Quasar Horizon: Hybrid Post-Quantum Finality over DAG-Native Nebula Roots

Zach Kelling

Lux Industries

: A consensus paper composing Pulsar (LP-073), Lens (LP-103), LSS (LP-077), Beam (LP-075), and ML-DSA (LP-070) under Prism (LP-105) over a Nebula DAG

Abstract

We present Lux, a hybrid post-quantum consensus architecture for DAG-native chains. Lux separates fast operational finality from durable post-quantum finality. Validators emit Photons over a Nebula DAG substrate; classical BLS aggregates form Beams for low-latency confirmation, while ML-DSA attestations and Pulsar lattice-threshold Pulses provide post-quantum finality over Nebula roots. Prism binds all certificate lanes to the same transcript, and Quasar reaches Horizon finality when the bound lanes verify.

This paper. Quasar Horizon is the consensus-level composition theorem of the Lux stack. We show how four independent threshold artifacts — a BLS Beam (LP-075 [26]), a per-validator ML-DSA cert set (LP-070 [24]), a Pulsar Pulse (LP-073 [25]), and an optional Lens classical-threshold control-plane certificate (LP-103 [33]) — compose into a single Prism-bound (LP-105 [34]) finality artifact through transcript binding rather than aggregation. We do not rederive the underlying threshold mathematics; the cryptographic kernels are specified in the LP-073 Pulsar paper [25] and the LSS lifecycle paper [38, 27]. We do specify (i) the formal model in which Horizon makes sense over a DAG substrate, (ii) the Prism verification predicate, (iii) the asynchronous PQ-finality discipline that lets Pulse follow Beam without blocking the hot path, (iv) the L1/L2/L3 tier model under co-validation, (v) grouped Pulsar deployment for large validator sets, and (vi) the exact code path that makes the 258-test `protocol/quasar` suite witness the composition. Soundness is by reference to four canonical proof-bucket theorems[1] which carry the per-lane reductions and the composition argument.

Contents

[1] `proofs/quasar/horizon-soundness.tex` (Theorem 1), `proofs/quasar/l1-l2-covalidation.tex` (Theorem 4), `proofs/pulsar/unforgeability.tex` (Theorem **??**), `proofs/quasar-cert-soundness.tex` (Thm. 7.5–7.7).

1 Introduction

Lux is not merely adding post-quantum signatures to a chain; it defines a hybrid finality architecture for DAG-native consensus, with protocol-agnostic threshold lifecycle, post-quantum threshold sealing, and cross-chain propagation of Horizon finality.

This paper specifies the consensus-level composition that makes that thesis precise. The cryptographic kernels — BLS12-381 [3, 1, 26], ML-DSA-65 (FIPS 204) [36, 24], the Pulsar lattice threshold (LP-073) [17, 25], and the Lens curve threshold (LP-103) [33] — are specified elsewhere. The threshold lifecycle that rotates Pulsar and Lens shares across validator generations (LSS, LP-077) [38, 27] is specified elsewhere. What this paper specifies is *Horizon*: the finality boundary that exists when, and only when, every required certificate lane verifies against a single shared transcript bound to the same Nebula DAG root.

1.1 The DAG-native motivation

A linear-chain consensus finalizes blocks. A DAG-native consensus finalizes *frontiers* or *bundle roots*: cuts in the partial order that the local node has decided are committed. Lux's Quasar engine (LP-020 [22]) sits inside the latter family, alongside Narwhal–Tusk [8], Bullshark [39], and Aleph [11]. A NebulaRoot is the canonical commitment to such a cut: a 32-byte digest binding the validator-set hash, the parent epoch, and the bundled vertices.

The post-quantum question for a DAG-native chain is therefore not "how do we sign a block?" but "how do we seal a NebulaRoot under post-quantum hardness, while preserving the operational properties (sub-second finality, leaderless operation, large rotating validator sets) that motivated the DAG choice in the first place?" Horizon is the answer.

1.2 Three lanes, one transcript

A Horizon certificate (Definition **??** from [18]) carries:

- a **BLS Beam** (LP-075) — a 48-byte aggregate signature giving classical fast-path finality;
- an **ML-DSA cert set** (LP-070) — per-validator FIPS-204 attestations carrying PQ accountability, optionally compressed via a Z-Chain Groth16 rollup (LP-307 [29]);
- a **Pulsar Pulse** (LP-073) — one lattice-threshold signature over the same NebulaRoot under the active key era's GroupKey;
- optionally, a **Lens classical-threshold control-plane certificate** (LP-103) — a curve-side threshold signature used for governance and control-plane operations that benefit from the smaller artefact and faster verify of a Schnorr family signature.

The composition rule is *not* "BLS OR Pulsar." Horizon-final is the conjunction over a shared transcript: each lane signs the same canonical `QuasarTranscript` (Definition **??** from `proofs/definitions/transcript-binding.tex`). The verifier — *Prism* (LP-105 [34]) — checks that every lane references the same chain id, network id, epoch, round, validator-set hash, KeyEraID, generation, hash-suite identifier, and NebulaRoot. Failure of any lane, or transcript drift across lanes, fails the composite.

1.3 What is novel

Each piece is well-established in isolation:

- BLS aggregates over independent keypairs are the standard classical fast path [3, 1].
- Two-round MAC-authenticated lattice threshold signing is established [2, 9, 37].
- Per-validator ML-DSA over FIPS-204 [36] is a NIST-standard PQ identity scheme.
- Proactive secret sharing [12] and verifiable resharing [10, 40] are decades old.

What is novel is the *composition into a DAG-native consensus*: the shape of a single finality artefact that binds a fast classical aggregate, a per-validator PQ accountability set, and a

compact PQ threshold seal to one NebulaRoot through one canonical transcript, and ships in a permissionless engine with rotating validator sets, no per-epoch trusted dealer, and a tested no-slashing-dependency rollback ladder. Section 2 states the model. Section 3 states the verifier. Section 4 states the composition theorem (the proof lives in the canonical bucket file). Section 5 documents the asynchronous-finality discipline that allows the Pulse to follow the Beam without blocking the hot path. Section 6 states the L1/L2/L3 model. Section 7 documents the grouped-Pulsar deployment for large validator sets. Section 8 cites the running 258-test code path. Sections 9 and 10 compare to prior consensus protocols and outline open work.

1.4 What this paper does not do

This paper does not redefine or rederive Pulsar, Lens, LSS, or ML-DSA. Where the underlying threshold mathematics is needed, we cite the canonical proof-bucket files (`proofs/pulsar/*`, `proofs/lss/*`, `proofs/quasar/*`) and the LP-073 paper [17]. We are explicit about what is and is not post-quantum: in particular, the Z-Chain Groth16 wrapper around the ML-DSA cert set is a *classical* succinct proof of post-quantum signature verification, useful for compression and EVM verifier compatibility, but **not** a contribution to the post-quantum reduction (Remark 1; see also `proofs/quasar/horizon-soundness.tex`). Horizon's PQ finality lives in the ML-DSA cert set and the Pulsar Pulse, not in any pairing-based wrapper.

2 Formal Model

This section fixes notation. We follow the canonical definitions of the proof bucket; we do not redefine algorithms. The single source of truth for algorithm signatures is `proofs/definitions/algorithms.tex`; the single source of truth for transcript binding is `proofs/definitions/transcript-binding.tex`; the single source of truth for finality state and certificate composition is `proofs/definitions/finality-defini`. We summarize what is needed for the composition argument.

2.1 Validator set and key era

Let V_t be the validator set at chain time t. A **key era** is a maximal interval over which the Pulsar GroupKey $(\mathbf{A}, \tilde{\mathbf{b}})$ and the Lens GroupKey (g, h, X) remain byte-identical. Eras are indexed by a monotone scalar $\eta = \mathsf{KeyEraID}$, incremented only on *Reanchor* (LSS lifecycle, [38, 27]). Within an era, a generation index $g \geq 0$ tracks LSS-managed share rotations; g increments on every Refresh (HJKY97 [12]) or Reshare (Desmedt–Jajodia [10]; verifiable per Wong–Wang–Wing [40]). A rollback marker $r = \mathsf{RollbackFrom}$ records the prior generation reverted from when the chain rolls back; $r = 0$ on forward transitions.

Every validator $i \in V_t$ holds a Pulsar *shard* (one Shamir share over $R_q^{N_{\text{vec}}}$ of the Pulsar master secret $\mathbf{s}$), an independent BLS keypair, an independent ML-DSA-65 keypair, and optionally a Lens shard. The four key materials are intentionally independent: a quantum adversary breaking BLS does not break the ML-DSA, Pulsar, or Lens components.

2.2 Nebula DAG substrate

The chain produces a partial order of vertices over the cycle of epochs and rounds; this is the **Nebula** substrate (LP-020 [22], LP-105 [34]). At each epoch's bundle interval (the Lux mainnet uses a $\sim$ 3-second interval grouping ~ 6 blocks), a deterministic DAG-finalization rule selects a cut and emits a 32-byte $\mathsf{NebulaRoot}_b$:

$$\mathsf{NebulaRoot}_b \;:=\; H_T\big(\mathsf{header}, \mathsf{vertexSet}_b, \mathsf{validatorSetHash}, \eta, g, b\big) \tag{1}$$

where H_T is the canonical TupleHash256 [35]. We write $\mathsf{NebulaRoot}_b$ to mean the canonical bundle root of bundle b; for a non-bundle frontier or epoch-level cut the same notation is used with the appropriate canonical content.

