Hybrid Post-Quantum Cryptographic Architecture for Blockchain Systems

Zach Kelling

Lux Industries

2024

Abstract

We present a hybrid cryptographic architecture that combines classical elliptic curve primitives (BLS12-381, ECDH on Curve25519) with NIST post-quantum standards (ML-DSA FIPS 204, ML-KEM FIPS 203) via dual-signature and key-combiner constructions. The core security property is *compositional resilience*: an attacker must break *both* the classical and post-quantum components simultaneously to compromise any authenticated message or key agreement. We prove this property formally under standard assumptions (co-CDH for BLS, Module-LWE for ML-DSA/ML-KEM) and show that the hybrid construction is strictly stronger than either component during the transition period when the relative security of lattice assumptions remains uncertain. The architecture is implemented as native EVM precompiles on the Lux C-Chain, adding 678 μs to signature generation (ML-DSA-65 keygen) and 1.2 KB to signature size (dual BLS + ML-DSA), while preserving backward compatibility with existing BLS-only validators. We provide concrete benchmarks, a migration schedule, and an analysis of the harvest-now-decrypt-later threat that motivates immediate hybrid deployment rather than a wait-and-see approach.

Keywords: hybrid cryptography, post-quantum, ML-DSA, ML-KEM, BLS, key combiner, blockchain

Contents

1 Introduction

The development of cryptographically relevant quantum computers (CRQCs) poses an existential threat to the elliptic curve cryptography underpinning every major blockchain. Shor's algorithm factors integers and computes discrete logarithms in polynomial time on a quantum computer, rendering ECDSA (secp256k1), EdDSA (Ed25519), and BLS (BLS12-381) insecure. NIST standardized three post-quantum algorithms in August 2024: ML-KEM (FIPS 203) [1], ML-DSA (FIPS 204) [2], and SLH-DSA (FIPS 205) [3]. These are believed secure against quantum adversaries, but their underlying lattice and hash-based assumptions lack the decades of cryptanalytic scrutiny enjoyed by elliptic curves.

This creates a dilemma. Deploying post-quantum algorithms alone risks relying on assumptions that may prove weaker than expected. Retaining classical algorithms alone leaves the system vulnerable to quantum attack. The *hybrid* approach— requiring both a classical and a post-quantum signature to validate, or combining both a classical and post-quantum key exchange to derive session keys—eliminates this dilemma entirely: security degrades only if *both* assumptions fail simultaneously.

We formalize this intuition for the blockchain setting. Our contributions:

1. A formal threat model for harvest-now-decrypt-later (HNDL) attacks against blockchain infrastructure (Section 2).
2. Dual-signature construction: BLS12-381 + ML-DSA-65, with a proof that compromise requires breaking both co-CDH and Module-LWE (Section 3).
3. Hybrid key exchange: ECDH (Curve25519) + ML-KEM-768, with a key combiner construction preserving IND-CCA2 security (Section 4).
4. Forward secrecy analysis for the hybrid construction (Section 5).
5. Lux precompile implementation and benchmarks (Section 6).

2 Threat Model

2.1 Adversary Classes

We consider three adversary classes, ordered by increasing capability:

Definition 2.1 (Classical adversary)**.** *A probabilistic polynomial-time (PPT) adversary with access to classical computation only. Cannot break co-CDH on BLS12-381 or Module-LWE with standard parameters.*

Definition 2.2 (Quantum adversary)**.** *An adversary with access to a fault-tolerant quantum computer running* $\geq$ 4,000 *logical qubits. Can execute Shor's algorithm to solve discrete logarithms on elliptic curves in polynomial time. Cannot break Module-LWE (no known quantum speedup beyond Grover's square root on search, which is addressed by parameter selection).*

Definition 2.3 (Transitional adversary)**.** *An adversary that may or may not have quantum capability, but is actively performing* harvest-now-decrypt-later *(HNDL) attacks: recording ciphertexts and signed messages today for retrospective analysis when quantum computers become available.*

2.2 Harvest-Now-Decrypt-Later

Blockchain is uniquely vulnerable to HNDL because:

1. **Public keys are exposed in the mempool.** Every transaction broadcast reveals the sender's public key before confirmation. An adversary recording mempool traffic accumulates a database of public keys paired with signed transactions.

