ZAP-Native PQ-Secured Rollups via the P3Q Precompile

Zach Kelling

Lux Industries

2026-06-03

Abstract

We present a rollup pattern where the sequencer signs each batch with a Pulsar threshold signature, the parent L1 verifies that signature via the P3Q precompile at EVM slot `0x012205`, and the rollup state inherits the parent L1's QuasarCert finality. No rollup-side validator set, no fraud-proof bond for the cryptographic path, no codec hop on the sequencer-to-L1 wire. Optimistic and ZK variants share the same envelope; the difference is whether the batch carries a STARK/Groth16 proof of correct execution. End-to-end commit latency from sequencer batch close to parent L1 finality at PQ-strict: 10 ms–50 ms. This paper expands on LP-218 (`lps/LP-218-zap-p3q-pq-rollup.md`) with the six-step pattern as a numbered list (alongside the user's five-step framing), the four wire envelopes (`0xE0`–`0xE3`), the P3Q precompile interface and honest gas accounting, the two rollup modes (optimistic vs ZK), and the inherited cert mode contract that ties rollup finality to the parent L1's LP-217 mode.

Contents

1 Introduction

This paper expands on LP-218 (`lps/LP-218-zap-p3q-pq-rollup.md`) with the six-step pattern as a numbered list (alongside the user's five-step framing), the four wire envelopes, the P3Q precompile interface and honest gas accounting, the two rollup modes (optimistic vs ZK), and the inherited cert mode contract that ties rollup finality to the parent L1's LP-217 mode.

1.1 Setup: what a rollup is here

A ZAP-native PQ rollup is a Tier-3 chain (per LP-204) whose sequencer collects N transactions as ZAP frames, applies them against the rollup state, computes a new state root over the ZAP-buffer Merkle structure defined in LP-203, and signs the tuple (prev_root, new_root, batch_hash, sequencer_id) with a Pulsar threshold signature (LP-073). The sequencer submits `RollupBatchTx{prev_root, new_root, batch_hash, pulsar_sig}` as a ZAP frame to Z-Chain (the parent L1 Tier-1 hosting the rollup-VM at Tier-3 in the LP-204 hierarchy).

1.2 What was wrong before

Traditional optimistic rollups (Arbitrum, Optimism) require a rollup-side validator set or sequencer + fraud-proof window of ≥ 7 days, plus a bridge contract on the parent L1 that maintains its own state. Three concrete failures:

1. **Cryptographic authenticity is implicit.** The sequencer is a single entity (or unaudited multisig) with no cryptographic attestation that survives parent-L1 verification. The bridge trusts the sequencer's signature on a classical secp256k1 key.
2. **Finality window is operational, not cryptographic.** The 7-day challenge window is a function of operator policy, not a cryptographic property. Reducing it requires a fraud-proof game executed by validators who must observe and challenge within the window.
3. **PQ posture is absent.** The bridge contract verifies secp256k1 ECDSA, broken by Shor's algorithm. A store-now-decrypt-later adversary can wait until quantum hardware is available to forge sequencer batches retroactively.

1.3 What this LP solves

LP-218 ties rollup finality to the parent L1's QuasarCert. Z-Chain validators verify the Pulsar signature via the **P3Q precompile at EVM slot `0x012205`**. P3Q is the on-chain verifier defined in HANZO-CRYPTO-SUITE ğ5.2 – a single precompile that takes a Pulsar threshold signature, a message hash, and a group public key, and returns true iff the signature validates. The verification is part of normal Z-Chain block validation; no second consensus is needed. The Z-Chain block then finalizes via the parent L1's QuasarCert (LP-202 tier degradation: PQ-off → PQ-fast → PQ-strict → PQ-heavy per LP-217).

The rollup state inherits the parent L1's full cryptographic finality because the rollup commit IS a Z-Chain transaction and Z-Chain IS the parent L1's rollup substrate (LP-063 + LP-204). No rollup-side validator set exists. No fraud-proof bond is required for the cryptographic path – the Pulsar signature is sufficient for the authenticity of the state transition; an optimistic-rollup challenge window is only needed if a fraud-proof system is the integrity model of the execution itself.

