

Post Quantum Blockchain Protocol White Paper

A Cryptographically Agile Proof of Stake Network

Revised Edition 2.2 Post Quantum PoS and Public Launch Design

10 October 2026

This paper proposes a new post-quantum blockchain protocol without assuming an existing codebase, test network, or software framework. In this proposal, “must” identifies a protocol requirement and “should” identifies an engineering recommendation. Security properties depend on the stated assumptions, refinement proofs, and implementation validation. This paper is not a test report for a deployed network.

Standards and research reviewed through 10 October 2026

Abstract

This paper proposes a proof-of-stake network built on standardized post-quantum cryptography and weighted Byzantine fault-tolerant consensus. It treats transaction authorization, validator voting, network identity, shared-key establishment, state commitments, and upgrade authorization as distinct trust paths. Versioned objects, coordinated activation, and verifiable migration connect these paths. The objective is to establish explicit cryptographic boundaries and executable evolution rules for a long-lived public ledger without introducing a new cryptographic primitive.

The baseline uses ML-DSA-87 for account and consensus signatures, 48-byte account identifiers, and 64-byte ledger commitments, with a standardized TLS hybrid key-agreement profile for transport. Stable account identities are separated from rotatable authorization policies. Transactions bind the chain identifier, policy epoch, sequence number, and expiry height. Validator sets change at previously committed boundaries. Consensus adopts the locking and voting mechanisms defined in existing Tendermint research. Its safety assumptions require Byzantine voting power to remain strictly below one third in each relevant set, while a commit certificate covers strictly more than two thirds of the voting power.

The protocol accounts for large signatures through separate budgets for bytes, verification, execution, and state growth. It distinguishes final transaction inclusion, successful business execution, and the availability of verifiable state proofs. Governance manages changes through delayed activation, separated authority, and security constraints that ordinary parameter proposals cannot remove. The paper also defines economic conservation relationships, light-client trust boundaries, cryptographic migration, experimental methods, and release conditions so that independent reviewers can assess design completeness, implementation consistency, and reproducibility.

The cumulative-issuance cap is at most 10 million tokens with nine decimal places. The review configuration allocates 75% to public initial distribution and 25% to long-term PoS security issuance. Founders, contributors, advisers, institutions and foundations receive no reserved free allocation, and the protocol levies no developer fee. A uniform-price initial subscription is capped by a disclosed cost budget, with proportional allocation and refunds on oversubscription. BTC receipt, custody and genesis delivery have separately disclosed risks. Later BTC purchases are market trades, without a fixed exchange rate, BTC reserve or redemption promise. Issuance must pass both dilution and validator-cost evaluation; it does not establish prices or returns.

The references are primary materials from standards bodies and researchers, including university research. They supply methods and parameter foundations; citation does not imply participation, audit, or endorsement by the authors or institutions. Throughput, yields, and security certifications without supporting evidence are not presented as achievements.

Keywords post-quantum cryptography Byzantine fault tolerance proof of stake cryptographic agility verifiable state

Contents

01 Design Objectives and System Architecture	4
02 Quantum Threats and Security Objectives	5
03 Cryptographic Suite and Encoding	6
04 Identity and Key Lifecycle	8
05 Protocol Objects and Ledger Commitments	9
06 Consensus and Finality	12
07 State Machine and Transaction Lifecycle	13
08 Light Clients and Cross-Chain Boundaries	14
09 Resource and Performance Cost Model	16
10 Native Asset Economics and Governance	17
11 Public Launch and BTC Purchase Path	22
12 Cryptographic Evolution and Protocol Upgrades	24
13 Validation Plan and Release Conditions	26
14 Research Roadmap and Protocol Scope	28
Appendix A Baseline Profile and Parameter Registry	29
Appendix B Security Claims and Proof Obligations	31
Appendix C Terms and Symbols	32
References	33

01 Design Objectives and System Architecture

1 1 The Protocol Problem

A public blockchain must preserve authorization, finality, and historical verification over many years. Once public keys enter the ledger or consensus messages, they can remain exposed indefinitely. Dependence on elliptic-curve signatures therefore cannot be removed solely through a wallet upgrade: validators, light clients, node authentication, and software release roots also need migration paths. This paper treats cryptographic replacement as a protocol lifecycle problem and places the rules, costs, and trust changes of each replacement within one system model. [1][2]

The intended readers include BTC holders seeking to reduce long-term quantum authorization risk. This protocol offers an independent native asset for voluntary participation after considering its consensus, price, liquidity and delivery risks. Bitcoin market capitalization, holder count and historical credibility do not automatically transfer with a trade. No exchange listing, liquidity or institutional adoption is committed. Bitcoin quantum-risk mitigation and migration proposals continue to evolve; BIP 360 and BIP 361 were drafts when reviewed, so inability to upgrade Bitcoin is not assumed. [28][29]

The target network permits public transaction submission and independent full-node operation. Stake determines voting power in a bounded active validator set. The initial application layer supports native transfers, stake delegation, withdrawals, authorization-policy updates, and protocol governance. General-purpose smart contracts, private transactions, and trust-minimized external bridges are outside the first execution profile. Subsequent specifications must define their verification and resource boundaries; support for account signatures does not automatically secure those extensions.

1 2 Architecture and Responsibilities

A wallet constructs a deterministically encoded transaction intent and signs it under the account policy. Network admission limits length and rate before authorization and resource checks. The proposer selects ordered transactions; validators check complete block data and application validity before participating in consensus. After commitment, the deterministic state machine computes balances, stake, authorization, and governance state. The next finalized header commits to the state and receipts resulting from this height. Proof services derive independently verifiable query results from those commitments. Each layer must preserve explicit error semantics: RPC acceptance does not establish successful transfer.

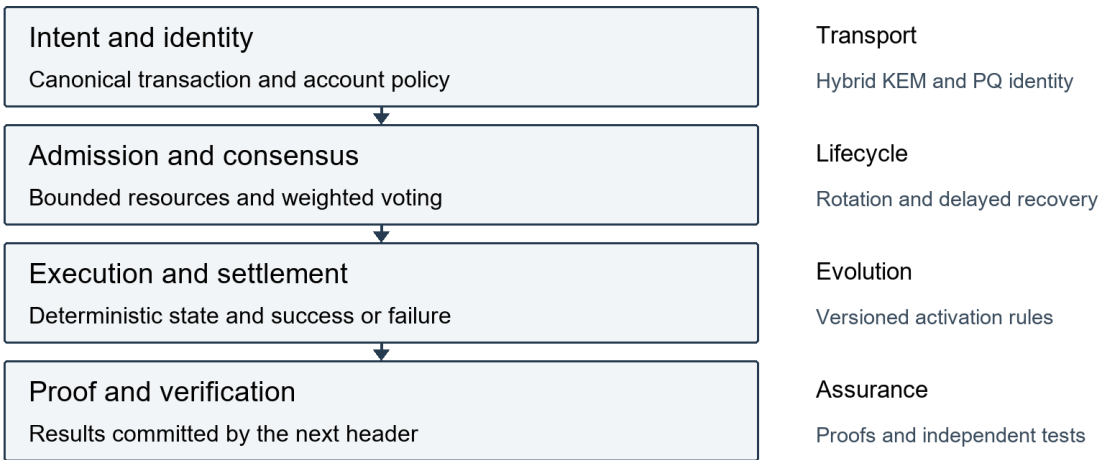


Figure 1 Principal protocol trust paths. Execution results at height h are committed by the finalized header at height $h+1$.

Full nodes independently verify consensus and execution. Validators additionally hold consensus keys bound to their staked identities. Light clients verify headers, set changes, and proofs after a trusted checkpoint. A node's TLS identity authenticates a connection peer rather than granting voting eligibility. A consensus signature authenticates a validator vote rather than establishing transaction success. A state proof authenticates a value under a commitment rather than guaranteeing continued service availability. These responsibilities allow each path to change algorithms with separately assessable consequences.

1 3 Design Principles and Review Questions

The security priorities are non-bypassable authorization, consensus-state consistency, conservation of assets and fees, recoverability, and continued progress under stated conditions. Performance optimizations must preserve these invariants. Every material design decision should identify its assumptions, the property it claims, an observable counterexample, and how old history remains verifiable after an upgrade. This white paper supplies the protocol structure and supporting rationale. Byte-level specifications, test vectors, and refinement proofs are companion materials that must be frozen before implementation can be accepted.

02 Quantum Threats and Security Objectives

2 1 Threat Model

The protocol targets a permanently public ledger and an open network. An adversary may collect public keys, transactions, and handshake transcripts; submit malicious messages; replay or modify traffic; and control some participants. The quantum threat model additionally permits access to a future fault-tolerant quantum computer. Compromised key devices, failed randomness sources, and validator corruption beyond the consensus fault bound are implementation, operational, and consensus risks, respectively; replacing the signature algorithm alone cannot address them.

Shor's algorithm provides quantum polynomial-time algorithms for integer factorization and discrete logarithms. RSA and conventional finite-field and elliptic-curve public-key mechanisms therefore cannot provide long-term post-quantum authorization. Grover's algorithm instead gives a quadratic query speedup for unstructured search. This differs from Shor's structural attack, and its query complexity does not directly determine practical attack time. [1][3]

Quantum resource studies for elliptic curves show that logical circuits, error-correction overhead, physical error rates, and the available attack time all affect resource requirements. Such estimates do not predict the year of a quantum break, and the protocol does not assume a particular arrival date. Network confidentiality must also address the risk of collecting ciphertext now for later decryption; the public ledger itself does not promise transaction privacy. [4][5]

2 2 Layered Security Objectives

The cryptographic layer requires that unauthorized parties cannot produce accepted account or consensus signatures, that commitments bind unambiguously to canonical objects, that communication provides both key establishment and peer authentication, and that keys and algorithms can migrate through controlled procedures. Its security arguments depend on the selected algorithms' computational assumptions, their composition, and implementation conditions. Consensus safety and liveness, correct state execution, and governance authority require separate arguments.

