# PQ profile

## The Strict Post-Quantum Profile of Lux Quasar

---

Zach Kelling

*Lux Industries*

Genesis-pinned. Fail-closed. NIST-forward.

# PQ profile

The Strict Post-Quantum Profile of Lux Quasar

Lux Industries Inc.

Genesis-pinned. Fail-closed. NIST-forward.

**Abstract**

**Quasar** is the Lux post-quantum cryptographic stack: PQ wallets, PQ transaction authorization, PQ contract authorization, PQ validator identity, PQ DKG, PQ finality, PQ rollup proofs, PQ networking, and PQ governance. **PQ** is the strict-PQ *boundary* of Quasar: the ring that excludes classical crypto and enforces SHA3_NIST, ML-DSA, Pulsar finality, STARK / FRI proofs, ML-KEM networking, and fail-closed verification.

PQ is not the whole stack and not a new primitive. It is the visible boundary of the eclipse — the ring of fire where Quasar eclipses the legacy classical stack. Toolkits remain configurable for forks and benchmarks; the PQ boundary itself is fail-closed.

This specification pins the boundary rules: the canonical `ProfileID = QUASAR_STRICT_PQ` byte, the field-by-field strict-PQ posture, the proof-envelope verifier gate, the TxAuth / contract-auth / DKG / finality / networking subprofiles, and the genesis-pinning discipline that makes operator flags incapable of weakening the boundary at runtime.

## Contents

# 1 Profile, not primitive

The architectural decision is the entire story:

> *Quasar is the PQ stack; PQ is its strict-PQ boundary. Pulsar, Corona, P3Q, and the FIPS PQ primitives are toolkits.*

Forks can use BLAKE3, Keccak, Groth16, PLONK, ECDSA, BLS, fake proofs, dev receipts, unsafe bridge wrappers. None of those are PQ. PQ accepts only:

- ML-DSA-65 / 87 identities (FIPS 204).
- ML-DSA-65 / 87 transaction and contract authorization, or Z-Chain batched authorization proofs.
- Pulsar-M threshold finality (M-LWE / FIPS 204-compatible).
- ML-KEM-768 / 1024 networking channels (FIPS 203).
- SHA3_NIST hashes (cSHAKE, KMAC, TupleHash, ParallelHash; FIPS 202 + SP 800-185).
- STARK / FRI proof policy with transparent setup; no pairings, no KZG, no Groth16/PLONK wrappers.
- SLH-DSA cold-root / recovery (FIPS 205).

That single distinction — toolkit vs profile — is the entire PQ architecture.

# 2 Lineage

| name | role | status | basis |
|---|---|---|---|
| **Nasua** | academic fork of Corona / Raccoon | archived / academic-only | Ring-LWE u |
| **Corona** | Lux R-LWE production-hardened threshold | enabled (non-default S-class) | forks Nasua |
| **Pulsar** | Lux M-LWE / ML-DSA threshold | enabled (PQ profile core) | lifts Corona |
| **PQ profile** | full-stack strict-PQ profile (*this spec*) | *the profile* that binds the production stack | no new prim |

# 3 Canonical profile

| field | value |
|---|---|
| ProfileName | `PROFILE_QUASAR_STRICT_PQ` |
| HashSuiteID | `SHA3_NIST` |
| WalletSchemeID | ML-DSA-65 |
| TxAuthSchemeID | ML-DSA-65 \| ZCHAIN_AUTH_PROOF |
| ContractAuthSchemeID | ML-DSA-65 \| ML-DSA-87 \| SLH-DSA \| ZCHAIN_AUTH_PROOF |
| ValidatorIdentityID | ML-DSA-65 |
| FinalitySchemeID | Pulsar-65 (was Pulsar-M-65) |
| HighValueFinalityID | Pulsar-87 (was Pulsar-M-87) |
| KeyExchangeID | ML-KEM-768 default; ML-KEM-1024 high-value |
| ProofPolicyID | `ProofPolicySTARKFRISHA3PQ` |
| RootHashBits | 384 minimum |
| CheckpointHashBits | 512 |
| CommitteeSizeDefault | 64 |
| ThresholdDefault | 43-of-64 |
| DKGCadence | once per epoch |
| ClassicalFallbacks | forbidden |

