Post Quantum Proof

A Post-Quantum Security Architecture for EVM-Compatible Execution

Abstract. This document specifies a research design for an EVM-compatible blockchain and account-control layer whose critical authorization paths are protected by post-quantum cryptographic proofs. The system is not presented as "unbreakable." Its security claim is reductionist: if an adversary can cause an unauthorized protected action, conflicting finality event, unauthorized validator-set mutation, or unauthorized key rotation to be accepted, then the adversary can be used to forge a post-quantum signature, violate the stated threshold-corruption bound, find a hash collision/preimage outside the chosen security margin, or exploit an explicitly declared bypass in the control graph. The design uses NIST-standardized post-quantum primitives as the intended algorithm family: ML-DSA for signatures, ML-KEM for key establishment, and SLH-DSA as a conservative hash-based signature fallback.

Technical Whitepaper Draft v0.2

May 30, 2026

This is a technical architecture draft. It is not an audit, implementation guarantee, or claim of absolute security.

Contents

1 Scope and Design Goals **2**
1.1 Non-Goals . . . 2

2 Background **2**
2.1 EVM as a State Transition System . . . 2
2.2 Post-Quantum Primitive Families . . . 3

3 Notation **3**

4 The Quantum Safety Layer **4**

5 Action Encoding **4**

6 Quantum Shield Proofs **5**

7 ML-DSA Mathematical Core **5**

8 Security Assumptions **6**

9 Authorization-Forgery Game **7**

10 Quantum Safety Registry **8**

11 Key Epochs and Rotation **8**

12 Control Graph and the No-Bypass Criterion **9**

13 Validator Set and Finality **9**
13.1 Conflicting Finality . . . 10

14 Validator Admission and Removal **11**

15 Governance Safety **12**

16 Bridge and External Message Safety **12**

17 Cryptographic Agility **13**

18 Precompile Interface **13**

19 Proof Object Encoding **14**

20 Safety Classes **14**

21 Threat Model **14**

22 Limitations and Open Research **15**

23 Implementation Roadmap as Protocol Work **15**

24 Conclusion **16**

Scope and Design Goals

Post Quantum Proof (PQP) is an EVM-compatible execution network with a post-quantum authorization and finality layer. The central objective is not to replace the EVM. The objective is to define a stronger security boundary around the EVM state transition function.

The design goals are:

1. Preserve EVM execution compatibility for contracts, storage, gas-metered computation, logs, and deterministic state transitions.
2. Replace critical authorization predicates with post-quantum verification predicates.
3. Support a governance-voted authority set for block finality, validator admission, validator removal, and key rotation.
4. Make quantum safety auditable through public registries, versioned proof objects, algorithm identifiers, and policy roots.
5. Define formal security games for unauthorized action acceptance, conflicting finality, validator-set mutation, and key-substitution attacks.
6. Maintain cryptographic agility so algorithms can be upgraded by policy rather than embedded as permanent assumptions.

Non-Goals

This paper does not claim:

- that any cryptographic system is absolutely unbreakable;
- that post-quantum primitives remove all smart-contract risk;
- that a quantum-safe signature alone makes a system quantum-safe;
- that all EVM software is secure by inheritance;
- that the final system can avoid implementation audits, side-channel review, parameter review, client testing, and operational controls.

The security target is narrower and more rigorous: protected state transitions should require proof objects whose forgery reduces to breaking defined post-quantum assumptions or violating explicit trust bounds.

Background

EVM as a State Transition System

The EVM may be modeled as a gas-bounded, stack-based virtual machine operating over persistent account and contract storage. Let the global state be S_t. Let tx be a transaction or system operation. The EVM transition function is:

$$S_{t+1} = \delta_{\mathrm{EVM}}(S_t, tx). \tag{1}$$

Post Quantum Proof does not redefine the contract bytecode execution model. Instead, it introduces a pre-execution acceptance predicate:

$$S_{t+1} = \begin{cases} \delta_{\mathrm{EVM}}(S_t, tx), & \text{if } \mathsf{Accept}_{\mathsf{QSL}}(S_t, tx) = 1, \\ S_t, & \text{otherwise.} \end{cases} \tag{2}$$

The predicate $\mathsf{Accept}_{\mathsf{QSL}}$ is the Quantum Safety Layer acceptance function. It binds syntax validity, nonce validity, epoch validity, registry state, policy state, and post-quantum proof verification.