The design selects category 5 public-key parameters for additional resource margin while accepting larger bandwidth and storage costs. NIST categories compare specified attack resources; they do not guarantee a fixed bit-security level for the entire blockchain. Hash preimage resistance, collision resistance, signature forgery, and protocol attacks are likewise not interchangeable measures. Selecting FIPS algorithms does not itself constitute cryptographic-module validation, an independent audit, or a security proof of the complete protocol. [6][7][8][9]

03 Cryptographic Suite and Encoding

3 1 Baseline Algorithms and Costs

Account authorization, consensus votes, and online node identities use ML-DSA-87, with independent keys for each role. Account identifiers use SHA3-384; transaction, block, state, and evidence commitments use SHA3-512. The baseline introduces neither a custom signature-aggregation scheme nor a lattice-based threshold-signature scheme. Multi-party authorization uses a collection of complete, independently verifiable signatures. [10][7]

Purpose	Algorithm	Standard Encoding Length
Online signatures	ML-DSA-87	Public key 2592 B; expanded private key 4896 B; signature 4627 B
Account identifiers	SHA3-384	48 B

Purpose	Algorithm	Standard Encoding Length
Ledger commitments	SHA3-512	64 B
Transport KEM component	ML-KEM-1024	Public key 1568 B; private key 3168 B; ciphertext 1568 B; shared secret 32 B
Optional offline signatures	SLH-DSA-SHAKE-256s	Public key 64 B; private key 128 B; signature 29792 B

Table 1 Cryptographic Suite and Encoding.

All three public-key parameter sets in the table are in NIST category 5. An ML-DSA 32 B key-generation seed and its 4896 B expanded private key are different storage representations; private-key backups must identify the format, algorithm, and version. Public keys, signatures, and KEM inputs undergo the standard's length and encoding checks. Pre-standard Dilithium or Kyber encodings are not accepted as FIPS objects. [6][7][8]

3 2 Canonicalization and Domain Separation

Consensus encoding C, also denoted ENC, is a restricted deterministic CBOR profile under RFC 8949 Section 4.2.1. It uses only schema-ordered definite-length arrays, unsigned integers, and byte strings, with shortest-form integer encodings. Floating-point numbers, indefinite-length items, unknown tags, and noncanonical representations are forbidden. Arrays with set semantics follow their schema's ordering and reject duplicates; purpose tags are ASCII byte strings. Account transactions sign the version, suite_id, auth_mode, G, auth_epoch, nonce, and expiry. Votes bind the applicable set epoch, height, round, step, block, and the signed timestamp used for BFT Time. Parsers check length, range, and cardinality bounds before signature verification. [11]

Ordinary outer transactions, including ordinary rotations, use pure ML-DSA over the complete C(body) with context TX-v1; votes use VOTE-v1. Independent new-key possession proofs and ancillary key-management records use KEY-v1 and bind G, the account, the new-policy digest, and the activation height. Recovery-specific envelopes use RECOVER-v1. These contexts are fixed ASCII byte strings; no custom prehash-signature variant is introduced. Signers use hedged mode by default. Hash inputs are C-encoded arrays containing a purpose tag, with distinct tags for Merkle leaves, internal nodes, and the empty tree. [7]

3 3 Transport Profile and Standards Status

Node transport specifies TLS 1.3, TLS_AES_256_GCM_SHA384, and SecP384r1MLKEM1024 from RFC 10024, preserving the prescribed key shares, combination order, and TLS key schedule. SHA3 must not replace SHA384 inside the handshake. This hybrid group became a Proposed Standard in August 2026, but its IANA Recommended value is N. It is an explicit protocol-profile choice requiring separate interoperability and resource-cost verification. [12][13]

Hybrid key establishment does not by itself provide post-quantum identity authentication. As of 2026-10-10, RFC 9881 specifies ML-DSA X.509 encodings, while TLS usage remains draft-ietf-tls-mldsa-06. The candidate mutual-authentication profile pins that revision and requires interoperability

testing and a standards review before release. For TLS CertificateVerify, the FIPS context is empty as required by the draft; on-chain contexts must not be reused. Failed negotiation terminates the connection, with no silent fallback to classical authentication. State-changing requests disable 0-RTT. [14][15] Keys for known node roles must come from independently trusted configuration or verified on-chain registration. A self-authenticating node identifier may prove key possession but grants no trusted role. If a CA is used, the complete authentication-signature path and independent trust anchor must satisfy the post-quantum identity profile; a classical certificate-signature path cannot authorize a new trusted node identity.

04 Identity and Key Lifecycle

4 1 Stable Accounts and Authorization Policies

The initial account identifier is $A = \text{SHA3-384}(\text{C}([\text{ACCOUNT-v1}, G, P0]))$, where $G = \text{SHA3-512}(\text{C}(\text{Genesis}))$ and Genesis excludes the derived field G . $P0$ is the complete initial public-key policy, including algorithm identifiers, public keys, the authorization threshold, and any enabled recovery configuration. Creation supplies $P0$ and the corresponding proof of key possession, recomputes A , and validates the policy; creation is rejected if the same A is bound to a different $P0$. Account state stores the current policy and a monotonically increasing `auth_epoch`. Rotation leaves A unchanged.

Account keys, validator voting keys, node TLS identities, and recovery keys are separate. Multi-signature authorization counts distinct public keys; duplicates cannot increase weight. Ordinary policy updates satisfy the current online policy, whereas recovery updates satisfy the registered recovery policy; both prove possession of the new keys. The wire encoding of `auth_mode` uses integer values 0 for ordinary and 1 for recovery. TLS identities authenticate communication peers; account authorization and validator eligibility must additionally match the on-chain policy and current validator set.

4 2 Generation Custody and Rotation

Key generation uses a cryptographic randomness source meeting the selected parameter set's requirements. The baseline generates independent random seeds for independent roles and does not transplant conventional elliptic-curve HD derivation into ML-DSA. Private keys have encrypted, versioned backups and undergo offline restoration exercises that compare the restored public key with its registered value. Logs, diagnostic interfaces, and on-chain objects must not contain private keys or recovery seeds.

A rotation record binds the old `auth_epoch`, new policy, and activation height. Activation rechecks the old online epoch and increments it atomically, after which old-epoch transactions are invalid. Ordinary rotations cannot activate during a pending recovery; successful recovery invalidates scheduled rotations bound to an old epoch. Historical policies remain available to verify confirmed transactions and consensus evidence; rotation cannot erase existing accountability. Ephemeral KEM keys are freshly generated for each handshake and destroyed promptly after use. Signing and decapsulation implementations address side channels, faults, and secret erasure, avoiding

observable secret-dependent branches. [9] At each height, deterministic processing activates due recoveries first, then ordinary rotations that still match the old epoch and are not frozen, before executing the block transactions.

4 3 Optional Offline Recovery and Cryptographic Migration

An account may preregister offline SLH-DSA-SHAKE-256s recovery keys and a k-of-n policy. Recovery uses pure SLH-DSA with RECOVER-v1 and collects k distinct complete signatures over the full C(body), without requiring the old online signature. Only recovery, cancellation, and explicitly defined recovery-policy administration operations are permitted; ordinary transfers cannot be mixed into the envelope. In recovery mode, the public auth_epoch and nonce fields match recovery_epoch and the independent recovery sequence; in ordinary mode, they match the online-policy version and online sequence. Recovery binds the new policy and key-possession proof and activates after D_recovery. An account without enrolled recovery has no protocol-level universal recovery key. [8]

Ordinary rotation cannot change an enabled recovery configuration; recovery-policy administration follows the recovery allowlist and increments recovery_epoch. Ordinary rotation does not reset the independent recovery sequence. Each account has at most one pending recovery, during which the recovery policy is locked: no separate recovery-key or threshold update is permitted before cancellation or scheduled activation. Submission and cancellation each consume the next recovery sequence. Cancellation signs the pending-request digest and releases the freeze without changing the online policy. Ordinary online-authorized policy changes and debits are frozen, while recovery fees signed by the recovery threshold and mandatory slashing remain enforceable. D_recovery is measured in blocks, fixed at confirmation, and cannot be shortened retroactively. Automatic activation does not consume another sequence. Success increments the online auth_epoch and releases the freeze. The recovery threshold holds final replacement authority granted in advance by the user.

The protocol explicitly manages algorithm identifiers, policy versions, and activation heights. A new suite first publishes its specification and test vectors; governance then establishes a migration window before new authorizations under the old suite cease. Nodes must not skip verification for an unknown algorithm. If the old algorithm is no longer secure, migration cannot rely solely on an old signature to establish legitimacy; it requires a previously authorized alternative path or emergency governance. This provides cryptographic agility without promising automatic remediation of unknown cryptanalytic breakthroughs. [2]

05 Protocol Objects and Ledger Commitments

5 1 Deterministic Encoding and Chain Identity

Let C(x) denote a restricted CBOR encoding satisfying the core deterministic encoding requirements in RFC 8949 Section 4.2.1. Consensus objects use fixed-length arrays, unsigned integers, and byte strings. Floating-point values, indefinite-length items, duplicate map keys, and undefined tags are prohibited. Domain labels are fixed ASCII byte strings. The structural version determines each field's position and type; unknown versions and non-minimal integer encodings must be rejected. Amounts,

sequence numbers, and heights are unsigned 64-bit integers. Intermediate arithmetic must use sufficient width and check the stored range without overflow or wraparound. [11]

The genesis manifest contains the protocol version, initial assets and authorization records, validator set, parameter table, time rules, and governance authorities. The chain identifier G is the SHA3-512 digest of its deterministic encoding. The manifest excludes G and fields derived from G , avoiding a circular definition. References to assets, authorities, and validators within the manifest use manifest indexes or labels independent of G , rather than account identifiers not yet derived. Genesis accounts are represented by initial policies and balances; identifiers are derived after G is known, followed by the initial state root. Transactions, votes, proofs, and upgrade objects must bind G . Matching display names do not establish matching chain identities. Genesis also binds final public-subscription accepted amounts and allocations, payment-verification anchors and the initial voluntary stake set, using initial policies and indexes independent of G .

5 2 Transaction Intent and Authorization Envelope

Field	Type	Constraint
version, suite_id, and chain_id	Integers and 64 bytes	Identify structure, suite, and target chain
expiry_height and sender_id	Integer and 48 bytes	Bound validity and identify account
auth_epoch and nonce	64-bit integers	Match the mode-specific policy epoch and independent sequence
fee_cap and compute_cap	64-bit integers	Bound total fee plus new bond debits and execution
storage_cap and messages	Integer and message array	Bound state growth and order operations
tip	64-bit integer	Signed voluntary validator-pool tip; no inclusion-priority guarantee
auth_mode	Mode integer	Ordinary = 0; enrolled recovery = 1

Table 2 Thirteen fixed fields in the transaction body. The complete envelope is [body, authorization], with authorization satisfying the account's current policy.

