Quantum Resistance in the QuantumCoin Blockchain

Authors: The QuantumCoin Community

Publication Date: October 2021

Revision 5: March 2026

Authors: The QuantumCoin Community Publication Date: October 2021 Revision 5: March 2026

Contents

Disclaimer

QuantumCoin is a community driven project. All visions and projects are aspirational. There is no value attributed to anything. All projects are community driven and there is no guarantee of delivery. QuantumCoin is not intended to be, or to be the subject of, an investment opportunity, investment contract, or security of any type.

Introduction

Public Key Cryptography (asymmetric cryptography) is essential for blockchains to secure accounts and to enable validations in Proof-of-Stake systems. Digital signatures, made possible by public key cryptography, allow the authenticity of blockchain transactions to be verified. Widely deployed digital signature schemes today — RSA and Elliptic Curve-based schemes such as ECDSA and EdDSA — are the foundation of Bitcoin, Ethereum, and most other blockchain protocols.

These classical schemes derive their security from the computational hardness of integer factorisation (RSA) and the hardness of the elliptic curve discrete logarithm problem (ECDLP) — the problem underlying ECDSA and EdDSA. Neither problem is believed to be hard for a sufficiently powerful quantum computer.

Quantum Computer Threat to Blockchains

With classical computers, deriving a private key from its corresponding public key is computationally infeasible for standardised schemes — requiring sub-exponential time for RSA (general number field sieve) and exponential time for ECDSA/EdDSA (Pollard's rho, achieving O(2^{n/2}) operations for an n-bit group order) under best-known classical attacks. A cryptographically relevant quantum computer (CRQC) changes this calculus fundamentally: Shor's algorithm solves both integer factorisation and the discrete logarithm problem in polynomial time on a quantum computer, rendering RSA, ECDSA, and EdDSA insecure.

An adversary with access to a CRQC could forge arbitrary transaction signatures, impersonate validators in Proof-of-Stake consensus, and rewrite the ledger by producing valid signatures for fabricated history. Unlike centralised systems that can temporarily shut down, re-key, and re-sign their data stores, blockchains are append-only and immutable-by-design: once the signature scheme underpinning a chain is broken, the authenticity of every past and future transaction is in question. This asymmetry — the inability to retroactively re-sign the ledger — makes blockchains uniquely vulnerable and motivates the adoption of post-quantum cryptography before a CRQC materialises. The point in time at which a CRQC becomes available is often referred to as Y2Q (Years to Quantum), by analogy with the Y2K problem.

In a Proof-of-Stake blockchain, the threat is amplified: in addition to forging account-holder signatures, an adversary can forge validator signatures, enabling double-spending, equivocation, and consensus subversion.

Beyond blockchains, the cryptographic foundations of internet security protocols such as TLS are equally at risk. Migrating a global TLS ecosystem — including legacy clients, IoT devices, and embedded systems — to post-quantum cipher suites will require years. A CRQC available before that migration is complete could undermine the confidentiality and authenticity of all internet communications.

Inter-Node Communication

Inter-node communication between blockchain nodes — validators, data archivers, relay nodes — is also at risk. A CRQC capable of breaking the key exchange and authentication layer of node-to-node channels can inspect, alter, delay, or selectively drop protocol packets (man-in-the-middle attacks). Even when individual transactions are signed before transmission, unsigned protocol-level metadata and control messages remain vulnerable.

Shor's Algorithm

Peter Shor's 1994 algorithm [2] solves the integer factorisation problem and the discrete logarithm problem in polynomial time on a quantum computer. This directly breaks RSA (which relies on the hardness of factoring) and ECDSA / EdDSA (which rely on the hardness of the elliptic curve discrete logarithm problem). Shor's algorithm is the principal motivation for the global migration to post-quantum cryptography.

Grover's Algorithm

Grover's algorithm [3] provides a square-root speedup in unstructured search on a quantum computer, reducing the effective bit-security of an n-bit work factor to approximately n/2 bits. In the context of Proof-of-Work blockchains, this translates to a quadratic advantage in mining hash computations for a single quantum processor. While the speedup is only quadratic — and diminishes when classical parallelism is available — a sufficiently large network of quantum computers could mount a 51% attack on Proof-of-Work chains or silently dominate block production.