Post-Quantum Primitive Families

NIST finalized the first three post-quantum cryptography standards in August 2024: FIPS 203 for ML-KEM, FIPS 204 for ML-DSA, and FIPS 205 for SLH-DSA [1, 2, 3, 4]. This paper uses those standards as the intended algorithm families:

- **ML-DSA** for transaction, validator, governance, bridge, treasury, and registry signatures.
- **ML-KEM** for session establishment, encrypted coordination, protected validator messaging, and quantum-safe communication channels.
- **SLH-DSA** as a stateless hash-based fallback for conservative emergency policies and long-horizon archive signatures.

The protocol separates signature authentication from key establishment. A key encapsulation mechanism is not used as a transaction signature scheme. A signature scheme is not used as a session encryption system.

Notation

Symbol	Meaning
λ	Security parameter.
Π_{PQ}	Post-quantum signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.
(pk, sk)	Public and secret key pair.
$\mathsf{H}_{\mathsf{QSL}}$	Domain-separated hash used by the Quantum Safety Layer.
a	Protected action.
m	Action digest, generally $m = \mathsf{H}_{\mathsf{QSL}}(a)$.
σ	Signature or proof material.
P	Quantum Authorization Policy.
$\mathsf{QSP}_P(a)$	Quantum Shield Predicate for action a under policy P.
$\mathcal{V}_e$	Active validator/finality set in epoch e.
W	Total finality weight.
$W_{\min}$	Minimum finality or authorization threshold.
C_{pk}	Public-key commitment, usually $C_{pk} = \mathsf{H}_{\mathsf{QSL}}(pk)$.
Reg	Registry state.
$\mathcal{A}^Q$	Quantum polynomial-time adversary.
$\mathsf{negl}(\lambda)$	Negligible function in the security parameter.

The Quantum Safety Layer

The Quantum Safety Layer is the protocol component that decides whether a state-changing operation is allowed to reach EVM execution or system-contract execution. It is defined as:

$$\mathsf{QSL} = (\mathsf{QSP}, \mathsf{QAP}, \mathsf{QSR}, \mathsf{QFV}, \mathsf{AlgReg}, \mathsf{Precompiles}). \tag{3}$$

Where:

- QSP is the proof object and verification predicate family.
- QAP is the policy language for authorization thresholds, epochs, domains, and key classes.
- QSR is the Quantum Safety Registry.
- QFV is Quantum Finality Verification.
- AlgReg maps algorithm identifiers to approved verification semantics.
- Precompiles are native verification interfaces for post-quantum proof checking.

The layer accepts an operation only if:

$$\mathsf{Accept}_{\mathsf{QSL}}(op) = \mathsf{Syntax}(op) \wedge \mathsf{Domain}(op) \wedge \mathsf{Nonce}(op) \wedge \mathsf{Epoch}(op) \wedge \mathsf{Registry}(op) \wedge \mathsf{Policy}(op) \wedge \mathsf{Proof}(op). \tag{4}$$

The security idea is simple: the EVM remains a deterministic execution engine, but protected actions must cross a post-quantum proof boundary first.

Action Encoding

A protected action is encoded as a typed tuple:

$$a = (d, c, s, u, p, n, e, \tau), \tag{5}$$

where d is a domain separator, c is the chain identifier, s is the subject being controlled, u is the function or operation class, p is the parameter commitment, n is the nonce, e is the active key epoch, and τ is the validity deadline.

The action digest is:

$$m_a = \mathsf{H}_{\mathsf{QSL}}(\texttt{PQP_ACTION} \parallel d \parallel c \parallel s \parallel u \parallel p \parallel n \parallel e \parallel \tau). \tag{6}$$

This prevents replay across domains, chains, contracts, function classes, epochs, and nonce spaces. A signature on one action is not a signature on another action because the encoded digest is domain-bound.

Quantum Shield Proofs

Definition 6.1 (Quantum Shield Proof)**.** *A Quantum Shield Proof for action a under policy P is a tuple:*

$$\pi_a = (v, alg, C_{pk}, e, r_P, \sigma, \eta), \tag{7}$$

where v is the proof version, alg is an algorithm identifier, C_{pk} is a public-key commitment, e is the key epoch, r_P is the policy root, σ is post-quantum signature material, and η is optional metadata such as signer index, threshold path, Merkle proof, or aggregation transcript.