# 4 Subprofiles

PQ has the right scope when its subprofiles are addressed separately. Every subprofile inherits the strict-PQ guarantees of §3.

| subprofile | scope |
|---|---|
| **PQ profile-ID** | PQ identity and account layer |
| **PQ profile-TX** | PQ transaction authorization |
| **PQ profile-VM** | PQ contract authorization and precompiles |
| **PQ profile-Z** | Z-Chain PQ proof / auth rollup profile |
| **PQ profile-Q** | Q-Chain finality-block profile |
| **PQ profile-DKG** | Pulsar epoch DKG profile |
| **PQ profile-Net** | ML-KEM secure networking profile |
| **PQ profile-Gov** | PQ governance / bridge / upgrade profile |

# 5 Component map

| layer | PQ choice | purpose |
|---|---|---|
| hash / transcript | SHA3_NIST: SHAKE, cSHAKE, KMAC, TupleHash, ParallelHash | domain separation, root |
| wallet identity | ML-DSA-65 | user account keys |
| validator identity | ML-DSA-65 | validator registration + |
| high-value identity | ML-DSA-87 / SLH-DSA | governance, bridge, eme |
| tx auth | ML-DSA-65 direct *or* Z-Chain batched auth proof | user transaction author |
| contract auth | PQ precompile / Z-Chain auth proof | replaces ECDSA permit |
| finality | Pulsar-65 (M-LWE) | threshold ML-DSA fina |
| high-value finality | Pulsar-87 (M-LWE) | checkpoints, bridges, go |
| DKG | Pulsar epoch DKG | group key for Q-Chain |
| proofs | STARK / FRI / SHA3 (P3Q + peers) | PQ rollup proofs |
| proof backends | SP1, RISC0, P3Q, Stone, Stwo behind one ABI | benchmarkable backend |
| P2P / private channels | ML-KEM-768 / 1024 (FIPS 203) | PQ key establishment |
| recovery | SLH-DSA (FIPS 205) | cold, rare, long-lived tru |

# 6 PQ profile-ID: identity and account layer

PQ accounts are not Ethereum-style 20-byte addresses. The canonical account identity is:

$\mathsf{AccountID} = \textsc{TupleHash256}\big(\text{“PQ-ACCOUNT-ID-V1”}, \mathit{profile\_id}, \mathit{chain\_id}, \mathit{scheme\_id}, \mathit{pubkey}\big)[0{:}32]$.

For high-value accounts, the SHAKE256-384 variant binds “PQ-HIGH-VALUE-ACCOUNT-V1” and outputs 48 bytes. Account records live on Z-Chain:

- `account_id`
- `scheme_id` (ML-DSA-65 / 87 / SLH-DSA only)
- `compact_public_key`, `expanded_public_key_hash` (opt.)
- `status` $\in$ {active, rotated, revoked, frozen}
- `valid_from_epoch`, `valid_until_epoch`
- `rotation_policy_hash`, `recovery_policy_hash`

**PQ profile accepts** only ML-DSA-65, ML-DSA-87, SLH-DSA keys. **PQ profile rejects** ECDSA, secp256k1-only, P-256-only WebAuthn, and BLS account-auth keys.

# 7 PQ profile-TX: PQ transaction authorization

Every transaction carries a self-describing auth envelope:

```
TxAuthEnvelope {
  version
  profile_id           = PROFILE_QUASAR_STRICT_PQ
  chain_id, network_id
  account_id
  nonce
  expiry_height
  auth_scheme_id       = ML_DSA_65 | ZCHAIN_AUTH_PROOF
  hash_suite_id        = SHA3_NIST
  public_key_ref
  signature_or_proof_ref
  tx_digest
}
```

The digest is TupleHash256(‘‘PQ-TX-AUTH-V1’’, profile_id, chain_id, network_id, account_id, nonce, expiry_height, fee_payer, gas_limit, max_fee, call_target, call_value, call_data_hash, zchain_state_root). Two paths accepted: direct ML-DSA-65 signature, or a Z-Chain batched authorization proof that lets wallets sign with ML-DSA while the hot path stays small.