Ordinary transactions sign $C(\text{body})$ using pure ML-DSA with the TX-v1 context. Enrolled recovery requests, cancellations, and recovery-configuration management use the recovery policy and pure SLH-DSA with RECOVER-v1, without requiring the old online key. Activation after the delay follows the committed request automatically and does not consume the recovery sequence again.

auth_mode, suite_id, and all body fields are signed. auth_epoch denotes the online-policy or recovery-policy epoch according to the mode; nonce selects the independent online or recovery sequence. Recovery mode permits only the recovery operations enumerated by the specification. Authorization records algorithms, key identifiers, and signatures in policy-defined order, counting no key twice. Policy resolution uses sequential state immediately before the transaction, so in-block

rotation, recovery freezes, and later transactions affect validity according to order. G, policy epochs, sequences, and expiry jointly separate replay contexts.

The intent identifier is $\text{SHA3-512}(\text{C}([\text{domain label}, \text{body}]))$ and identifies the same signed business intent. A separate envelope digest covers all bytes, including authorization. Randomized signatures can produce different envelopes for the same body, so the block transaction root must commit to complete envelopes rather than intent identifiers alone. Receipts are located by intent identifier, height, and transaction index, with sequence consumption preventing repeated execution of an intent.

5 3 Block Headers and Proof Objects

A header commits at minimum to the version, G, height, epoch, parent identifier, current and next-height validator-set roots, the previous height's commit-certificate root, the root of current complete transaction envelopes, the previous height's receipt root, the post-execution state root for the previous height, the parameter root, and protocol time. At a set transition, the previous certificate is still verified under the previous height's set. Validator leaves cover identity, algorithm, public key, and voting power, with deterministic ordering and duplicate rejection. The block identifier is SHA3-512 over the complete deterministic header. Consensus votes additionally bind height, round, step, and that identifier, preventing reinterpretation across rounds or steps. `next_validator_root` identifies only the set effective at the next height: it equals the current root within an epoch and identifies the new set only at the final height before the boundary. A future approved manifest is retained separately in state commitments. A precommit timestamp used by BFT Time is part of the signed complete $\text{C}(\text{vote})$ and certificate commitment, represented as integer UTC milliseconds since the Unix epoch; the corresponding reserved field is zero in other steps.

Commitment trees use distinct domain labels for empty trees, leaves, and internal nodes, committing to node type, subtree length, and left-right order to remove structural ambiguity. The first profile uses a versioned sparse Merkle state tree with 512-bit state-key paths. Leaves commit to the complete key, value, and namespace. A proof carries the tree version, key, value or absence claim, and a path represented by a bitmap and sibling digests. Verifiers bound path depth and total bytes. Exact node encodings and empty-subtree constants must be frozen with test vectors; a tree-version change requires a protocol upgrade.

Index services may accelerate queries, but a verifier relies on finalized headers, valid set proofs, and matching state proofs. A transaction root authenticates ordered envelopes rather than independently proving application results. Receipts cover status, actual fees, resource use, and event digests. A query about post-execution state or receipts at height h must state that its proof is anchored in the finalized header at $h+1$; an envelope-membership proof for height h can anchor directly in its finalized header, avoiding premature treatment of uncommitted execution results as verifiable facts.

06 Consensus and Finality

6 1 Baseline and Fault Model

The protocol proposes a Tendermint-style weighted BFT design, using Algorithm 1 in Buchman, Kwon, and Milosevic's 2018 paper as the starting point for consensus refinement. It does not introduce a new consensus algorithm. [16] The validator set is fixed at each height. With total voting power W and Byzantine voting power B , the assumption is $B < W/3$; a valid quorum must strictly satisfy $Q > 2W/3$. Exactly two thirds is insufficient. Voting power arithmetic must use bounded integers with overflow checks.

$$B < W/3 \quad Q > 2W/3$$

Safety additionally depends on secure signatures and commitments, correct locking and deterministic validation by honest nodes, and preservation of signing state through recovery. Liveness uses partial synchrony: after an unknown global stabilization time, communication among honest nodes becomes reliable and bounded, with sufficient honest voting power online and suitable timeouts. [17] A partition can halt the chain. The protocol must not lower its threshold when validators are offline. Block finality does not guarantee timely inclusion of every transaction.

6 2 Voting, Locking, and Round Advancement

Each round contains proposal, prevote, and precommit steps, with proposers selected by a deterministic weighted rotation. Nodes maintain `lockedValue/lockedRound` and `validValue/validRound`. A locked node may prevote its locked block, or vote for another block only after verifying a quorum of $>2W/3$ prevotes for that block in the proposal's claimed earlier round, subject to `validRound` $\geq$ `lockedRound` and `validRound` $<$ the current round. A current-round prevote quorum for a block causes locking and precommit. A commit certificate requires $>2W/3$ precommits for the same block, height, and round.

The historical prevote proof above is the polka in this design. The complete state machine must preserve the baseline algorithm's rules for nil votes, timeouts, and catching up to later rounds; a timeout alone must not clear a lock. Vote signatures bind the chain identifier, epoch, height, round, step, block identifier, and precommit timestamp, with purpose separation for ML-DSA-87. Lock and valid-round state must be durable or safely reconstructed from a log that cannot be rolled back. The signer durably records its protection against double signing before releasing a signature. Recovery or device replacement must not roll back that record. [16]

6 3 Epoch Handoffs and Resource Limits

The validator set is frozen within an epoch. The next-set manifest contains canonically ordered full public keys, voting powers, the algorithm version, and an activation height, and must be finalized in advance by the old set. The announcement lead time and epoch length are chain parameters. If no valid manifest is finalized by the deadline, the next epoch inherits the current set under the same deterministic rule at every node. Only the new set votes from the activation height onward; the preceding height's certificate is still verified under the old set. Incoming validators must synchronize

in advance and must not change the activation height when unready. Each set must satisfy the fault assumption. A proof for one height does not directly establish correctness under arbitrary reconfiguration.

Commit certificates retain individual ML-DSA-87 signatures; the baseline assumes no available signature aggregation. Coordinated limits are required for validator count, block bytes, certificate size, verification queues, and traffic per connection, with measurements covering multiple rounds and recovery. Before release, the design requires refinement arguments, model checking, and fault tests for weighted thresholds, persistent locks, epoch handoffs, and the cryptographic substitution. Citing an existing proof does not prove this protocol or an implementation, and supplies no measured TPS.

07 State Machine and Transaction Lifecycle

7 1 Deterministic State Transitions

The state machine produces identical results from the same initial state and ordered inputs. [18] Transactions use canonical encoding. Their signatures bind the protocol version, chain identifier, account, policy version, nonce, validity period, compute_cap, storage_cap, fee cap, and complete business messages. Blocks execute in a deterministic order. Transaction and state commitments use SHA3-512. The specification must define integer widths, rounding, error codes, resource metering, and storage ordering. Execution must not read local wall clocks, external networks, or uncommitted randomness.

Execution takes time only from the block's time field after consensus validation. The baseline adopts a pinned version of BFT Time: a voting-power-weighted median of signed timestamps in the preceding height's commit certificate, with Genesis time used for the first block. Validation checks monotonicity and specifies the assumed error bound for honest clocks. [19] The preceding commit data used in this computation must be covered by the block commitment. This rule and light-client time validation require separate tests. The precommit UTC-millisecond timestamp is signed as part of the vote. The median sorts by timestamp and validator identifier, then selects the first timestamp whose cumulative weight is strictly more than half the certificate weight. Honest signers use timestamps later than the previous block time and within the specified clock-error bound.

7 2 Inclusion, Execution, and Failure Semantics

Stage	Defined meaning
Local reception	Check format, signatures, and resource limits; mempool admission creates no commitment.
Proposal validation	Check authorization, nonce, validity period, and fee reservation in block order; an invalid transaction invalidates the proposal.
Consensus commit	A complete valid block obtains a commit certificate, finalizing transaction order.

Stage	Defined meaning
Execution and settlement	Execute deterministically, produce success or failure receipts, and settle fees and nonces.
Result proof	Wait for the finalized header at $h+1$ to commit to the state and receipts for height h .

Table 3 State Machine and Transaction Lifecycle.

Proposal validation must sequentially simulate the same rules as final execution in isolated state, or perform provably equivalent validation, including prior writes, failure rollback, fees, sequences, and policy changes, without publishing external effects. A transaction with invalid format, authorization, nonce, or fee reservation cannot be validly included and cannot charge an impersonated account. If a validly included transaction subsequently encounters a business failure or runs out of gas, only its business writes revert; nonce consumption and metered fees remain. All business messages within a transaction succeed or revert together. Fee reservation, pricing, refunds, and the out-of-gas bound must be deterministic, and the amount charged must not exceed the signed fee cap. On execution failure, the new business-state bond D is zero and its reservation is refunded. Tentative refunds for deleting old state also revert; only fee F and consumption of the applicable sequence remain.

7 3 Result Commitments and Data Availability

Header _{h} commits to tx_root over the current complete transaction envelopes, previous_state_root($S_{\{h-1\}}$), and previous_receipt_root($R_{\{h-1\}}$). After block h is finalized, execution produces S_h and R_h , whose roots are finalized in Header _{$\{h+1\}$} ; empty blocks also advance this process. APIs and wallets must therefore distinguish finality of inclusion at h from verifiability of the resulting state at $h+1$. Receipts contain fixed fields for intent identifier, execution height, transaction index, status, resource use, actual fee, and event digest. Uncommitted logs must not be presented as state proofs.

An honest validator must obtain and validate the complete block before prevoting for it. The baseline uses neither sharding nor data availability sampling. [20] State snapshots must be checked against a finalized commitment. Retention periods, redundant services, and retrieval tests for blocks, receipts, evidence, and proofs should be public and cover the trusting period, unbonding period, and evidence submission window. A signature certificate cannot guarantee indefinite access to historical data. When required data are missing, nodes suspend the corresponding verification rather than infer business success.

08 Light Clients and Cross-Chain Boundaries

8 1 Trusted Starting Points and Extended Offline Periods

A light client starts from a chain identifier and trusted checkpoint confirmed through independent channels. The checkpoint binds the height, block identifier, time, and validator-set commitment. An internally consistent header chain supplied only by an unfamiliar server cannot establish initial trust. The trusting period T must be shorter than the unbonding period U , with margins for clock error,

attack detection, and evidence submission. [21] Retired validators' old keys can still sign; post-quantum cryptography does not eliminate long-range attacks or weak subjectivity.