2 The Pattern

2.1 Six-step framing

1. **Collect.** Sequencer collects N transactions; each is a ZAP frame; no re-encode.

2. **Apply.** Sequencer applies the batch; computes the new state root over the ZAP-buffer Merkle structure (LP-203 GPU Merkle kernel).

3. **Sign.** Sequencer Pulsar threshold-signs the tuple (prev_root, new_root, batch_hash, seq_id).

4. **Submit.** Sequencer submits `RollupBatchTx` as a ZAP frame to Z-Chain over QUIC.

5. **Verify.** Z-Chain validators call P3Q (`0x012205`) to verify the Pulsar signature over $H(\text{prev\_root} \| \text{new\_root} \| \text{batch\_hash})$.

6. **Finalize.** Z-Chain block finalizes via the parent L1 QuasarCert; rollup state inherits parent L1 finality at the configured LP-217 mode.

2.2 Five-step framing (user's framing)

The user's mental model groups steps 1–2 together as "the sequencer constructs the batch":

1. Sequencer constructs the batch (collect + apply) and computes the new root.

2. Sequencer Pulsar-signs the root tuple.

3. Sequencer submits `RollupBatchTx` to Z-Chain.

4. Z-Chain validators call P3Q to verify the sig.

5. Z-Chain block finalizes via parent L1 QuasarCert.

The two framings are equivalent; the six-step framing breaks the "construct" step into Collect and Apply for stage-cost accounting, which is useful when measuring per-stage GPU utilization. Both are canonical.