# 8 PQ profile-VM: PQ contracts

Contracts must stop depending on `ecrecover`. PQ VM host functions:

```
pq_verify_mldsa65(pk_ref_or_pk, digest, sig) -> bool
pq_verify_mldsa87(pk_ref_or_pk, digest, sig) -> bool
pq_verify_slh_dsa(pk_ref_or_pk, digest, sig) -> bool
pq_verify_z_auth_proof(account_id, digest, tx_auth_root, proof_ref) -> bool
pq_account_id_from_pubkey(scheme_id, pubkey) -> account_id
```

Contract auth profile defaults under PQ: `admin = upgrade = ML-DSA-87 or SLH-DSA`; `pause = ML-DSA-65 multisig or Pulsar committee`; `permit = ML-DSA-65 or ZCHAIN_AUTH_PROOF`; `allow_classica = false`.

The `PQPermit` envelope replaces ECDSA EIP-2612 permits and carries {*profile_id*, *chain_id*, *verifying_contract*

## 9 PQ profile-Z: PQ Z-Chain profile

Z-Chain is not a general EVM rollup in PQ core. It is a PQ authorization and state-proof rollup. It proves wallet key registration, rotation, revocation, session-key authorization, contract-policy updates, batched transaction-signature validity, batched PQ permits, validator identity validity, committee selection, DKG transcript-root acceptance, group public-key binding, and epoch commitment construction.

Public roots committed by Z-Chain:

- `identity_root`, `account_key_root`, `revocation_root`, `session_key_root`
- `contract_policy_root`, `tx_auth_root`, `permit_auth_root`
- `validator_registry_root`, `stake_weight_root`, `committee_root`
- `dkg_transcript_root`, `group_public_key_hash`

Proof policy: `ProofPolicyID = ProofPolicySTARKFRISHA3PQ`. Allowed proof backends: SP1_COMPRESSED_S RISC0_SUCCINCT_STARK, P3Q_PLONKY3_STARK_FRI, STONE_CAIRO_STARK_CPP, STWO_CIRCLE_ST STARK / FRI is the right proof family for PQ: hash-based, transparent setup, no pairings, no KZG, no trusted setup. SP1 and RISC0 are allowed only along their STARK paths; their Groth16 / PLONK wrappers are classical-SNARK fallbacks and PQ rejects them.

## 10 P3Q: canonical long-term proof backend

P3Q is the Lux strict-PQ Plonky3 profile / fork. P3Q permits STARK, FRI, Merkle / hash commitments, transparent setup, STARK-native recursion, and external SHA3 / SHAKE / cSHAKE roots. P3Q forbids BN254, KZG, Groth16, PLONK/KZG, pairings, elliptic-curve recursion, and trusted setup. The distinction matters because Plonky3 is a general polynomial-IOP toolkit; PQ pins a strict subset of it rather than inheriting the whole toolkit unchecked.

Stone (Cairo CPU AIR, C++, Apache-2.0) and Stwo (Circle STARK, Rust) are useful independent STARK backends and slot into the same `ProofBackendID` surface alongside P3Q.

## 11 PQ profile-Q: Q-Chain finality profile

Q-Chain is the compact finality lane. Each Q-Block carries no raw wallet keys, no raw validator keys, no raw transaction signatures, no raw DKG messages, no full Z-Chain witness data, and no classical fallback certs. Its transcript binds (no unbound field, no silent fallback, no "best effort"):

- `profile_id`, `hash_suite_id`, `proof_policy_id`, `finality_scheme_id`
- `network_id`, `chain_id`, `epoch`, `height`, `round_or_view`, `parent_qblock_hash`
- `lux_state_root`, `zchain_state_root`, `tx_auth_root`, `payload_root`, `da_root`
- `committee_root`, `dkg_transcript_root`, `group_public_key_hash`
- `signer_bitmap_or_weight_commitment`

Each Q-Block carries a single Pulsar-65 threshold finality certificate over the digest. High-value checkpoints use Pulsar-87.