2.3 Adversary and corruption model

The adversary is computationally bounded with possibly quantum-aided lower-bound queries; it adaptively corrupts up to f validators with $f < n/3$ for the standard Byzantine bound, where $n = |V_t|$. The adversary controls the network (delay, drop, reorder) but not the chain's deterministic state-machine acceptance rule. The adversary's PQ access is independent of its classical access: a quantum oracle for elliptic-curve discrete log (which breaks BLS and Groth16) does not provide a Module-LWE oracle (which would break ML-DSA, Pulsar, and Lens to the extent Lens runs on a curve — see below).

2.4 Generation as the canonical lifecycle scalar

The Pulsar share lineage is (η, g, r). A Pulsar Pulse emitted at (η, g) verifies under the GroupKey that is byte-identical for every g within η (Lemma **??**, `proofs/lss/lifecycle-safety.tex`). Therefore: a verifier need only know the active η and the active GroupKey; it does not need to track g for verification *within* an era, only across eras.

This is the technical reason Horizon can compose Beam, ML-DSA cert set, and Pulse: each lane references the same $(\eta, g, \mathsf{HashSuiteID})$ tuple in its transcript, and that tuple is enough to identify which BLS validator-set, which ML-DSA validator-set, and which Pulsar GroupKey to verify against. The transcript binding makes the lanes mutually self-locating without out-of-band registry lookups beyond the chain's own state.

2.5 Generation, validator-set, and tier coupling

Definition 1 (Coupled state at (η, g))**.** *The chain's coupled state at (η, g) is the tuple*

$$\big(V_t, \mathsf{BLSRegistry}_t, \mathsf{MLDSARegistry}_t, \mathsf{PulsarGroupKey}_\eta, \mathsf{LensGroupKey}_\eta, \mathsf{HashSuiteID}\big)$$

recorded on chain at the moment Bootstrap or Reshare for (η, g) activated successfully under `QUASAR-PULSAR-ACTIVATE-v1` *[17]. A Horizon certificate at (η, g) is a Prism-bound composite over this coupled state.*

2.6 What we hold fixed

The following are protocol parameters fixed by the chain configuration:

- Pulsar parameters: $R_q = \mathbb{Z}_q[X]/(X^{256}+1)$, $q =$ `0x1000000004A01`, matrix dimensions $(M, N_{\text{vec}}) = (8, 7)$ (`proofs/definitions/algorithms.tex` Definition `def:pulsar-setup`; mirror in `pulsar/sign/config.go`);
- ML-DSA parameter set: ML-DSA-65 (FIPS 204);
- BLS curve: BLS12-381;
- Hash suite: Pulsar-SHA3 by default (`proofs/definitions/transcript-binding.tex` Definition `def:pulsar-sha3`); Pulsar-BLAKE3 allowed but not normative for production;
- Bundle interval: ~ 3 seconds on Lux mainnet (LP-020 [22]).

The composition theorem (Section 4) is parametric in these choices: any FIPS-equivalent classical aggregate, any FIPS-204-conformant per-validator PQ signature, and any Module-LWE-secure two-round threshold lattice scheme satisfying the LP-073 systems contract works in this slot. We name the production choices for concreteness; the architecture is not tied to them.

3 Prism: Certificate Composition

Prism (LP-105 [34]) is the certificate-composition verifier. It is a verifier, not a key, not a signature, not a kernel. Its single responsibility is to check that every certificate lane in a Horizon certificate refracts from the same Nebula transcript and verifies under the active validator set's keys. This section states the Prism predicate; Section 4 states what predicate acceptance buys.

3.1 The Horizon certificate shape

A Horizon certificate is the canonical composite (Definition ?? in `proofs/definitions/finality-definitions.tex`, equivalently `HorizonCertificate` in the Go canonical [20], `warp/pulsar/HorizonCertificate`):

```
type HorizonCertificate struct {
    Beam        bls.AggregateCertificate    // LP-075
    MLDSA       mldsa.CertificateSet        // LP-070; signer bitmap +
                                            //   sigs (or Z-Chain Groth16
                                            //   rollup of LP-307)
    Pulsar      pulsar.Certificate          // LP-073; one Pulse
    NebulaRoot  ids.ID                      // 32-byte canonical root
    KeyEraID    pulsar.KeyEraID
    GroupID     pulsar.GroupID              // partitioned-set deployments
                                            //   (Section sec:grouped)
    Generation  uint64
    HashSuiteID string                      // e.g., "Pulsar-SHA3"
}
```

The optional Lens classical-threshold control-plane certificate (LP-103 [33]) is carried alongside this shape on chains where governance and control-plane operations benefit from a Schnorr-family threshold; it does not enter the Prism predicate for ordinary Horizon finality but binds the same transcript when present.

3.2 The shared transcript

Each lane signs the same canonical `QuasarTranscript` value. We restate the canonical definition for reference:

```
QuasarTranscript = TupleHash256(
    "QUASAR-TRANSCRIPT-v1",
    NebulaRoot,
    chain_id, network_id,
    epoch, round,
    validator_set_hash,
    KeyEraID, Generation,
    hash_suite_id
)
```

This is Definition ?? of `proofs/definitions/transcript-binding.tex`, repeated here only because it is load-bearing for the Prism predicate. The personalization tag is canonical and *not* re-used across protocols; in particular, the Warp 2.0 envelope's Pulse uses `WARP-PULSAR-ENVELOPE-v1` (LP-021v2 [23, 19]), and the activation cert uses `QUASAR-PULSAR-ACTIVATE-v1` ([17]); a Pulse over a NebulaRoot cannot be replayed as a Pulse over a Warp envelope or an activation cert.

3.3 The Prism predicate

Definition 2 (Prism). $\mathsf{Prism}(\mathsf{cert}) \to \{0,1\}$ *accepts a Horizon certificate iff all six conditions hold (this is Definition* **??** *of* `proofs/definitions/finality-definitions.tex` *restated):*

1. ***Domain separation and transcript binding.*** $\mathsf{HashSuiteID}$, $\mathsf{KeyEraID}$, $\mathsf{Generation}$, *and* $\mathsf{NebulaRoot}$ *in the certificate match the values that each lane signed. Compute* $\tau := \mathsf{QuasarTranscript}(\mathsf{cert})$*; this* τ *is the message verified by every lane below.*
2. ***Beam.*** *BLS aggregate* $\mathsf{cert.Beam}$ *verifies under the BLS public-key registry of the validator set indicated by the certificate's* $\mathsf{validator\_set\_hash}$*, on message* τ*, signer set indicated by the aggregate's signer bitmap with weight* $\geq 2/3$*.*
3. ***ML-DSA cert set.*** *For every signer in the cert-set's bitmap, the per-validator FIPS-204 ML-DSA-65 signature verifies on* τ *under the validator's registered ML-DSA public key (context string* `lux-quasar-horizon-v1`*); OR the optional Z-Chain Groth16 rollup proof verifies the per-validator signature batch against a public-inputs hash that commits to* τ *(LP-307 [29]). Threshold* $\geq 2/3$*.*
4. ***Pulse.*** $\mathsf{Pulsar.Verify}(\mathsf{GroupKey}_\eta, \tau, \mathsf{cert.Pulsar}) = 1$ *where* $\mathsf{GroupKey}_\eta$ *is the active key-era GroupKey that the chain has on record at* (η, g) *(Definition 1).*
5. ***Validator-set binding.*** *The validator-set hash, BLS registry digest, and ML-DSA registry digest in* cert *match the chain's epoch state.*
6. ***Nebula linkage.*** $\mathsf{NebulaRoot}$ *is the canonical bundle root for the indicated* $(\mathsf{epoch}, \mathsf{round})$ *(Section 2). Reorg / fork cases are rejected at this step.*

A bundle root is Horizon-final *iff* $\mathsf{Prism}(\mathsf{cert}) = 1$ *for some certificate carrying it.*

3.4 Why "no Beam-OR-Pulse" is essential

A common informal reading is "Horizon = (Beam ∨ Pulse)." That reading is wrong, and removing it is one of this paper's contributions.

If Horizon were $\mathsf{Beam} \vee \mathsf{Pulse}$**.** Under a quantum adversary that breaks BLS in polynomial time, the adversary can produce a valid Beam over any chosen (τ, τ') pair, including a τ' that disagrees with the actual chain history. If acceptance only required "Beam OR Pulse," a quantum-induced Beam forgery would single-handedly produce a Horizon-final certificate over τ', and any honest Pulse-sealed certificate over τ would also be Horizon-final — giving us conflicting Horizon-finals, i.e. a safety break. The disjunctive reading is non-PQ-safe.

Horizon $= \mathsf{Beam} \wedge \mathsf{Pulse} \wedge \mathsf{MLDSA}$ **over a shared** τ**.** Under the conjunctive Prism predicate, a quantum adversary that breaks only BLS still has to produce a valid Pulse over the conflicting τ'. By Pulsar SUF-CMA (Theorem **??**, `proofs/pulsar/unforgeability.tex`), this requires breaking Module-LWE / Module-SIS, not just discrete log. The conjunction is what makes the composite PQ-sound (Theorem 1, Section 4).

The Beam-only and Pulse-only intermediate states are still useful. They are public observable states (`Beam-final`, `Pulse-sealed`) in the canonical finality state lattice (Definition **??** of `proofs/definitions/finality-definitions.tex`); the chain's API can return them as intermediate confirmations, and clients with classical-only threat models can choose to act on Beam-final alone. But `Horizon-final` is the strongest state and is the conjunctive one. Specifications that pretend the disjunctive reading is acceptable fail the post-quantum soundness check (`proofs/pq-finality-no-bls.tex`; [16]).