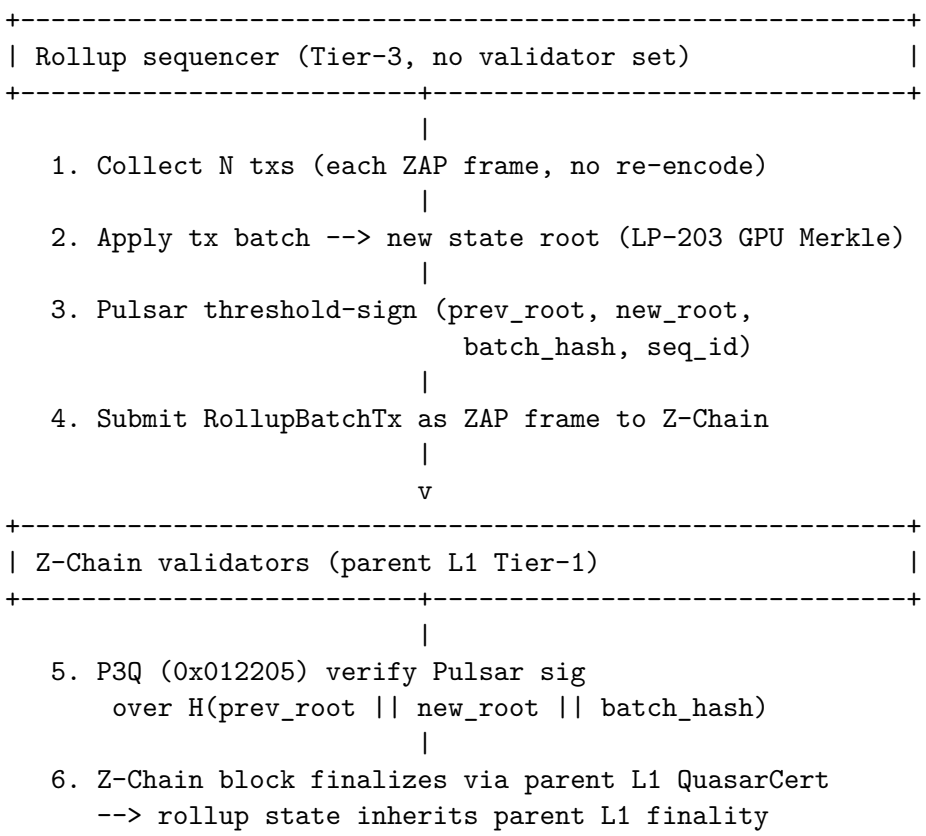
2.3 Stage-by-stage cost table

#	Stage	Where it runs	Cost target
1	Tx collection	Sequencer node(s)	NIC-bound, ZAP frames at LP-201 wire speed
2	State transition + Merkle root	Sequencer GPU (LP-203 UMA buffer)	∼100 µs per 1k-tx batch
3	Pulsar threshold sign	Sequencer group (1-of-1 or t-of-n)	∼5 ms at N=64 on H100
4	RollupBatchTx submission	ZAP frame over QUIC (LP-201)	∼1 ms RTT to nearest Z-Chain validator
5	P3Q verify on Z-Chain	Z-Chain validator EVM precompile	∼50 µs per verify on Blackwell sm_120
6	Parent L1 finality	Z-Chain QuasarCert	LP-217 cert mode: PQ-fast ≈ 5 ms, PQ-strict ≈ 15 ms

Table 1: Stage-by-stage cost. Source: LP-203 measured benches at `/tmp/spark-gpu-bench/` (stages 2–3 Pulsar GPU) and the LP-217 cert table (stage 6).

End-to-end commit latency from sequencer batch close to parent L1 finality at PQ-strict: 10 ms–50 ms. End-to-end at PQ-off (BLS only): 1 ms–5 ms.

2.4 Diagram



3 The P3Q Precompile

P3Q is an EVM precompile at slot `0x012205`. It is the unified PQ-verifier family that dispatches on a 1-byte `kind` discriminator: $0x01$ Pulsar (FIPS-204 ML-DSA threshold), $0x02$ Corona (Ring-LWE threshold), $0x03$ Magnetar (FIPS-205 SLH-DSA threshold). The earlier draft of LP-218 allocated separate slots for each kind; the 2026-06-03 decomplection collapsed them onto a single slot with kind-byte dispatch (LP-220 §"Architectural decision – single slot, kind-byte dispatch"). One slot, one ABI, three kinds.

3.1 Solidity ABI

```
// EVM precompile at slot 0x012205 -- unified PQ verifier family
function verify(
    uint8    kind,         // 0x01 Pulsar | 0x02 Corona | 0x03 Magnetar
    bytes calldata sig,    // primitive-specific threshold sig
    bytes32 messageHash,   // H(prev_root || new_root || batch_hash)
    bytes calldata pk      // primitive-specific group public key
) external view returns (bool);
```

The wire format used by the reference implementation places the `kind` byte at offset 0, the primitive-specific `mode` byte at offset 1, then the existing `(sigLen, sig, pkLen, pk, messageHash)` layout. Canonical Pulsar / ML-DSA-65 input is 5303 B (5302 B pre-decomplection + 1 B for the kind byte).

3.2 Properties

- **Stateless.** No storage reads, no storage writes. Function of inputs only.

- **Pure cryptographic check.** Returns true iff `sig` is a valid threshold signature of the named `kind` on `messageHash` under `pk`.
- **Constant-time.** No data-dependent branches; no side channels.
- **Slot.** `0x012205` per HANZO-CRYPTO-SUITE §5.2. Single slot hosts all three kinds; kind-byte dispatch routes to the primitive's verifier.
- **Kind dispatch.** Unknown or reserved-not-yet-wired `kind` values return `abiFalse` – no revert – so the Solidity caller can detect via the boolean return rather than via an exception path.

3.3 Gas cost (honest accounting)

Verifier	Gas	Blackwell time	Notes
`ecrecover` (secp256k1)	3000	∼3 µs	Classical; broken by Shor's
BLS pairing verify	5000	∼10 µs	Classical; broken by Shor's
P3Q (Pulsar / ML-DSA-65)	50 000	∼50 µs	**PQ-secure**
Re-running rollup consensus	N/A	5 ms–50 ms	What we are NOT doing

Table 2: Honest gas comparison.