## 12 PQ profile-DKG: Pulsar finality

| | |
|---|---|
| primitive | Pulsar (was Pulsar-M) |
| base | ML-DSA / FIPS 204-compatible |
| NIST target | MPTC Class N1 + N4 |
| default profile | Pulsar-65 |
| high-value profile | Pulsar-87 |
| committee size | 64 |
| threshold | 43-of-64 |
| DKG cadence | once per epoch |
| abort model | identifiable abort |
| fallback | bounded previous-epoch grace only |
| classical fallback | forbidden |

Flow: (1) Z-Chain proves validator registry and committee root; (2) committee runs Pulsar epoch DKG; (3) DKG transcript root and group public-key hash are committed to Z-Chain; (4) Q-Chain accepts finality blocks only under the committed group key; (5) each Q-Block carries a Pulsar-65 threshold finality certificate.

The NIST-facing statement stays narrow: *Pulsar is a threshold ML-DSA signing and DKG system targeting NIST MPTC Class N1 + N4, with final signatures intended to verify under unmodified FIPS 204 ML-DSA verification.* That is much stronger and cleaner than claiming the whole chain is FIPS-validated.

## 13 PQ profile-Net: PQ networking

| | |
|---|---|
| node identity | ML-DSA-65 |
| validator identity | ML-DSA-65 |
| session KEX | ML-KEM-768 (FIPS 203) |
| validator / DKG KEX | ML-KEM-1024 preferred |
| transcript hash | SHA3_NIST / cSHAKE |
| channel binding | KMAC / TupleHash |
| transport AEAD | FIPS-approved AEAD inside module boundary |

**PQ profile rejects** ECDH-only validator channels, classical-only node identity, unauthenticated DKG transport, and unbound session transcripts.

## 14 PQ profile-Gov: governance, bridge, upgrades

Stronger / slower keys for rare high-value actions:

- Governance key: ML-DSA-87 or SLH-DSA.
- Bridge admin key: ML-DSA-87 or SLH-DSA.
- Upgrade key: SLH-DSA or ML-DSA-87 multisig.
- Emergency pause: Pulsar committee or ML-DSA-87 multisig.
- Checkpoint root: Pulsar-87.

SLH-DSA (FIPS 205) is the right cold-root because it is stateless and hash-based. A bridge is PQ-grade only if *both ends* are PQ; otherwise the bridge must carry `POST_QUANTUM_END_TO_END = false` and `QUASAR_STRICT = false`.

## 15 Proof envelope

Every proof backend maps into one canonical envelope:

```
ZProofEnvelope {
  version
  profile_id, chain_id, network_id, epoch
  proof_policy_id, proof_backend_id, proof_format_id, verifier_id
  hash_suite_id, identity_scheme_id, finality_scheme_id
  public_inputs_hash, verifier_key_hash
  program_or_air_id, program_or_air_hash
  soundness_bits_claimed, hash_output_bits
  transparent_setup
  uses_pairings, uses_kzg, uses_trusted_setup,
  uses_classical_snark_wrapper
  proof_bytes
}
```

PQ verifier gate: refuse the envelope unless `profile_id == QUASAR_STRICT_PQ`, `hash_suite_id == SHA3_NIST`, `proof_policy_id == STARK_FRI_SHA3_PQ`, `proof_backend_id` is in the allowed list, `soundness_bits_claimed` $\geq 128$, `hash_output_bits` $\geq 384$, `transparent_setup` is true, `uses_pairings = uses_kzg = uses_trusted_setup = uses_classical_snark_wrapper = false`.

The backend cannot self-certify these flags. They are pinned by a `VerifierManifest` signed by the chain governance:

```
VerifierManifest {
  verifier_id, backend_id, source_commit, build_profile
  verifier_key_hash, program_or_air_hash
  supports_policy_ids
  reviewed_soundness_bits
  uses_pairings, uses_kzg, uses_trusted_setup,
  uses_classical_snark_wrapper
}
```

## 16 Genesis profile

PQ is locked at genesis. Operator flags must not override this:

```
GenesisSecurityProfile {
  profile_id, profile_name, profile_hash
  hash_suite_id
  wallet_scheme_id, tx_auth_scheme_id, contract_auth_scheme_id
  validator_identity_id, finality_scheme_id, high_value_scheme_id
  proof_policy_id, allowed_proof_backends
  min_soundness_bits, min_hash_output_bits
  forbid_pairings, forbid_kzg, forbid_trusted_setup
  forbid_classical_snarks, forbid_classical_signatures
  forbid_bls_fallback, forbid_dev_proofs
}
```