3.5 Where Lens fits

Lens (LP-103 [33]) is a curve-side threshold kernel sharing the LSS lifecycle with Pulsar (LP-077 [27]). For the consensus-finality predicate, a Lens certificate is *not* a substitute for a Pulse: Lens is classical (curve discrete log) and adds no PQ reduction. Lens earns its place in two production roles:

- **Control-plane / governance.** Validator-set updates, parameter-changes, fee-config votes can take a Lens threshold cert under a curve-side group key for compactness and cheap on-chain verification. Such certs ride the same LSS lifecycle (KeyEraID / Generation / RollbackFrom) as Pulsar, sharing the no-trusted-dealer-after-genesis story.
- **Cross-rail compatibility.** Where downstream verifiers (EVM precompiles, light clients on classical-only chains) cannot afford a Pulse verify, a Lens cert is the natural classical companion — compact, fast, and orthogonal to the BLS Beam.

Lens certs do *not* contribute to the Horizon PQ reduction. They live in the Lux stack as an option, not a requirement; a chain may run Horizon with no Lens cert at all.

3.6 Prism as Go code

The shipping verifier is in `github.com/luxfi/consensus` [20], file `protocol/quasar/horizon.go`. The verifier returns explicit per-lane error codes (`ErrBeamFailed`, `ErrMLDSAFailed`, `ErrPulseFailed`, `ErrTranscriptDrift`, `ErrValidatorSetMismatch`) so callers can distinguish "Beam OK, Pulse not yet arrived" (in which case the bundle is `Beam-final` but not `Horizon-final`) from "Pulse rejected, retransmit needed." This explicit per-lane error surface is the implementation expression of the conjunctive predicate; nothing in the shipping code lets a single lane carry the bundle to Horizon-final on its own.

4 Horizon Soundness

The composition theorem is `proofs/quasar/horizon-soundness.tex`. We restate it here for reference and connect it to the per-lane reductions.

4.1 The composition theorem

Theorem 1 (Horizon-final implies all lanes verify the same transcript)**.** *Let* cert *be a Horizon certificate (Definition* **??** *in the canonical bucket) with* $\mathsf{Prism}(\mathsf{cert}) = 1$*. Then the underlying validator set of size* n *with* $f < n/3$ *corruptions cannot produce two Horizon-final certificates on distinct* $\mathsf{NebulaRoot}$ *values at the same* $(\mathsf{epoch}, \mathsf{round})$*, except with negligible probability under:*

1. *BLS12-381 co-CDH (for the Beam lane);*
2. *Module-LWE / Module-SIS (for the Pulsar lane, Theorem* **??***);*
3. *EUF-CMA security of ML-DSA-65 / FIPS 204 (for the ML-DSA cert-set lane).*

The proof is by independent reduction: any pair of Prism-accepted certs on conflicting $\mathsf{NebulaRoot}$*s at the same* $(\mathsf{epoch}, \mathsf{round})$ *forces a forgery in at least one lane. The lanes are independent (different hardness assumptions); a quantum adversary breaking BLS does not break the Pulsar or ML-DSA components, and vice versa.*

The full proof, including the formal definitions of the three reductions, appears as Theorem 1 of `proofs/quasar/horizon-soundness.tex` and is companion-cited from `proofs/quasar-cert-soundness.tex` ([18]) Theorem 7.5 (classical soundness), Theorem 7.6 (parallel-lane liveness), and Theorem 7.7 (PQ safety).

4.2 Per-lane reductions

Beam (BLS aggregate). Forging a BLS aggregate requires an adversary who controls more than $2n/3$ keys (since the aggregate weighs the signer set), or a forgery in the underlying BLS12-381 signature scheme. The latter reduces to co-CDH ([1]) over BLS12-381. Under quantum advice, BLS is broken in polynomial time by Shor; under classical attack, the standard $> 2^{120}$ security applies. This is why Beam alone is *not* the PQ root of trust.

Pulse (Pulsar SUF-CMA). Forging a Pulse with at most $t-1$ corrupted parties reduces to MLWE / MSIS, by Theorem **??** of `proofs/pulsar/unforgeability.tex` (cf. [17] Theorem 5.2). Concretely, with the LP-073 parameters [17], classical security $\geq 2^{142}$ and quantum security $\geq 2^{130}$ via BDGL sieving plus Grover speedup (`quasar-cert-soundness.tex` App. E). The Lean mechanization is `Crypto.Pulsar.corona_pq_unforgeability` [32].

ML-DSA cert set (FIPS 204 EUF-CMA). Each per-validator signature reduces to MLWE/MSIS by FIPS 204's standard analysis. For the cert-set construction (a bitmap plus per-validator signatures, each over τ with the canonical context string `lux-quasar-horizon-v1`), forging a $\geq 2/3$-weight cert set requires forging $\geq 2/3$-weight worth of validator-specific ML-DSA signatures. Each forgery individually is hard under MLWE/MSIS; the bitmap construction adds no structural shortcut.

4.3 Cross-lane independence

Lemma 2 (Lane independence under quantum advice)**.** *Let $\mathcal{A}$ be a polynomial-time adversary with quantum oracle access to BLS12-381's discrete log. Then $\mathcal{A}$'s success probability against the Pulsar lane is bounded by* $\mathrm{Adv}^{\mathsf{MLWE}}_{q,\sigma_E}(\mathcal{A}) + \mathrm{Adv}^{\mathsf{MSIS}}_{q,[\mathbf{A}|-c\tilde{\mathbf{b}}],2B}(\mathcal{A}) + \mathrm{negl}(\lambda)$. *The DL oracle gives $\mathcal{A}$ no MLWE / MSIS-relevant computational power, since neither problem reduces to elliptic-curve DL. Symmetrically, an MLWE-oracle-equipped $\mathcal{A}$ enjoys no advantage against BLS.*

This is the formal statement of "BLS contributes no PQ security to a multi-lane cert," which is the central observation of `proofs/pq-finality-no-bls.tex` ([16]). The implication for Quasar Horizon is: the Pulse lane is the post-quantum root of trust; the Beam lane provides classical fast-path performance; the ML-DSA cert set provides per-validator PQ accountability; the optional Groth16 wrapper around the ML-DSA cert set is a classical compression artifact, not a PQ contribution.

4.4 The Groth16 wrapper

Remark 1 (Groth16 wrapper does not contribute PQ security)**.** *A Z-Chain Groth16 rollup proof of ML-DSA cert-set verification (LP-307 [29], Definition* **??** *of* `proofs/definitions/finality-definitions.t` *is a* classical succinct proof of post-quantum signature verification. *Pairing-based SNARKs over BLS12-381 break under Shor's algorithm. Horizon's PQ soundness (Theorem 1 parts 2 and 3) flows from the ML-DSA signatures and Pulsar threshold signature alone. The Groth16 proof is verified for compatibility / compression only and is not part of the PQ reduction. See Remark* **??** *of* `proofs/quasar/horizon-soundness.tex` *and Remark* **??** *of* `proofs/definitions/finality-definitions.`

The Lux specification corpus refuses to call the Groth16 wrapper "the PQ proof lane." It is a wrapper, not a root of trust. Future PQ-friendly proof systems (P3Q [13] or lattice-based) replace Groth16 in the wrapper role; PQ finality continues to flow from ML-DSA + Pulsar.

4.5 Liveness

Theorem 3 (Parallel-lane liveness)**.** *If V_t has $\geq 2n/3$ honest validators, then in any interval of length at least T (a chain-configured upper bound on per-lane signing latency, currently $T = 3\,\mathrm{s}$ on*

Lux mainnet), all three lanes produce signatures, and Prism accepts. The lanes are independent, so liveness in each is independent: any single-lane stall (e.g., a Pulse coordinator timeout) does not prevent a future round's Beam from being formed and re-anchored to the new τ.

This is Theorem 7.6 of [18]. The implementation uses an asynchronous-finality discipline (Section 5) so that Pulses follow Beams without forcing the Beam path to wait. The chain's safety does not require synchrony; parallel-lane liveness is a partial-synchrony property.

4.6 Why we do not assume synchrony

We do not require any synchronous round. The Prism predicate is asynchronous-safe: a verifier accepts a Horizon-final certificate when all three lanes' signatures arrive over τ, irrespective of intra-round timing. Asynchronous PQ finality (Section 5) is the operational discipline that uses this fact. A non-asynchronous design that required all three lanes to land in the same wall-clock window would fail liveness on a WAN-distributed validator set under reasonable network failure models.

4.7 Mechanization status

- **Pulsar SUF-CMA.** Lean: `Crypto.Pulsar.corona_pq_unforgeability` [32].
- **Reshare preserves master secret.** Lean: `Crypto.Threshold.Reshare.reshare_preserves_public_ke` `Crypto.Threshold.Reshare.reshare_same_secret`.
- **Triple consensus.** Lean: `Consensus/Quasar.lean` carries `triple_requires_all`, `quantum_safe_triple`, `independent_pq_from_classical`, `cert_intersection`, `unique_per_round`.
- **BFT safety / liveness / finality.** Lean: `Consensus/{Safety,Liveness,Finality,BFT}.lean`.
- **TLA+.** `tla/Quasar.tla` model-checked (`tla/MC_Quasar.cfg`).
- **Tamarin.** `tamarin/QuasarConsensus.spthy` verified against the canonical handshake / vote / bundle protocol.

The paper-level theorems (Theorem 1, Theorem **??**, Lemma 2) are companion-cited from these mechanizations; the canonical theorem names cited above are verbatim from the proof tree.