For a single-signature policy, verification is:

$$\mathsf{QSP}_P(a, \pi_a) = 1 \tag{8}$$

if and only if:

$$\mathsf{Reg}[s].status = \mathsf{ACTIVE}, \tag{9}$$
$$\mathsf{Reg}[s].policyRoot = r_P, \tag{10}$$
$$\mathsf{Reg}[s].keyEpoch = e, \tag{11}$$
$$\mathsf{Reg}[s].pubKeyCommitment = C_{pk}, \tag{12}$$
$$\mathsf{H}_{\mathsf{QSL}}(pk) = C_{pk}, \tag{13}$$
$$\mathsf{Verify}_{alg}(pk, m_a, \sigma) = 1, \tag{14}$$
$$P(a, e, r_P, \eta) = 1. \tag{15}$$

For a threshold policy over a signer set $\mathcal{K} = \{pk_1, \ldots, pk_N\}$ with weights w_i, verification is:

$$\mathsf{QSP}_P(a, \Pi_a) = \mathbb{1}\left[\sum_{i=1}^{N} w_i \cdot b_i \geq W_{\min}\right] \cdot \mathbb{1}[P(a) = 1], \tag{16}$$

where:

$$b_i = \begin{cases} 1, & \mathsf{Verify}_{alg_i}(pk_i, m_a, \sigma_i) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

This defines protected authorization as a Boolean predicate over post-quantum signature verification, policy state, registry state, and threshold state.

ML-DSA Mathematical Core

This section gives a simplified mathematical model of the module-lattice structure used by ML-DSA. It is not a replacement for FIPS 204; it is a protocol-level explanation of the assumption class.

Let:

$$R_q = \mathbb{Z}_q[X]/(X^{256} + 1) \tag{18}$$

with modulus $q = 8380417$ in the ML-DSA family. Let:

$$A \in R_q^{k \times l}, \quad s_1 \in R_q^l, \quad s_2 \in R_q^k. \tag{19}$$

The public relation has the form:

$$t = As_1 + s_2 \pmod{q}. \tag{20}$$

The public key commits to A and t; the secret key contains short vectors related to s_1 and s_2. The security intuition is that, given (A, t), finding short (s_1, s_2) satisfying the relation is computationally infeasible under module-lattice assumptions.

A simplified Fiat-Shamir-with-aborts signing outline is:

$$\mu = \mathsf{H}(\mathsf{H}(pk) \parallel m), \tag{21}$$
$$y \leftarrow S_y, \tag{22}$$
$$w = Ay \pmod{q}, \tag{23}$$
$$w_1 = \mathsf{HighBits}(w), \tag{24}$$
$$c = \mathsf{H}(\mu \parallel w_1), \tag{25}$$
$$z = y + cs_1. \tag{26}$$

The signature contains values related to:

$$\sigma = (z, c, h), \tag{27}$$

with rejection sampling and hint mechanisms ensuring distributional and correctness properties. Verification reconstructs a high-bit value from:

$$Az - ct \pmod{q} \tag{28}$$

and accepts only if the challenge is consistent and norm bounds are satisfied:

$$\|z\| < B, \tag{29}$$
$$c = \mathsf{H}(\mu \parallel \mathsf{HighBits}(Az - ct)). \tag{30}$$

The protocol-level dependency is not that the EVM understands the entire algebra. The dependency is that the native verification function correctly implements the standardized verification relation for the chosen algorithm identifier.

Security Assumptions

The following assumptions define the security boundary of the whitepaper.

Assumption 8.1 (Post-Quantum Signature Unforgeability)**.** *For the selected post-quantum signature scheme* Π_{PQ}*, no quantum polynomial-time adversary can win the existential unforgeability under adaptive chosen-message attack game except with negligible probability:*

$$\Pr[\mathsf{EUF\text{-}CMA}^{\mathcal{A}^Q}_{\Pi_{\mathsf{PQ}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda). \tag{31}$$

Assumption 8.2 (Hash Robustness)**.** *The domain-separated hash* $\mathsf{H}_{\mathsf{QSL}}$ *provides collision and preimage resistance at the selected output size against quantum attackers up to the intended security margin.*

