Lux MPTC Technical Specification (NIST IR 8214C-style)

Zach Kelling

Lux Industries

2026-03-03

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 document. The high-level mathematical specification of the Lux MPTC stack: Pulsar (Module-LWE two-round threshold signature with key-era lifecycle), Lens (FROST 2-round Schnorr threshold signature), LSS (protocol-agnostic dynamic-resharing lifecycle framework), Quasar Horizon (consensus-side Prism-bound finality), and Warp 2.0 (cross-chain Pulse-backed envelope). The document follows the NIST IR 8214C submission structure; mathematical detail is summarized to the level needed for review, with pointers to the canonical proof-bucket files for full proofs.

Contents

1 Pulsar (lattice threshold)

Pulsar is a (t, n) post-quantum threshold signature scheme over the cyclotomic ring $R_q = \mathbb{Z}_q[X]/(X^{256}+1)$ with $q = \texttt{0x1000000004A01}$ (48-bit NTT-friendly prime). The signing math is derived byte-equal from Ringtail ([1], IEEE S&P 2025); Lux's contribution is the key-era lifecycle that wraps the static signing primitive with permissionless validator-set rotation, activation-gated resharing, and rollback. The full mathematical specification is in the LP-073 paper [3]; here we summarize the externally-visible surface.

1.1 Public algorithms

- $\mathsf{Pulsar.Setup}(1^\lambda) \to \mathsf{params}$: deterministic parameter derivation. (See `proofs/definitions/algorithms.te` Definition `def:pulsar-setup`.)
- $\mathsf{Pulsar.Gen}(\mathsf{params}, t, n; \rho) \to (\mathsf{GroupKey}, \{\mathsf{Share}_i\})$: trusted-dealer key generation (foundation MPC ceremony at chain genesis), or $\mathsf{Pulsar.DKG2}(\mathsf{params}, t, n)$: Pedersen-style DKG over R_q, producing the same output without a trusted dealer.
- $\mathsf{Pulsar.Sign1}, \mathsf{Pulsar.Sign2}, \mathsf{Pulsar.Combine}$: two-round MAC-authenticated signing protocol; see Definition `def:pulsar-sign`.
- $\mathsf{Pulsar.Verify}(\mathsf{GroupKey}, m, \sigma) \to \{0, 1\}$: standalone verification under the persistent group public key.
- $\mathsf{Pulsar.Refresh}(\{\mathsf{Share}_j\}, t)$: HJKY97 same-set zero-poly refresh (Definition `def:pulsar-refresh`).
- $\mathsf{Pulsar.Reshare}(\{\mathsf{Share}_i\}_{i \in Q}, t_{\text{old}}, V_{\text{new}}, t_{\text{new}})$: Desmedt-Jajodia old-qualified-subset resharing (Definition `def:pulsar-reshare`).
- $\mathsf{Pulsar.Activate}$: new committee threshold-signs the canonical $\mathsf{ActivationMsg}$ under the unchanged GroupKey (Definition `def:pulsar-activate`); the chain accepts the new generation iff the activation cert verifies.

1.2 Key-era lifecycle

A Pulsar share state carries three lineage scalars: $\mathsf{KeyEraID}$ (η, persistent group-key lineage), $\mathsf{Generation}$ (g, LSS resharing version within an era), $\mathsf{RollbackFrom}$ (r, audit lineage; 0 on forward transitions, > 0 records prior generation reverted from). The GroupKey $(\mathbf{A}, \tilde{\mathbf{b}})$ is byte-identical for every g within η (Lemma 4, `proofs/lss/lifecycle-safety.tex`); η increments only on Reanchor.

1.3 Security

Theorem 1 (Pulsar SUF-CMA under MLWE/MSIS)**.** *Let $\mathcal{A}$ be a probabilistic polynomial-time adversary against Pulsar with at most q_S signing queries and at most $t-1$ corrupted parties. Then*

$$\mathrm{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{Pulsar}}(\mathcal{A}) \;\le\; q_S \cdot \mathrm{Adv}^{\mathsf{MLWE}}_{q,\sigma_E}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{MSIS}}_{q,[\mathbf{A}|-c\tilde{\mathbf{b}}],2B}(\mathcal{B}_2) + \mathrm{negl}(\lambda).$$

The full proof lives in `proofs/pulsar/unforgeability.tex`; concrete parameters give classical security $\ge 2^{142}$ and quantum security $\ge 2^{130}$ via BDGL sieving plus Grover speedup (`quasar-cert-soundness.tex` App. E). Lean mechanization: `Crypto.Ringtail.ringtail_pq_unforgeability` [10].