5 Asynchronous PQ Finality

A naive composition would require Beam, ML-DSA cert set, and Pulse to all land in the same hot-path round before the bundle is finalized. That design fails liveness on a permissionless WAN-distributed validator set: the lattice two-round signing is structurally heavier than BLS aggregation, and forcing it into the hot path pushes consensus latency from $\sim$ 1s to multiple seconds.

The shipping discipline is different: **Beam fast, Pulse-sealed, Horizon-final on a delay**. The bundle is observable in the chain's API at three distinct states; clients consume the state appropriate to their threat model.

5.1 The state lattice

We restate Definition **??** of `proofs/definitions/finality-definitions.tex`:

$$\mathsf{Observed} \subset \mathsf{Beam\text{-}final} \subset \mathsf{PQ\text{-}attested} \subset \mathsf{Pulse\text{-}sealed} \subset \mathsf{Horizon\text{-}final} \tag{2}$$

where each containment is strict and each transition requires the additional certificate specified in Definition **??**.

Observed. Photons (validator votes / attestations) on the bundle have been seen through the Lumen transport. No finality.

Beam-final. BLS aggregate over $\geq 2/3$-weight signers has formed and verified. Sub-second on Lux mainnet (target $\sim$ 250–500 ms from first photon). This is the state that a classical-only client (e.g., an EVM dApp on an L1 with classical-only threat model) acts on.

PQ-attested. The per-validator ML-DSA cert set has reached $\geq 2/3$ weight. Adds per-validator PQ accountability without yet adding the compact PQ threshold seal. Useful as evidence material for slashing / forensics if a Pulse is later disputed.

Pulse-sealed. A Pulsar Pulse over the same NebulaRoot has been formed and verified under the active GroupKey. This is the strongest single-lane PQ confirmation.

Horizon-final. Prism accepts the composite. This is the strongest finality state, and is the state that bridges, cross-chain settlement, and downstream chains depend on.

5.2 Bundle interval

A Quasar bundle aggregates several Beam-finalized blocks. The Lux mainnet bundle interval is $\sim$ 3 s, grouping $\sim$ 6 blocks. The bundle root signed by the Pulse is the canonical NebulaRoot for that interval.

Why bundles, not blocks. Two-round MAC-authenticated lattice threshold signing has a non-trivial online phase, even with Round 1 preprocessing. At $K = 21$ active validators per group, the median end-to-end signing time on the Go canonical is $\sim$ 250 ms (Section 8). At one Pulse per block (sub-second blocks), the signing phase would be the bottleneck. At one Pulse per bundle ($\sim$ 3 s), the online phase comfortably fits within the bundle interval with order-of-magnitude headroom; preprocessing is amortized.

Implementation: `protocol/quasar/quasar.go`. The shipping engine triggers Pulse signing on the chain's bundle commit event, not on individual block commits. Sign1 is preprocessed off the hot path. Sign2 + Combine run as the bundle commit fires. The 258-test `protocol/quasar` suite ([20], `quasar_bench_test.go`) exercises this path.

5.3 The relationship between Beam latency and Pulse latency

- **Beam latency.** Dominated by BLS aggregate signing + WAN 2/3-weight collection. On Lux mainnet, $\sim$ 250 ms at $n = 21$.
- **Pulse latency.** Dominated by Pulsar Round 1 (offline-preprocessable) and Round 2 (online, message-dependent). At $K = 21$ on the Go canonical: Round 1 $\sim$ 185 ms per party, Round 2 + Combine $\sim$ 30 ms, plus WAN coordination. Total $\sim$ 1–2 s end-to-end (cf. Section 8; full numbers in the benchmark report [15]).
- **Horizon latency.** Pulse latency dominates. On Lux mainnet, Horizon-final lags Beam-final by $\sim$ 1–2 s.

This delay is intentional. A client that needs sub-second finality acts on `Beam-final`, accepting the classical-only threat model. A client that needs PQ finality (e.g., a long-archival contract or a cross-chain bridge to a high-value chain) waits for `Horizon-final`. The delay is not a bug; it is the price of delivering compact PQ threshold finality on a permissionless WAN validator set.

5.4 Bundle re-anchoring under failed Pulse

If the Pulse coordinator times out or the threshold quorum cannot be assembled, the bundle stays at `PQ-attested` (or `Beam-final`) and the next bundle's NebulaRoot retroactively commits

to the previous bundle's NebulaRoot via the chain's standard parent linkage. The next round's Pulse therefore covers *both* bundles via transitive transcript inclusion. We do not require a successful per-bundle Pulse; we require successful Pulses on a sufficient density of bundles to bound how long a NebulaRoot can wait for Horizon-finality.

The chain's governance configures the "Pulse-sealing density bound" — e.g., "Pulse must be observed within K bundle intervals." Failure to meet the bound triggers an alert; persistent failure triggers the LSS rollback ladder (`proofs/lss/rollback-safety.tex`; [38, 27]) for the failing key era.

5.5 Why this is the right design

Three independent design pressures point to the asynchronous discipline:

1. **Validator WAN topology.** Lux mainnet validators run in 5 regions across 3 continents. Cross-WAN p99 latency is ~ 200 ms. A single round of two-round threshold signing requires at least 2 WAN crossings; at 21 validators, the worst-case round-trip dominates and a per-block PQ Pulse would be the bottleneck.
2. **Heterogeneous client threat models.** Classical EVM dApps with low-value short-time-horizon transactions can act on Beam-final; high-value cross-chain bridges with long-time-horizon archival need Horizon-final. One state lattice serves both.
3. **Future-proofing for quantum.** If a quantum capability appears, classical-only clients can switch from Beam-final to Pulse-sealed or Horizon-final without a chain-level upgrade. The state lattice is forward-compatible with shifts in client risk tolerance.

This is the consensus-level analogue of LP-105 [34] §"Quasar finality state machine": finality is not a single bit. It is a lattice with public, observable, well-defined intermediate states.

6 L1 / L2 / L3 Tier Model

This section specifies the Lux network-tier model. The L1 / L2 / L3 distinction here is *not* the rollup-vs-settlement distinction of other ecosystems; it is a co-validation model rooted in Pulsar's KeyEraID lineage and validator-set inclusion.

6.1 Tier definitions

Definition 3 (Lux network tiers)**.** *A Lux-architecture chain occupies exactly one of three tiers (cf. Definition* **??** *in* `proofs/definitions/finality-definitions.tex`*, restated here):*

L1 (sovereign). *No* primary_network_id *declared. Own validator set, own KeyEraID lineage, own Beam, own ML-DSA cert set, own Pulse. Whether other chains anchor to this L1 is a graph property, not a protocol distinction.*

L2 (co-validating). *Declares* primary_network_id $= N_P$*. Activation cert binds* primary_nebula_root *from N_P; validator set is required to be a subset of N_P's active set at the era's Bootstrap or Reanchor. Not a rollup — L2 produces its own Beam, ML-DSA cert set, and Pulse over its own NebulaRoot.*

L3 (rollup / validity / app-execution). *Settles or anchors to an L1 or L2 via a succinct proof (Groth16, STARK), an FHE circuit, or a fraud / validity proof. The proof system is a compatibility / compression / privacy adapter, not the PQ root of trust. PQ finality of an L3 inherits from the underlying L1/L2's Pulse + ML-DSA, never from a pairing-based SNARK alone.*

A chain may transition between tiers via a Reanchor-class governance event.

6.2 L1: the default

L1 is the baseline tier. Every Lux-architecture network is implicitly L1; no opt-in is required. There is no "L0" or "settlement layer" below L1. Two distinct deployment shapes share the L1 protocol:

- **Primary network.** A Lux-architecture network that other chains may opt to co-validate with (Lux mainnet itself is the canonical example).
- **Independent L1.** A chain that runs entirely on its own validator set, never co-validates with any primary, and is never declared as a primary by other chains. Operationally identical to a primary network with no downstream L2s.

The two shapes are protocol-equivalent. A chain becomes a "primary" simply by being declared as $\mathsf{primary\_network\_id}$ by some L2; it is a graph property of who anchors to whom.

6.3 L2: co-validation, not subordination

A chain becomes L2 of a primary network N_P by declaring $\mathsf{primary\_network\_id} = N_P$ and committing, at the consensus layer, to co-validation:

```
L2 chain_id            = X
L2 network_id          = N_X
L2 primary_network_id  = N_P    (declared)
L2 validator set       subseteq N_P validator set
```

Co-validation is enforced by binding N_P's NebulaRoot into the L2's activation transcript and bundle Pulses. The L2 activation message extends the canonical `QUASAR-PULSAR-ACTIVATE-v1` prefix with two fields:

```
L2 activation message:
    QUASAR-PULSAR-ACTIVATE-v1 ||
        chain_id              = X
        network_id            = N_X
        primary_network_id    = N_P
        primary_nebula_root   = NebulaRoot_P
        key_era_id            = ...
        old_generation, new_generation, ...
        nebula_root           = NebulaRoot_X
        ...
```

The L2's validators must be members of N_P's validator set (or a configurable subset thereof). Their Pulsar shares for the L2 derive from the same Bootstrap Dealer / Signature Coordinator roles that validate N_P ([38, 27]), so a Prism accepting an L2 Horizon-final implicitly verifies the validators were active on N_P at the bound primary epoch.