Assumption 8.3 (Bounded Byzantine Weight)**.** *For finality and validator-set safety, less than one-third of active finality weight equivocates within an epoch:*

$$W_{\mathsf{Byz}} < \frac{1}{3}W. \tag{32}$$

Assumption 8.4 (Correct Native Verification)**.** *Native precompiles or host functions for post-quantum verification implement the referenced standard faithfully and are constant-time or otherwise hardened enough for the deployment threat model.*

Assumption 8.5 (No Undeclared Bypass)**.** *Every critical control path is registered in the control graph and gated by a Quantum Shield Predicate. If an undeclared path exists, the system must be classified below full quantum safety.*

Authorization-Forgery Game

Let $\mathcal{O}_{\mathsf{Sign}}$ be a signing oracle. The challenger runs:

$$(pk, sk) \leftarrow \mathsf{KeyGen}(1^{\lambda}) \tag{33}$$

and gives pk to $\mathcal{A}^{Q}$. The adversary may query $\mathcal{O}_{\mathsf{Sign}}(m_i)$ and receive:

$$\sigma_i \leftarrow \mathsf{Sign}(sk, m_i). \tag{34}$$

The adversary wins if it outputs (m^*, σ^*) such that:

$$\mathsf{Verify}(pk, m^*, \sigma^*) = 1, \tag{35}$$

$$m^* \notin \{m_1, \ldots, m_q\}. \tag{36}$$

For action authorization, each protected action a maps to a digest m_a. An unauthorized accepted action implies a valid signature on a digest not authorized by the legitimate secret holder.

Theorem 9.1 (Protected Action Unforgeability)**.** *Assume* Π_{PQ} *is post-quantum EUF-CMA secure,* $\mathsf{H}_{\mathsf{QSL}}$ *is domain-separated and collision resistant under the chosen margin, and every critical action requires* $\mathsf{QSP}_P(a) = 1$*. Then no quantum polynomial-time adversary can cause acceptance of an unauthorized protected action except with negligible probability.*

Proof. Suppose an adversary $\mathcal{A}^{Q}$ causes the system to accept an unauthorized protected action a^*. Since the action is protected:

$$\mathsf{Accept}_{\mathsf{QSL}}(a^*) = 1 \Rightarrow \mathsf{QSP}_P(a^*) = 1. \tag{37}$$

By definition of QSP, there exists signature material σ^* and a registered public key pk such that:

$$\mathsf{Verify}_{\mathsf{PQ}}(pk, \mathsf{H}_{\mathsf{QSL}}(a^*), \sigma^*) = 1. \tag{38}$$

If a^* was not authorized, then the legitimate signer did not sign $\mathsf{H}_{\mathsf{QSL}}(a^*)$. Thus $\mathcal{A}^{Q}$ has produced a valid signature on a new message. This is an EUF-CMA forgery against Π_{PQ}, except in the case

where $\mathsf{H}_{\mathsf{QSL}}$ maps distinct authorized and unauthorized actions to the same digest. The latter is a hash collision. Therefore:

$$\Pr[\mathsf{UnauthorizedAccept}] \leq \Pr[\mathsf{Forge}_{\Pi_{\mathsf{PQ}}}] + \Pr[\mathsf{Collide}_{\mathsf{H}}] + \Pr[\mathsf{Bypass}]. \tag{39}$$

Under the assumptions, each term is negligible or explicitly outside the safety classification. Hence the theorem follows. □

Quantum Safety Registry

The Quantum Safety Registry records subjects and their active proof requirements. A subject may be an account, contract, treasury, validator, bridge, identity, system contract, or policy module.

A registry entry is:

$$R_s = (s, alg, C_{pk}, r_P, e, status, metaRoot). \tag{40}$$

The registry commitment is:

$$C_R = \mathsf{H}_{\mathsf{QSL}}(s \parallel alg \parallel C_{pk} \parallel r_P \parallel e \parallel status \parallel metaRoot). \tag{41}$$

A proof against the registry verifies:

$$\mathsf{Reg}[s].status = \mathsf{ACTIVE}, \tag{42}$$
$$\mathsf{Reg}[s].alg = alg, \tag{43}$$
$$\mathsf{Reg}[s].pubKeyCommitment = C_{pk}, \tag{44}$$
$$\mathsf{Reg}[s].policyRoot = r_P, \tag{45}$$
$$\mathsf{Reg}[s].keyEpoch = e. \tag{46}$$

This provides public inspectability. Any client, indexer, explorer, or application can evaluate the security posture of a subject by checking registry state and proof validity.