QuantumCoin mitigates the Grover threat by employing a Proof-of-Stake consensus mechanism, eliminating the Proof-of-Work mining attack surface entirely.

Hybrid Post-Quantum Digital Signature Schemes

Design Rationale

NIST has standardised post-quantum digital signature algorithms — ML-DSA (FIPS 204, lattice-based) and SLH-DSA (FIPS 205, stateless hash-based) — yet these schemes have not been subjected to the decades of real-world cryptanalysis that classical schemes have undergone. It remains possible that novel classical or quantum attacks against lattice-based assumptions will emerge.

To hedge against this risk, QuantumCoin employs hybrid digital signature schemes that combine:

1. A lattice-based PQC component (ML-DSA) — conjectured to be resistant to both classical and quantum attacks.
2. A stateless hash-based PQC component (SLH-DSA) — whose security rests solely on the standard security properties of the underlying hash function family, providing defence-in-depth from a fundamentally different mathematical family.
3. A classical elliptic-curve component (Ed25519) — providing a hedge against the possibility that lattice-based or hash-based PQC is broken by classical means before a CRQC exists.

The hybrid construction ensures that all component verifications must succeed for a signature to be accepted. Consequently, to forge a composite signature an adversary must produce valid forgeries under every component verifier. If any single algorithm family (classical, lattice, or hash-based) remains secure, the composite signature remains unforgeable. This property follows from the security analysis of hybrid/composite signature constructions, which shows that a concatenation-based combiner requiring all component verifications to succeed retains at least the security of its strongest component under the EUF-CMA model (see IETF draft-ietf-lamps-pq-composite-sigs [18]).

This approach is consistent with NIST guidance on combining NIST-approved and post-quantum signature algorithms as a transition strategy to post-quantum cryptography (see NIST IR 8547 [13]).

NIST Conformance

The underlying cryptographic primitives are invoked exactly as specified by their respective NIST standards. No algorithm modifications or non-standard parameters are introduced. A legacy hybrid scheme built against pre-final draft versions of ML-DSA and SLH-DSA exists in the codebase but is deprecated and out of scope for this whitepaper.

Component	Specification	Standard
ML-DSA-44	Module-Lattice-Based Digital Signature Standard	FIPS 204 [5]
ML-DSA-87	Module-Lattice-Based Digital Signature Standard	FIPS 204 [5]
SLH-DSA-SHAKE-256f	Stateless Hash-Based Digital Signature Standard	FIPS 205 [6]
SLH-DSA-SHAKE-256s	Stateless Hash-Based Digital Signature Standard	FIPS 205 [6]
Ed25519	Digital Signature Standard (EdDSA), §§ 7.1–7.7	FIPS 186-5 [7]

The hybrid layer defines only:

- Concatenation of component public keys and component signatures into composite keys and signatures.
- A requirement that verification of all components must succeed for the composite signature to be valid.
- Context strings for domain separation in ML-DSA and SLH-DSA signing (see Domain Separation and Context Strings).

Component Algorithms

ML-DSA (Module-Lattice-Based Digital Signature Algorithm) is a lattice-based signature scheme standardised by NIST in FIPS 204. ML-DSA achieves SUF-CMA (strong unforgeability under chosen-message attack) security: EUF-CMA is proven in the random oracle model (ROM) via a reduction to the Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS) hard problems, and the stronger SUF-CMA property follows from the deterministic derivation of the commitment and challenge from the message and key material, together with the tightness of the verification equation. The mechanism of hashing the public key into the message-dependent hash input (FIPS 204 §3.3) provides what the academic literature terms beyond-unforgeability (BUFF) properties. Extensions to the quantum random oracle model (QROM) have been studied (Kiltz, Lyubashevsky, and Schaffner, EUROCRYPT 2018 — for a modified Dilithium variant with lossy public keys; Jackson, Miller, and Wang, 2024 [21]), though the QROM security analysis of the exact Fiat-Shamir with aborts construction used in ML-DSA remains an active research area. ML-DSA-44 targets NIST security category 2; ML-DSA-87 targets NIST security category 5.