6.4 The L2 co-validation soundness theorem

The L2 inheritance theorem is `proofs/quasar/l1-l2-covalidation.tex`. We restate it:

Theorem 4 (L2 co-validation security inheritance)**.** *Let* Chain_X *be an L2 with* $\mathsf{primary\_network\_id} = N_P$*, and let* $V_X \subseteq V_{N_P}^{(\eta_P, g_P)}$ *at the moment of* Chain_X*'s key-era Bootstrap. If a Horizon-final certificate on* Chain_X *binds* $\mathsf{primary\_nebula\_root} = R_P$ *(a NebulaRoot of* N_P *at era* η_P *generation* g_P*), then any adversary forging conflicting finality on* Chain_X *must:*

1. *Compromise enough of* V_X *to forge an L2-local Pulse, AND*
2. *Either compromise enough of* V_{N_P} *to forge an alternative* R_P *(whence the security reduction inherits Theorem 1 applied to* N_P*), OR*

3. Break the transcript binding (collision on TupleHash256 [35]).
Therefore the L2's safety bound is at least the minimum of N_P's safety bound and the L2's own validator-set safety bound.

This is co-validation security, not rollup security. There is no fraud-proof window, no validity-proof anchoring; the inheritance is purely via shared validator-set membership and transcript binding ([17] Section 11; `proofs/quasar/l1-l2-covalidation.tex`).

6.5 L3: the proof-system tier

Remark 2 (L3 is a different layer model)**.** Chain_X *at L3 inherits security via a succinct proof of state transitions, not via co-validation membership. L3 chains may use Groth16 or STARKs for compression. PQ finality of an L3 inherits from the underlying L1/L2's Pulse + ML-DSA; the proof system itself is not the PQ root of trust (Remark 1). A Z-Chain instance, for example, can be L1 (sovereign validator set), L2 (co-validating with another primary), or L3 (privacy/validity execution layer anchored to an L1/L2). The tier is a deployment role, not a property of the proof system in use.*

The mapping:

Tier	**Anchors via**	**PQ root of trust**
L1	own validator set	own Pulse + ML-DSA
L2	co-validation with primary N_P	primary's Pulse + ML-DSA, transitively (Theorem 4)
L3	succinct proof / fraud proof / FHE	underlying L1/L2's Pulse + ML-DSA (the proof system itself does not provide PQ security)

6.6 Tier transitions

Tier changes are governance events gated by Reanchor:

- L1 → L2: declare a $\mathsf{primary\_network\_id}$, bind to that primary's NebulaRoot at the next Reanchor; this creates a fresh KeyEraID under the primary's validator set.
- L2 → L1: drop the $\mathsf{primary\_network\_id}$ declaration; future eras run on the chain's own validator set.
- L1/L2 → L3: declare a settlement target and proof system; future Horizon-finality on the L3 inherits from the underlying L1/L2.
- L3 → L1/L2: graduate to running its own validator set.

Every transition is gated by the same Reanchor mechanism that handles fresh KeyEraID creation; the underlying KeyEraID stops at the old tier and a fresh KeyEraID starts under the new tier.

6.7 Multiple primaries

A chain may declare any one Lux-architecture network as its primary (Lux mainnet, Hanzo, Zoo, or another L1). The choice is deployment configuration, not protocol-level constraint. If the primary network undergoes a Reanchor (fresh GroupKey, new validator authority), every L2 anchored to it must either Reanchor to follow, Reanchor to a different primary, or drop to L1. This is by design: the security inheritance is explicit, not silent.

6.8 What L2 buys (and does not buy)

L2 buys. Shared validator security (compromising the L2 requires compromising enough of N_P's validators). Cross-chain Prism binding (a Horizon-final L2 cert proves the L2 state was

sealed at a specific N_P primary state). Operational economy (no separate validator-acquisition path).

L2 does not buy. Free liveness: an L2 still needs $\geq 2/3$-weight of its own validators to be live. Free safety on N_P: if N_P itself is compromised, the L2's primary-bound transcripts inherit that compromise. Free upgrade path: L2 has its own KeyEraID lineage, and Reanchors must be coordinated explicitly with N_P's lifecycle.

This is co-validation, not custody. The L2 is a sovereign chain whose validator-set authority happens to overlap with N_P's, by explicit declaration, by Reanchor-class governance, with cryptographic transcript binding to the primary's NebulaRoot. It is the consensus-level analog of multi-sig delegation, not of upgrade-proxy admin.

7 Grouped Pulsar for Large Validator Sets

A single global threshold key over thousands of validators is operationally brittle: every rotation requires every validator to participate in JVSS broadcasts, Pedersen-commit verification, and activation. Pulsar supports *grouped* certificates as the deployment shape for large validator sets. This section composes grouped Pulsar with Prism, so that a Horizon certificate over a large validator set carries one Pulse per group rather than one Pulse over all signers.

7.1 Group construction

Validators are partitioned into G groups of (approximate) size $k = N/G$. Each group has its own GroupKey lineage, its own KeyEraID, its own resharing schedule. A Pulsar bundle certificate at the consensus layer is a quorum of group certificates:

$$\mathsf{HorizonCert.Pulsar} \;=\; \big\{(\mathsf{group}_j, \sigma_{\mathrm{Pulsar},j}) \;:\; j \in \mathsf{quorum}(G)\big\} \tag{3}$$

with $|\mathsf{quorum}(G)| \geq \lfloor 2G/3 \rfloor + 1$ for the standard Byzantine bound, or a tighter quorum if individual groups are known to be more reliable. Each group's $\sigma_{\mathrm{Pulsar},j}$ is a per-group threshold signature signed under group_j's GroupKey.

7.2 What grouping buys

Local resharing. A group rotates its shares only when its own validators rotate; cross-group churn does not force every group to reshare. JVSS bandwidth scales as $\mathcal{O}(k^2)$ per rotation rather than $\mathcal{O}(N^2)$.

Independent failure domains. A failed activation in group j does not block other groups. The chain emits a Pulsar quorum-of-groups certificate from the surviving groups; group j rolls back independently via the LSS rollback ladder ([38, 27]).

Compact bundle pulses. Each group emits one Pulsar pulse (size on the order of the canonical Pulsar signature). G groups produce G pulses; aggregating across groups via Merkle is straightforward. Total bundle-cert overhead is $\mathcal{O}(G)$, not $\mathcal{O}(N)$.

7.3 Composition with Prism

The Prism predicate (Definition 2) extends to grouped Pulsar by replacing the single Pulse verify step with a quorum-of-groups verify:

Step 4 (revised, grouped form). For each group_j in the certificate's group quorum, $\mathsf{Pulsar.Verify}(\mathsf{GroupKey}_{\mathsf{gr}}$ 1, where $\mathsf{GroupKey}_{\mathsf{group}_j,\eta_j}$ is the group's active key-era GroupKey at the chain's bound state for (η_j, g_j). The number of accepting groups must be $\geq \lfloor 2G/3 \rfloor + 1$, and each accepting group's (η_j, g_j) must match the chain's epoch state for that group.

Steps 1, 2, 3, 5, 6 are unchanged. The transcript τ is unchanged: every group signs the same Nebula bundle root.

7.4 Soundness in the grouped form

Theorem 5 (Grouped Horizon soundness)**.** *Let* cert *be a grouped Horizon certificate with* $\mathsf{Prism}(\mathsf{cert}) = 1$ *in the form just specified. Suppose validators are assigned to groups by a process* $\mathcal{G}$ *with the property that, for any subset* S *of corrupted validators with* $|S| \leq f$ *and group size* k*, the probability that* $\mathcal{G}$ *assigns* $\geq t_{\text{group}}$ *corrupted validators to any single group is bounded by a function* $p_{\text{group}}(f, n, k, t_{\text{group}})$*. Then forging conflicting Horizon-final certificates requires:*

1. *Forging* $\geq \lfloor 2G/3 \rfloor + 1$ *group Pulses: each requires either Pulsar SUF-CMA (Theorem* **??***) on that group, or a corruption density that exceeds the group threshold (probability bounded by* p_{group}*); AND*
2. *Forging the Beam aggregate: requires BLS forgery or* $> 2/3$*-weight corruption; AND*
3. *Forging the ML-DSA cert set: requires* $> 2/3$*-weight ML-DSA forgery.*

The argument is by per-group reduction: the conjunctive Prism predicate reduces a forgery to a forgery in some group; the per-group SUF-CMA reduction completes the lattice argument. Cross-group independence holds because each group has its own GroupKey and its own resharing schedule; a corruption-quorum in one group does not propagate to another.

7.5 Open systems-research questions

Grouped Pulsar is where dynamic-PQ-threshold consensus becomes a research contribution rather than an engineering exercise. The open questions, in priority order:

1. Given validator corruption fraction f, and group assignment process $\mathcal{G}$, what is the closed form of $p_{\text{group}}(f, n, k, t_{\text{group}})$? Hypergeometric vs binomial; balanced vs adversarial assignment.
2. How should group assignment respond to validator churn while preserving in-group share integrity?
3. How does group resharing interact with epoch transitions? Per-group epochs vs chain-global epochs; leader staleness.
4. How many group certificates are required for a Horizon-final boundary at a given safety target (2^{128}, 2^{142}, etc.)?
5. What is the bandwidth/latency curve of Pulsar pulses as a function of k and G, under wide-area churn? (Section 8 cites measured numbers from the benchmark report [15]; the open work is the closed-form trade-off curve.)

These questions connect cryptographic threshold assumptions to consensus-level quorum assumptions, and are the natural follow-on formal-systems paper to this one.

7.6 Comparison to unscaled threshold schemes

Threshold Raccoon [9]. Reports practical thresholds up to 1024 signers with $\sim$ 13 KiB signatures and $\sim$ 40 KiB communication per user. Targets the cryptographic-primitive layer.

Hermine [37]. DKG, identifiable aborts, and proactive refresh in a single package. Targets the cryptographic-primitive layer, with broader lifecycle coverage than Threshold Raccoon.