A client whose trusting period has expired must obtain a fresh independently trusted checkpoint rather than recover automatically from the largest signature count or longest history. Continued updates also require the specified honesty assumption for the relevant validator sets during the trusting period. Bond penalties provide incentives and a recovery mechanism; they do not replace cryptographic or protocol assumptions. On conflicting finalized headers, the client freezes the affected path and preserves verifiable evidence. Recovery follows a separately published governance procedure.

8 2 Header Verification and Proof Availability

The baseline verifies adjacent heights. It checks the chain identifier, version, parent link, time, validator-set commitments, and signatures from strictly $>2W/3$ of the applicable set for the same block and round. Duplicate signers contribute voting power only once. At an epoch boundary, the client first verifies the manifest finalized by the old set and then changes its verification set. A certificate signed only by a newly declared set cannot establish that set's authority to take over. Skipping or bisection verification requires a separate specification for trusted-set signature overlap and failure handling; it cannot be inferred from adjacent verification. [21]

Business verification additionally requires transaction, state, or receipt membership proofs. Absence requires a non-membership proof for the specified tree structure. Proofs bind the key, path, root, and execution height; state and receipt proofs for h are anchored in the finalized header at $h+1$. Header validity and proof availability are separate conditions. A server timeout or refusal to provide a proof means verification is currently unavailable, not that the record is absent. Clients may request proofs for the same commitment from multiple independent services.

8 3 Cross-Chain Verification and Upgrade Boundaries

Cross-chain operation is a separate security protocol; adopting post-quantum signatures at the base layer does not establish it automatically. The receiving chain must verify the sending chain's finalized header and a business proof under its commitment, binding both chain identifiers, the channel, packet sequence, recipient, and timeout conditions, with replay protection and deterministic acknowledgements. Relayers transport data; their signatures cannot substitute for source-chain proofs. Relevant cross-chain processing pauses when a client expires, a conflict is detected, or required data are unavailable. [22]

Both sides must separately migrate account authorization, validator certificates, node identities, transport authentication, and governance upgrade trust roots, and specify the versions of ML-DSA-87, SHA3-512, and proof encodings. ICS-007 informs the design of trusting periods, set updates, and freezing on conflict, but existing clients must not be assumed compatible with this protocol. [22] Changes to algorithms, tree structures, verifiers, or recovery checkpoints require dedicated upgrade proofs and tests. Cross-chain custody, economic security, and compromise of an external chain require separate assessment.

09 Resource and Performance Cost Model

9 1 Resource Budgets and Analytical Scenarios

Capacity planning imposes separate limits on serialized bytes, signature-verification work, execution and state growth. An ML-DSA-87 public key occupies 2,592 bytes and a signature occupies 4,627 bytes; a transaction carrying both therefore incurs 7,219 bytes for these two fields alone, before its payload and encoding overhead. [7] A hash commitment cannot replace a signature, and caching a public key cannot replace verification of its binding to the account. Standard ML-DSA supplies no aggregate-signature construction directly adopted by this protocol. Parallel or batched scheduling must preserve the verification semantics of every signature; any construction that reduces cryptographic checks requires a separate proof and review.

The following is an analytical scenario with no measured performance: assume 100 validators, one block every five seconds and 100 signatures in each commit certificate. Raw signatures then occupy 462,700 bytes per block and 7,995,456,000 bytes per day, approximately 8.00 GB in decimal units. This is no bandwidth, storage or TPS commitment. The actual commit threshold is strictly more than two thirds of voting power, so the signature count is not fixed. Public-key repetition, proposal dissemination, voting rounds, retransmissions, block indexes and replica counts require separate accounting.

9 2 Deterministic Charging and Admission Control

The proposed fee is $F_i = \lceil p_B B_i + p_V V_i + p_G G_i \rceil + tip_i$. B denotes the canonical encoded bytes of the complete transaction envelope, V calibrated signature-verification work units and G execution units excluding verification work already charged. V and G accumulate according to deterministic metering schedules, rather than local elapsed execution time. Prices are determined from the common parent state, represented as rational numbers with a common denominator q , and rounded upward once after summation. Multiplication and addition must be bounded and checked for overflow. The signed transaction body covers tip and fee_cap . The latter is the payer's maximum authorized protocol debit, covering fee F and new-state bond D . The protocol reserves fee_cap , enforces $F + D \leq fee_cap$ at settlement and refunds the unused reservation. Transfer amounts and other application payments are checked separately against the balance available after reservation. Failed execution still pays for consumed resources while application writes are rolled back. A transaction that passes authentication and enters execution consumes its sequence number. Transactions failing parsing or authentication are invalid for block inclusion.

$$F_i = \text{ceil}\left(p_B B_i + p_V V_i + p_G G_i\right) + tip_i$$

New persistent state additionally locks a bond $D_i = p_S \cdot \sum_e \max(\Delta S_{i,e}, 0)$, summing positive growth per entry without offsetting it against deletion of other entries. The bond is accounted for as escrowed funds rather than fee revenue. Each state entry records its funding attribution; deletion refunds only the amount actually locked, and $storage_cap$ bounds the sum of per-entry positive growth in bytes. The bond and fee jointly remain subject to fee_cap , without an additional bond-amount-cap field. State created and deleted within the same block still incurs byte and execution

charges. Zero net growth must not exempt computation and writes from metering. A block must satisfy simultaneous limits on bytes, verification work, execution and state growth; pricing cannot replace these hard limits. On-chain fees cannot constrain malicious inputs that never reach the chain, so nodes also require length checks, connection quotas, bounded queues and local rate limits. Fee and bond parameters must be frozen after cost measurement, without claiming that they fund indefinite archival retention. p_S is restricted to a nonnegative integer number of smallest asset units per byte; refunds always use the integer amount actually escrowed for the entry. A failed transaction retains no new business-state bond and realizes no tentative refund for deleting old state.

10 Native Asset Economics and Governance

10 1 Objectives and Supply Units

The economic model supports native settlement, resource payments, staking liability and protocol maintenance. Long-term value depends on continued demand for these services and on a reliable, secure network that is practical to integrate. Supply, unit price and market capitalization describe different quantities: denomination count, traded price and a valuation calculated from price and a stated supply measure. Changing 100 million tokens to 10 million does not create network value. Nine decimal places preserve small-payment granularity while fees remain integer amounts. Post-quantum security derives from cryptography and protocol design rather than a particular supply number.

This edition adopts a cumulative-issuance cap $S_{\max}$ of 10,000,000 tokens. One token equals 10^9 atomic units, so $S_{\max}$ is 10^{16} atomic units, below the uint64 limit. The cumulative counter C_h includes genesis issuance and every later mint; burning never reduces it. Outstanding supply is $M_h = C_h - B_h$, where B_h is cumulative burning. Every mint must satisfy both the published schedule and $C_h \leq S_{\max}$. Burning creates no new issuance entitlement. Ordinary governance cannot increase the cap, reset the counter or count a transfer from an already minted reserve as new issuance.

$$S_{\max} = 10^{16} \quad M_h = C_h - B_h \quad 0 \leq M_h \leq C_h \leq S_{\max}$$

One native asset pays fees, secures stake and settles transfers. The evaluation distinguishes viability, decentralization, stability and feasibility as in the cited research. [26] The 75% and 25% shares are an explicit review configuration, not an optimum proved by university researchers. The selection preference is to keep first-reference-year new issuance below 2% of genesis distribution at a 20-reference-year curve scale, while retaining as much security allowance as possible among the compared five-percentage-point configurations. The 2% limit is a design preference, not a price-stability law. Failure of validator-cost evaluation requires parameter reassessment and postponement of launch.

10 2 Public Distribution and No Reserved Free Allocations

Purpose	Share and amount	Ownership and constraints
Public initial distribution	75% 7,500,000 tokens	Issued at genesis only after full subscription and satisfied release conditions; entirely mapped to verified participant PQ accounts.
Long-term PoS security issuance	25% 2,500,000 tokens	Unissued at genesis; enters a separate reward pool per height. The allowance has no owner, stake or vote.
Founders contributors advisers institutions foundations	0% reserved free allocation	No insider discount, guaranteed allocation or protocol developer levy. Participation follows the same public subscription rules.

Table 4 Public distribution and security issuance. No reserved allocation does not establish dispersed control or the absence of organizational sales revenue.

All 7.5 million initial tokens must be assigned directly in the genesis manifest to participants with completed public allocations. There is no unsold, unclaimed or project-held allocation pool. Staking is voluntary, and unstaked balances have no additional vesting schedule. Undersubscription, incomplete delivery or unmet release conditions cancel this configuration as a whole and trigger the refund procedure; a small participating group does not receive a partial launch. Unsold shares cannot become team funds, treasury assets or additional security allowance. Chapter 11 defines public launch and funding boundaries.

Contributors may purchase on the same terms, stake and earn duty-based rewards, or receive development-service payments under disclosed contracts. Contributor status confers no issuance entitlement. Research funding and protocol issuance are separate. Pre-launch work requires committed front funding; post-launch budget payments, external grants and voluntary donations disclose control and related parties. Bitcoin block rewards provide no developer-status allocation, while Brink demonstrates external donation-funded development. [27][31]

10 3 Long-Term Issuance and Dilution Measures

Schedules use completed post-genesis height h , with genesis at $h=0$. H_{year} is a positive reference-year height span frozen before genesis from measurements; it does not guarantee calendar duration. A halted chain advances no heights and receives no wall-clock catch-up issuance. Set $H_{\text{tau}}=20H_{\text{year}}$, $R=2.5 \times 10^{15}$ atomic units and genesis cumulative issuance $C_0=7.5 \times 10^{15}$. At $h \geq 1$, I_h is minted exactly once into the reward pool, so $C_h=C_0+A(h)$. The valid H_{year} range is 1 through $\text{floor}((2^{64}-1)/20)$. Multiplication and $h+H_{\text{tau}}$ require checked intermediates of at least 128 bits.

$$A(h) = \text{floor}\left(\frac{R h}{h + H_{\text{tau}}}\right) \quad I_h = A(h) - A(h - 1)$$

The unrounded marginal issuance declines smoothly without scheduled halving steps; integer per-height increments may fluctuate between adjacent values or be zero. Half the security allowance is issued after 20 reference years, which is not a subsidy expiry. At every finite height $A(h) < R$, and residual atomic units may never be issued. Ten million is a cap, not a promised final top-up. Burning, slashing transfers, reward claims and delegation never replenish used allowance.