P3Q is more expensive than BLS verify because PQ verification is inherently heavier – lattice operations dominate elliptic-curve operations on every CPU and most GPUs. The 10× factor is honest, not pessimistic. Blackwell sm_120 closes the absolute-time gap (50 µs P3Q vs 10 µs BLS verify), but the gas accounting reflects CPU-class verify cost as the gas calibration baseline.

P3Q is dramatically cheaper than re-running consensus to attest to the rollup batch. The rollup pays 50 000 gas per batch instead of running an entire validator set per batch. This is the economic case for the pattern.

3.4 Verifier reference

Artifact	Path	Purpose
Go reference impl	`lux/standard/precompiles/p3q/p3q.go`	EVM precompile at slot `0x012205`
Solidity consumer ABI	`lux/standard/precompiles/p3q/p3q.sol`	Solidity shim
Pulsar verifier core	`lux/pulsar/verify/`	Constant-time FIPS 204 verifier – P3Q wraps this
GPU kernel	`lux/accel/cuda/p3q.cu`	Blackwell sm_120 batched verify path

Table 3: P3Q implementation artifacts.

P3Q's verifier core IS the FIPS-204 ML-DSA verifier – Pulsar signatures serialize byte-identically to FIPS-204 ML-DSA signatures per LP-073 wire format [1]. Any conformant FIPS-204 verifier suffices. The precompile exists to give EVM-side callers a stable ABI and an audited gas charge.

3.5 Pulsar-only verifier (not generic PQ)

P3Q verifies Pulsar / FIPS-204 ML-DSA signatures specifically. It is NOT a generic PQ-signature precompile. A Corona (Ring-LWE) threshold signature verified on-chain requires a separate precompile – planned as LP-220.

Rationale: Pulsar is the threshold-PQ signature already in production across the Lux stack (consensus legs, validator keys, treasury authorization). Wallets, sequencers, and tooling already

produce Pulsar signatures. A second PQ verifier proliferates the surface area for no immediate benefit. When Corona-threshold-signed rollups become operationally relevant, LP-220 adds the dedicated precompile slot.

4 Wire Schemas

Rollup transaction envelopes are ZAP frames in the `0xE0`–`0xEF` rollup schema range. The umbrella LP-200 reconciles the range against LP-211 (cross-shard wire `0xE2`–`0xE7`) and LP-208 (DAG header/body, current allocation `0xF0`–`0xF1`).

Schema	Envelope	Kind	Purpose
`0xE0`	`RollupBatchTx`	`0x40`	Sequencer commit of a batch
`0xE1`	`RollupChallengeTx`	`0x41`	Fraud-proof challenge for optimistic rollups
`0xE2`	`RollupExitTx`	`0x42`	User exit from rollup to parent L1
`0xE3`	`RollupGenesisTx`	`0x43`	Rollup-creation tx at deploy time
`0xE4`–`0xEF`	reserved	—	Future rollup-VM envelopes

Table 4: Rollup wire schemas. Note: per LP-211, schemas `0xE2`–`0xE7` are reserved for cross-shard wire; the umbrella allocation in LP-200 resolves this so that LP-218 rollup envelopes use `0xE0`, `0xE1`, plus higher slots in the rollup range. The exact final schema-ID-to-name mapping is reconciled in the LP-200 schema map.

4.1 RollupBatchTx (`0xE0`, Kind `0x40`)

Fixed section (offset table):

Offset	Field	Size	Description
0	Kind	1	`0x40`
1	Version	1	`0x01`
2	SequencerID	20	Sequencer node ID (or threshold-group ID for t-of-n)
22	PrevRoot	32	State root before this batch
54	NewRoot	32	State root after this batch
86	BatchHash	32	`sha256(concat(tx_i bytes))`
118	BatchNum	8	Monotonic batch counter per rollup
126	TxCount	4	N – number of txs in batch
130	PulsarSigOff	4	Offset to Pulsar threshold sig in tail
134	GroupPubKeyOff	4	Offset to Pulsar group public key in tail
138	ZKProofOff	4	Offset to STARK / Groth16 proof (zero if optimistic)
142	TxFramesOff	4	Offset to packed list of tx ZAP frames

Table 5: `RollupBatchTx` field layout.