Grouped Pulsar. Composes the underlying lattice threshold scheme into a deployment shape that scales across permissionless validator sets where committee membership is not fixed at protocol design time. The contribution is the system-level composition.

The three approaches are complementary, not substitutes: Hermine's primitive could in principle be the kernel that grouped Pulsar deploys (Hermine's lattice math + LSS lifecycle + grouped Horizon composition). The Lux production stack uses Pulsar's Corona-derived primitive, but the composition argument is kernel-agnostic.

7.7 Default deployment

- **Lux mainnet ($n = 21$).** $G = 1$ (no grouping needed). One global Pulsar key era; Pulse latency dominated by the WAN $K = 21$ signing path.
- **Mid-scale ($n \sim 100$).** $G = 5$, $k = 20$. Quorum-of-groups ≥ 4.
- **Large-scale ($n \sim 1000$).** $G = 50$, $k = 20$. Quorum-of-groups ≥ 34.

The Go canonical ([14]) parameterizes the group construction; the consensus engine ([20], `protocol/quasar/grouped_threshold.go`) consumes group-level signatures and assembles the quorum-of-groups Horizon Pulse. The 258-test `protocol/quasar` suite includes `grouped_threshold_test.go` exercising the assembly path; production $G > 1$ deployments are gated on the open-research questions listed above.

8 Implementation

The shipping consensus implementation is in `github.com/luxfi/consensus` [20]. This section locates the Horizon composition in that code base and points at the test surface that witnesses the composition argument.

8.1 Repository layout

Path	Role
`protocol/photon`	K-of-N committee selection, luminance tracking
`protocol/wave`	Threshold voting + FPC
`protocol/focus`	Beta consecutive-successes counter
`protocol/prism`	DAG geometry: cuts, frontiers, uniform sampling
`protocol/horizon`	DAG reachability, LCA, transitive closure, skip lists
`protocol/flare`	DAG cert/skip via 2f+1 quorum
`protocol/ray`	Linear chain finality driver
`protocol/field`	DAG finality driver with safe-prefix commit
`protocol/nova`	Linear chain mode (wraps `ray`)
`protocol/nebula`	DAG mode (wraps `field`)
`protocol/quasar`	BLS Beam + ML-DSA cert set + Pulsar Pulse composition

The Horizon composition lives in `protocol/quasar/horizon.go` (Prism predicate) and `protocol/quasar/qu` (lifecycle: bundle interval, Beam-final, Pulse-sealed, Horizon-final state transitions). The Pulse signing path is wired through `epoch.go` and `reshare_epoch.go` to the LSS-Pulsar adapter at `threshold/protocols/lss/lss_pulsar.go` [30], which delegates to the Pulsar kernel at `github.com/luxfi/pulsar` [14].

8.2 The Quasar engine flow

- Photons (validator votes) arrive over the Lumen transport (forthcoming Lumen LP). The transport is ZAP [22] (zero-copy wire protocol; not p2p / gRPC).

- Wave threshold-voting runs per-vertex on the DAG frontier (`protocol/wave`). Vertices that gather $\geq 2/3$-weight Wave votes become Wave-final.
- Focus tracks consecutive-successes counters for each vertex; once Focus's β threshold is met, the vertex is Focus-final (`protocol/focus`).
- The DAG safe-prefix commit (`protocol/field` / `protocol/nebula`) collects Focus-final vertices into a bundle; the bundle root is the canonical NebulaRoot.
- The BLS Beam aggregates over the bundle's $\geq 2/3$-weight signers. Bundle moves to Beam-final.
- The ML-DSA cert set assembles per-validator FIPS-204 attestations over the same NebulaRoot transcript τ. Bundle moves to PQ-attested.
- The Pulsar engine signs τ via Sign1/Sign2/Combine under the active key era's GroupKey. Bundle moves to Pulse-sealed.
- Prism (`protocol/quasar/horizon.go`) verifies the composite. Bundle moves to Horizon-final.

8.3 Test surface: 258 tests

The Quasar test suite at `protocol/quasar/` contains 258 tests as of the v0.1.0-rc1-pq-consensus-freeze tag (`git log -oneline | grep "rc1-pq-consensus-freeze"`). The suite breaks down as:

File	Coverage
`adversarial_test.go`	Byzantine adversary cases on Beam aggregation
`bls_test.go`	BLS Beam component
`comprehensive_test.go`	End-to-end multi-lane Horizon
`core_test.go`	Core engine state machine
`dual_threshold_test.go`	BLS + Pulse without ML-DSA
`dynamic_test.go`	Validator rotation across bundles
`engine_test.go`	The Quasar engine driver
`epoch_test.go`	Epoch / KeyEra transitions
`grouped_threshold_test.go`	Grouped Pulsar quorum-of-groups path
`horizon_test.go`	Prism predicate, lane composition
`quantum_block_test.go`	PQ-only mode regression
`corona_death_test.go`	Pulse failure / rollback paths
`security_fixes_test.go`	Specific known issues
`security_regression_test.go`	Soundness regressions
`threshold_compat_test.go`	LSS-Pulsar adapter contract
`triple_sign_test.go`	Beam + ML-DSA + Pulse all together
`witness_test.go`	Slashing-evidence witness production

The build / test gate is `GOWORK=off go test -count=1 -short -timeout 300s ./protocol/quasar/...`. All 258 tests pass on the v0.1.0-rc1-pq-consensus-freeze tag (the one known flake, `TestQuantumBundle_ChainInt` is a Pulsar threshold-signing nondeterminism artifact that passes on retry; the underlying signing is correct, the test asserts a non-canonical structural property that the retry exercises).

8.4 Quasar certificate

The on-wire `QuasarCert` (the Horizon composite, with the legacy Quasar 3.0 type name retained for compatibility) is at `protocol/quasar/types.go:40`:

```
type QuasarCert struct {
    BLS        []byte  // BLS12-381 aggregate, 48 bytes
    Pulsar   []byte  // Pulsar Pulse (variable size)
```

```
    MLDSAProof []byte  // Z-Chain Groth16 over N x ML-DSA, ~192 bytes
    Epoch      uint64
    Finality   time.Time
    Validators int
}
```

The newer canonical type at `warp/pulsar/HorizonCertificate` (cf. Section 3) carries the explicit transcript-binding fields (NebulaRoot, KeyEraID, Generation, HashSuiteID) introduced in LP-105. Older callers see the legacy three-byte-blob layout; new callers see the explicit-field layout. Both bind to the same canonical τ.

8.5 Quasar mode toggles

Each lane is independently toggleable:

- BLS-only: classical fast path. Beam-final is the strongest state. Used in development and on chains that explicitly accept the classical-only threat model.
- BLS + Pulse: dual PQ. Pulse-sealed is the strongest single-lane state. Used when ML-DSA cert-set bandwidth is impractical.
- BLS + Pulse + ML-DSA: full Quasar Horizon. The production mode on Lux mainnet.
- BLS + Pulse + ML-DSA + Z-Chain Groth16: production mode with succinct ML-DSA cert-set compression. The wrapper is classical (Remark 1); the underlying PQ root of trust is unchanged.

`IsTripleMode()` on the engine returns true when all three signing layers are configured.

8.6 Domain separation registry

Every ML-DSA / SLH-DSA caller in the Lux stack binds signatures to a context string per FIPS 204/205:

Context	Used by	File
`lux-x-chain-utxo-v1`	UTXO Fx plugins	`utxo/mldsafx`, `utxo/slhdsafx`
`lux-evm-precompile-mldsa-v1`	EVM precompile	`precompile/mldsa/contract.go`
`lux-evm-precompile-slhdsa-v1`	EVM precompile	`precompile/slhdsa/contract.go`
`lux-reshare-v1`	Key resharing HKDF	`threshold/mpc`
`lux-wave-v1`	Wave voting	`consensus/protocol/wave`
`lux-quasar-horizon-v1`	ML-DSA cert set on Horizon transcript	`protocol/quasar/horizon.go`

No collisions. The full registry, with normative wire encodings for each context, is in the NIST submission package ([15]; companion to this paper).

8.7 Performance summary

Per-component CPU costs for `QuasarCert` verification on Apple M1 Max, single core (full numbers in the benchmark report [15]):

Operation	Time
BLS aggregate verify ($n = 100$)	875 μs
ML-DSA-65 verify (single, Fx context)	254 μs
ML-DSA-65 verify (cached)	3 μs
Pulsar threshold verify	variable, amortized $\mathcal{O}(1)$ after DKG
Quasar full block (BLS + ML-DSA + Pulse)	1.85 ms

The benchmark report [15] carries the full microbenchmark table, the LSS resharing benchmarks, the activation-cert latency, and the cross-WAN $K = 21$ Pulse-latency projection.

8.8 Cross-implementation parity

The Go canonical [20, 14, 30, 21, 31] is the byte oracle. The C++ port at `luxcpp/crypto/corona/cpp/` [28] mirrors the Go canonical layer for layer; KAT vectors gate every release. SDK status (honest assessment): Go is production-ready; Python (`pkg/python/`) is the only complete non-Go SDK with real consensus logic; C, Rust (FFI wrapper around C), and C++ are at varying levels of completion. We do not ship cross-language production paths beyond Go.

8.9 Wire transport

Inter-node communication uses ZAP (`github.com/luxfi/zap`; LP-022). Not p2p, not gRPC/protobuf. ZAP delivers $\sim$ 114 K TPS single-connection, $\sim$ 376 K TPS at 20 parallel connections, and $\sim$ 20.26 M TPS at 50 connections + batch 1000 (`bench/`; full numbers in [15]). The ZAP design is not the focus of this paper; the relevant fact is that Beam aggregation and Pulse signing are not network-bound on Lux mainnet at $n = 21$.