2. **Validator public keys are static.** BLS public keys used for consensus are published in the validator set and do not rotate. A future quantum computer can forge validator signatures for any historical epoch.

3. **Chain history is immutable and public.** Unlike TLS sessions that are ephemeral, every signature ever produced on a blockchain is permanently recorded and publicly accessible.

4. **Economic incentives are persistent.** Assets secured by ECDSA/BLS signatures retain value indefinitely, making decade-old keys worth attacking.

2.3 Shor's Algorithm: Concrete Impact

Table 1: Impact of Shor's algorithm on blockchain cryptographic primitives.

Primitive	**Use**	**Attack**	**Broken?**
ECDSA (secp256k1)	Transaction signing	DLP on curve	Yes
BLS12-381	Consensus signatures	DLP on $\mathbb{G}_1$, $\mathbb{G}_2$	Yes
EdDSA (Ed25519)	Bridge/MPC signing	DLP on twisted Edwards	Yes
ECDH (Curve25519)	Key exchange	DLP on Montgomery	Yes
SHA-256	Merkle trees, addresses	Grover (2^{128} search)	No*
Keccak-256	EVM hashing	Grover (2^{128} search)	No*

*Grover's algorithm reduces security from 2^{256} to 2^{128}, which remains computationally infeasible.

2.4 Grover's Algorithm: Hash Function Impact

Grover's algorithm provides a quadratic speedup for unstructured search, reducing the security of an n-bit hash function to $n/2$ bits. For SHA-256 and Keccak-256, this means 128-bit quantum security—adequate by all current standards. Blockchain Merkle trees, address derivation, and proof-of-work (where applicable) remain secure against quantum adversaries with current hash function parameters.

The critical distinction: Shor's algorithm is *exponential* speedup (rendering DLP-based schemes completely broken), while Grover's is *quadratic* (manageable by doubling key lengths). Hybrid deployment addresses the former; hash function parameters address the latter.

3 Dual-Signature Construction

3.1 Construction

Let $\mathsf{BLS} = (\mathsf{BLS.KeyGen}, \mathsf{BLS.Sign}, \mathsf{BLS.Verify})$ be BLS12-381 signatures [4] and $\mathsf{PQ} = (\mathsf{PQ.KeyGen}, \mathsf{PQ.Sign}, \mathsf{PQ.V}$ be ML-DSA-65 [2].

Definition 3.1 (Hybrid signature scheme)**.** *The hybrid signature scheme* $\mathsf{Hyb} = (\mathsf{Hyb.KeyGen}, \mathsf{Hyb.Sign}, \mathsf{Hyb.Veri}$ *is defined as:*

- *Hyb.KeyGen*(1^λ)*: Run* $(pk_B, sk_B) \leftarrow$ *BLS.KeyGen*(1^λ) *and* $(pk_P, sk_P) \leftarrow$ *PQ.KeyGen*(1^λ)*. Output* $pk = (pk_B, pk_P)$ *and* $sk = (sk_B, sk_P)$*.*
- *Hyb.Sign*(sk, m)*: Compute* $\sigma_B \leftarrow$ *BLS.Sign*(sk_B, m) *and* $\sigma_P \leftarrow$ *PQ.Sign*(sk_P, m)*. Output* $\sigma = (\sigma_B, \sigma_P)$*.*
- *Hyb.Verify*(pk, m, σ)*: Parse* $\sigma = (\sigma_B, \sigma_P)$ *and* $pk = (pk_B, pk_P)$*. Accept iff* *BLS.Verify*$(pk_B, m, \sigma_B) = 1$ ***and*** *PQ.Verify*$(pk_P, m, \sigma_P) = 1$*.*

The AND-composition in verification is critical: the verifier rejects unless *both* signatures are valid.