Variable-length tail: Pulsar sig, group pubkey, optional ZK proof, packed tx frames. The tx frames are the exact same ZAP buffers the sequencer received from clients – no re-encode, no re-frame.

4.2 RollupChallengeTx (`0xE1`, Kind `0x41`)

Submitted by any peer to challenge an optimistic rollup batch during its challenge window.

Offset	Field	Size	Description
0	Kind	1	`0x41`
1	Version	1	`0x01`
2	ChallengedBatchID	32	`sha256` of the challenged `RollupBatchTx`
34	ChallengerID	20	Challenger node ID
54	FraudWitnessOff	4	Offset to fraud witness in tail
58	BondAmount	8	Challenger bond (slashable if invalid)

Table 6: `RollupChallengeTx` field layout.

Tail: fraud witness – typically a single tx ZAP frame plus a state witness showing the sequencer applied the tx incorrectly. The Z-Chain re-executes the tx witness; if the sequencer's claimed new root disagrees, the batch is rejected and the sequencer is slashed. Omitted entirely from ZK rollups (no challenge window).

4.3 RollupExitTx (`0xE2`, Kind `0x42`)

User exits a rollup back to the parent L1 by proving inclusion of a withdrawal record in a finalized batch.

Offset	Field	Size	Description
0	Kind	1	`0x42`
1	Version	1	`0x01`
2	UserAddress	20	Parent L1 address receiving the exit
22	BatchID	32	Anchored batch ID
54	ExitAmount	32	uint256 – amount being exited
86	TokenID	20	Token contract address on rollup
106	MerkleProofOff	4	Offset to state-membership proof

Table 7: `RollupExitTx` field layout.

4.4 RollupGenesisTx (`0xE3`, Kind `0x43`)

Submitted once at rollup deploy time.

Offset	Field	Size	Description
0	Kind	1	`0x43`
1	Version	1	`0x01`
2	RollupID	20	Globally-unique rollup ID
22	DeployerAddress	20	Parent L1 deployer address
42	RollupMode	1	`0x00` optimistic, `0x01` ZK
43	CertModeFloor	1	LP-217 cert mode required on parent L1
44	ChallengeWindowBlocks	8	Optimistic window (zero for ZK)
52	InitialRootOff	4	Offset to initial state root
56	SequencerPubKeyOff	4	Offset to sequencer Pulsar group pubkey
60	ConfigOff	4	Offset to rollup-specific config blob

Table 8: `RollupGenesisTx` field layout.

5 Two Rollup Modes and Inherited Cert Mode

5.1 Optimistic vs ZK

Mode	Integrity model	Challenge tx	ZK proof	Withdrawal latency	Use case
Optimistic	Sequencer asserts; anyone may challenge during window	Required	Absent	$\geq$ challenge window ($\sim$ 7 days)	High-throughput general-purpose rollups; gaming; social
ZK	Each batch carries STARK / Groth16 proof of correct execution	Absent	Required	Immediate ($\sim$ one parent L1 block)	Custody-grade rollups; payment finality; cross-rollup composition

Table 9: The two rollup modes share the same envelopes and same P3Q precompile verification path. The difference is the integrity witness.

Both modes share the same envelopes and the same P3Q precompile verification path. The difference is whether the integrity of the state transition is established by:

- (a) sequencer Pulsar-signed attestation plus optimistic-challenge window, OR
- (b) sequencer Pulsar-signed attestation plus STARK / Groth16 proof of correct execution.

The Pulsar-sig + P3Q path is identical; the integrity witness differs.

