EVM.fail

The shortest path to post-quantum authentication on an EVM

Christopher Smith

November 2026

This report examines the shortest practical path for an EVM-compatible chain to offer post-quantum authentication. It focuses on the protocol features required to let accounts and privileged keys use lattice signatures, and it deliberately separates that minimal path from longer-term work on consensus, aggregation, and pairing-based proof systems.

Contents

Key takeaways **1**

1 Core concepts **2**
1.1 Pre-quantum cryptography on EVM chains 2
1.2 Post-quantum cryptography 4

2 Relevant EIPs **4**
2.1 Signature agility and key registration 5
2.2 Post-quantum signature verification 5

3 Ethereum roadmap items not required for a minimal path **5**

4 Large EVM chains, L2s, and projects **6**

5 Recommendation **7**
5.1 Execution layer 7
5.2 Consensus layer 8
5.3 Later steps 8
5.4 Scope and adoption 9

6 Conclusion **9**

Key takeaways

- Quantum computers break the discrete-log, elliptic-curve, and pairing assumptions used by ECDSA, BLS, KZG, and pairing-based SNARKs. Hash functions and symmetric primitives are only quadratically weakened and remain usable.
- The shortest path for an EVM to offer post-quantum authentication does not require a new consensus layer or a forced cut-over. It requires two execution-layer features: a lightweight Keystore or native account-abstraction mechanism (EIP-8130 or equivalent) and ML-DSA verification precompiles covering the higher NIST parameter sets (EIP-8355).

- Those two features let a chain offer post-quantum transaction authorization before Ethereum finishes its own roadmap. Consensus signatures are covered only if that chain also migrates validator deposits and registration. Bridges, existing contracts, and funds held by keys whose public keys are already exposed are not. The chain makes the signatures available. Applications and holders still have to use them.
- Security is prioritized over performance. The recommended path therefore uses lattice signatures at NIST levels 3 and 5, and accounts can adopt them before aggregation or leanVM exists. Retiring ECDSA and replacing pairing-based proofs come later. Both belong to a finished migration.
- Pure rollups inherit L1 security and need only the execution-layer steps. Chains that run their own validators need additional deposit and registration changes. No generic multi-authenticator validator account exists today.
- Pairing-based blob commitments and on-chain SNARK verifiers remain a separate, longer-term migration. For any check that occurs on Ethereum L1, an L2 cannot complete that replacement unilaterally.
- Most large EVM projects have not yet published independent post-quantum roadmaps beyond Ethereum's own timeline. The execution-layer path described in this report can be adopted by any EVM without waiting for those roadmaps.

1 Core concepts

1.1 Pre-quantum cryptography on EVM chains

Ethereum and EVM-compatible chains rely on several pre-quantum asymmetric cryptographic primitives whose security rests on the hardness of the elliptic-curve discrete logarithm problem (ECDLP) or related pairing problems. These are efficient on classical computers but are broken by Shor's algorithm on a sufficiently powerful quantum computer. The main attack surfaces are execution-layer account signatures, consensus-layer validator signatures, and pairing-based proof systems.

Execution-layer account signatures (ECDSA over secp256k1). Externally owned accounts (EOAs) authorize transactions with the Elliptic Curve Digital Signature Algorithm on the secp256k1 curve. The private key is a 256-bit scalar; the corresponding public key is a point on the curve. Addresses are derived by taking the Keccak-256 hash of the uncompressed public key and keeping the last 20 bytes.

The public key is not stored on-chain until the account first spends. When a transaction is submitted, the signature is included and the public key is recovered via `ecrecover`. Once the public key is known, a quantum adversary that can solve the ECDLP can recover the private key and forge signatures. Accounts that have never spent, or whose public key has never been revealed, remain protected until that moment. This exposure model is the same one used by Bitcoin and most other ECDSA-based chains.

Since the Pectra upgrade (May 2025), EIP-7702 allows an EOA to set a delegation designator in its code field, pointing to a smart-contract implementation. The original ECDSA key remains the ultimate authority that can set or clear the delegation. The underlying signature scheme for

ordinary transactions and for the authorization tuples themselves is still secp256k1 ECDSA.

Consensus-layer validator signatures (BLS over BLS12-381). Ethereum's proof-of-stake consensus uses Boneh–Lynn–Shacham (BLS) signatures on the BLS12-381 curve. Each validator has a BLS public key (48 bytes on G_1) and signs with a corresponding private key. BLS was chosen because signatures aggregate efficiently: hundreds of thousands of attestations can be combined into a single aggregate signature that is cheap to verify.