SLH-DSA (Stateless Hash-Based Digital Signature Algorithm) is a stateless hash-based signature scheme standardised by NIST in FIPS 205. Its security relies on the standard security properties of the underlying hash function family (SHAKE-256) — rather than on number-theoretic or lattice assumptions — specifically pseudorandom function (PRF) security and second-preimage resistance of the hash primitives, together with the multi-target second-preimage resistance properties of the tweakable hash function constructions (T_l, F, H), the PRF-security properties of the keyed functions (PRF, PRF_msg), and the properties of the randomised message hash (H_msg), all built from those primitives. (The academic SPHINCS+ security analysis further decomposes these requirements into properties such as interleaved target subset resilience (ITSR) and undetectability; FIPS 205 formulates its security requirements using its own PRF and second-preimage-resistance framework, though the fundamental properties identified in the academic analysis remain necessary.)

SLH-DSA-SHAKE-256f is the "fast" parameter set; SLH-DSA-SHAKE-256s is the "small" parameter set offering smaller signatures at the cost of slower signing. Both target NIST security category 5 in the parameter sets used by QuantumCoin.

Ed25519 is an Edwards-curve Digital Signature Algorithm (EdDSA) instantiated over Edwards25519 (the twisted Edwards curve birationally equivalent to Curve25519), standardised in FIPS 186-5 §§ 7.1–7.7. It provides approximately 128 bits of classical security (FIPS 186-5 §7.1): Pollard's rho on the ~2^{252} group order requires ~2^{126} operations.

Key Usage and Identity

The composite key pair — comprising the Ed25519, ML-DSA, and SLH-DSA component keys — constitutes the signer's sole cryptographic identity on the blockchain. Individual component keys MUST NOT be reused in any other signing context — whether standalone, in a different hybrid combination, or across protocol boundaries. This constraint prevents cross-context attacks and preserves the security assumptions of the hybrid construction.

Scheme Overview

QuantumCoin defines two hybrid signature schemes with three scheme IDs:

Scheme ID	Scheme	Mode	Components
3	hybrid-ML-DSA-44-SLH-DSA-SHAKE-256f-Ed25519	Compact	Ed25519 + ML-DSA-44 (SLH-DSA key present, not signed)
4	hybrid-ML-DSA-44-SLH-DSA-SHAKE-256f-Ed25519	Full (break-glass)	Ed25519 + ML-DSA-44 + SLH-DSA-SHAKE-256f
5	hybrid-ML-DSA-87-SLH-DSA-SHAKE-256s-Ed25519	Full only	Ed25519 + ML-DSA-87 + SLH-DSA-SHAKE-256s (NIST Category 5)

Scheme 1: hybrid-ML-DSA-44-SLH-DSA-SHAKE-256f-Ed25519 (Scheme IDs 3 and 4)

This scheme combines ML-DSA-44, SLH-DSA-SHAKE-256f, and Ed25519. It supports two operating modes — compact and full — using the same key pair. The mode selection is a policy decision, not a key rollover event.

Public key layout (1,408 bytes):

Offset	Length (bytes)	Content
0	32	Ed25519 public key
32	1,312	ML-DSA-44 public key
1,344	64	SLH-DSA-SHAKE-256f public key

Private key layout (4,064 bytes):

Offset	Length (bytes)	Content
0	64	Ed25519 private key seed (32 B) with cached public key (32 B)
64	2,560	ML-DSA-44 secret key
2,624	1,312	ML-DSA-44 public key
3,936	128	SLH-DSA-SHAKE-256f secret key (includes public key)

Compact signature layout (2,518 bytes for a 32-byte message):

Offset	Length (bytes)	Content
0	1	Scheme ID (0x03)
1	1	Message length (32)
2	64	Ed25519 signature
66	2,420	ML-DSA-44 signature
2,486	32	Original message

Full signature layout (52,374 bytes for a 32-byte message):

Offset	Length (bytes)	Content
0	1	Scheme ID (0x04)
1	1	Message length (32)
2	64	Ed25519 signature
66	32	Original message
98	2,420	ML-DSA-44 signature
2,518	49,856	SLH-DSA-SHAKE-256f signature

Scheme 2: hybrid-ML-DSA-87-SLH-DSA-SHAKE-256s-Ed25519 (Scheme ID 5)

This scheme combines ML-DSA-87, SLH-DSA-SHAKE-256s, and Ed25519. Both PQC components target NIST security category 5. It operates in full mode only — every signature incorporates all three components, and verification requires all three to succeed. This scheme is suitable when maximum assurance is required and the larger key and signature sizes are acceptable (e.g. for high-value transactions or governance operations).