Reference year	Cumulative security issuance	Cumulative issued C
0	0.000	7,500,000.000
1	119,047.619	7,619,047.619
2	227,272.727	7,727,272.727
5	500,000.000	8,000,000.000
8	714,285.714	8,214,285.714
20	1,250,000.000	8,750,000.000
40	1,666,666.667	9,166,666.667
100	2,083,333.333	9,583,333.333

Table 5 No-burning scenario in tokens, displayed rounded to three decimals; protocol arithmetic floors atomic units. Genesis distribution is fully delivered, but reward-pool balances need not be circulating.

First-reference-year new issuance is 119,047.619047619 tokens, approximately 1.587302% of genesis issuance. A non-staking holder with an unchanged balance loses approximately 1.5625% of their share of issued supply, calculated as $1 - C_0/C_1$; this is not a price forecast. Reports must distinguish cumulative issuance, distributed balances, stake and unbonding, reward pools and transferable balances. Maximum supply cannot replace the relevant denominator, and pre-minted but undelivered inventory cannot be used to understate dilution.

At the same curve scale, first-year issuance relative to genesis is approximately $(1-g)/(21g)$, where g is the public genesis share. The 2% preference requires g of at least approximately 70.4225%. Seventy-five percent is the smallest five-percentage-point configuration in the table meeting that preference, while retaining 25% more security allowance than the 80% configuration. This comparison explains the selection process; it does not establish real user preferences or an adequate security budget.

Public and security shares	Year-one security issuance	Year-one issuance growth
40/60	285,714.286	7.142857%
60/40	190,476.190	3.174603%
70/30	142,857.143	2.040816%
75/25	119,047.619	1.587302%

Public and security shares	Year-one security issuance	Year-one issuance growth
80/20	95,238.095	1.190476%

Table 6 Allocation sensitivity at a fixed 10-million cap and 20-reference-year curve scale. The 75/25 configuration is under review; other rows are comparisons.

With the same 2.5-million security allowance, curve scales of 10, 20 and 40 reference years produce approximately 227,272.727, 119,047.619 and 60,975.610 new tokens in year one. Slower issuance lowers early validation income; faster issuance increases holder dilution. Production H_{year} , resource capacity and operating costs require joint calibration. Lower issuance alone establishes neither security nor growth.

10 4 Fees and Security Reward Settlement

All fees F_i defined in Chapter 9, including base fees and tip_i , initially enter the validator reward pool. Protocol developer levies, public-treasury fee levies and default fee burning are zero. State bond D remains separate escrow rather than a fee. Tips enter the shared validator reward pool, without an individual recipient selected from an uncommitted proposal round. Public development uses the disclosed Chapter 11 budget or voluntary external funding and receives no automatic share of consensus fees.

A security subsidy becomes issued supply when it enters the reward pool. Settlement must use commonly committed data, fixed validator and weight snapshots, and verifiable participation rules. Fixed rewards per node and payouts based on local message-arrival times are prohibited. Execution at height h cannot depend on an h -height commit certificate that has not yet been commonly committed. If the preceding certificate is used, it must be bound to the corresponding historical validator set, with evidence that proposer-selected certificate subsets cannot arbitrarily deny honest participants their rewards. Mere appearance in a selected certificate is not equivalent to all eligible work. Participation proofs, delegation splits, commission bounds, slashing order and claim deadlines require byte-level specifications and bias tests before genesis.

Unearned amounts, rounding residues and expired unclaimed shares remain in the dedicated reward pool. They cannot become unissued allowance again, be automatically claimed retroactively by later participants, or be redirected to insider allocations or operating funding through ordinary governance. Any pool redistribution conditions must be specified in advance and remain auditable. Only the function in this section authorizes new issuance. Staking rewards depend on active weight, participation, actual fees, subsidies and commissions. No fixed APR is promised, and issuance does not target a token price. Reducing the validator fee share, introducing burning or changing the issuance function requires a new version of the economic rules and renewed security-budget review; ordinary budget transfers cannot implement such changes.

Holding alone earns no security reward. A holder must register a PQ validator identity and voluntarily lock stake, or delegate under frozen rules; commissions, exit delays and slashing liability must be disclosed in advance. Full nodes recompute rewards from fixed stake snapshots, committed duty evidence and deterministic formulas rather than project approval. With equal participation, an

illustrative weight-based allocation of 100 available tokens across total effective weight 10,000 assigns one token to weight 100. This example is neither a fixed return nor a substitute for the production settlement specification.

10 5 Asset Conservation and Validator Accountability

At every height, $M_h = M_{(h-1)} + I_h - \text{burn}_h$. M sums disjoint transferable balances, stake, unbonding balances, state bonds and protocol pools. Claimable liabilities must be fully backed by pool assets and cannot be counted again as assets. Transfers, delegations and slashing transfers create no assets; any slashing burn is recorded explicitly. Genesis freezes slashing fractions, evidence and unbonding windows, and every pool authorization and accounting rule.

Stake confers voting power and enforceable liability; it establishes no fiat attack-cost bound and does not replace BFT assumptions. Equivocation and absence are distinct, and a network partition is not automatically equivocation. Concentration is assessed through administrative, key, funding and delegation control and shared failure domains. Address counts, node counts and per-address limits do not establish independent control. Unissued allowance has no stake or vote. Team purchases on public terms remain aggregated in common-control disclosures.

10 6 Operating Budgets and Adoption Conditions

Security income comes from current issuance and actual fees, allocated under delegation and commission rules. A necessary operating check compares income retained after delegator payments against independent operating costs and risk compensation under explicit price and fee scenarios. This is not a complete attack-cost proof. New issuance during reference years 20, 40 and 100 is approximately 32,051.282, 14,124.294 and 3,501.401 tokens, respectively, computed as differences between cumulative integer issuance at consecutive reference years rather than fixed staking APRs.

Before release, stress tests must cover near-zero fees, low demand, an 80% price decline, doubled costs, concentrated delegation and correlated exits. Compare 70/30, 75/25 and 80/20, and curve scales of 10, 20 and 40 reference years. Production prices and operating data do not yet exist; this edition does not claim that 75/25 has passed these tests. Insufficient budgets require lower resource costs or a publicly revised pre-genesis configuration and postponed launch, not silent cap violations, insider issuance or a lower strictly-greater-than-two-thirds commit threshold.

A fixed cumulative cap with subsidies tending to zero requires recurring paid demand. Fixed supply, perpetual fixed staking yields and no need for fees cannot all be promised. Adoption metrics include fees net of subsidies and identifiable self-dealing, successful settlements, wallet and custody integrations, independent active voting power and operator net income. Algorithms, supply denominations, free claims and high nominal yields do not automatically create purchase demand. Governance establishes no price floor, guaranteed buyback or fixed BTC exchange target.

10 7 Unbonding, Evidence and Light-Client Trust

An unbonding request must not immediately extinguish liability for historical signatures. Using the last accountable signature as a common reference point, the design must ensure $T_{\text{trust}} + T_{\text{check}} + T_{\text{report}} + T_{\text{process}} + \Delta_{\text{clock}} < T_{\text{slashable}}$. Evidence-submission and retention periods must

cover the corresponding verification, reporting and processing windows. Exiting funds remain slashable throughout the applicable evidence window; timely submitted pending evidence must not expire merely because unbonding reaches its deadline. Funds are released only after liabilities are resolved. Height-based and time-based deadlines must both be modeled, with explicit rules for processing after a chain halt. An expired light-client trust period requires a new trusted checkpoint; signatures from old keys alone must not extend trust. [21]

10 8 Separation of Powers and Security Boundaries

Proposal submission, voting, technical review and execution after a timelock are separated. Emergency authority may pause specified operations only within predefined scope and duration; reasons must be public and authority must expire automatically. It must not arbitrarily issue assets, confiscate balances or rewrite finalized history. Ordinary parameter governance cannot remove the strictly greater-than-two-thirds commit threshold, full signature verification, replay protection or minimum cryptographic requirements. Changing these boundaries requires a new protocol version, renewed justification and review, and explicit acceptance. Algorithm replacement requires versioned registration, migration windows and downgrade-rejection rules. [2] Validator independence is evaluated through administrative authority, key control, cloud providers and network failure domains; node count is no substitute.

11 Public Launch and BTC Purchase Path

11 1 Native Asset and Exchange Boundaries

The network uses PoS without a PoW distribution prerequisite. Initial tokens come from the public distribution below. After launch, new users may buy from existing holders at market prices and choose whether to hold or stake. The protocol does not custody BTC for all market trades, set a fixed BTC exchange rate for post-launch market trades, promise a listing or immediate execution, or provide BTC reserves and redemption claims for the native asset. BTC payment and PQ balance delivery occur on separate chains and are not represented as an implemented atomic cross-chain exchange.

A free BTC-snapshot claim is a different mechanism: recipients can keep all their BTC, so a claim does not establish capital migration. Historical-balance allocation also inherits concentration and custodial control, with lost-key, unclaimed-allocation and legacy-signature risks. [32] This edition creates no snapshot-airdrop pool. Wrapping BTC does not change authorization of collateral on Bitcoin, and this chain's PQ signatures cannot secure that collateral.

11 2 Uniform Price and Budget Cap

The review configuration publicly allocates all 7.5 million tokens in one adequately announced window. At least 30 days for participation and 14 days for allocation review are recommended; actual UTC deadlines, confirmation anchors and conflict handling are frozen in the launch manifest. The price follows a publicly reviewed cost budget for subsequent development, audits, integration and basic services rather than forecast market capitalization. Let L_BTC be the accepted BTC cap in

satoshis and $J_0 = 7.5 \times 10^{15}$ the initial atomic units. The uniform subscription price is L_BTC/J_0 satoshis per atomic unit. A real budget must support L_BTC ; this paper does not invent that budget.

