Secure Enclave, HSM, and Threshold Boundary Design for Validator Key Infrastructure

Zach Kelling

Lux Industries

2025

Abstract

Hardware Security Modules (HSMs) and Trusted Execution Environments (TEEs) provide tamper-resistant boundaries for key material, but their security properties are frequently misunderstood. We analyze the boundary design for Lux Network's validator key infrastructure, specifying when hardware protection helps, when it does not, and how it interacts with threshold cryptography. The analysis covers key material locality (which keys reside in hardware, which in software), side-channel assumptions for BLS and ML-DSA implementations, attestation models for TEE-hosted signers, and the interaction between HSM-enforced access policies and threshold signature protocols (FROST for BLS, Pulsar for post-quantum). We define a reference architecture where HSM boundaries protect individual key shares while threshold protocols protect against HSM compromise, providing defense-in-depth that neither mechanism achieves alone.

1 Introduction

A Lux validator holds multiple categories of key material:

- **BLS signing key**: Used for consensus signatures, Warp message attestation, and block finalization [1].
- **ML-DSA signing key**: Post-quantum identity key used in hybrid certificates [2].
- **Pulsar key shares**: Threshold post-quantum signature shares for triple-proof finality.
- **TLS staking key**: X.509 certificate private key for P2P authentication.
- **BLS signer key**: 32-byte key for BLS operations, separate from staking.

The instinct is to put all key material into an HSM. This is neither necessary nor sufficient. HSMs protect against key extraction but introduce latency, limit throughput, and create a single point of failure if the HSM itself is compromised (firmware vulnerability, supply chain attack). Threshold cryptography distributes trust but requires online shares that may be vulnerable to memory extraction. The correct design uses both: HSMs protect individual shares, and the threshold protocol ensures that compromising any $< t$ HSMs does not compromise the aggregate key.

2 Boundary Model

2.1 What Goes in Hardware

Definition 1 (HSM Boundary)**.** *The HSM boundary $\mathcal{H}$ is the set of operations and data elements that execute within tamper-resistant hardware. For Lux validators:*

$$\mathcal{H} = \{sk_{BLS\_share}, sk_{MLDSA}, Sign(\cdot), DKG\_share(\cdot)\}$$

Key material that resides in hardware:

2.2 What Stays in Software

The following deliberately remain outside the HSM boundary:

1. **FHE evaluation keys**: These are public parameters ($\sim$100 MB for PN10QP27 [5]). Placing them in HSM would exceed storage limits and add unusable latency.

Key	Location	Extractable	Rationale
BLS share (sk_i)	HSM	No	Signing hot path, high-value
ML-DSA key	HSM	No	Identity key, long-lived
Pulsar share	HSM or TEE	No	Threshold PQ, latency-tolerant
TLS staking key	HSM	No	Node identity anchor
Evaluation keys (FHE)	Software	N/A	Public, large ($\sim$100 MB)
Bootstrap keys (FHE)	Software	N/A	Public, used by evaluator

Table 1: Key material placement. Private keys in HSM; public/evaluation keys in software.

2. **Ephemeral session keys**: Short-lived keys for peer connections, rotated every 60 seconds via SPIFFE SVIDs [6].

3. **Verification keys**: Public keys of other validators, used only for signature verification.

3 Side-Channel Considerations

3.1 BLS12-381

BLS signing involves a hash-to-curve operation and a scalar multiplication on the G1 group. The scalar multiplication is the side-channel-sensitive operation.

Attack	HSM Mitigation	Software Mitigation
Timing	Constant-time multiply	Constant-time library
Power analysis	Shielded enclosure	N/A
EM emanation	Faraday cage	N/A
Cache timing	No shared cache	Process isolation

Table 2: Side-channel mitigations for BLS signing.

HSMs provide physical shielding that software cannot replicate. For BLS signing at consensus speed (1000+ signatures/second), the HSM must support batched signing to avoid becoming a throughput bottleneck.

3.2 ML-DSA-65

ML-DSA signing involves polynomial arithmetic over $\mathbb{Z}_q[X]/(X^{256}+1)$. The NTT (Number Theoretic Transform) is the performance-critical path. Known side-channel vulnerabilities in lattice-based schemes include rejection sampling leakage and coefficient timing.

HSMs with ML-DSA support (FIPS 140-3 Level 3) are expected from 2026. In the interim, Lux uses TEE-based software signing with Intel TDX or AMD SEV-SNP, which provides memory encryption but not physical side-channel resistance.

4 Threshold + HSM Interaction

4.1 FROST-BLS with HSM Shares

FROST (Flexible Round-Optimized Schnorr Threshold) is used for BLS threshold signing [3]. Each validator holds a share sk_i inside its HSM. The FROST protocol operates as follows:

1: **Round 1**: Each HSM generates nonce pair (d_i, e_i) and publishes commitments.
2: **Round 2**: Coordinator distributes aggregated commitments and message m.
3: **Inside HSM**: Compute partial signature $z_i = d_i + e_i \cdot \rho_i + \lambda_i \cdot sk_i \cdot c$ where $c = H(R, \text{pk}, m)$.
4: **Outside HSM**: Aggregate partial signatures $\sigma = \sum z_i$.