Public key layout (2,688 bytes):

Offset	Length (bytes)	Content
0	32	Ed25519 public key
32	2,592	ML-DSA-87 public key
2,624	64	SLH-DSA-SHAKE-256s public key

Private key layout (7,680 bytes):

Offset	Length (bytes)	Content
0	64	Ed25519 private key seed (32 B) with cached public key (32 B)
64	4,896	ML-DSA-87 secret key
4,960	2,592	ML-DSA-87 public key
7,552	128	SLH-DSA-SHAKE-256s secret key (includes public key)

Full signature layout (34,517 bytes for a 32-byte message):

Offset	Length (bytes)	Content
0	1	Scheme ID (0x05)
1	1	Message length (32)
2	64	Ed25519 signature
66	32	Original message
98	4,627	ML-DSA-87 signature
4,725	29,792	SLH-DSA-SHAKE-256s signature

Compact vs Full Mode and the Break-Glass Mechanism

Compact mode (Scheme ID 3) signs the message with ML-DSA-44 and Ed25519 only. The SLH-DSA public key is part of the composite public key and of address derivation, but is not used for signing in normal operation. This keeps on-chain signatures small (2,518 bytes) and verification fast (~9,980 verify ops/s on a single thread; see Size and Performance Summary below for hardware details).

Full mode (Scheme IDs 4 and 5) signs the message with all three component algorithms. For Scheme ID 4, verification throughput drops to ~290 ops/s and signature size increases to 52,374 bytes. For Scheme ID 5, verification throughput is ~470 ops/s and signature size is 34,517 bytes.

The transition from compact to full mode is a policy switch, not a key rollover. The same key pair is used in both modes. A blockchain may activate full mode via:

- Break-glass / emergency: When an imminent threat is detected (e.g. a CRQC announcement or a published break of ML-DSA), the chain governance or a hard-coded block height triggers a requirement for full-mode signatures. Since the SLH-DSA component was already bound to the composite key, the signer can produce new full-mode (Scheme ID 4) signatures over the same transaction data, providing SLH-DSA-backed attestations of those transactions using the same composite key pair.
- Regular full-mode signing: A chain may elect to require full mode for all transactions from genesis, trading throughput for maximum assurance.

The break-glass mechanism is particularly valuable because SLH-DSA's security rests on fundamentally different mathematical assumptions (hash function security) from ML-DSA's (module lattice hardness). Even if ML-DSA and Ed25519 are both broken — whether by quantum or classical means — full-mode signatures that include SLH-DSA components remain valid, and signers can re-sign previously compact-only transactions in full mode to provide SLH-DSA-backed authenticity guarantees for historical data.

Size and Performance Summary

Property	Scheme 1 Compact (ID 3)	Scheme 1 Full (ID 4)	Scheme 2 Full (ID 5)
Public key	1,408 B	1,408 B	2,688 B
Private key	4,064 B	4,064 B	7,680 B
Signature (32 B msg)	2,518 B	52,374 B	34,517 B
Verify ops/s	~9,980	~290	~470

Approximate single-threaded verify operations per second on a modern x86-64 processor (AMD Ryzen 7 5800X).

Signature Scheme Multiplexing

The QuantumCoin blockchain supports multiple signature schemes simultaneously. The first byte of every signature is the scheme ID, enabling verifiers and nodes to dispatch to the correct verification procedure and byte layout without out-of-band metadata or a separate type field. Unknown scheme IDs are rejected.

This design allows the blockchain to evolve its cryptographic posture over time. If a vulnerability is discovered in any scheme, the chain can introduce a replacement scheme with a new ID, and users can rotate their accounts to the new scheme with minimal protocol disruption.