5.2 Authenticity vs correctness

Sequencer-side Pulsar signing is REQUIRED; ZK proof is OPTIONAL. Every `RollupBatchTx` MUST carry a valid Pulsar threshold signature verified by P3Q. There is no path where a rollup batch skips P3Q verification.

The ZK proof field is optional. If present, Z-Chain verifies it inline (Groth16 verify $\approx$ 200 000 gas, STARK verify $\approx$ 500 000 gas on a generic verifier). If absent, the batch is optimistic and subject to challenge.

Rationale: the Pulsar signature is the **authenticity** witness (the sequencer attests to the batch). The ZK proof is the **correctness** witness (the batch was applied correctly). Authenticity is always required; correctness can be optimistic.

5.3 Inherited cert mode

Each rollup pins one field of the LP-217 `CertPolicy` struct at genesis: `CertPolicy.Mode`. Historically this LP referred to this field as `CertModeFloor`; semantically they are the same – the minimum cert mode the parent L1 must be operating at for the rollup batch to be considered final. The other three `CertPolicy` fields (`Variant`, `TimeoutMs`, `Fallback`) are inherited from the parent L1 by reference; Tier-3 is the only tier where a partial-`CertPolicy` declaration is valid (see LP-204 §"Tier-3 – L3 rollups on Z-Chain").

A rollup cannot ENFORCE a stronger cert mode than its parent L1 operates at. If parent L1 is at PQ-fast, the rollup cannot synthesize PQ-strict finality – it would have to wait for legs the parent L1 never produces. A rollup that requires PQ-strict MUST deploy on a parent L1 operating at PQ-strict or higher.

The rollup MAY pin its floor lower than the parent L1's mode and finalize earlier in the LP-202 tier-degradation timeline. A rollup with `CertModeFloor = PQ-fast` running on a PQ-

CertModeFloor	Mode	Required parent-L1 legs
`0x00`	PQ-off	BLS
`0x01`	PQ-fast	BLS + Pulsar
`0x02`	PQ-strict	BLS + Pulsar + Corona
`0x03`	PQ-heavy	BLS + Corona + Pulsar + Magnetar
`0x10`	strict-PQ-fast	Pulsar
`0x11`	strict-PQ-strict	Pulsar + Corona
`0x12`	strict-PQ-heavy	Corona + Pulsar + Magnetar

Table 10: Per-rollup `CertModeFloor`.

strict parent L1 finalizes at the parent L1's PQ-fast preliminary tier ($\sim$5 ms), not the PQ-strict final tier ($\sim$15 ms).

5.4 Rollups SHOULD NOT run at PQ-off for value-bearing state

The protocol allows `CertModeFloor = PQ-off` (BLS only). This is the right choice for ephemeral or non-financial rollups: gaming session state, social-graph mutations, real-time bidding, telemetry.

For **value-bearing state** – money, securities, identity, custody – the recommended floor is **PQ-strict** (BLS + Pulsar + Corona). This matches the floor recommended by LP-217 for "custody, treasury, regulated securities, store-of-value."

This LP does not introduce a protocol-level requirement that rollups hit PQ-strict. The parent L1's cert mode is the security ceiling; the rollup pins its floor anywhere up to that ceiling. Codifying a "must be PQ-strict for tokens $> \$X$" rule belongs in operator-policy, not in this LP.

5.5 Inheritance from parent L1

Per LP-204 "Tier-3 rollups have NO own validator set," a rollup's security model IS the parent L1's security model.

Property	Source
Block finality cryptographic strength	Parent L1 QuasarCert under its LP-217 cert mode
Sequencer payment gas token	Parent L1 native gas (Z-Chain settlement gas)
Rollup state commit	Embedded in a parent L1 block (Z-Chain block on Lux primary)
Tier degradation behavior	Parent L1 LP-202 tier degradation applies; rollup users see parent L1's degraded tier with no separate degradation
Liveness	Bounded by parent L1 liveness; if Z-Chain stops, rollup stops; if rollup sequencer fails, parent L1 continues, rollup pauses
Slashing	Sequencer slashable on `RollupChallengeTx` (optimistic) or on invalid Pulsar sig; slashing settles on parent L1

Table 11: Properties inherited from the parent L1.