This is a fixed-price subscription, not a price-discovery auction. Valid deposits D_BTC below L_BTC cancel the batch. Otherwise first allocate accepted satoshis b_i proportionally using floors, largest remainders and stable order identifiers to break ties, so their sum is L_BTC and no order exceeds its deposit. Refund each deposit less b_i . Then allocate J_0 in proportion to b_i/L_BTC using the same deterministic remainder method. Orders with $b_i=0$ receive no tokens and a full refund. Only disclosed integer-rounding error affects the uniform price. There are no first-come preferences, insider discounts or hidden reservations. Larger funders may acquire more; per-address limits prevent neither splitting nor common control. This remainder rule is not split-resistant. Order bounds, minimum applications and aggregate relative-rounding bounds must be disclosed and tested; less than one satoshi per order is not a general fairness guarantee.

11 3 Receipts Orders and Genesis Mapping

Before payment, an order binds `launch_id`, `batch_id`, a unique nonce, PQ public key and proof of possession, an explicitly chosen BTC refund script and terms version. Receipt is verified from the actual deposit outpoint, amount, Bitcoin confirmation anchor and deduplication record. A BIP 322 message signature neither proves that its signer sent a historical transaction nor permanently proves control of funds, and cannot replace receipt evidence. [30] No seed phrase or secret key is requested. Refunds must not default to input addresses, which may belong to exchanges or reflect multi-input and change structures.

Publicly recomputable data must produce identical accepted amounts, refunds, participant balances, authorization policies and genesis allocation roots. Participants confirm destination accounts and separately choose staking, delegation or transferable balances. A purchase cannot force delegation to team validators. The genesis manifest contains the complete initial stake set, consensus public keys and actual-control disclosures. Two independent implementations must reproduce the same results; unverifiable or incomplete delivery prevents launch. Subscriptions bind initial authorization policies and indexes independent of G. Account identifiers are derived only after final G, avoiding a circular genesis definition.

11 4 BTC Custody and Development Funding

Sales revenue is not collateral backing the token. The launch manifest identifies the receiving organization, custodians, signature threshold, receiving-script descriptor, budget items, beneficiaries and related parties, milestones, fee sources and treatment of unused funds. The project organization and development-service providers may receive income; zero team-token allocation cannot be advertised as zero economic benefit. Common control and actual remuneration remain disclosed, and funding administration grants no power to mint, rewrite the ledger or freeze user balances.

This proposal relies on custodial fund handling. Bitcoin scripts do not themselves determine whether this chain launched correctly. Custodians may collude, become unavailable or release funds incorrectly; multisignature custody does not remove this trust boundary. Subscription principal cannot be spent before launch-success conditions are satisfied. Independent front funding pays pre-

launch development and audits. After success, funds are released in disclosed milestone tranches, and treatment of unused funds cannot be changed retrospectively. Collection cannot begin before budgets, custody, dispute handling and liability terms are frozen.

11 5 Failure Refunds and Delivery Ordering

The launch manifest specifies a complete state machine for deposits, settlement, objections, allocation confirmation, initial finalized blocks, fund release and failure refunds, including deadline conflicts, deep reorganizations and chain halts. An order cannot receive both effective delivery and an effective failure refund. Success requires independently verifiable genesis and finality evidence. Custodial processes inspect that evidence; Bitcoin does not thereby enforce cross-chain success conditions automatically. Before activation, allocations are conditional receipts rather than transferable mainnet balances. Once the frozen success criteria are met, the network enters its operating phase; later halts or price changes do not reactivate initial-subscription principal refunds.

Batch failure requires return of received principal; miner fees on the original sending transaction are not received principal. A separately pre-funded reserve pays refund miner fees. Any fee-deducting alternative must disclose its items, calculation and cap before payment and cannot simultaneously promise unconditional full refunds. This edition includes no implicit deduction. Refund reserves, excess refunds, dust limits and custodian failures require testing; inadequate reserves or unenforceable commitments prevent collection.

11 6 PoS Activation and the Quantum-Risk Window

Full subscription does not establish sufficient secure stake. Activation separately verifies voluntarily locked stake, validator keys, delegation and operational readiness, and reviews common control, custody and failure domains. Under the assumption that Byzantine power is strictly below one third in each relevant set, commitment still requires strictly more than two thirds. A few addresses under one controller do not meet the dispersed-launch objective. Independence reviews cannot prove the absence of hidden collusion, and thresholds cannot be reduced because participation is small. [32]

The Bitcoin side of BTC deposits and trades retains the authorization security of Bitcoin at that time. Draft BIP 360 P2MR mitigates long-exposure risk but does not itself provide full short-exposure protection; the BIP 361 migration draft is not a deployed upgrade. [28][29] Once quantum attacks can defeat the relevant legacy signatures, new deposits depending on those authorizations must stop and the published pause and refund procedures apply. This proposal is not represented as a risk-free BTC rescue route after a cryptographic break. Refund execution still depends on Bitcoin security and availability and on custodial funds remaining spendable.

12 Cryptographic Evolution and Protocol Upgrades

12 1 Versioned Capabilities and Deterministic Activation

The protocol maintains separate versions for object structure, cryptographic suites, account policies, state trees, and transport profiles. An algorithm identifier resolves to a complete parameter set and

encoding rather than a generic “post-quantum signature.” An upgrade proposal commits to the new specification digest, activation height, historical-verification window, migration plan, and resource limits. Governance approval does not immediately change validity rules: honest nodes activate at the same committed height, not according to local installation time. [2]

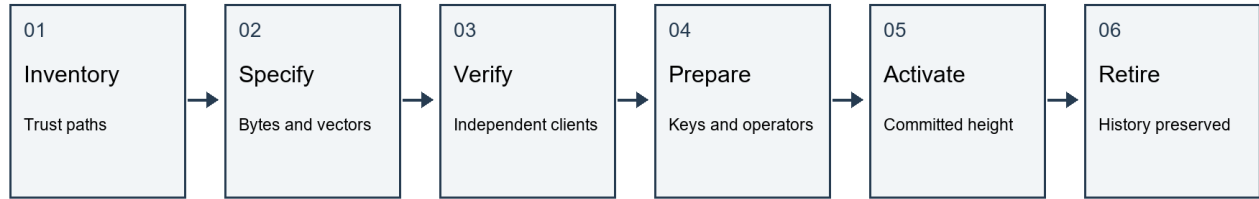


Figure 2 Protocol evolution. Each stage has inspectable inputs, artifacts, and rejection conditions.

Migration proceeds through a cryptographic inventory, byte-level specifications and vectors, independent implementation verification, account and validator preparation, deterministic activation, retirement of old authorization, and maintenance of historical verification. A compatibility window may recognize old and new formats, but authorization rules at any height must be unambiguous. If an AND combination is used during transition, both signatures must verify. Replacing it with OR permits authorization by the weaker component alone and constitutes a different security design. The baseline does not add classical signatures to ordinary transactions.

12 2 Account and Validator Migration

Ordinary account rotation uses the currently valid online policy to authorize a new policy and increments `auth_epoch`. The transaction signs the new policy digest and activation height. A previously registered delayed-recovery policy may operate within its original authority when the primary key fails. If an attacker can already forge a signature family, continued use of that family to prove ownership cannot restore the lost trust. Recovery enrollment and migration should therefore precede exposure, rather than depending on an ad hoc governance judgment about asset ownership.

A validator-key replacement binds the same staking identity, activation boundary, and new set commitment. The old key cannot retain eligibility for new heights after the boundary. Anti-double-signing state migrates with the key and height; restoring a backup must not let two devices sign conflicting messages. Light clients update sets, algorithm allowlists, and historical-verification rules together, preventing a silent split in which full nodes can verify a chain that clients cannot.

12 3 Emergency Response and Historical Integrity

Emergency pause authority must be defined in advance by scope, maximum duration, and recovery procedure. It is limited to stopping specified hazardous operations or entering read-only mode; it cannot forge user authorization or rewrite finalized history. If the model's assumptions fail, socially coordinated recovery must document affected heights, checkpoints, asset-treatment rules, and

disagreements. It cannot retain an unconditional security claim for the original chain. Recovery must preserve evidence for subsequent review.

Archive nodes retain required historical verifiers, version descriptions, and commitment data. Being able to parse an old signature does not establish that it retains current security. Long-term confidence in history depends on timely accepted checkpoints, consensus evidence valid at the time, and an explicit migration chain. The protocol should publish each retirement decision and its consequences so that reviewers can identify historical intervals that depend on assumptions no longer accepted.

13 Validation Plan and Release Conditions

13 1 Protocol Modeling and Implementation Validation

Each release candidate fixes a protocol version, threat model, source digest and build digest. Formal models examine weighted-quorum intersections, locking rules, validator changes, account rotation, replay protection and asset conservation. Liveness claims must state assumptions of partial synchrony and sufficient honest online voting power. Model-checking and proof results apply to their models and assumptions and do not automatically establish implementation correctness. [23] At least two independent implementations undergo differential testing of encoding, state transitions and cryptographic verification. Standard test vectors are combined with fuzzing of malformed lengths, noncanonical encodings, integer boundaries, unknown versions and invalid signatures, retaining minimized failure cases.

Fault experiments cover delay, packet loss, partitions, clock skew, crash and restart, database recovery, and recovery of the signer's anti-equivocation state. They also cover key rotation, expired checkpoints and forged snapshots. Offline and malicious fractions are configured separately by voting power. Tests examine absence of forks below the safety threshold, safe stalling when an online quorum is unavailable, and resumed progress after synchrony is restored. Experiments near or beyond the Byzantine threshold expose the boundary of guarantees; they do not claim unconditional safety in that region.

13 2 Reproducible Performance Measurement

Following BLOCKBENCH's layered evaluation approach, cryptography, networking, execution and storage are measured separately before end-to-end evaluation. [24] Reports identify hardware, software versions, geographical distribution, operational control domains, workload composition, warm-up and measurement intervals. Successful TPS is the number of finalized transactions with successful execution during the interval divided by its duration in seconds. TPS for finalized failed transactions is reported separately, alongside submission and admission rates, timeouts and backlog. Latency runs from first submission to finalization, with p50/p95/p99 and the timeout fraction reported; failed transactions are reported separately. Timeouts must not be silently discarded. Open-loop and closed-loop workloads are distinguished, cross-machine timestamps disclose synchronization error, and raw data and scripts permit independent checking. Because this protocol commits the post-h execution state and receipts in the h+1 header, reports distinguish finality latency

for inclusion at h from state-verifiability latency to an $h+1$ commitment and a valid proof. A successful RPC response cannot substitute for either metric.

$$TPS_{success} = N_{finalized, success} / \Delta t$$

13 3 Staged Release Gates

Stage	Required evidence
Protocol freeze	Complete encoding and state rules; reviewable model assumptions, Genesis parameters and migration paths; no unresolved issue affecting interoperability.
Independent testnet	Operation across independent control domains; passing differential, vector, fuzz and fault tests; published failures, recovery records and complete measurement methods.
Production candidate	Completed independent cryptographic and protocol reviews, with blocking findings fixed and rechecked; pinned dependencies, an SBOM, reproducible builds, signed releases and completed key-recovery exercises.
Launch authorization	Achievement of previously published resource, latency and recovery targets; finalization and interoperability testing of the candidate TLS authentication specification; explicit resolution of material outstanding risks.