Key Epochs and Rotation

A key epoch prevents stale-key replay and provides explicit identity continuity. Let subject s have active public-key commitment:

$$C_e = \mathsf{H}_{\mathsf{QSL}}(pk_e). \tag{47}$$

A rotation to $C_{e+1} = \mathsf{H}_{\mathsf{QSL}}(pk_{e+1})$ is encoded as:

$$r_e = \mathsf{H}_{\mathsf{QSL}}(\texttt{PQP_KEY_ROTATION} \parallel s \parallel C_e \parallel C_{e+1} \parallel e \parallel r_P). \tag{48}$$

The default strong rotation rule requires signatures from both the outgoing and incoming key:

$$\mathsf{Verify}(pk_e, r_e, \sigma_e) = 1, \tag{49}$$
$$\mathsf{Verify}(pk_{e+1}, r_e, \sigma_{e+1}) = 1. \tag{50}$$

This prevents unilateral substitution of a new key by an operator who only controls registry access. The control path itself must remain QSP-gated.

Theorem 11.1 (Key-Substitution Resistance)**.** *Assume the rotation policy requires both old-key and new-key post-quantum signatures. An adversary who does not control sk_e or sk_{e+1} cannot rotate C_e to C_{e+1} except by forging at least one post-quantum signature or exploiting a registry bypass.*

Proof. The registry accepts a rotation only if both verification equations hold. If neither secret key signed the rotation digest, at least one accepted signature is a forgery. If the registry updates without those equations, the registry bypass assumption has been violated. Therefore the result follows. □

Control Graph and the No-Bypass Criterion

A system may use post-quantum signatures but still fail quantum safety if a hidden administrative edge bypasses them. PQP models control as a directed graph:

$$G = (V, E). \tag{51}$$

Vertices V are states, subjects, authorities, contracts, and system modules. Edges E are state-changing control actions.

Let $E_{crit} \subseteq E$ be critical edges. These include transfers from protected treasuries, minting, pausing, unpausing, upgrades, validator admission, validator removal, bridge release, bridge mint, key rotation, recovery, policy mutation, and finality-set mutation.

Definition 12.1 (No-Bypass Safety)**.** *A system satisfies No-Bypass Safety if:*

$$\forall e_i \in E_{crit} : \quad \mathsf{Exec}(e_i) = 1 \Rightarrow \mathsf{QSP}_{P_i}(e_i) = 1. \tag{52}$$

Definition 12.2 (Path Safety)**.** *A path $\rho = (e_1, \ldots, e_k)$ from public state to critical state is QSP-safe if:*

$$\exists e_j \in \rho \text{ such that } \mathsf{QSP}_{P_j}(e_j) = 1. \tag{53}$$

Definition 12.3 (Quantum-Sovereign Control)**.** *A control graph is quantum-sovereign if every path from public state to critical state is QSP-safe and every critical edge is No-Bypass Safe.*

This criterion is stricter than saying that a single account or key is post-quantum. It requires the entire administrative graph to cross post-quantum proof gates.

Validator Set and Finality

The network uses a governance-voted authority set. Validators are admitted, removed, and rotated through QSP-gated registry operations. Block finality is represented by threshold signatures or threshold proof sets over the active validator set.

Let the active validator set in epoch e be:

$$\mathcal{V}_e = \{(id_i, pk_i, w_i)\}_{i=1}^{N}. \tag{54}$$

Let total finality weight be:

$$W = \sum_{i=1}^{N} w_i. \tag{55}$$

For block header commitment:

$$B_h = \mathsf{H}_{\mathsf{QSL}}(\texttt{PQP_BLOCK} \parallel h \parallel parent \parallel stateRoot \parallel txRoot \parallel receiptRoot \parallel e \parallel validatorSetHash). \tag{56}$$

Each validator may sign:

$$\sigma_i = \mathsf{Sign}(sk_i, B_h). \tag{57}$$

The block is quantum-final if:

$$\mathsf{QFV}(B_h) = 1 \tag{58}$$

where:

$$\mathsf{QFV}(B_h) = \mathbb{1}\left[\sum_{i=1}^{N} w_i \cdot \mathsf{Verify}(pk_i, B_h, \sigma_i) \geq W_{final}\right] \tag{59}$$

and:

$$W_{final} > \frac{2}{3}W. \tag{60}$$

Conflicting Finality

Two blocks at the same height conflict if:

$$h(B) = h(B') \quad \wedge \quad B \neq B'. \tag{61}$$

Let S_B and $S_{B'}$ be the validator subsets whose valid signatures finalized B and B'.