Scheme ID	Scheme	Mode
3	hybrid-ML-DSA-44-SLH-DSA-SHAKE-256f-Ed25519	Compact
4	hybrid-ML-DSA-44-SLH-DSA-SHAKE-256f-Ed25519	Full / break-glass
5	hybrid-ML-DSA-87-SLH-DSA-SHAKE-256s-Ed25519	Full only

Domain Separation and Context Strings

To prevent cross-mode and cross-scheme signature reuse, ML-DSA and SLH-DSA signing operations employ context strings for domain separation:

- Compact mode (Scheme ID 3): ML-DSA-44 signs with context = 0x03 || pk_SLH-DSA (1-byte scheme ID concatenated with the 64-byte SLH-DSA public key, 65 bytes total). This binds the compact ML-DSA-44 signature to the specific SLH-DSA key in the composite, preventing a "sliding" substitution of the hash-based component. Ed25519 signs the message directly without a context parameter.
- Full mode (Scheme ID 4): Both ML-DSA-44 and SLH-DSA-SHAKE-256f sign with context = 0x04 (1-byte scheme ID). Ed25519 signs the message directly.
- Full mode (Scheme ID 5): Both ML-DSA-87 and SLH-DSA-SHAKE-256s sign with context = 0x05 (1-byte scheme ID). Ed25519 signs the message directly.

Hedged signing is used for ML-DSA and SLH-DSA in all modes: fresh random values are mixed with deterministic key-and-message-derived material via the rnd parameter (ML-DSA, FIPS 204) and the opt_rand parameter (SLH-DSA, FIPS 205), mitigating the impact of poor-quality or biased randomness.

Address Derivation

The account address is derived from the hash of the concatenated composite public key:

```
address_input = pk_Ed25519 || pk_ML-DSA || pk_SLH-DSA
address = Hash(address_input)
```

The hash function used for address derivation is Keccak-256, retained for legacy compatibility with Ethereum-derived tooling. Because the SLH-DSA public key is included in address derivation, any change to the hash-based component invalidates the address — even in compact mode where SLH-DSA does not participate in signing. This prevents substitution attacks against the dormant SLH-DSA key.

State Proof Signing

To strengthen the long-term quantum resistance of the blockchain ledger, validators sign the proposal packet every 4,096 blocks using full mode (Scheme ID 4 or 5). This means that SLH-DSA — whose security rests on well-studied hash function properties rather than lattice assumptions — periodically attests to the integrity of the chain. Even if ML-DSA and Ed25519 are broken at a future date, these periodic full-mode attestations anchor the blockchain's authenticity to hash-based cryptography.

Audit and Independent Verification

QuantumCoin provides tooling for independent audit and conformance verification of hybrid signatures. The audit interface supports:

- Signature parsing: Verifying a hybrid signature and extracting per-component public keys and signatures for independent re-verification against FIPS 204, FIPS 205, and FIPS 186-5.
- Component-level cross-checking: Reconstructing the signature from parsed components and verifying using both the composite hybrid verifier and each component's individual verifier, confirming correctness of context bindings and domain separation.

Auditors can decode the extracted components and pass them to independent implementations — such as PQClean [14], liboqs [15], or other algorithm-conformant reference implementations — to perform differential testing and conformance audits.

Guardrails

Side-Channel Resistance

Blockchain nodes must be online to participate in consensus, creating opportunities for timing-based side-channel attacks on signing operations. QuantumCoin requires that security-critical cryptographic operations are implemented using constant-time techniques to avoid secret-dependent timing behaviour. The risk is particularly acute in scenarios involving high-frequency signing (e.g. exchange hot wallets), where both remote timing attacks (e.g. HertzBleed [12]) and physical-proximity side-channel attacks must be considered. Constant-time execution and resistance to power-analysis attacks are required properties for any signing implementation deployed in the QuantumCoin ecosystem.

Key Rotation

Validators can rotate their keys to a new key pair within the same or a different signature scheme. Periodic key rotation is a general cryptographic hygiene best practice; in the QuantumCoin context it provides the additional benefit that if a novel attack is discovered against one scheme, affected accounts can migrate to a different scheme ID with minimal protocol impact.

Code Checkpoints

In the highly unlikely but non-zero probability event that all current PQC algorithms are compromised before users can rotate keys, the integrity of historical blocks is at risk. As a proactive hedge, the QuantumCoin client node software is periodically updated with hardcoded checkpoint hashes from random blocks, enabling runtime verification of ledger integrity. While not a complete defence, this measure limits the window of vulnerability for retroactive forgery of historical chain data.

