threading; before it the two rows' prove times were equal. The bytes are the proof's as shipped, without its preprocessed openings; row 6 adds the transaction's body in parentheses.

Table 11: The tree's proofs on the closeout build.

proof	extension, blowup, queries	regime, bits	bytes	verify, s
node ($K = 8$)	quartic, 8, 120	unique decoding, 100	2,870,578	0.101
join-of-nodes	quartic, 8, 120	unique decoding, 100	1,775,336	0.074
join-of-joins	quartic, 8, 120	unique decoding, 100	1,863,400	0.054
wrap	degree 8, 32, 42	Johnson, 102	425,678	0.015

Table 12: The four circuits on the RTX PRO 6000, the device off and on.

	CPU build: fill, prove	GPU build: fill, prove	proof bytes	verify, s
node (eight leaves)	1.29 s, 7.2 s	0.26 s, 0.4 s	2,870,578	0.101
join-of-nodes	1.27 s, 3.6 s	0.31 s, 0.2 s	1,775,336	0.074
join-of-joins	0.69 s, 3.0 s	0.12 s, 0.2 s	1,863,400	0.054
wrap	0.34 s, 7.4 s	0.09 s, 0.5 s	425,678	0.015
per transaction at $K = 8$, prove only	1.35 s	0.075 s		
per transaction, fill and prove	1.67 s	0.16 s		

The fills are unchanged since the closeout.

Table 13: The three authorisation choices under the built delivery: what each costs, measured or estimated.

authorisation	bytes per transaction	scan per output	leaf prove, s	recovery from the seed	measured
the signature outside the proof (built)	1,725	11.2 ms	11.9 (1.4)	yes	yes
a hash-based signature in the circuit	n/e	11.2 ms	n/e	yes	no
a lattice signature, aggregated by a lattice proof	about 5,168 (ML-DSA-44), 2,999 (Falcon-512)	11.2 ms	11.9 (1.4)	yes	estimated

The leaf prove is on one thread with sixteen threads in parentheses; n/e is not built and not estimated. The lattice bytes are the body with ML-DSA-44's 2,420-byte signature and 1,312-byte key [Nat24a], or Falcon-512's 666, the padded format, and 897 [FHK+20], in place of the signature's 206 and 83; aggregated, the signatures cost about 42 bytes each in the aggregate [CDBT26].

The cards. Table 22 is one column per card, each card's byte gate passed against its own CPU build. The 32 GB and 24 GB cards copy every low-degree extension down. The 5090's per-transaction prove is 2.7 times the server card's with memory bandwidth close to it, so the gap is the copies, not the silicon. A consumer card is within a factor of two to three of the server card on this prover, and the factor is memory capacity. The full tree has run on the RTX PRO 6000 alone, a 96 GB card. On a 40 GB card the tree with a builder in flight exceeds the device at any resident budget. The fill stores, the stream sets' scratch and the Merkle trees sit outside the budget the admission rule counts. The reference wrap at blowup 32 holds the device memory Table 14 lists at its peak, and a running proof's copies are never evicted, so a 32 GB card cannot produce it under any budget. The measured alternative is the wrap at blowup 16, in the same table. The change that removes the limit is a running proof's extension copies made evictable, or its quotient chunks and FRI layers streamed.

Table 14: The wrap's blowup and arity at 105 proven bits without grinding in the record's reading (102 with η, Section 7), on the RTX PRO 6000.

blowup, arity	queries	prove, s	proof bytes	verify, s	peak VRAM, GiB
8, 4	70	0.3	618,138	0.020	19.1
8, 8	70	0.2	612,849	0.019	19.1
16, 4	53	0.3	511,201	0.016	23.9
16, 8	53	0.4	504,843	0.017	24.0
32, 4	42	0.6	436,579	0.016	33.1
32, 8 (the reference)	42	0.6	425,678	0.015	33.0

The transcripts' κ. The transcripts and their κ are three. Keccak-256 serves a leaf proof the verifier checks directly, the standalone configuration, $\kappa = 256$. The Poseidon2 duplex with one capacity lane serves a leaf the recursion consumes, a 326-bit lane on the shared prime and 381 on the rigorous one. So $\kappa = 326$ or 381, and 163 or 190 bits against collisions. Plonky3's Poseidon2 duplex over Baby Bear serves the nodes, the joins and the wrap, eight capacity lanes of 30.9 bits, $\kappa = 247$. The term uses the smallest, $\kappa = 247$: every transcript of the batch is one random oracle under the one query budget Q, and the smallest capacity bounds the collision term. At $Q = 2^{64}$ that term is 2^{-117}, and the batch's bound is Q times the union; Table 16 gives it at $Q = 2^{64}$ and at $Q = 2^{80}$.