BLS security relies on the hardness of the discrete logarithm in pairing-friendly groups. Shor's algorithm also breaks this assumption. Unlike EOA keys, validator public keys are published on-chain at deposit time and remain permanently visible in the beacon state. A quantum adversary that can run Shor's algorithm can therefore forge attestations and block proposals for any validator whose key has been deposited.

Zero-knowledge and polynomial commitments (pairing-based). Pairing-based systems (KZG, Groth16, PLONK) rely on the same discrete-log and pairing assumptions. Once those assumptions fall, an attacker can forge a proof of a false statement that a correct verifier will accept. The verifier still runs the same pairing check; it simply no longer distinguishes true proofs from forged ones.

These appear in several places:

- **Blob commitments (KZG over BLS12-381).** EIP-4844 commits to blob data with KZG. The point-evaluation precompile (`0x0A`) checks openings. A quantum break lets an adversary open a commitment to data other than what was published, breaking the binding between the on-chain commitment and the posted blob.
- **ZK-rollup state transitions.** Validity proofs that a batch of L2 transactions is correct. The L1 contract verifies a Groth16 or PLONK proof, via the pairing precompiles, and accepts the new state root if the proof passes. A forged proof lets an attacker force an invalid state transition.
- **Privacy applications.** Mixers, privacy pools, and private transfer systems that verify a SNARK proving a valid spend or membership. A forged proof lets an attacker spend without a real witness or violate the protocol's constraints.
- **Bridges and cross-chain verification.** Contracts that accept a SNARK as evidence of a state or consensus claim on another chain. A forged proof can be used to withdraw assets or accept a false claim.
- **Recursive aggregation.** Many proofs are compressed off-chain into a single final Groth16 or PLONK proof that is then verified on L1. A quantum break compromises both the intermediate pairing-based steps and the final on-chain check.

In every case the failure mode is the same: a forged proof is accepted by a correct verifier.

Hash functions (Keccak-256). Addresses, Merkle proofs, and most integrity checks use Keccak-256. Grover's algorithm gives a quadratic speedup against collision and preimage resistance, reducing 256-bit security to roughly 128-bit security. The runtime for a single iteration of Grover's algorithm is also typically very long for hash functions, because the hash function must

be implemented in quantum gates. This renders the attack effectively useless. Hash functions such as Keccak-256 are therefore generally regarded as plausibly post-quantum.

1.2 Post-quantum cryptography

Quantum computers do not threaten all cryptographic primitives equally. Public-key systems whose security does not rest on the discrete logarithm problem are generally considered post-quantum. This includes symmetric primitives and hash functions, as well as a suite of public-key cryptosystems designed to remain secure against both classical and quantum adversaries. The two families most relevant to blockchains are lattice-based signatures and hash-based signatures.

Lattice-based signatures (ML-DSA / Dilithium, Falcon). Security is based on the hardness of problems over module lattices (Module-LWE, Module-SIS, and related variants). ML-DSA was standardized by NIST (FIPS 204) in three parameter sets targeting NIST security levels 2, 3, and 5. Signatures are relatively compact, roughly 2–5 KB, and verification consists of a small number of modular multiplications plus a short rejection-sampling check. Falcon is smaller still but has a more complex signing procedure. Both are stateless.

Hash-based signatures (XMSS / leanXMSS, SPHINCS+ / SLH-DSA). Security reduces to the preimage and collision resistance of a hash function. XMSS and its variants, including the leanXMSS parameters under discussion for Ethereum consensus, are stateful: each signature consumes a one-time key, and the signer must never reuse a leaf index. SPHINCS+ / SLH-DSA is stateless but produces larger signatures, tens of kilobytes. Verification is dominated by hash evaluations and is therefore expensive in gas when performed naively on the EVM.

Practical constraints on an EVM. Signature and public-key size affect calldata cost and bandwidth. Verification cost appears as gas or as consensus-critical time. Aggregation properties matter for consensus: neither family aggregates natively the way BLS does. Key-management complexity differs as well. Lattice schemes are stateless, while XMSS-style schemes require tracking used indices. These constraints determine which schemes are realistic for user accounts, sequencer keys, and validators.