3.2 Security Proof Sketch

Theorem 3.1 (Hybrid EUF-CMA security)**.** *If* BLS *is EUF-CMA secure under the co-CDH assumption in the random oracle model, and* PQ *is EUF-CMA secure under the Module-LWE assumption (NIST security level 3), then* Hyb *is EUF-CMA secure as long as at least one of the two assumptions holds.*

Proof sketch. Suppose adversary $\mathcal{A}$ forges a hybrid signature $\sigma^* = (\sigma_B^*, \sigma_P^*)$ on message m^*. By the AND-verification rule, both σ_B^* and σ_P^* must be valid. Therefore $\mathcal{A}$ has simultaneously:

1. Forged a BLS signature (breaking co-CDH), *and*
2. Forged an ML-DSA signature (breaking Module-LWE).

By contrapositive: if co-CDH holds, then σ_B^* cannot be forged, so the hybrid scheme is secure regardless of ML-DSA's status. If Module-LWE holds, then σ_P^* cannot be forged, so the hybrid scheme is secure regardless of BLS's status. The hybrid is therefore at least as secure as the stronger of the two components.

More formally, let ε_{Hyb} be the advantage of any PPT adversary against Hyb. Then:

$$\varepsilon_{\text{Hyb}} \leq \min(\varepsilon_{\text{BLS}}, \varepsilon_{\text{PQ}})$$

where ε_{BLS} and ε_{PQ} are the advantages against the individual schemes. This follows directly from the fact that a successful hybrid forgery implies successful forgery against *both* components. □

Remark 3.1. *The inequality* $\varepsilon_{Hyb} \leq \min(\varepsilon_{BLS}, \varepsilon_{PQ})$ *shows the hybrid is* strictly stronger *than either component in the transition period where we are uncertain which assumption is stronger. It can never be weaker.*

3.3 Aggregation Compatibility

BLS signatures support aggregation: n individual signatures on the same message compress to a single 48-byte aggregate signature. ML-DSA does not support aggregation natively. We address this asymmetry in the consensus layer:

1. **Per-block**: BLS aggregate signature (48 bytes) for fast verification.
2. **Per-epoch**: STARK proof compressing n ML-DSA signatures into ∼200 bytes, verified once per epoch.

This two-tier approach preserves BLS aggregation benefits while adding post-quantum security at the epoch boundary.

4 Hybrid Key Exchange

4.1 Construction

For node-to-node encrypted communication (P2P gossip, validator coordination), we deploy a hybrid key exchange combining X25519 (ECDH on Curve25519) with ML-KEM-768 (FIPS 203, NIST security level 3) [1].

Definition 4.1 (Hybrid KEM)**.** *Let* $\mathsf{X25519}$ *be Diffie-Hellman key exchange on Curve25519 and* MLKEM *be ML-KEM-768. The hybrid KEM is:*

- $\mathsf{HybKEM.KeyGen}()$*: Generate* $(pk_X, sk_X) \leftarrow \mathsf{X25519.KeyGen}()$ *and* $(pk_M, sk_M) \leftarrow \mathsf{MLKEM.KeyGen}()$*. Output* $pk = (pk_X, pk_M)$ *and* $sk = (sk_X, sk_M)$*.*
- $\mathsf{HybKEM.Encaps}(pk)$*: Compute* $(ct_X, ss_X) \leftarrow \mathsf{X25519.Encaps}(pk_X)$ *and* $(ct_M, ss_M) \leftarrow \mathsf{MLKEM.Encaps}(pk_M$*. Derive* $ss = \mathsf{HKDF}(ss_X \| ss_M \| ct_X \| ct_M)$*. Output* $(ct_X \| ct_M, ss)$*.*
- $\mathsf{HybKEM.Decaps}(sk, ct)$*: Parse* $ct = (ct_X, ct_M)$*. Compute* $ss_X \leftarrow \mathsf{X25519.Decaps}(sk_X, ct_X)$ *and* $ss_M \leftarrow \mathsf{MLKEM.Decaps}(sk_M, ct_M)$*. Derive* $ss = \mathsf{HKDF}(ss_X \| ss_M \| ct_X \| ct_M)$*. Output* ss*.*

4.2 Key Combiner Security

The HKDF-based combiner includes both shared secrets *and* both ciphertexts in the key derivation. Including ciphertexts binds the derived key to the specific exchange instance, preventing substitution attacks [7].