The cost of 105 bits at the batch. For every term of the measured batch's union to sit under 2^{-105} together, each of its proofs needs about 115.6 bits. That is 140 queries for

Table 15: The forged-proof tables.

suite	mutant classes	rows
the leaf	a trace cell edited after witness generation; a public value; a selector; a value outside $\mathbb{F}_p$; every relation of the statement; the proofs exported for the Python verifier	every relation
the node over one leaf	the leaf proof's cells, public values and caps; a non-canonical opening failing the canonical check alone	
the node over eight slots	an inactive slot with a real proof's values; an active slot with an inactive leaf's zeros; two slots sharing one leaf proof; one slot's public values swapped with another's	
the leaf verifier in the node	a byte of the child proof flipped in the FRI openings; a child public input flipped; the child key's commitment flipped; a flipped opened cell	6
the joins (both kinds)	a child swapped for a valid proof of another tree; a child's public-input chain flipped; the children's order swapped; a byte of a child flipped; a child's key commitment flipped	10
the earlier batch aggregate	a tampered inner proof; a wrong public input; a duplicated inner proof; a proof under the wrong key	4
the demo's negative runs	a leaf with a flipped nullifier refused at the gate; a flipped node proof after the tree is built, the batch still accepted as designed; a flipped wrap refused, the state unchanged	3

Table 16: Theorem 1 instantiated: the union bound at the measured batch and, as a remark, over a sequence of batches.

what is accepted	the union	bits
one node, one join, the wrap	per proof	100, 100, 102
a two-node batch's recursion ($K = 8$, 16 transactions)	$4 \cdot 2^{-100} + 2^{-102}$	98
the 1,248-transaction batch's recursion	$311 \cdot 2^{-100} + 2^{-102}$	91.7
the same batch with its 1,248 leaves	$+\ 1{,}248 \cdot 2^{-100.1}$	89.6
the probability at $Q = 2^{64}$ oracle queries under Fiat–Shamir	$2^{64} \cdot 2^{-89.6} + 3(2^{128}+1)/2^{247}$	25.6
the probability at $Q = 2^{80}$	$2^{80} \cdot 2^{-89.6} + 3(2^{160}+1)/2^{247}$	9.6
a sequence of 2^{22} such batches, recursion alone (remark)	2^{22} times the batch's	69.7
a sequence of 2^{22} such batches, with the leaves (remark)		67.6

the nodes and joins, 47 for the wrap at the record's 2.5 bits a query and 48 at 2.43, and 66 for the leaf. Table 17 is the measured cost. The queries add openings, not prover work, so the node's and the leaf's prove are unchanged and the bytes grow by about a sixth. The joins and the wrap at the higher counts were not measured, since a join over a 140-query child is a different circuit. The batch's wall would move with the joins' verification rows.

Proven against conjectured. Under the list-decoding-capacity conjecture a query at blowup 2^b would buy b bits, 210 for the wrap's queries and twice the proven reading for the leaf's standalone configuration. The conjecture is refuted as stated [CS25, KKH26], and the paper claims the proven figures only. Grinding [BLB+21] is implemented and not used by the reference configuration. A wrap at blowup 8 with 57 queries and 20 grinding bits reads 101.5 bits of work and 81.5 of probability at $\eta = \sqrt{\rho}/20$, the record's 105.5 and 85.5 at $\eta \to 0$. A union bound is a statement about probability.

Table 17: The reference configuration against the one that puts a 1,248-transaction batch's union under 2^{-105}.

proof	queries, reference / 105 at the batch	prove	bytes	verify
node	120 / 140	0.5 s / 0.5 s	2,870,578 / 3,326,275	0.101 s / 0.114 s
leaf	57 / 66	11.9 (1.4) s / n/m	609,515 / about 703,100	
join-of-nodes	120 / 140	0.2 s / n/m	1,775,336 / about 2.07 MB projected	0.074 s /
join-of-joins	120 / 140	0.2 s / n/m	1,863,400 / about 2.17 MB projected	0.054 s /
wrap	42 / 47	0.6 s / n/m	425,678 / about 476 KB projected	0.015 s /

The leaf's prove is on one thread with sixteen threads in parentheses; n/m is not measured; the projected bytes are the record's own projections.