Migration style. An EVM can move to post-quantum signatures either by opt-in signature agility or by a forced cut-over. Agility lets accounts keep their addresses and voluntarily add or rotate to a new authenticator, leaving ECDSA available. A cut-over is a hard fork that stops accepting the old scheme, usually paired with a recovery path. Agility can be deployed incrementally and avoids stranding users. A cut-over removes residual exposure but requires coordinated upgrades and a clear deadline. Most current designs start with agility, then follow it with a cut-over once enough accounts have moved that a deadline will not strand the rest.

2 Relevant EIPs

Various EIPs relevant to the quantum threat already exist. Of those discussed below, only EIP-7702 is live on a public network. EIP-8130 and EIP-8141 have client implementations and have run on devnets. The verification precompiles and the consensus-key proposals are drafts, and none of them has been activated on a network. They fall into two groups: signature agility and key registration, and execution-layer verification of post-quantum signatures.

2.1 Signature agility and key registration

- **EIP-7702 (Final, Pectra 2025).** Lets an EOA set code and delegate to a contract. This is an already-live stepping stone that turns an EOA into a smart account without changing its address. The original ECDSA key remains the authority that can set or clear the delegation.
- **EIP-8130 (Keystore accounts).** A lightweight account-abstraction design from Base. An account registers one or more authenticators, with rules for scope, expiry, and policy, in an on-chain keystore. New signature schemes can be added later without changing the account address, so an account can switch to a post-quantum authenticator by rotating keys. It works with ordinary transactions, ERC-4337, or a native account-abstraction transaction, and it is simpler and more sequencer-friendly than the broader Frame Transactions proposal.
- **EIP-8141 (Frame Transaction).** A broader native account-abstraction proposal, and a must-ship candidate for Hegotá. It decomposes a transaction into programmable frames for validation, gas payment, and execution. It provides signature agility, and it is heavier and more general than EIP-8130.
- **EIP-8450 (Post-quantum validator key registry).** The consensus-layer parallel. It adds a beacon-state registry so each existing validator can register one dormant leanXMSS public key bound to its validator index. Keys stay unused until a later fork. The design is currently scheme-specific (leanXMSS), not a general multi-authenticator model.
- **EIP-8394 (deposit-contract redesign).** Makes *new* validator deposits multi-scheme capable, with variable-length public keys and an explicit scheme identifier (scheme 0 is BLS). This is the primary flexibility mechanism for the validator entry path. No fully generic validator account with pluggable authenticators, comparable to EIP-8130, exists today.

2.2 Post-quantum signature verification

- **EIP-8355 (precompiles for ML-DSA verification).** Three precompiles covering ML-DSA-44, ML-DSA-65, and ML-DSA-87 (NIST levels 2, 3, and 5). Each takes a concatenated public key, signature, and message and returns success or failure. This proposal directly supports the higher security levels.
- **EIP-8051.** An earlier ML-DSA precompile proposal, mainly level 2 plus an EVM-optimized variant.
- **EIP-7932 (secondary signature algorithms).** An algorithm registry and a `SIGRECOVER`-style precompile. A foundation for adding new schemes without rewriting transaction formats.

There is no hash-based precompile EIP, largely because hash-based signature verification can already be done with existing opcodes (mainly `KECCAK256`).

3 Ethereum roadmap items not required for a minimal path

Several major roadmap items improve throughput, state management, or fairness, but they are not prerequisites for authenticating with post-quantum signatures. They can be deferred without blocking a minimal post-quantum migration.

- **Scaling.** PeerDAS, full danksharding, enshrined proposer-builder separation (EIP-7732), and

inclusion lists (FOCIL) increase data throughput and censorship resistance. None of them changes the signature schemes used by accounts or validators.

- **State management.** Verkle trees, state expiry, and statelessness reduce the cost of storing and verifying state. They are orthogonal to the choice of signature algorithm.
- **Privacy and MEV mitigation.** Most privacy protocols and MEV-mitigation designs, including encrypted mempools and inclusion lists beyond FOCIL, operate above the signature layer. They can adopt post-quantum signatures later through the same agility mechanisms used by ordinary accounts.
- **Consensus redesigns.** Proposals that restructure fork choice, finality, or validator duties for reasons other than post-quantum security (for example, faster finality gadgets) are independent of the signature-scheme transition.

These items remain valuable. None of them is required before an EVM can offer opt-in post-quantum authentication for user accounts, sequencers, or, where applicable, validators.

4 Large EVM chains, L2s, and projects

Post-quantum planning is uneven. A few projects have published timelines or concrete EIPs. Most others have not yet separated execution-layer agility from consensus-layer migration.