Allowed mainnet command-line shapes:

```
luxd --network lux-mainnet --role validator --datadir /var/lib/lux
zchain-prover --backend sp1 --profile pq-strict-pq
zchain-bench --all-backends --profile pq-strict-pq
```

Forbidden on mainnet:

```
--hash-suite blake3
--allow-ecdsa
--allow-groth16
--allow-kzg
--allow-bls-fallback
--accept-dev-proof
--lower-threshold
```

## 17 What PQ profile eclipses

| prior path | PQ status |
|---|---|
| Nasua | archived / academic tracker only |
| Corona | legacy production R-LWE line; optional non-NIST Class S; *not* PQ default |
| Pulsar | core PQ finality primitive; M-LWE / ML-DSA-compatible; NIST-facing |
| BLS | removed from strict finality; compatibility only outside PQ |
| ECDSA | removed from wallets / contracts in strict mode; compatibility only outside PQ |
| Groth16 / PLONK / KZG | removed from strict Z-Chain; compatibility only outside PQ |
| BLAKE3 | non-normative legacy / benchmark profile only |

PQ is the ring of fire around the old stack: the eclipse boundary where classical assumptions disappear.

## 18 Implementation order

From the current state, the implementation order is:

1. **Close F22** — Pulsar must use SHA3_NIST HashSuite internally by default.
2. **Close F34** — Q-Chain digest must bind `hash_suite_id`, `sig_scheme_id`, `proof_policy_id`, `committee_root`, `dkg_transcript_root`, `group_public_key_hash`, and `profile_id`.
3. Add the PQ profile object; genesis-pinned, hash-checked, zero-value-invalid.
4. Split proof IDs: `ProofPolicyID`, `ProofBackendID`, `ProofFormatID`, `VerifierID`.
5. Add `ZProofEnvelope`; one envelope for SP1, RISC0, P3Q, Stone, Stwo.
6. Add `TxAuthEnvelope`; ML-DSA-65 direct and `ZCHAIN_AUTH_PROOF` paths.
7. Add PQ account registry; account keys, rotations, revocations, sessions, contract policy.
8. Add PQ VM host functions; ML-DSA / SLH-DSA verification, Z-auth proof verification.
9. Disable classical auth in PQ; `ecrecover` / ECDSA / BLS / KZG / Groth16 rejected.
10. Add PQ status labels to RPC, explorer, metrics, block headers.

## 19 Final definition

PQ is Lux's strict end-to-end post-quantum cryptographic stack. It binds ML-DSA wallets, PQ transaction authorization, PQ contract authorization, Z-Chain STARK / FRI / SHA3 proofs, Q-Chain Pulsar finality, epoch DKG, ML-KEM networking, and SLH-DSA high-value recovery under one genesis-pinned profile. Toolkits remain configurable for forks; PQ itself is fail-closed, SHA3_NIST, FIPS-forward, NIST-friendly, and rejects every classical authorization or proof path.

## References


[1] National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (FIPS 202). 2015.

[2] National Institute of Standards and Technology. SP 800-185: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, ParallelHash. 2016.

[3] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard (FIPS 203, ML-KEM). 2024.

[4] National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard (FIPS 204, ML-DSA). 2024.

[5] National Institute of Standards and Technology. Stateless Hash-Based Digital Signature Standard (FIPS 205, SLH-DSA). 2024.

[6] National Institute of Standards and Technology. First Call for Multi-Party Threshold Schemes (NIST IR 8214C). 2026.

[7] Lux Industries Inc. Pulsar: Threshold ML-DSA over Module-LWE with FIPS 204 output interchangeability. `github.com/luxfi/pulsar`.

[8] Lux Industries Inc. Corona: Production R-LWE threshold signatures (legacy non-default S-class path). `github.com/luxfi/corona`.

[9] Lux Industries Inc. Nasua: archived academic fork of Corona / Raccoon. `github.com/luxfi/nasua`.

[10] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. IACR ePrint 2018/046.

[11] Polygon Labs et al. Plonky3. `github.com/Plonky3/Plonky3`.