The HSM never exposes sk_i. The partial signature z_i does not reveal the share (it is blinded by the ephemeral nonces d_i, e_i). The aggregation is performed in software because it operates only on public values.

4.2 Pulsar with TEE Shares

Corona threshold signatures use lattice-based shares [4]. The share combination involves polynomial addition over the ring $\mathbb{Z}_q[X]/(X^N + 1)$. Due to the larger share size ($\sim$1 KB vs. 32 B for FROST-BLS), HSM storage is a concern for large threshold sets. TEE-based execution (with attestation) is the preferred boundary for Corona shares.

Protocol	Share Boundary	Share Size	Signing Latency
FROST-BLS	HSM (PKCS#11)	32 B	< 5 ms
Pulsar	TEE (SEV-SNP)	$\sim$1 KB	~ 7 ms
ML-DSA (non-threshold)	HSM	4,032 B	< 1 ms

Table 3: Signing protocol boundaries and performance.

4.3 Defense-in-Depth Analysis

Theorem 1 (Threshold + HSM Defense)**.** *An adversary who compromises $< t$ HSMs and the software environment of $< t$ validators cannot forge a valid aggregate signature.*

Proof. Compromising an HSM yields the share sk_i stored within it. Compromising the software environment of a different validator j yields at most the partial signature z_j for a specific message (the share sk_j is inside j's HSM). With $< t$ shares recovered from HSM compromise and $< t$ partial signatures from software compromise, the adversary has fewer than t shares and cannot reconstruct the aggregate signing key. The threshold protocol's security guarantee (t-of-n unforgeability) holds. □

5 Attestation Model

TEE-hosted signers produce attestation reports that bind the signer's public key to the TEE measurement:

```
Attestation {
  tee_type:    "SEV-SNP" | "TDX" | "SGX"
  measurement: <sha384 of signer binary>
  public_key:  <BLS + ML-DSA public keys>
  nonce:       <challenge from verifier>
  signature:   <TEE-signed report>
}
```

Validators publish their attestation reports to Q-Chain. Delegators can verify that a validator's signer runs the expected binary in an attested TEE before delegating stake. This is complementary to the device identity layer in hybrid certificates [2].

6 When Hardware Does Not Help

Hardware protection is ineffective against:

1. **Authorized misuse**: The HSM signs whatever the authorized caller requests. If the caller is compromised and sends a malicious block for signing, the HSM complies. Mitigation: rate limiting and anomaly detection in the signer service [6].

2. **Firmware vulnerabilities**: HSM firmware is software running on a microcontroller. Vulnerabilities have been demonstrated in major HSM vendors. Mitigation: threshold cryptography ensures that a single HSM compromise is insufficient.

3. **Supply chain attacks**: A tampered HSM can exfiltrate keys during manufacturing. Mitigation: multi-vendor HSM deployment with threshold shares distributed across vendors.

7 Conclusion

The correct answer to "should we use an HSM?" is "for what, and combined with what?" HSMs protect individual key shares against extraction. Threshold protocols protect against individual HSM compromise. TEE attestation provides evidence that the signer software is unmodified. None of these mechanisms is sufficient alone; together, they provide a layered defense where an adversary must simultaneously compromise multiple HSMs across multiple vendors, defeat threshold cryptography, and bypass continuous authorization—a combination that is operationally infeasible.

References

[1] Kelling, Z. (2025). *Quasar: Quantum-Secure Multi-Engine Consensus with Triple-Proof Quantum Finality*. Lux Industries Research.

[2] Kelling, Z. (2024). *Hybrid Certificate Chains and Post-Quantum Trust Anchors for Validator Networks*. Lux Industries Research.

[3] Kelling, Z. (2021, revised 2025). *Threshold MPC: FROST Unforgeability, CGGMP21 UC-Security, and Taproot Integration*. Lux Industries Research.

[4] Lux Industries. *Corona: Two-Round Module-LWE Threshold Signatures*. Lux Industries, 2026. `github.com/luxfi/corona`. Module-LWE (Ringtail-derived), two-round.

[5] Kelling, Z. (2025). *FHE API and Developer Model for Practical Private Applications*. Lux Industries Research.

[6] Kelling, Z. (2025). *Zero-Trust Validator Operations and Service Identity*. Lux Industries Research.

[7] Kelling, Z. (2024). *TEE Computing Mesh for Attestable Validator Infrastructure*. Lux Industries Research.

[8] NIST (2024). *FIPS 140-3: Security Requirements for Cryptographic Modules*. National Institute of Standards and Technology.