LP	Composition
LP-063	Z-Chain; the parent L1 hosting rollup commits
LP-073	Pulsar threshold signature (the scheme P3Q verifies)
LP-078	EVM precompiles; precompile registry P3Q joins
LP-182	QuasarCert wire format (the cert the rollup state inherits)
LP-200	ZAP Stack umbrella; the no-codec wire guarantee
LP-202	Pipelining and cert tier degradation; the finality timeline the rollup inherits
LP-203	GPU-native verify; where P3Q runs on Blackwell
LP-204	Network of blockchains (Tier-3 rollup definition, "no own validator set")
LP-217	Cert profile modes (per-rollup `CertModeFloor` selection)

Table 12: Composition of LP-218 with sibling LPs.

6 Composition With Sibling LPs

7 Performance Characteristics

Metric	Value	Source
P3Q precompile verify, single sig	∼50 µs on Blackwell	LP-203 Pulsar verify bench
P3Q precompile verify, batched (N=64)	∼5 ms aggregate	LP-203 Pulsar aggregate row
Rollup batch finalization end-to-end (PQ-fast parent)	∼5 ms	PQ-fast preliminary + P3Q verify
Rollup batch finalization end-to-end (PQ-strict parent)	∼15 ms	PQ-strict final + P3Q verify
Sequencer throughput (single node)	100 ktx/s–1 Mtx/s	NIC-bound; not Z-Chain-bound
Aggregate throughput, M rollups on one Z-Chain	10+Mtx/s	M sequencers compose
Gas cost per rollup batch on Z-Chain	∼50 000 gas (P3Q) + ∼21 000 gas base	One P3Q verify per batch

Table 13: Performance characteristics. **Projected** where labeled; measured P3Q verify time pending Blackwell sm_120 production kernel.

Throughput scales horizontally in M (rollups per Z-Chain). Z-Chain's verify capacity is the bottleneck only at $M \gg 100$ – at M=100, Z-Chain verifies 100 P3Q sigs per block (∼5 ms total verify, well within block budget).

8 Cross-Rollup Atomicity

Two rollups deployed on the same Z-Chain instance can commit atomically: both `RollupBatchTx`es land in the same Z-Chain block and share the same QuasarCert. Either both finalize or neither does. This is automatic – no additional protocol is needed.

Cross-L1 atomicity (rollup-on-Lux ↔ rollup-on-Hanzo-L1) requires the bridgevm path defined in LP-204 "Tier-1 ↔ Tier-1 bridge" and is out of scope for this LP. A future LP-219 will specify rollup-to-rollup cross-L1 atomic composition.

9 Activation Marker

```
activates:      2025-12-25T16:20:00-08:00
activates-unix: 1766708400
```

The P3Q precompile is callable at slot `0x012205` from the genesis block of the new final Lux network. Rollup deployments via `RollupGenesisTx` are accepted by Z-Chain from height 0.

10 Backwards Compatibility

None. The pre-Quasar Edition Lux network had no P3Q precompile and is a separate network out of scope.

11 Future Work

- **LP-219** – Rollup-to-rollup atomic composition across L1 boundaries (cross-Tier-3 atomicity via bridgevm).
- **LP-220** – P3Q-Corona: Ring-LWE PQ-proof precompile (sibling of P3Q for Corona-threshold-signed rollups).
- **LP-221** – Dedicated STARK/Groth16 verifier precompile (alternative to the generic EVM verifier for ZK rollups; closes the ∼500 000 gas STARK-verify cost gap).

References

[1] NIST. FIPS 204: Module-lattice-based digital signature standard (ML-DSA), 2024. https://csrc.nist.gov/pubs/fips/204/final.