The balance lift. The balance is lifted from the field identity to the integers. For witnesses with at most 2^{16} terms and $p > 2^{82}$, an integer below 2^{81} in absolute value whose image is the value balance equals the witnesses' sum over the integers.

The Lean build. Lean `v4.34.1` with mathlib at the same tag. The development is 84 modules and 36,932 lines, of which the generated modules are 20,000. A fresh-checkout build with mathlib from source is clean in 24 min 15 s on the 32-thread workstation.

Table 18: The Lean coverage table.

claim	in Lean	evidence otherwise
1. proof-system soundness of the four leaf systems	the subfield check's polynomial core: `G_eval_eq_zero_iff`, `E_ne_zero`, `card_roots_le`, `exists_Q`; not the STARK	the published soundness argument; a Python verifier agreeing on 178 exported proofs (4 accepted, 174 rejected)
2. the leaf constraints mean the spend statement	all four systems: `sound_2x2`, `sound_4x4` on each prime; the balance lifted to the integers (`balance_int`); the relation's three non-literal encodings: the position by its closed form (`rev32`), the balance as a field identity lifted by `balance_int`, the booleans by type	the leaf mutant suite
3a. verifier equivalence	none	the mirror, event by event (69,385 events of a node proof); the honest suites; the forged-proof tables
3b. the small-field bound at the recursion's parameters	none	the arithmetic of Table 3; the byte gate
3c. the gadgets: 31-limb arithmetic, range checks, Poseidon2 rows, LogUp buses	none	the mutant tables; the bus audit; the cell-for-cell trace gate
4. the union bound over the tree	none; a stated computation	Table 16
5. the verifier's state invariants	none; the per-transaction balance as an integer	the verifier's checks and tests; the demo's negative runs
6. model to code	the constraints extracted from the prover by a hook a test keeps current; the schedule decided row by row; the node key hashes the extraction	no Lean model of the verifier; correspondence by inspection
7. not claimed		zero knowledge beyond the hiding bound; the signature's and the NIKE's security beyond their papers; constant time; the GPU prover beyond byte identity; the hash's parameters

Row 1's published argument is [BCI+20, BGKS19, BGK+23]; row 7's hiding bound is [HK24].

Table 19: The gates between the CPU and the GPU provers, on the closeout build.

gate	what passed
the check phase	every device step against its Plonky3 reference on a one-leaf node; the row-by-row checks on the wrap and on an eight-leaf node
the byte gate	five proof files byte-identical between the CPU and the GPU builds, sequentially and four at once: node 2,870,578 bytes, a second node 2,878,258, join-of-nodes 1,775,336, join-of-joins 1,863,400, the wrap 425,678
the fill gate	the device fill against the CPU filler cell for cell, at the hashing step on the host and on the device; the byte gate on the five files and the six honest suites under the device fill
the batch digest	FNV-1a over the node, join and wrap proof bytes in batch order, 38e9fb8ddd411922, the same at any worker count

Table 20: Claims and proof sketches: each property as a claim under its hypotheses, with the notion of Section 4.1 it instantiates and the sketch of Section 7.3 that argues it.

claim	hypotheses	notion	argument
anonymity of outputs	H_2 hiding under uniform r; MIKE's NIKE security; RandPK key indistinguishability; AES-GCM confidentiality	Definition 1	Sketch 1
anonymity of spends	the STARK's zero knowledge, Theorem 6 of the hiding construction at the budget of Section 7.3; H_3 a PRF in nk; rk unlinkable under fresh r_{rk}; the signature leaks nothing beyond rk	Definition 1	Sketch 1
dummies hide the structure	a dummy's nf, rk and signature as a real note's under a fresh key; en a witness	Definition 1	Sketch 1
balance	H_2 and tree-hash binding; the STARK's soundness at the stated bits; the range check and the field-size argument	Definition 2	Sketch 2
non-malleability	identity by txh; the signatures over txh; the transcript absorbs txh; existential unforgeability under randomised keys and the STARK's knowledge soundness (Section 7.3); the instance identifier	Definition 3	Sketch 3
spend authority	the signature's existential unforgeability under the randomised key, over txh (Section 7.3); the proof binds rk to a walk from E_{out}; H_2 binding	Definition 4	Sketch 4
nullifier integrity	H_4 ties nk to the nkc in cm; H_2 ties ρ to it; the path bits tie the position to the anchor	Definition 5	Sketch 5
outgoing records	AEAD confidentiality under ock; with ovk, checked against the output through eph_pk and cm	none	none
payment disclosure	verified from the published transaction alone; reveals one output; the signed form adds spend authority	none	none

The hiding construction of the second row is [HK24]; the specification's security section states every row as a claim with its assumptions. The outgoing records and the disclosure have no notion in Section 4.1; their claims are stated against their own checks.