Communication Security

QuantumCoin secures inter-node communication using a post-quantum hybrid authenticated key establishment and encrypted transport protocol (RLPx V2). The protocol structure closely follows the TLS 1.3 key schedule (RFC 8446 [17]), adapted to use post-quantum primitives for both key establishment and authentication, and with SHA3-256 substituted for SHA-256 throughout the HMAC and HKDF constructions. While KMAC (NIST SP 800-185) is the NIST-recommended keyed MAC for SHA-3, the protocol uses HMAC-SHA3-256 and HKDF-SHA3-256 to preserve structural alignment with the TLS 1.3 key schedule; HMAC instantiated with SHA3-256 remains a secure PRF — while the standard HMAC security proofs (Bellare, Canetti, and Krawczyk, 1996; strengthened by Bellare, 2006) target Merkle-Damgard hash functions, the Keccak sponge construction underlying SHA3-256 has been proven indifferentiable from a random oracle in the ideal permutation model (Bertoni, Daemen, Peeters, and Van Assche [20]), which implies that HMAC-SHA3-256 inherits the security guarantees of the generic HMAC construction within the ideal permutation model.

Key Establishment

Key establishment uses a hybrid construction combining X25519 ECDH key agreement and ML-KEM-768 key encapsulation (FIPS 203 [16]). The hybrid construction runs both X25519 (Curve25519 ECDH, providing approximately 128-bit classical security — ~2^{126} operations via Pollard's rho) and ML-KEM-768 (targeting NIST security category 3) in parallel and combines their shared secrets internally. An adversary must break both X25519 and ML-KEM-768 to recover the session shared secret, applying the same hedging rationale used in the signature layer. Ephemeral key pairs (both X25519 and ML-KEM-768) are generated per session, providing forward secrecy: assuming timely erasure of ephemeral key material, compromise of long-term keys does not expose past session traffic.

Handshake Authentication

Both peers authenticate via the hybrid Ed25519 + ML-DSA-44 signature scheme (compact mode), with the SLH-DSA-SHAKE-256f public key cryptographically bound into the signed data. The handshake is mutually authenticated: both the client and the server sign a running transcript hash, and the counterparty verifies the signature against the expected public key before deriving application keys.

Explicit key confirmation is provided via HMAC-SHA3-256 Finished messages (following TLS 1.3 § 4.4.4), ensuring that both peers have derived identical keying material and that the transcript has not been

tampered with. Finished values are compared using constant-time comparison to prevent timing side-channels.

Symmetric Encryption (Record Layer)

After handshake completion, all application data is encrypted under AES-256-GCM (with an untruncated 128-bit authentication tag) with HKDF-SHA3-256-derived directional keys and IVs. AES-256 is chosen because Grover's algorithm provides a square-root speedup in brute-force key search, reducing the bit-security of an n-bit symmetric cipher to approximately n/2 bits; AES-256 therefore retains approximately 128-bit security against quantum adversaries. Nonces are constructed per RFC 8446 § 5.3: the 12-byte IV is XORed with the 64-bit sequence counter zero-padded on the left to 12 bytes (96 bits) per direction, with an overflow guard that rejects the maximum sequence number to prevent nonce reuse. Separate key and IV pairs are maintained for each direction (client-to-server and server-to-client) in each epoch (handshake and application).

Each record header carries an explicit 32-byte Additional Authenticated Data (AAD) field. Before passing the record to AES-256-GCM, the receiver reconstructs the AAD from the header's version and record-length fields and rejects the record early if it does not match the transmitted header value. The reconstructed AAD is then supplied as the associated data input to AES-256-GCM decryption; the authentication tag cryptographically verifies the integrity of both the ciphertext and the AAD. The AAD uses a fixed content type regardless of the actual packet type, concealing whether a record contains handshake or application data.

Security Properties Summary

Property	Mechanism
Quantum-resistant key establishment	Hybrid X25519 ECDH key agreement + ML-KEM-768 key encapsulation
Quantum-resistant authentication	Hybrid Ed25519 + ML-DSA-44 signatures
Forward secrecy	Ephemeral key pairs (X25519 + ML-KEM-768) per session
Mutual authentication	Both parties sign the running transcript
Key confirmation	HMAC-SHA3-256 Finished messages (TLS 1.3 style)
Replay protection	Monotonic per-direction sequence numbers
Nonce reuse prevention	IV XOR sequence counter with overflow guard
Record integrity and confidentiality	AES-256-GCM AEAD (untruncated 128-bit tag)
Transcript binding	All signatures and Finished MACs cover the cumulative transcript hash
Traffic analysis resistance	Random padding on serialised messages
Allocation DoS mitigation	Record size limits enforced before reading ciphertext
Constant-time verification	Constant-time comparison for Finished messages

---

Conclusion

QuantumCoin employs hybrid post-quantum digital signature schemes that combine NIST-standardised lattice-based (ML-DSA, FIPS 204), hash-based (SLH-DSA, FIPS 205), and classical (Ed25519, FIPS

186-5) components. The underlying primitives are invoked without modification, and the hybrid construction requires all component verifications to succeed — ensuring that the scheme remains secure as long as any one component algorithm family is unbroken.