2 Lens (curve threshold)

Lens is the FROST 2-round Schnorr threshold signature [2] over a prime-order group $\mathbb{G}$ of order ℓ. Lux supports three groups: Ed25519, secp256k1, Ristretto255. Lens inherits the RFC 9591 algorithm byte-equal; the Lux contribution is the key-era lifecycle wrapping (identical in shape to Pulsar's, with R_q replaced by $\mathbb{Z}_\ell$ and lattice Pedersen replaced by curve Pedersen $g^s \cdot h^r$).

2.1 Public algorithms

Lens.Sign1, Lens.Sign2, Lens.Aggregate, Lens.Verify, Lens.Refresh, Lens.Reshare, Lens.Activate, all mirroring Pulsar's lifecycle (Definition `def:lens-sign` of `proofs/definitions/algorithms.tex`). The LSS-Lens adapter (`lss_lens.go`) shares the contract with LSS-Pulsar; 10/10 acceptance tests pass.

2.2 Security

Standard FROST EUF-CMA reductions to the discrete-log assumption on $\mathbb{G}$ [2]. Lens is classical (curve discrete log), broken in polynomial time by Shor; Lens is the right primitive for control-plane operations and cross-rail compatibility, not for the consensus PQ floor.

3 LSS (lifecycle framework)

LSS (Linear Shamir Secret Sharing, LP-077 [8]) is the protocol-agnostic dynamic-resharing lifecycle framework. LSS owns generations, snapshots, rollback, and dealer / coordinator roles; the underlying threshold kernels (Pulsar, Lens, ECDSA-CMP) own their own signing math. The contract is documented in `proofs/lss/adapter-contract.tex`.

3.1 Adapter contract

Definition 1 (LSS adapter contract)**.** *An LSS adapter MUST satisfy:*

1. *Lifecycle delegation: the adapter does not implement share-secret arithmetic; it calls the kernel (*Pulsar.Reshare*,* Lens.Reshare*, etc.) and wraps the output.*
2. *Generation increment: $g \mapsto g+1$ on every successful* DynamicReshare*. On* Rollback(g_{target})*, $g \mapsto g_{\text{current}} + 1$ with $\mathsf{rb} = g_{\text{current}}$.*
3. *KeyEraID preservation: η is read-only across* DynamicReshare*. η increments only on* Reanchor*.*
4. *GroupKey identity: the kernel's GroupKey pointer flows through unmodified.*
5. *Failure-mode containment: on any verification failure (commit mismatch, activation rejection), return an error and retain the previous state.*
6. *No ECDSA-specific assumptions inherited beyond what the underlying kernel actually uses.*

3.2 Lifecycle invariants

Three lemmas hold for any LSS-managed kernel (`proofs/lss/lifecycle-safety.tex`):

Lemma 2 (Generation monotonicity)**.** *g increases monotonically by exactly 1 per* DynamicReshare *or* Refresh *within an era. On* Rollback*, the restored state carries $g = g_{\text{current}} + 1$ and $\mathsf{rb} = g_{\text{current}}$.*

Lemma 3 (KeyEraID stability)**.** *η is unchanged by* Refresh*,* Reshare*, or* Rollback*. η increments only on* Reanchor*.*

Lemma 4 (GroupKey byte-identity)**.** *The GroupKey pointer is byte-identical across every* Refresh *and* Reshare *within a key era. (Reduces to master-secret preservation, Theorem 5.)*

Theorem 5 (Reshare preserves master secret)**.** *The new shares $\{\mathsf{Share}'_j = G(\beta_j)\}$ are valid $(t_{\text{new}}, |V_{\text{new}}|)$-Shamir shares of the same master secret $\mathbf{s}$, where $G(X) = \sum_{i \in Q} \lambda_i^Q \cdot g_i(X)$ and $g_i(0) = \mathbf{s}_i$. (Lagrange identity; full proof in `proofs/pulsar/reshare-preservation.tex`.)*

4 Quasar Horizon (consensus composition)

Quasar Horizon ([4]) is the consensus-side composition of three independent threshold artifacts under transcript binding:

- Beam: BLS12-381 aggregate over independent validator BLS keypairs (LP-075 [7]); classical fast-path finality.
- ML-DSA cert set: per-validator FIPS-204 ML-DSA-65 attestations (LP-070 [6]); PQ accountability.
- Pulse: one Pulsar threshold signature over the canonical QuasarTranscript; PQ threshold finality.

The certificate-composition verifier is *Prism* (LP-105 [11]), which checks that all lanes refract from the same Nebula transcript (Definition 2).