Table 21: The scan rate on the workstation.

	median, 2,496 outputs	outputs a second	one output
MIKE trial decryption, one thread	27.897 s	89	11.2 ms
MIKE trial decryption, 31 threads	1.531 s	1,630	
ML-KEM-512 decapsulation alone, one thread	0.092 s	27,054	37 μs
ML-KEM-512 decapsulation alone, 31 threads	0.005 s	470,543	

Table 22: The cards, each column on its build; the wrap rows at blowup 8.

	RTX PRO 6000 (server; the build of record)	RTX PRO 6000 Max-Q (an earlier build)	RTX 4090 (the same)	RTX 5090 (the same)
host CPU, threads for the prover	Xeon 6767P, 30	Threadripper PRO 7995WX, 192	Threadripper PRO 7975WX, 64	EPYC 9755, 48
VRAM, resident budget, GiB	95, 52	95, 52	24, 13	32, 17
the LDEs	resident	resident	download mode	download mode
node: fill, prove, CPU / GPU	1.29, 7.2 / 0.26, 0.4 s	1.02, 4.9 / 0.21, 0.6 s	1.02, 7.2 / 0.89, 2.5 s	0.88, 4.4 / 0.65, 1.8 s
join-of-nodes, CPU / GPU	1.27, 3.6 / 0.31, 0.2 s	0.76, 1.9 / 0.21, 0.2 s	0.71, 3.2 / 0.55, 1.2 s	0.58, 1.4 / 0.44, 0.9 s
join-of-joins, CPU / GPU	0.69, 3.0 / 0.12, 0.2 s	0.34, 1.5 / 0.09, 0.2 s	0.31, 2.9 / 0.39, 1.1 s	0.28, 1.2 / 0.29, 0.8 s
wrap, CPU / GPU (the server card's at blowup 32, the others' at blowup 8)	0.34, 7.4 / 0.09, 0.5 s	0.24, 1.7 / 0.07, 0.5 s	0.21, 2.8 / 0.21, 1.1 s	0.21, 1.4 / 0.18, 0.9 s
per transaction, prove only, CPU / GPU	1.35 / 0.075 s	0.85 / 0.10 s	1.30 / 0.46 s	0.73 / 0.34 s
per transaction, fill and prove, CPU / GPU	1.67 / 0.16 s	1.07 / 0.153 s	1.52 / 0.64 s	0.91 / 0.47 s
peak VRAM, node and join-of-joins, GiB	18.5, 22.4	18.5, 22.4	18.0, 21.2	18.3, 22.6

Table 23: The kernel headroom on the closeout build's timeline: a measured ceiling a family, and what was done with each.

family	sequential s a batch	on the timeline, s	headroom	the outcome
quotient	22.9	38.7	11.1 by bandwidth, an upper bound	open: the counters first, then fewer passes over the LDE rows
NTT	25.3	42.7	6.2	taken: one launch a stage for a chunk of columns; a node proof's NTT kernels 0.15 to 0.06 s, the batch 105.3 to 100.5 s at the same digest
hash	22.8	38.5	2.0	measured and not kept: two states a thread slower; four states' bytes differ on the join-of-joins
DEEP	4.8	8.1	2.3	measured and not kept: one launch over every matrix of a height, within a run's spread
the fill's terms and multiply-accumulate	10.2	17.2	not measured	the serial big-integer arithmetic; the leaf verifier's change takes it out
the fill's recursion records	7.5	12.7	not measured	as the terms
LDE spread and transpose	5.0	8.4	1.2	at the ceiling

B What is left, with its reasons

The device items are the ones left open at the final build.