Lemma 13.1 (Intersection Bound)**.** *If $W(S_B) > 2W/3$ and $W(S_{B'}) > 2W/3$, then:*

$$W(S_B \cap S_{B'}) > W/3. \tag{62}$$

Proof. Using weighted set inclusion:

$$W(S_B \cap S_{B'}) = W(S_B) + W(S_{B'}) - W(S_B \cup S_{B'}). \tag{63}$$

Since $W(S_B \cup S_{B'}) \leq W$:

$$W(S_B \cap S_{B'}) > \frac{2W}{3} + \frac{2W}{3} - W = \frac{W}{3}. \tag{64}$$

□

Theorem 13.1 (Quantum Finality Safety)**.** *Assume less than one-third of finality weight equivocates and the post-quantum signature scheme is unforgeable. Then two conflicting blocks cannot both be quantum-final except with negligible probability.*

Proof. If two conflicting blocks finalize, signer sets for each must exceed $2W/3$. By the intersection bound, more than $W/3$ finality weight is in the intersection. Any signer in the intersection must have signed both conflicting block commitments or a signature must have been forged. If less than $W/3$ weight equivocates, then at least one honest signature on a conflicting commitment was forged. This contradicts post-quantum signature unforgeability. Thus conflicting finality occurs only with negligible probability under the stated assumptions. □

Validator Admission and Removal

A candidate validator registers:

$$C_v = \mathsf{H}_{\mathsf{QSL}}(pk_v) \tag{65}$$

and signs a registration digest:

$$m_{reg} = \mathsf{H}_{\mathsf{QSL}}(\texttt{PQP_VALIDATOR_REGISTER} \parallel id_v \parallel C_v \parallel nodeMetaRoot \parallel e). \tag{66}$$

The registry accepts the candidate only if:

$$\mathsf{Verify}(pk_v, m_{reg}, \sigma_v) = 1. \tag{67}$$

Admission to the active finality set is a protected action:

$$a_{add} = (\texttt{VALIDATOR_ADD}, id_v, C_v, e_{activate}, r_P, n). \tag{68}$$

and requires:

$$\mathsf{QSP}_{P_{gov}}(a_{add}) = 1. \tag{69}$$

Removal is analogous:

$$a_{remove} = (\texttt{VALIDATOR_REMOVE}, id_v, reasonRoot, e_{remove}, r_P, n). \tag{70}$$

with:

$$\mathsf{QSP}_{P_{gov}}(a_{remove}) = 1. \tag{71}$$

Validator-set changes activate only at epoch boundaries:

$$\mathcal{V}_{e+1} = \mathsf{ApplyQueuedChanges}(\mathcal{V}_e, \mathcal{Q}_e). \tag{72}$$

This prevents single-block mutation of the finality set and makes light-client verification deterministic by epoch.

Governance Safety

Governance actions are among the most sensitive operations. A chain cannot be quantum-safe if the finality layer is quantum-safe but the governance layer can mutate it without a QSP gate.

Let G be the governance state. Let a governance action be:

$$g = (target, selector, parameters, proposalId, nonce, epoch). \tag{73}$$

The execution predicate is:

$$\mathsf{Exec}_G(g) = 1 \Rightarrow \mathsf{QSP}_{P_G}(g) = 1 \wedge \mathsf{Delay}(g) \wedge \mathsf{Quorum}(g). \tag{74}$$

For high-risk actions, such as validator-set mutation, proof-verifier upgrade, algorithm registry upgrade, bridge-control mutation, and emergency policy mutation, the recommended form is conjunctive:

$$\mathsf{Exec}_G(g) = 1 \Rightarrow \mathsf{QSP}_{P_{vote}}(g) = 1 \wedge \mathsf{QSP}_{P_{council}}(g) = 1 \wedge \mathsf{Timelock}(g) = 1. \tag{75}$$

This means that voting, council control, and delay are separate conditions. If any one condition is compromised, the others remain barriers.

Bridge and External Message Safety

Bridge security is modeled as threshold authorization over message commitments.