9 Related Work

We compare Quasar Horizon to four neighboring consensus families: Tendermint / HotStuff (linear-chain BFT), Narwhal–Tusk / Bullshark / Aleph (DAG-native BFT), Threshold Raccoon / Hermine (post-quantum threshold primitives), and the lattice-vs-classical hybrid approaches in production research.

9.1 Tendermint / HotStuff: linear-chain BFT, classical only

Tendermint [4] and HotStuff [41] are the canonical linear-chain BFT designs. Both finalize blocks via three rounds (propose / prevote / precommit for Tendermint; the four phases of HotStuff). The signing primitive is conventionally a classical aggregate signature (typically Ed25519 / BLS).

Differences vs Quasar Horizon. (1) Linear chain vs DAG. Tendermint / HotStuff finalize linear blocks; Quasar Horizon finalizes Nebula DAG bundle roots. (2) Single-lane vs three-lane finality. Tendermint / HotStuff have one signing lane; Quasar Horizon composes a Beam, an ML-DSA cert set, and a Pulse. (3) Classical only vs hybrid PQ. Neither Tendermint nor HotStuff is post-quantum.

A natural question is whether one could "just bolt" a PQ threshold sig onto Tendermint / HotStuff. The answer is essentially yes (the threshold-Schnorr family fits in the precommit slot), with two caveats: (a) the lattice-threshold primitives have heavier online phases, which is a poor fit for a synchronous linear-chain hot path, and (b) without the validator-set-rotation lifecycle (LSS), the bolt-on becomes a static-committee scheme rather than a permissionless one. Quasar Horizon's contribution is the bolt-on done correctly, with asynchronous PQ finality and dynamic-resharing, integrated with a DAG substrate that absorbs the Pulse latency without blocking the hot path.

9.2 Narwhal–Tusk / Bullshark / Aleph: DAG-native BFT, classical only

Narwhal–Tusk [8] separates mempool / data-availability (Narwhal) from consensus (Tusk), with the consensus layer running over the DAG. Bullshark [39] optimizes the DAG-BFT path with synchronous and asynchronous fallbacks. Aleph [11] provides asynchronous atomic broadcast over a DAG.

Differences vs Quasar Horizon. (1) Classical only. None of these is post-quantum. (2) Single-lane vs three-lane. None composes BLS + ML-DSA + Pulse. (3) Static committees in their default deployments; LSS-style validator rotation is bolt-on work.

The Lux Quasar engine sits in the same DAG-BFT family operationally: Photons / Wave / Focus / Field / Nebula are the DAG-BFT machinery (LP-020 [22]; `protocol/` layout in Section 8). The novel contribution is the post-quantum hybrid finality wrapper around the DAG-BFT core.

9.3 Threshold Raccoon / Hermine: PQ threshold primitives, no consensus integration

Threshold Raccoon [9] is a two-round Module-LWE threshold signature scheme; Hermine [37] is a NIST MPTC-track lattice threshold scheme with DKG and proactive refresh. Both target the cryptographic primitive layer.

Differences vs Quasar Horizon. The primitives compose into the Pulse lane as kernels; they do not by themselves answer the consensus questions: how does a permissionless validator set rotate without a per-epoch trusted dealer (LSS)? How does the Pulse compose with a fast classical aggregate (Beam) and a per-validator PQ accountability lane (ML-DSA cert set) over a shared transcript (Prism)? How does the Pulse follow the Beam without blocking the hot path (asynchronous PQ finality)? How does grouped deployment scale to large validator sets (Section 7)? Threshold Raccoon and Hermine specify the signing primitive; this paper specifies the consensus composition.

Pulsar vs Threshold Raccoon vs Hermine. The Pulsar paper [17] discusses primitive-level differences. The composition argument here is kernel-agnostic: any two-round Module-LWE threshold signature satisfying the LP-073 systems contract works in the Pulse slot.

9.4 FROST / Lens: classical curve threshold, complementary not competing

FROST (RFC 9591 [7]) is the canonical Schnorr-family threshold signature. Lens (LP-103 [33]) is the Lux instantiation, with key-era lifecycle wrapping identical to Pulsar's (`proofs/definitions/algorithms.t` Definition `def:lens-sign`).

Lens in the Lux stack. Lens is the curve-side counterpart to Pulsar. It is classical, not post-quantum; it is the right primitive for control-plane operations (governance, parameter updates, validator-set transitions), where compactness and EVM-precompile-cheap verification matter and the ML-DSA cert set / Pulse already provide the PQ root of trust. Lens does not contribute to the Horizon PQ reduction.

LSS as the universal lifecycle. The LSS lifecycle paper [38] specifies the protocol-agnostic resharing framework that wraps Pulsar, Lens, and ECDSA-CMP under a single contract. The composition contribution of Quasar Horizon is the consensus-level cert; LSS is the orchestration framework underneath.

9.5 Hybrid classical / PQ approaches in production research

Replace BLS entirely ([16]). A maximalist position: drop BLS, run only ML-DSA + Pulse. Smaller cert in the common case; operationally simpler. Quasar Horizon takes the conservative position: keep BLS for fast classical finality, add ML-DSA + Pulse for the PQ floor. Defensible under the "defense-in-depth" argument: under a quantum break of BLS, the chain falls back to

Pulse-sealed / Horizon-final without a chain-level upgrade. The cost is operational complexity; the benefit is graceful degradation.

Z-Chain Groth16 wrapper (LP-307 [29]). A pairing-based SNARK over BLS12-381 compressing the ML-DSA cert set into $\sim$ 192 bytes plus a public-inputs hash. Useful for EVM verifier compatibility, useful for compression. Not post-quantum (Remark 1); not a contribution to the Horizon PQ reduction.

Future PQ-friendly proof systems. P3Q [13] or lattice-based transparent SNARKs replace Groth16 in the wrapper role once they are production-ready (cf. [19] for the Warp 2.0 future-work parallel). Until then, the Groth16 wrapper is honest about its classical security limit.

9.6 Summary table

Family	Substrate	PQ?	Lifecycle	Lanes
Tendermint	linear chain	no	static	1
HotStuff	linear chain	no	static	1
Narwhal–Tusk	DAG	no	static	1
Bullshark	DAG	no	static	1
Aleph	DAG	no	static	1
Threshold Raccoon	primitive only	yes	static	—
Hermine	primitive only	yes	proactive	—
FROST	primitive only	no	static	—
Quasar Horizon	**DAG (Nebula)**	**yes (hybrid)**	**LSS-managed dynamic**	**3 (Beam + ML-DSA + Pulse)**

The Quasar Horizon row is the only design that combines a DAG substrate, post-quantum threshold finality, dynamic validator-set rotation, and conjunctive multi-lane composition. That combination is the contribution.

10 Future Work

The Mar-3 freeze ships Quasar Horizon as the consensus composition over Pulsar / ML-DSA / BLS lanes, with LSS-managed dynamic lifecycle and asynchronous PQ finality discipline. This section names the open work that follows.

10.1 Lumen stream layer

Lumen is the post-quantum encrypted stream layer that carries Photons between validators (LP-105 [34]). The forthcoming Lumen LP specifies three modes:

- **Lumen Fast.** AEAD ratchet, no per-frame OTS. For high-throughput gossip / mempool / bundle data.
- **Lumen Audit.** OTS or publicly verifiable per-frame auth (LMS / XMSS / SLH-DSA). For votes, resharing messages, activation certs.
- **Lumen Bulk.** Bundle / state / chunk transport with optional erasure coding.

The Lumen handshake is hybrid (X-Wing combiner [34]; ML-KEM-768 + X25519 + SHA3, per the IETF specification under that name). The composition with Quasar Horizon is straightforward: Lumen carries Photons; Photons cohere into Beams; Beams compose with ML-DSA cert sets and Pulses via Prism into Horizon. Lumen is orthogonal to the Horizon composition; the open work is the LP, the wire format, and the production deployment.

10.2 FHE result roots

Z-Chain produces FHE-encrypted state transitions ([19] §"Warp Private"). The composition with Horizon: a Z-Chain result root commits to the FHE ciphertext digest; the Pulse signs that root via the standard Horizon transcript. The destination chain (or downstream consumer) verifies under the unchanged GroupKey without seeing plaintext. This is Warp Private as a finality lane.

Open work. The FHE result root is currently a research target. The Lux FHE benchmark report [15] cites measured numbers; the open work is the production circuit and the Pulse-over-FHE-result-root composition theorem.

10.3 P3Q proof lane

The Z-Chain Groth16 wrapper around the ML-DSA cert set is a classical compression artifact (Remark 1). A future PQ-friendly wrapper replaces Groth16 with P3Q [13] or a lattice-based transparent SNARK. The wrapper continues to compress the ML-DSA cert-set verification, but now without a pairing-based assumption.

Open work. (1) Confirm the P3Q parameter set for the ML-DSA-65 verification circuit ($\sim 2^{22.5}$ R1CS gates, App. B of [18]). (2) Benchmark it against Groth16 on the same circuit. (3) Provide a Warp Private PQ envelope variant in Warp 2.0 ([19] §"Future PQ-friendly proof system"). (4) Update LP-307 with the PQ wrapper as the default for new deployments. The state-machine for the wrapper choice is governance-gated; chains can switch wrappers without disturbing the Horizon PQ reduction.