Theorem 4.1 (Hybrid KEM IND-CCA2 security)**.** HybKEM *is IND-CCA2 secure if at least one of X25519 or ML-KEM-768 is IND-CCA2 secure and HKDF is modeled as a random oracle.*

Proof sketch. Assume X25519 is broken (quantum adversary). Then ss_X is known to the adversary, but ss_M remains pseudorandom (ML-KEM is IND-CCA2). The combined key $ss = \mathsf{KDF}(ss_X \| ss_M \| \cdot)$ is pseudorandom because HKDF extracts entropy from ss_M regardless of the adversary's knowledge of ss_X. The symmetric argument holds if ML-KEM is broken but X25519 remains secure. □

5 Forward Secrecy

Classical forward secrecy relies on ephemeral Diffie-Hellman: even if long-term keys are compromised, past session keys remain secret. Quantum adversaries break this guarantee because they can recover ephemeral DH secrets from recorded transcripts.

The hybrid construction restores forward secrecy against transitional adversaries:

1. Each session generates ephemeral X25519 *and* ephemeral ML-KEM key pairs.
2. The session key is derived from both ephemeral shared secrets via HKDF.
3. After the session, both ephemeral private keys are destroyed.

A future quantum adversary who records the session transcript can recover the X25519 ephemeral secret but *cannot* recover the ML-KEM ephemeral secret (ML-KEM decapsulation requires the private key, which was destroyed). The session key remains secret because it depends on ss_M, which the adversary cannot compute.

Proposition 5.1 (Hybrid forward secrecy)**.** *If ephemeral ML-KEM private keys are securely erased after each session, the hybrid key exchange provides forward secrecy against quantum adversaries.*

This is the strongest forward secrecy guarantee available during the transition period: it does not require trusting that X25519 is secure (it is not, against quantum), only that ML-KEM is secure and ephemeral keys are erased.

6 Implementation

6.1 Precompile Addresses

The hybrid primitives are deployed as native EVM precompiles on the Lux C-Chain, activated at genesis.

Table 2: Lux hybrid cryptography precompile addresses.

Address	Primitive	Gas Cost
`0x0600...01`	BLS12-381 sign	45,000
`0x0600...02`	BLS12-381 verify	55,000
`0x0600...03`	BLS12-381 aggregate	12,000
`0x0600...10`	ML-DSA-65 keygen	120,000
`0x0600...11`	ML-DSA-65 sign	95,000
`0x0600...12`	ML-DSA-65 verify	80,000
`0x0600...20`	ML-KEM-768 keygen	65,000
`0x0600...21`	ML-KEM-768 encapsulate	50,000
`0x0600...22`	ML-KEM-768 decapsulate	50,000
`0x0600...30`	Hybrid sign (BLS + ML-DSA)	150,000
`0x0600...31`	Hybrid verify (BLS + ML-DSA)	145,000
`0x0600...40`	Hybrid KEM (X25519 + ML-KEM)	120,000

6.2 Benchmarks

All measurements on AMD EPYC 7763 (single core, 2.45 GHz), Go 1.22 with SIMD-optimized NTT from the `luxfi/lattice` library.

Table 3: Cryptographic operation benchmarks.

Operation	Latency	Output Size
BLS keygen	10 μs	48 B (pk)
BLS sign	28 μs	96 B
BLS verify	1.2 ms	—
BLS aggregate ($n = 100$)	42 μs	48 B
ML-DSA-65 keygen	678 μs	1,952 B (pk)
ML-DSA-65 sign	1.4 ms	3,309 B
ML-DSA-65 verify	620 μs	—
ML-KEM-768 keygen	182 μs	1,184 B (pk)
ML-KEM-768 encapsulate	215 μs	1,088 B (ct)
ML-KEM-768 decapsulate	198 μs	32 B (ss)
Hybrid sign (BLS + ML-DSA)	1.43 ms	3,405 B
Hybrid verify (BLS + ML-DSA)	1.82 ms	—
Hybrid KEM (X25519 + ML-KEM)	415 μs	1,120 B (ct)

6.3 BLS Same-Message Aggregate Verification

The Quasar consensus path verifies n BLS signatures over the same 32-byte certificate subject (a leader-broadcast pattern). cevm v0.45 introduced a same-message batched aggregate verifier; cevm v0.47.1 (commit `6bf0e232`) added a 2-way set-associative 4096-slot pubkey-decompression cache (FNV1a-64 hash, 48-byte memcmp guard). At n=1024 same-message verify lifts from 9.24× to **16.51**× versus the flat host-blst baseline (LP-137 Phase-3 roll-up; the $\geq 10\times$ target is exceeded by 65%). Pairing math itself remains the canonical `luxcpp/crypto/bls` c-abi body; the batching plus pubkey-cache amortization is what produces the published number. Production binaries link zero blst symbols (cevm v0.46.0 onward, CI-asserted).