Let a bridge message be:

$$M_b = \mathsf{H}_{\mathsf{QSL}}(\texttt{PQP_BRIDGE} \parallel src \parallel dst \parallel eventRoot \parallel token \parallel amount \parallel recipient \parallel nonce). \tag{76}$$

Let bridge signers be:

$$\mathcal{B} = \{(pk_i, w_i)\}_{i=1}^{N_B}. \tag{77}$$

The bridge release or mint action is accepted only if:

$$\sum_{i=1}^{N_B} w_i \cdot \mathsf{Verify}(pk_i, M_b, \sigma_i) \geq W_{bridge}. \tag{78}$$

Theorem 16.1 (Bridge Forgery Bound)**.** *If fewer than the bridge threshold weight is corrupt and the signature system is unforgeable, a false bridge release cannot be accepted except with negligible probability or a source-chain verification failure.*

Proof. A false bridge release requires a threshold-valid signature set over M_b without a valid source event. If insufficient signer weight is corrupt, at least one required signature must be forged, unless the source-chain event verification itself is false. Thus the claim reduces to signature unforgeability plus source verification correctness. □

Cryptographic Agility

Quantum safety is not a permanent property of one algorithm. PQP therefore uses versioned algorithms and proof formats.

An algorithm registry maps:

$$algID \mapsto (name, version, securityLevel, verifyAddress, status, activationEpoch, retirementEpoch). \tag{79}$$

A proof domain includes the algorithm identifier:

$$D_{alg} = \mathsf{H}_{\mathsf{QSL}}(\texttt{PQP_ALG} \parallel algID \parallel version \parallel chainID). \tag{80}$$

The action digest includes D_{alg} to prevent replay between algorithm versions:

$$m_a = \mathsf{H}_{\mathsf{QSL}}(D_{alg} \parallel a). \tag{81}$$

Retirement is handled by policy:

$$\mathsf{AlgAllowed}(alg, e) = 1 \iff e_{activate}(alg) \le e < e_{retire}(alg) \wedge status = \mathsf{ACTIVE}. \tag{82}$$

This allows the protocol to add new algorithms or retire old algorithms without rewriting the conceptual security model.

Precompile Interface

Post-quantum verification is too expensive to implement naively in ordinary EVM bytecode. PQP therefore requires native precompiles or host functions for verification.

A minimal precompile interface is:

```
PQP_VERIFY(algID, publicKey, messageDigest, signature) -> bool
PQP_PUBKEY_HASH(algID, publicKey) -> bytes32
PQP_BATCH_VERIFY(algID, publicKeys[], digest, signatures[]) -> bitmap
```

A validator/finality verifier may use a system-level host function:

```
PQP_FINALITY_VERIFY(epoch, blockDigest, signerBitmap, signatures) -> bool
```

The implementation must define:

- exact byte encoding for keys and signatures;
- canonical digest length;
- maximum key/signature sizes per algorithm;
- error semantics;
- gas cost model;
- batch-verification semantics;
- constant-time or side-channel-hardened implementation requirements.

Proof Object Encoding

A canonical proof object can be encoded as:

```
struct QuantumShieldProof {
    uint16  version;
    uint16  algId;
    uint64  keyEpoch;
    bytes32 subject;
    bytes32 policyRoot;
    bytes32 publicKeyCommitment;
    bytes32 actionDigest;
    bytes   publicKey;       // optional if retrievable from registry
    bytes   signature;
    bytes   policyWitness;   // threshold index, Merkle branch, bitmap, etc.
}
```

The action digest is computed outside the proof object and must bind to chain, subject, nonce, domain, parameters, and deadline.

Replay resistance requires:

$$\mathsf{NonceUsed}(s, n) = 0 \tag{83}$$

before execution and:

$$\mathsf{NonceUsed}(s, n) \leftarrow 1 \tag{84}$$

after execution.

Safety Classes

PQP classifies subjects by the completeness of their proof-gated control graph.

Class	Definition
QSC-0	No registered quantum-safe key or proof policy.
QSC-1	Quantum-safe key registered, but critical paths are not fully QSP-gated.
QSC-2	All declared critical paths require at least one valid QSP.
QSC-3	Critical paths require threshold QSP authorization.
QSC-4	Critical paths, recovery paths, upgrade paths, finality paths, bridge paths, and key-rotation paths satisfy No-Bypass Safety and threshold authorization.

A subject cannot self-declare QSC-4. It must be derivable from registry state, policy state, and control-graph analysis.