10.4 Lattice-based universally composable security

The current proof bucket carries SUF-CMA reductions and BFT-style safety / liveness theorems. A universally composable (UC, Canetti [5]) treatment of the full Quasar Horizon composition — Beam + ML-DSA + Pulse + LSS lifecycle + DAG-BFT engine + Lumen transport — is open work. The current Tamarin spec at `tamarin/QuasarConsensus.spthy` verifies the protocol-level handshake / vote / bundle flow under standard symbolic abstractions; a UC treatment would carry concrete security under the ideal-functionality framework.

10.5 Grouped Pulsar at $n \geq 1000$

Section 7 stated five open systems-research questions. The most pressing is the closed-form trade-off between G (number of groups), k (group size), f (corruption fraction), and the per-group Pulse latency under WAN churn. Production deployments at $n \sim 100$ are tractable today; production deployments at $n \sim 1000$ are gated on this trade-off being formally characterized.

Empirical handle. The benchmark report [15] cites measured Pulse latency at $K = 21$ and projects $K = 64$, $K = 128$ from the per-party Round 1 / Round 2 cost curve. The closed-form analysis remains open.

10.6 Cross-chain Horizon propagation

Warp 2.0 [19, 23] carries source-chain Horizon finality across chain boundaries via Pulsar Pulses. The composition with Quasar Horizon is direct: a Warp 2.0 envelope carries the source chain's HorizonCertificate (Beam + ML-DSA + Pulse); the destination chain's verifier (Warp 2.0 receiver) re-runs Prism over the source's transcript using the source-chain key registry.

Open work. (1) Cross-chain validator-set registry: how does a destination chain learn the source chain's (V_t, BLSRegistry, MLDSARegistry, PulsarGroupKey) as it evolves? Currently via the destination's source-chain key-registry contract; this contract's update path is itself a Horizon-final on the destination, leading to a recursive trust structure. (2) Quantitative analysis of cross-chain latency: with 5 source chains and 5 destination chains, what is the worst-case cross-chain Horizon-final delay? (3) Cross-chain Horizon batching: can a Warp 2.0 envelope carry multiple source NebulaRoots in one Pulse? Yes in principle; details in [19].

10.7 Reanchor via DKG2 vs Foundation MPC

Pulsar's Bootstrap path is a foundation MPC ceremony. The Pedersen DKG over R_q (`pulsar/dkg2/`; LP-073 [17] §**??**) is the trusted-dealer-free alternative. Both ship in parallel.

Open work. The forthcoming DKG2 LP specifies the Pedersen DKG production path: KAT vectors, constant-time verifier, and the Reanchor governance flow that wires it into chain-level operation. Until DKG2 is production-ready, foundation MPC ceremonies remain the default for new key-eras.

10.8 Threshold ML-DSA

ML-DSA's per-validator signatures are independent (one keypair per validator); there is no threshold ML-DSA in production. [6] (Celi et al., USENIX Security 2025) is a research construction; [18] App. A discusses the rejection-sampling circular dependency that prevents straightforward composition.

Open work. If a production-ready threshold ML-DSA appears, it could replace the per-validator ML-DSA cert set with a single threshold signature. The composition with Horizon is direct (replace step 3 of the Prism predicate with a threshold-ML-DSA verify). Until then, per-validator ML-DSA + Z-Chain Groth16 compression is the production path.

10.9 Scope of these futures

None of these open items affect the core Horizon composition theorem (Theorem 1). They extend the surface (transport, FHE, cross-chain), tighten the analysis (UC, grouped trade-off), and replace classical-only wrappers with PQ-friendly ones. The Mar-3 freeze ships the production composition; the futures listed here are the natural extensions.

References

[1] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *ASIACRYPT*. Springer, 2001.

[2] Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. Ringtail: Practical two-round threshold signatures from LWE. Cryptology ePrint Archive, Paper 2024/1113; IEEE Symposium on Security and Privacy (S&P) 2025, 2024. Academic two-round lattice threshold (IACR ePrint 2024/1113, IEEE S&P 2025). Not a Lux artifact; do not relabel as "Corona" or "Pulsar".

[3] Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction. https://electriccoin.co/blog/new-snark-curve/, 2017.

[4] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. In *arXiv:1807.04938*, 2018.

[5] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*. IEEE, 2001.

[6] Andrea Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Efficient threshold ML-DSA. In *USENIX Security*, 2025.

[7] Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood. RFC 9591: The flexible round-optimized schnorr threshold (frost) protocol. RFC Editor, 2024.

[8] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus. In *EuroSys*, 2022.

[9] Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Mark-Henry Servera. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. Eurocrypt, 2024.

[10] Yvo Desmedt and Sushil Jajodia. Redistributing secret shares to new access structures and its applications. In *Technical Report ISSE TR-97-01, George Mason University*, 1997.

[11] Adam Gągol, Damian Leśniak, Damian Straszak, and Michał Świętek. Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes. In *AFT*, 2019.

[12] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO*, 1995. Updated 1997.

[13] Lux Industries Inc. luxfi/p3q: Post-quantum plonky3 with sha3 challenger. https://github.com/luxfi/p3q, 2026.

[14] Zach Kelling. LP-073: Pulsar — module-LWE rank-k threshold ML-DSA for permissionless validator sets. Technical Report LP-073, Lux Industries, Inc., 2026. Lux's own threshold ML-DSA (FIPS 204) over Module-LWE at general rank k; canonical LP-073. Distinct from Corona (rank 1) and from Ringtail ([2]).

[15] Zach Kelling. Lux post-quantum consensus benchmarks. Lux Industries, Inc.; `papers/lux-pq-consensus-benchmarks/`, 2026.

[16] Zach Kelling. Post-quantum finality for lux quasar: Replacing bls aggregation with ml-dsa + pulsar. Lux Industries; `lux/proofs/pq-finality-no-bls.tex`, 2026.

[17] Zach Kelling. Pulsar: Dynamic lattice threshold signatures for permissionless validator sets. Lux Industries, Inc.; `papers/lp-073-pulsar/`, 2026.

[18] Zach Kelling. QuasarCert soundness: Canonical formal specification, quasar 3.0. Lux Industries; `lux/proofs/quasar-cert-soundness.tex`, 2026.

[19] Zach Kelling. Warp 2.0: Pulse-backed cross-chain source finality. Lux Industries, Inc.; `papers/lux-warp-v2/`, 2026.

[20] Lux Core Team. Consensus: Quasar / nebula / photon / beam / pulse / prism / horizon. https://github.com/luxfi/consensus, 2026.

[21] Lux Core Team. Lens: Curve-based threshold signing (go canonical). https://github.com/luxfi/lens, 2026.

[22] Lux Core Team. Lp-020: Quasar consensus 3.0. https://github.com/luxfi/lps/blob/main/LP-020-quasar-consensus.md, 2026.

[23] Lux Core Team. Lp-021v2: Warp 2.0 cross-chain envelope. https://github.com/luxfi/lps/blob/main/LP-021v2-warp-v2.md, 2026.

[24] Lux Core Team. Lp-070: Fips 204 ml-dsa signatures. https://github.com/luxfi/lps/blob/main/LP-070-mldsa.md, 2026.

[25] Lux Core Team. Lp-073: Pulsar — dynamic lattice threshold signatures for permissionless validator sets. https://github.com/luxfi/lps/blob/main/LP-073-corona.md, 2026.

[26] Lux Core Team. Lp-075: Bls aggregate signatures for warp messaging. https://github.com/luxfi/lps/blob/main/LP-075-bls-aggregate.md, 2026.

[27] Lux Core Team. Lp-077: Linear shamir's secret sharing — universal threshold lifecycle. https://github.com/luxfi/lps/blob/main/LP-077-linear-shamir.md, 2026.

[28] Lux Core Team. Lp-137: Gpu-native crypto stack. https://github.com/luxfi/lps/blob/main/LP-137.md, 2026.

[29] Lux Core Team. Lp-307: Z-chain groth16 rollup of ml-dsa cert sets. https://github.com/luxfi/lps/blob/main/LP-307-z-chain-rollup.md, 2026.

[30] Lux Core Team. Threshold: Lss lifecycle framework + per-kernel adapters. https://github.com/luxfi/threshold, 2026.

[31] Lux Core Team. Warp: Cross-chain messaging protocol (go canonical). https://github.com/luxfi/warp, 2026.

[32] Lux Industries. Lean4 cryptographic proof tree. https://github.com/luxfi/proofs/tree/main/lean/Crypto, 2026. Mechanised companion to `quasar-cert-soundness.tex`.

[33] Lux Industries. LP-103: Lens — curve-based threshold signatures with dynamic resharing. Lux Improvement Proposal, 2026.

[34] Lux Industries. LP-105: The lux finality stack — public vocabulary and internal names. Lux Improvement Proposal, 2026.

[35] NIST. Nist sp 800-185: Sha-3 derived functions: cshake, kmac, tuplehash, parallelhash. https://csrc.nist.gov/pubs/sp/800/185/final, 2016.

[36] NIST. Fips 204: Module-lattice-based digital signature standard. https://csrc.nist.gov/pubs/fips/204/final, 2024.

[37] NIST Multi-Party Threshold Cryptography Project. Hermine: Threshold lattice signatures with dkg and proactive refresh. NIST MPTC presentation, 2025.

[38] Vishnu J. Seesahai and Zach Kelling. LSS: A universal lifecycle framework for threshold cryptographic protocols. Lux Industries, Inc., 2026.

[39] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: DAG BFT protocols made practical. In *CCS*, 2022.

[40] Theodore M. Wong, Chenxi Wang, and Jeannette M. Wing. Verifiable secret redistribution for archive systems. In *SISW*, 2002.

[41] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *PODC*, 2019.