The compact/full mode architecture with a break-glass mechanism, periodic state proof signing with SLH-DSA, single-byte scheme ID multiplexing, context-string domain separation, and audit tooling for independent per-component verification together provide a defence-in-depth posture against both known and unforeseen threats from quantum and classical adversaries.

Combined with a TLS 1.3-style transport protocol secured by hybrid X25519 + ML-KEM-768 key establishment with forward secrecy, mutual authentication via hybrid post-quantum signatures, AES-256-GCM record encryption, key rotation, code checkpoints, and a Proof-of-Stake consensus that eliminates the Grover attack surface, QuantumCoin is designed to maintain cryptographic integrity through the transition to the post-quantum era.

References

1. QuantumCoin Vision Paper — https://quantumcoin.org/whitepapers/Quantum-Coin-Vision-Paper-latest.pdf
2. P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," 1994 — https://arxiv.org/pdf/quant-ph/9508027.pdf
3. L. Grover, "A fast quantum mechanical algorithm for database search," STOC 1996 — https://arxiv.org/abs/quant-ph/9605043
4. NIST Post-Quantum Cryptography Standardization — https://csrc.nist.gov/Projects/post-quantum-cryptography
5. NIST FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA) — https://doi.org/10.6028/NIST.FIPS.204
6. NIST FIPS 205: Stateless Hash-Based Digital Signature Standard (SLH-DSA) — https://doi.org/10.6028/NIST.FIPS.205
7. NIST FIPS 186-5: Digital Signature Standard (DSS) — https://doi.org/10.6028/NIST.FIPS.186-5
8. Ed25519: High-speed high-security signatures — https://ed25519.cr.yp.to/
9. CIRCL: Cloudflare Interoperable, Reusable Cryptographic Library (QuantumCoin fork) — https://github.com/quantumcoinproject/circl
10. Hybrid ML-DSA-44 + SLH-DSA-SHAKE-256f + Ed25519 (Scheme 1) — https://github.com/quantumcoinproject/circl/tree/main/sign/hybridedmldsaslhdsa
11. Hybrid ML-DSA-87 + SLH-DSA-SHAKE-256s + Ed25519 (Scheme 2) — https://github.com/quantumcoinproject/circl/tree/main/sign/hybridedmldsaslhdsa5
12. HertzBleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86 — https://www.hertzbleed.com
13. NIST IR 8547: Transition to Post-Quantum Cryptography Standards (Initial Public Draft, November 2024) — https://csrc.nist.gov/pubs/ir/8547/ipd
14. PQClean: Clean, portable, tested implementations of post-quantum cryptography — https://github.com/PQClean/PQClean
15. liboqs: Open Quantum Safe library — https://github.com/open-quantum-safe/liboqs
16. NIST FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM) — https://doi.org/10.6028/NIST.FIPS.203
17. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 — https://doi.org/10.17487/RFC8446
18. IETF draft-ietf-lamps-pq-composite-sigs: Composite ML-DSA for use in X.509 PKI and CMS (Internet-Draft, work in progress) — https://datatracker.ietf.org/doc/draft-ietf-lamps-pq-composite-sigs/
19. QuantumCoin Blockchain Data Availability Whitepaper — https://quantumcoin.org/whitepapers/Quantum-Coin-Blockchain-Data-Availability-Whitepaper.pdf
20. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "On the Indifferentiability of the Sponge Construction," EUROCRYPT 2008 — https://doi.org/10.1007/978-3-540-78967-3_11
21. S. Jackson, C. Miller, and D. Wang, "On the QROM Security of the Dilithium Signature Scheme," NIST, 2024 — https://doi.org/10.6028/NIST.IR.8528