Table 7 Validation Plan and Release Conditions.

Supply-chain and release processes use the SSDF to organize traceable evidence. [25] Citing research, passing test vectors or using standardized algorithms does not constitute an audit, certification or endorsement by a research institution. The table sets conditions to be completed; performance results, external-review conclusions and production-readiness statements may be populated only after the corresponding evidence is published.

Economic release conditions include independent reproduction of the cap and issuance table; checks that pool balances and claims are not double-counted, burns do not replenish issuance allowance, and unissued allowance cannot vote and no insider allocation exists; integer-boundary, evidence-deduplication and certificate-selection-bias tests; disclosure of distribution beneficiaries and common control; and operating-budget analysis under low fees and prices. Integer arithmetic makes the curve reproducible but does not validate demand, operator profitability or future token prices.

13 4 Public Launch Release Conditions

Public launch release materials include a real cost budget and BTC cap, fixed subscription price, participation window, custody and refund agreements, orders and genesis manifest, common-control disclosures, initial voluntary stake and source-audit evidence. Tests cover under- and oversubscription, duplicate and late deposits, reorganizations, rounding, wrong refund scripts, unavailable custodians, inadequate refund fees and failed activation. Independent implementations must recompute both launch settlement and consensus execution. Without this evidence, the paper remains a design proposal rather than an operating system ready to accept user funds.

14 Research Roadmap and Protocol Scope

Development follows technical dependencies. Object encodings, signing domains, state and fee semantics, consensus, and set-transition models are frozen first. A minimal reference implementation and independent verifier follow, then measurements of cryptographic and data paths, and finally a testnet run by multiple independent operators. Entry into a public network depends on reviewable evidence rather than a predetermined launch date. A virtual machine, privacy module, bridge, or signature-aggregation extension requires renewed analysis of authorization, proof size, deterministic execution, and resource accounting.

The protocol is suited to research on long-lived public settlement ledgers and post-quantum identity infrastructure. Public state carries no confidentiality guarantee. Light clients require an externally obtained initial trust anchor. Concentrated stake, censorship, key compromise, and governance failures can still damage the network. The design addresses these issues through mechanisms and validation obligations rather than claims that the risks cannot occur.

The next stage should publish the byte-level specification, parameter rationale, reference vectors, model-checking results, raw experimental data, and independent review findings. Each security conclusion should identify the version, environment, adversary capabilities, and unresolved issues it covers. This publication model makes the white paper a design foundation that can be implemented, tested, and revised.

Appendix A Baseline Profile and Parameter Registry

The table distinguishes standardized object lengths from protocol choices. Freeze and verify the candidate transport-authentication profile before release; standardized key agreement does not establish a standardized complete node-identity path. Economic choices and remaining pre-genesis decisions follow Chapters 10 and 11.

Purpose	Baseline profile	Status
Account and consensus signatures	ML-DSA-87, pure, on-chain domain context	Standard algorithm; protocol choice
Raw ML-DSA-87 objects	Public key 2592 B, expanded secret key 4896 B, signature 4627 B	FIPS 204 Table 2
Stable account identifier	SHA3-384, 48 B, binding G and full initial policy	Protocol choice
Ledger and state commitments	SHA3-512, 64 B, separate domain labels	Protocol choice
TLS key agreement	SecP384r1MLKEM1024	Explicit application choice under RFC 10024
TLS record protection	TLS_AES_256_GCM_SHA384	Preserves TLS transcript and key schedule
TLS identity authentication	ML-DSA-87 under a pinned TLS profile	Candidate; authentication specification evolving
Optional offline recovery	SLH-DSA-SHAKE-256s, enrolled in advance	Standard algorithm; optional policy

Table 8 Cryptographic baseline. A category applies to its primitive and does not automatically establish a security level for the complete protocol. [7][6][8]

Before genesis, operational parameters are frozen in a versioned registry: block and transaction bytes; messages and authorizations per transaction; verification and execution budgets; state growth and proof length; validator bound; epoch length; set-transition lead time; clock-drift bounds; evidence and unbonding windows; fees; and issuance schedule. Each record states its unit, range, rationale, dependencies, and modification authority. Measurements of hardware costs and fault experiments determine values. Analytical scenarios in this paper do not become network defaults. Transaction and authorization bounds must accommodate the worst-case valid envelope under every enabled policy; a policy that cannot satisfy this constraint must not be created. The public launch registry additionally covers costs, BTC cap, fixed price, order binding, integer settlement, custody and refunds, success criteria and initial voluntary stake. Collection is prohibited while production receipt fields remain unfrozen.

The encoding profile must freeze message types, fixed-array field order, logical integer bounds, domain labels, algorithm identifiers, empty-tree constants, and error codes. Contexts used for

account and vote signatures must not be copied into TLS, whose authentication messages are defined by the transport specification. A release should include a machine-readable registry and rejection tests so that two independent clients reach the same conclusion on identical bytes.

Appendix B Security Claims and Proof Obligations

The following claims are properties the proposed protocol must satisfy, rather than completed proofs or audit findings. Primary papers support models and methods. The new protocol composition, set transitions, recovery policies, and fee semantics require separate refinement and validation.

Property	Key assumptions	Evidence required
Non-bypassable authorization	Unforgeable signatures; unambiguous policy and encoding	Vectors, differential verification, malformed authorization and recovery models
Consistent finality at a height	$B < W/3$ in relevant sets; correct locking and persistence	Consensus refinement proof, set-transition and crash-recovery experiments
Replay and duplicate-execution resistance	Joint binding of G , auth_epoch , nonce , and expiry	Cross-chain, rotation, in-block ordering and re-signing tests
Conservation of assets and fees	Deterministic integer arithmetic; disjoint escrow accounting	Whole-state invariants, boundary and failure-path tests
Correct state proofs	Secure commitments; correct header and tree-version verification	Membership and absence vectors; $h/h+1$ checks
Continued progress	Partial synchrony; sufficient honest online power and resources	Partition recovery, load and Byzantine-behavior experiments
No silent downgrade on upgrade	Common activation; retirement and client coordination	Version-boundary and rollback-rejection tests

Table 9 Security claims, assumptions, and corresponding validation obligations.

Composition issues requiring separate review include epoch-based set updates under the selected Tendermint locking rules; TLS identity and node-certificate revocation; ordering between delayed recovery and pending transactions; state-bond refund ownership; and coexistence of state-tree encodings and historical versions. A formal model supports a concrete release claim only after the refinement relationship between implementation and model has been examined.

Appendix C Terms and Symbols

Symbol or term	Definition
G	64-byte genesis-manifest digest and chain identifier
C(x)	Restricted core-deterministic CBOR encoding
W, B, Q	Total, Byzantine, and commit-certificate voting power
M _h	Outstanding supply at height h, equal to cumulative issuance less cumulative burning
auth_epoch	Monotonic version of an account authorization policy
nonce	Next consumable transaction sequence for an account
epoch	Protocol interval with a fixed active validator set
Final inclusion	The envelope appears in a block satisfying conditional commit rules
Execution success	Business writes commit under state-machine semantics
State-proof confirmation	The next finalized header commits execution results for proof verification
Cryptographic agility	Versioned algorithm replacement preserving authorization and historical verification
C _h	Cumulative issuance counter, never reduced by burning
H _{year}	Genesis-frozen reference-year height span, not a calendar-duration guarantee
A(h)	Cumulative security-reward issuance in atomic units

Table 10 Terms used in this paper. Voting-power thresholds are distinct from node-count thresholds.

References

Primary standards, original research, and official specifications. Internet-Draft status and version-specific specifications are stated in the entries. Accessed 10 October 2026.

- [1] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. 1997. SIAM Journal on Computing 26(5), 1484–1509, October 1997; revised preprint 1996-01-25. DOI 10.1137/S0097539795293172. Establishes the structural threat from quantum factoring and discrete-logarithm algorithms. <https://arxiv.org/abs/quant-ph/9508027>
- [2] Elaine Barker; Lily Chen; David Cooper; Dustin Moody; Andrew Regenscheid; Murugiah Souppaya; William Newhouse; Russ Housley; Sean Turner; William Barker; Karen Kent. NIST CSWP 39-upd1 Considerations for Achieving Crypto Agility: Strategies and Practices. 2026. Originally published 2025-12-19; this updated edition is dated 2026-06-29. Discusses replacing algorithms while preserving security and operation, including governance, compatibility, and lifecycle tradeoffs; the migration rules here are protocol-design choices. <https://csrc.nist.gov/pubs/cswp/39/upd1/considerations-for-achieving-crypto-agility/final>
- [3] Lov K. Grover. A fast quantum mechanical algorithm for database search. 1996. STOC 1996, 212–219, May 1996; revised preprint 1996-11-19. DOI 10.1145/237814.237866. Supports the quadratic query speedup for unstructured search. <https://arxiv.org/abs/quant-ph/9605043>
- [4] Martin Roetteler; Michael Naehrig; Krysta M. Svore; Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. 2017. First posted 2017-06-21; v3 dated 2017-10-31; ASIACRYPT 2017. Estimates logical quantum resources for elliptic-curve discrete logarithms; it is not a direct forecast of physical-computer size or a break date. <https://arxiv.org/abs/1706.06752>
- [5] Mark Webber; Vincent Elfvig; Sebastian Weidt; Winfried K. Hensinger. The Impact of Hardware Specifications on Reaching Quantum Advantage in the Fault Tolerant Regime. 2022. Preprint first posted 2021-08-27, v3 dated 2021-11-11; AVS Quantum Science 4, 013801, 2022. DOI 10.1116/5.0073075. Analyzes how hardware, error correction, and attack duration affect resource requirements through conditional models, not date forecasts. <https://arxiv.org/abs/2108.12371>
- [6] National Institute of Standards and Technology. FIPS 203 Module-Lattice-Based Key-Encapsulation Mechanism Standard. 2024. Final publication 2024-08-13. ML-KEM-1024 is category 5; Table 3 specifies public-key, private-key, ciphertext, and shared-secret lengths. The 2025-11-17 page note flags potential corrections, not an already effective replacement standard. <https://csrc.nist.gov/pubs/fips/203/final>
- [7] National Institute of Standards and Technology. FIPS 204 Module-Lattice-Based Digital Signature Standard. 2024. Final publication 2024-08-13. Tables 1 and 2 define the ML-DSA-87 category and encoding lengths; pure signatures support a context and signing is hedged by default. Potential corrections announced on 2026-07-31 do not change the listed parameter lengths or category. <https://csrc.nist.gov/pubs/fips/204/final>
- [8] National Institute of Standards and Technology. FIPS 205 Stateless Hash-Based Digital Signature Standard. 2024. Final publication 2024-08-13. Table 2 defines SLH-DSA-SHAKE-256s as category 5, with a 64 B public key and 29792 B signature; its private key comprises four 32 B components. <https://csrc.nist.gov/pubs/fips/205/final>
- [9] Gorjan Alagic; Elaine Barker; Lily Chen; Dustin Moody; Angela Robinson; Hamilton Silberg; Noah Waller. NIST SP 800-227 Recommendations for Key-Encapsulation Mechanisms. 2025. Final publication 2025-09-18. Covers KEM implementation, authentication conditions, key confirmation, and hybrid composition; distinguishes theoretical, implementation, and deployment security, and requires single-use ephemeral keys and prompt destruction. <https://csrc.nist.gov/pubs/sp/800/227/final>