6.4 Signature Size Analysis

The dominant cost of hybrid signatures is size, not computation. A pure BLS consensus signature is 48 bytes (aggregated). The hybrid construction adds ML-DSA overhead:

- Per-block: 48 B (BLS aggregate only). No ML-DSA per block.
- Per-epoch ($E = 100$ blocks): 48 B (BLS) + ∼200 B (STARK-compressed ML-DSA proof) = 248 B total.
- Annual storage at 1,000 epochs/day: $248 \times 1{,}000 \times 365 \approx 90$ MB/year for epoch proofs. Negligible.

6.5 Migration Path

The hybrid architecture supports a three-phase migration:

1. **Phase 1 (current)**: Validators generate both BLS and ML-DSA key pairs. Both are registered in the validator set. Blocks require BLS aggregate only; ML-DSA signatures are collected but not enforced.
2. **Phase 2 (enforcement)**: Epoch finality requires both BLS aggregate and STARK-compressed ML-DSA proof. Blocks still finalized by BLS alone for speed.
3. **Phase 3 (post-transition)**: If BLS is demonstrably broken, drop BLS and rely on ML-DSA/Corona only. The protocol supports this gracefully because the ML-DSA infrastructure is already deployed and tested.

7 Related Work

Hybrid constructions have been studied extensively in the TLS context. Stebila, Fluhrer, and Gueron [6] formalize hybrid key exchange for TLS 1.3. RFC 9180 (HPKE) [7] defines a KEM combiner that our construction follows. Bindel et al. [8] prove that hybrid signatures achieve the "best of both worlds" security guarantee we formalize in Theorem 3.1.

In the blockchain context, QRL [9] uses XMSS (stateful hash-based signatures) exclusively, providing post-quantum security but abandoning classical assumptions entirely. Ethereum's EIP-7562 [10] proposes account abstraction with custom signature verification but does not mandate post-quantum primitives. Our approach differs by making hybrid verification a protocol-level requirement, not an application-level option.

8 Conclusion

We have presented a hybrid post-quantum cryptographic architecture for blockchain systems that combines BLS12-381 with ML-DSA-65 for signatures and X25519 with ML-KEM-768 for key exchange. The architecture is provably at least as secure as the stronger of the two components, adds acceptable overhead (1.43 ms signing, 248 bytes/epoch), and supports a graceful migration path.

The key insight is that hybrid deployment is not a hedge—it is *strictly dominant*. During the transition period, we do not know whether lattice assumptions or elliptic curve assumptions are stronger. The hybrid construction is secure regardless of which assumption ultimately fails. The cost is modest. The benefit is existential. Immediate hybrid deployment is the only rational choice.

References

[1] NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard (FIPS 203)," National Institute of Standards and Technology, August 2024.

[2] NIST, "Module-Lattice-Based Digital Signature Standard (FIPS 204)," National Institute of Standards and Technology, August 2024.

[3] NIST, "Stateless Hash-Based Digital Signature Standard (FIPS 205)," National Institute of Standards and Technology, August 2024.

[4] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[5] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[6] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid Key Exchange in TLS 1.3," Internet-Draft, draft-ietf-tls-hybrid-design, 2020.

[7] R. Barnes, K. Bhargavan, B. Lipp, and C. Wood, "Hybrid Public Key Encryption," RFC 9180, IETF, February 2022.

[8] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange," *Post-Quantum Cryptography (PQCrypto)*, pp. 206–226, Springer, 2019.

[9] P. Waterland et al., "The Quantum Resistant Ledger," QRL Foundation, 2018.

[10] Ethereum Foundation, "EIP-7562: Account Abstraction Validation Scope Rules," Ethereum Improvement Proposals, 2023.

[11] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 212–219, 1996.

[12] M. Mosca, "Cybersecurity in an Era with Quantum Computers: Will We Be Ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.