Definition 2 (Quasar transcript)**.** QuasarTranscript := H_T(*"QUASAR-TRANSCRIPT-v1"*, NebulaRoot, chain_i
where H_T = TupleHash256.

Theorem 6 (Horizon soundness)**.** *A Prism-accepted Horizon certificate cannot exist for two distinct* NebulaRoot*s at the same* (epoch, round) *except with negligible probability under (i) BLS12-381 co-CDH, (ii) Module-LWE / Module-SIS, and (iii) FIPS 204 EUF-CMA. The lanes are independent (different hardness assumptions); a quantum break of one does not break the others. (Full proof:* `proofs/quasar/horizon-soundness.tex`*.)*

5 Warp 2.0 (cross-chain envelope)

Warp 2.0 ([5]) is the cross-chain envelope carrying source-chain Horizon finality. Wire format:

```
0x02 || RLP([
    Message,                    // v1 Beam (BLS aggregate)
    SourceNebulaRoot   [32]byte,
    SourceKeyEraID     uint64,
    SourceGeneration   uint64,
    HashSuiteID        string,
    PulsarPulse        []byte,     // empty if absent
    MLDSACertSet       []byte,     // empty if absent
])
```

The Pulse signs H_T("WARP-PULSAR-ENVELOPE-v1", sourceChainID, payload, sourceNebulaRoot, sourceKe
The destination resolves (sourceChainID, sourceKeyEraID, sourceGeneration) to the source's GroupKey + HashSuiteID via the destination's source-chain key registry.

Theorem 7 (Warp 2.0 envelope unforgeability)**.** *A successful* Verify_{V2} *implies any forgery on the same* (sourceChainID, sourceNebulaRoot, sourceKeyEraID, sourceGeneration) *tuple with a different* payload *requires breaking either co-CDH (Beam) or MLWE/MSIS (Pulse). (Full proof:* `proofs/quasar/warp-pq-soundness.tex`*.)*

6 Composition under transcript binding

The unifying structural fact: all three certificate lanes (Beam, ML-DSA, Pulse) and the cross-chain Pulse share one canonical transcript. Domain-separation tags (Section `spec/domain-separation-registr` prevent cross-protocol replay. Hash-suite identifiers (`Pulsar-SHA3` default) prevent cross-suite forgery (`proofs/pulsar/hash-suite-separation.tex` Theorem 8):

Theorem 8 (Hash-suite separation)**.** *A Pulsar signature under one hash suite does not verify under a different hash suite, with overwhelming probability over the verifier's hash function.*

7 Network tier model

Lux defines L1 / L2 / L3 by validator co-validation, not by rollup vs settlement (Definition `def:network-tiers` of `proofs/definitions/finality-definitions.tex`):

- L1: sovereign; own validator set, own KeyEraID lineage.
- L2: declares primary_network_id $= N_P$; validator set $\subseteq V_{N_P}$. Activation cert binds primary_nebula_root. Inheritance via Theorem 9.
- L3: settles via succinct proof; PQ finality inherits from underlying L1/L2.

Theorem 9 (L2 co-validation security inheritance)**.** *An L2's safety bound is at least the minimum of its primary's safety bound and its own validator-set safety bound. (Full proof:* `proofs/quasar/l1-l2-co`

8 Honest classification of the Groth16 wrapper

The optional Z-Chain Groth16 rollup proof of ML-DSA cert-set verification (LP-307 [9]) is a *classical succinct proof of post-quantum signature verification.* Pairing-based SNARKs over BLS12-381 break under Shor; the Groth16 wrapper is a compatibility / compression / privacy adapter, not a PQ root of trust. PQ finality lives in ML-DSA + Pulsar.

This classification is enforced in code: `warp/pulsar/IsPQRootOfTrust` returns false for Groth16-wrapped lanes (`warp/pulsar/groth16_classification_test.go`).

References

[1] Ringtail Authors. Ringtail: A lattice-based threshold signature for post-quantum consensus. In *IEEE S&P*, 2025.

[2] Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood. RFC 9591: The FROST protocol. RFC Editor, 2024.

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

[4] Zach Kelling. Quasar horizon: Hybrid post-quantum finality over dag-native nebula roots. Lux Industries, Inc.; `papers/lux-quasar-horizon/`, 2026.

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

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

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

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

[9] Lux Core Team. Lp-307: Z-chain groth16 rollup. https://github.com/luxfi/lps/blob/main/LP-307-z-chain-rollup.md, 2026.

[10] Lux Industries. Lean4 cryptographic proof tree. https://github.com/luxfi/proofs/tree/main/lean/Crypto, 2026.

[11] Lux Industries. LP-105: The lux finality stack. Lux Improvement Proposal, 2026.