Threat Model

The adversary may:

- run quantum polynomial-time algorithms;
- observe all public chain data;
- request signatures from compromised or authorized users on chosen messages;
- corrupt a bounded fraction of validator/finality weight;
- corrupt non-critical infrastructure;
- attempt replay, key substitution, stale epoch use, domain confusion, and proof malleability;
- exploit smart-contract bugs in unprotected application logic;
- exploit implementation bugs in precompiles or clients.

The adversary is not assumed able to:

- forge the selected post-quantum signature scheme beyond negligible probability;
- break the selected hash function beyond the declared margin;
- corrupt threshold weight above the stated bound;
- bypass registered control paths without such bypass being treated as a failure of the system classification.

Limitations and Open Research

The design has practical challenges:

- Post-quantum signatures and public keys are larger than classical signatures and require careful calldata, storage, and bandwidth design.
- Native verification must be implemented and audited. A faulty precompile invalidates the proof model.
- Batch verification semantics must be selected carefully; incorrect batching can introduce soundness hazards.
- Side channels matter. Validator signing infrastructure must harden key generation, signing, and storage.
- Control-graph completeness is non-trivial. A contract may have indirect upgrade or administrative paths that are difficult to statically detect.
- Cryptographic agility creates governance risk. Algorithm-registry changes must themselves be QSP-gated and timelocked.
- Interoperability with non-PQ chains can reduce end-to-end security to the weakest linked system.

These limitations are not peripheral. They define the engineering work required to make the architecture real.

Implementation Roadmap as Protocol Work

This paper avoids marketing milestones and instead defines required protocol artifacts:

1. Formal byte encoding for QSP objects.

2. Canonical action digest specification.
3. Registry system contract specification.
4. Algorithm registry and retirement rules.
5. ML-DSA verification precompile.
6. SLH-DSA verification precompile for fallback policies.
7. ML-KEM session protocol for protected off-chain coordination.
8. Epoch-based validator-set update rules.
9. Finality proof format.
10. Control-graph scanner for QSC classification.
11. Client test vectors.
12. Fault-injection tests for invalid signatures, stale epochs, wrong domains, threshold failures, and equivocation evidence.

Conclusion

Post Quantum Proof is a proposed EVM-compatible security architecture in which critical state transitions are accepted only when they satisfy post-quantum proof predicates. The system's core claim is reductionist rather than rhetorical:

$$\mathsf{UnauthorizedAccept} \Rightarrow \mathsf{PQSignatureForgery} \vee \mathsf{HashFailure} \vee \mathsf{ThresholdCorruption} \vee \mathsf{DeclaredBypass}. \quad (85)$$

This gives the architecture a precise audit target. Quantum safety is not a label applied to a chain. It is a property of the authorization graph, finality graph, registry state, algorithm registry, and proof-verification implementation.

An EVM chain becomes post-quantum credible only when every critical path - transaction control, validator admission, validator rotation, finality, bridge release, treasury control, recovery, registry mutation, and upgrade authority - crosses a post-quantum proof boundary.

That is the design principle of Post Quantum Proof.

References

[1] National Institute of Standards and Technology, "NIST Releases First 3 Finalized Post-Quantum Encryption Standards," August 13, 2024. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

[2] National Institute of Standards and Technology, FIPS 203, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," August 13, 2024. https://csrc.nist.gov/pubs/fips/203/final

[3] National Institute of Standards and Technology, FIPS 204, "Module-Lattice-Based Digital Signature Standard," August 13, 2024. https://csrc.nist.gov/pubs/fips/204/final

[4] National Institute of Standards and Technology, FIPS 205, "Stateless Hash-Based Digital Signature Standard," August 13, 2024. https://csrc.nist.gov/pubs/fips/205/final

[5] Gavin Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Yellow Paper. https://ethereum.github.io/yellowpaper/paper.pdf

[6] Ethereum Foundation, "Ethereum Execution Layer Specifications." https://github.com/ethereum/execution-specs

[7] Miguel Castro and Barbara Liskov, "Practical Byzantine Fault Tolerance," Proceedings of the Third Symposium on Operating Systems Design and Implementation, 1999. https://www.usenix.org/conference/osdi-99/practical-byzantine-fault-tolerance

[8] Ethereum Improvement Proposal 4337, "Account Abstraction Using Alt Mempool." https://eips.ethereum.org/EIPS/eip-4337