- [10] National Institute of Standards and Technology. FIPS 202 SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. 2015. Final publication 2015-08-04. Defines SHA3-384 and SHA3-512. An update was announced on 2025-03-13; the page records a typographical correction to nonnormative Appendix B that does not change the output lengths used here.
<https://csrc.nist.gov/pubs/fips/202/final>
- [11] Bormann C, Hoffman P. Concise Binary Object Representation (CBOR). 2020. RFC 8949, STD 94, December 2020. Section 4.2.1 defines core deterministic encoding requirements; this protocol further restricts data types.
<https://www.rfc-editor.org/rfc/rfc8949.html>
- [12] Eric Rescorla. RFC 9846 The Transport Layer Security (TLS) Protocol Version 1.3. 2026. Published July 2026 as a Proposed Standard; obsoletes RFC 8446. Specifies the TLS 1.3 handshake, CertificateVerify messages, key schedule, and replay limitations of 0-RTT.
<https://datatracker.ietf.org/doc/rfc9846/>
- [13] Kris Kwiatkowski; Panos Kampanakis; Bas E. Westerbaan; Douglas Stebila. RFC 10024 Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3. 2026. Published August 2026 as a Proposed Standard. Defines SecP384r1MLKEM1024 and other groups. This group's IANA Recommended value is N; it must not be presented as a general IETF deployment recommendation.
<https://datatracker.ietf.org/doc/rfc10024/>
- [14] Jake Massimo; Panos Kampanakis; Sean Turner; Bas E. Westerbaan. RFC 9881 Internet X.509 Public Key Infrastructure -- Algorithm Identifiers for the Module-Lattice-Based Digital Signature Algorithm (ML-DSA). 2025. Published October 2025 as a Proposed Standard. Specifies ML-DSA X.509 algorithm identifiers and encodings; this does not imply that the ML-DSA TLS authentication document has been published as an RFC.
<https://www.rfc-editor.org/rfc/rfc9881.html>
- [15] Tim Hollebeek; Sophie Schmieg; Bas E. Westerbaan. Use of ML-DSA in TLS 1.3 draft-ietf-tls-mldsa-06. 2026. Revision dated 2026-09-11. As of 2026-10-10 it remains an Internet-Draft in the RFC Ed Queue, intended Informational, and is work in progress. Requires an empty FIPS context for pure ML-DSA CertificateVerify signatures.
<https://datatracker.ietf.org/doc/draft-ietf-tls-mldsa/06/>
- [16] Ethan Buchman; Jae Kwon; Zarko Milosevic. The latest gossip on BFT consensus. 2018. Pinned to arXiv:1807.04938v1, 13 July 2018, Algorithm 1. DOI:10.48550/arXiv.1807.04938. Defines locking, valid rounds, voting, and timeouts. Its proof does not directly cover the new cryptographic profile, persistence implementation, or this design's epoch extension.
<https://arxiv.org/abs/1807.04938v1>
- [17] Cynthia Dwork; Nancy Lynch; Larry Stockmeyer. Consensus in the Presence of Partial Synchrony. 1988. Journal of the ACM 35(2):288–323. DOI:10.1145/42282.42283. The original paper defines partial synchrony and conditions for safety and termination; it is not a direct proof of this protocol's handoff mechanism.
<https://groups.csail.mit.edu/tds/papers/Lynch/jacm88.pdf>
- [18] Fred B. Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. 1990. ACM Computing Surveys 22(4):299–319. DOI:10.1145/98163.98167. The paper lists Cornell University's Department of Computer Science as the author's affiliation. Supports the deterministic state-machine replication method, not this protocol's fee or transaction semantics.
<https://www.cs.cornell.edu/fbs/publications/SMSurvey.pdf>
- [19] CometBFT contributors. BFT Time. 2026. Official specification pinned to v0.40.0, released 27 July 2026; accessed 10 October 2026. Defines the weighted median over preceding-height commit timestamps. The document also identifies PBTS as a replacement direction. This design selects a pinned BFT Time profile and retains separate refinement obligations.
<https://github.com/cometbft/cometbft/blob/v0.40.0/spec/consensus/bft-time.md>

- [20] Mustafa Al-Bassam; Alberto Sonnino; Vitalik Buterin. Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities. 2019. Pinned to arXiv:1809.09044v5, 25 May 2019. DOI:10.48550/arXiv.1809.09044. Supports the distinction between header verification and data availability, including the effect of withholding on proof construction. This design adopts neither its sampling protocol nor its adversarial model or performance results.
<https://arxiv.org/abs/1809.09044v5>
- [21] CometBFT contributors. Core Verification: Light Client Verification. 2026. Official specification pinned to v0.40.0; accessed 10 October 2026. Describes externally trusted initial headers, honesty assumptions during the trusting period, a trusting period shorter than unbonding, and distinct conditions for adjacent and skipping verification. This protocol's cryptographic and epoch extensions require separate proofs.
<https://github.com/cometbft/cometbft/blob/v0.40.0/spec/light-client/verification/README.md>
- [22] Christopher Goes; IBC contributors. ICS 007: Tendermint Client. 2019. Official ICS-007, created 10 December 2019; main branch accessed 10 October 2026. Informs trusting periods, validator updates, membership verification, and freezing on conflict. Citation does not establish compatibility with existing IBC encodings or algorithms.
<https://github.com/cosmos/ibc/blob/main/spec/client/ics-007-tendermint-client/README.md>
- [23] Sean Braithwaite; Ethan Buchman; Ismail Khoffi; Igor Konnov; Zarko Milosevic; Romain Ruetschi; Josef Widder. A Tendermint Light Client. 2020. Original author research formalizing the dynamic trust model and checking light-client safety and liveness with TLA+ and APALACHE; it does not prove an implementation of this design.
<https://arxiv.org/abs/2010.07031>
- [24] Tien Tuan Anh Dinh; Ji Wang; Gang Chen; Rui Liu; Beng Chin Ooi; Kian-Lee Tan. BLOCKBENCH: A Framework for Analyzing Private Blockchains. 2017. Original author paper, with affiliations at the National University of Singapore and Zhejiang University. Cited for overall and component-level throughput, latency, scalability and fault evaluation, without transferring its platform results.
<https://arxiv.org/abs/1703.04057>
- [25] Murugiah Souppaya; Karen Scarfone; Donna Dodson. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. 2022. NIST SP 800-218, DOI 10.6028/NIST.SP.800-218. Used to organize development, supply-chain and release evidence.
<https://csrc.nist.gov/pubs/sp/800/218/final>
- [26] Aggelos Kiayias; Philip Lazos; Paolo Penna. Single-token vs Two-token Blockchain Tokenomics. 2024; revised 2025, version 3. Original author research. Used to distinguish viability, decentralization, stability and feasibility; its mechanism assumptions and results do not prove this paper's supply choices, price or returns.
<https://arxiv.org/abs/2403.15429>
- [27] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Section 6 supports block rewards and initial distribution; early mining is not a developer-status allocation.
<https://bitcoin.org/bitcoin.pdf>
- [28] Hunter Beast; Ethan Heilman; Isabel Foxen Duke. BIP 360: Pay-to-Merkle-Root (P2MR). Draft version 0.12.1; accessed 2026-10-10. Draft. P2MR addresses long exposure and is not deployed full post-quantum signature protection.
<https://github.com/bitcoin/bips/blob/master/bip-0360.mediawiki>
- [29] Jameson Lopp; Christian Papathanasiou; Ian Smith; Joe Ross; Steve Vaile; Pierre-Luc Dallaire-Demers. BIP 361: Post Quantum Migration and Legacy Signature Sunset. Draft; accessed 2026-10-10. Migration draft in the official BIP repository, not activation or a commitment by Bitcoin.
<https://github.com/bitcoin/bips/blob/master/bip-0361.mediawiki>
- [30] Karl-Johan Alm; Oliver Gurger. BIP 322: Generic Signed Message Format. Complete version 2.0.0; accessed 2026-10-10. Official specification. Message signatures establish neither historical payment origin nor enduring control and cannot replace deposit outpoints.
<https://github.com/bitcoin/bips/blob/master/bip-0322.mediawiki>
- [31] Brink. Brink. Accessed 2026-10-10. Primary organizational description of donation-funded developers, without participation in or endorsement of this protocol.
<https://brink.dev/>

[32] Varul Srivastava; Sankarshan Damle; Sujit Gujar. Centralization in Proof-of-Stake Blockchains: A Game-Theoretic Analysis of Bootstrapping Protocols. 2024; arXiv 2404.09627 version 1. Original IIIT Hyderabad author research on initial stake, Sybil and concentration risks. Model ideality does not establish a real-world optimum or prove this proposal. <https://arxiv.org/abs/2404.09627>