- *The hash-leaf variant.* A leaf committing under a small-field hash would leave the recursion no big-field hashing to verify. It is designed and not built, because its leaf is a new circuit with a new Lean development.
- *The signature in the circuit.* Verifying the prime-degree PRISM signature over txh inside the leaf would remove the per-transaction check and the 206 bytes per spend from the transaction. Its verifier is about 18 megacycles of $\mathbb{F}_{p^2}$ arithmetic. The leaf would grow by an estimated one to two orders of magnitude in rows and the node phase by an estimated 1.5 to 2 times; neither is measured. The hash-to-prime step would stay outside the proof as a public input. The alternative is a dedicated aggregate argument for isogeny signatures, Aerie's shape on a different algebra, and none exists.
- *The quotient's ceiling.* The quotient is the largest kernel family on the batch, and its headroom by bandwidth (Table 23) is an upper bound. Its true ceiling needs the profiler's counters, which the rented hosts lock.
- *The leaf verifier's change.* The node fills the big-field leaf's verifier over Baby Bear while the leaf's commitments and transcript are the native field's, so the fill's largest item is the terms (Table 23). A small-field leaf would take them out of the recursion, a change to the leaf's proof system and not to the aggregator.
- *The leaf prover on the card.* The leaf's one-thread and sixteen-thread times (Table 10) are 151 and 18 times the aggregation's 0.079 s share per transaction, so a batch's leaves scale with the senders' cores. The preprocessed trace's transform and tree, a third of the one-thread leaf and the same for every proof under a key, are the next lever, and a port to the card is its own program.
- *The wrap on a 32 GB card.* The wrap at blowup 32 holds the device memory of Table 14 at its peak, and a running proof's copies are never evicted. The change is a running proof's extension copies made evictable with a re-upload, or the quotient chunks and FRI layers streamed.
- *A second card, as arithmetic.* The node phase is most of the wall on one card at 75 percent busy (Table 4). Two cards would take the nodes in parallel in about 46 s and the batch in about 53 s, 0.043 s per transaction, which is not measured and needs a device index per stream set.
- *Lean.* In order: the union bound as a theorem over stated per-proof inputs, the in-circuit verifier's equivalence to the native one with the mirror as the pivot, and the small-field bound pinned to the configuration constants. Then the verifier's state invariants over a sequence of batches and a Lean verifier for the leaf STARK, each its own development on top of the leaf theorems.

C Payment disclosure

The sender of an output can prove to a third party what the output is, from the published transaction alone, following [Gri26]. A disclosure is $(txid, k, eph_sk, v, \rho_{\text{bytes}}, r, memo, address)$ and optionally a signature under the spend-authorising key of one of the transaction's inputs over the disclosure's bytes under a domain separator of its own. The third party checks the ephemeral secret against the output's ephemeral key and derives the shared secret with the address's scanning curve. It then opens the ciphertext and checks the plaintext, ρ_{bytes}, E_{out} and cm in turn, each a named refusal, and the signature under the input's rk if present. It reveals the one output's recipient, value, memo and

randomness and the ephemeral secret used for that output alone. It does not reveal the transaction's other outputs, which notes the inputs spend, the sender's keys, or whether sender and discloser are one party. The signed form adds that whoever made it held an input's spend-authorising key. Its wire form's bytes are in Table 9.

D Autonomous methodology

Original contributions. We knew that Sapling-style shielded pools rest on primitives a quantum computer breaks, that post-quantum replacements exist for each, and roughly where each replacement is strong or weak once it is placed in a pool. We had been taught isogeny cryptography after the SIDH break and knew that its keys and signatures are the smallest in the post-quantum field, and from the block size debates that bytes per transaction are the cost a ledger feels. We knew which proof systems exist and their asymptotics, that isogeny arithmetic can be written as constraints, that large proofs compose and fold, and that folding turns the verification of many statements into one cheap check, which is the leverage this paper uses. We knew that a Sapling-shaped pool needs re-randomisable keys, that SQIsign cannot be strongly unforgeable and why, and that the prime-degree PRISM variant existed on paper with no implementation. We knew that the fixed cost of a quantum-safe signature could be shared with the key exchange if one key served both roles, and waited for MIKE for that reason. We knew that GPUs would change the proving numbers. We are also acquainted with the attacks on SIDH and chose these primitives with caution; we are aware too of the productive repurposing of those attacks into the next generation of isogeny cryptography, and seek their productive application.

What the agents were directed to write. In the absence of in-house CUDA, Plonky3 or Lean expertise, coding agents were directed to write the GPU kernels, the leaf and recursion constraints, the Lean development, the test harnesses, the measurement scripts and the first drafts of this text. We set the questions, the design, the parameters and the acceptance rules, and reviewed everything kept. A prover change was kept only if its proofs were byte-identical to the previous build's on fixed inputs, every forged input was rejected, and the number it was made for moved by more than the run-to-run spread. Agents' reports were checked against the record before use. That the signature's field admits native arithmetic in the leaf was found in the work, not planned.

Disclosure. The agents generated the code, the measurements and the first drafts of the prose; the authors generated the questions, the design, the security model, the acceptance rules and the final text, and accept responsibility for all of it.