Ethereum L1. The Ethereum Foundation's post-quantum team has stated a target of a quantum-resistant L1, covering execution, consensus, and data availability, by roughly December 2029. Near-term milestones on its planning roadmap (the Strawmap) are the post-quantum key registry in the I* fork (EIP-8450, currently leanXMSS-focused, allowing existing validators to register a dormant post-quantum key) and post-quantum signature precompiles in the J* fork, so contracts and wallets can verify post-quantum signatures natively. Later milestones cover validator attestations and aggregation. Current lean consensus prototypes favor hash-based signatures over lattice schemes.

Optimism / Superchain / Base. The Superchain has published an explicit 10-year ECDSA-EOA deprecation timeline ending around January 2036. The OP Stack already supports signature-scheme changes via hard forks. Base is advancing a Keystore-style design (EIP-8130) that lets accounts register scheme-agnostic authenticators. Sequencer and batcher keys are treated as in scope for the same migration path.

Arbitrum. As of late 2026, Arbitrum has not published a post-quantum roadmap or a public cryptographic inventory of its sequencer, batcher, or validator keys.

Starknet. Starknet already has native account abstraction. It has demonstrated Falcon-based accounts and continues to use STARK-friendly hashes (Poseidon and others) in its proving system. These experiments are execution-layer only. Starknet's own consensus is outside the EVM. Its finality rests on STARK proofs verified on Ethereum L1, not on an EVM validator set.

Injective. Injective is a Cosmos-based L1 with native EVM support. It uses secp256k1 for accounts and Ed25519 for consensus. In mid-October 2026 it announced that quantum-resistant

infrastructure work is under way, but it has not published chosen schemes, a roadmap, or a timeline.

zkSync. Native account abstraction is already live. Its current Airbender prover uses hash-based STARKs internally, claimed post-quantum at the proof-soundness layer, but the final on-chain verifier on Ethereum is still pairing-based. No date has been published for post-quantum account signatures.

Other L2s. Most other EVM L2s, including Polygon, Scroll, Linea, and Mantle, have not published independent post-quantum roadmaps and continue to rely on pairing-based validity proofs. They inherit Ethereum's timeline for any protocol-level changes.

5 Recommendation

Security is prioritized over performance. The first step is account authentication. Aggregation, retirement of ECDSA, and replacement of pairing-based proofs come afterward, and they do not all play the same role.

5.1 Execution layer

The shortest path for an EVM to offer post-quantum authentication does not require a new consensus layer or a forced cut-over. It requires two protocol features that can be adopted independently of Ethereum L1's own timeline. Privileged keys should then move onto the same path.

1. Lightweight native account abstraction / Keystore. A Keystore, EIP-8130 or a close equivalent, lets any account register one or more authenticators together with scope, expiry, and policy rules. The account address itself does not change. New signature schemes can be added later simply by registering a new authenticator contract. The same mechanism works for ordinary user accounts, including EOAs that have set code via EIP-7702, for smart accounts, and for privileged operator keys. Because the design is scheme-agnostic, an account can start with ECDSA and later add or rotate to an ML-DSA or hash-based authenticator without a further hard fork.

2. ML-DSA verification precompiles. EIP-8355 gives the Keystore, or any other account-abstraction validation logic, a cheap way to verify lattice signatures. The proposal specifies three parameter sets. The higher ones, ML-DSA-65 and ML-DSA-87, are preferred for long-lived account keys because they target NIST security levels 3 and 5. Verification cost is low enough that ordinary transactions and operator transactions remain practical. The recommendation favors ML-DSA for its smaller signatures, greater flexibility, and statelessness. The protocol features required for hash-based signatures are the same.

3. Migrate privileged keys onto the same path. Sequencer, batcher, bridge-operator, and admin keys should use the same Keystore and ML-DSA mechanism as ordinary user accounts. This removes the most valuable targets first and avoids a separate, one-off migration process for operator infrastructure. Once both features are live, any account on the chain can opt into post-quantum authentication without waiting for further protocol changes. Opting in is what protects

an account. Shipping the features does not migrate existing keys, or the applications built on the chain.

5.2 Consensus layer

This step applies only to chains that run their own validators. Pure rollups generally do not operate a beacon chain or a large independent validator set. They inherit finality and data-availability security from Ethereum L1, or from another base chain, and therefore need only the execution-layer steps for their sequencers, batchers, and user accounts.

Chains that run their own consensus face an additional migration.

- **Deposit path.** New validators must be able to register a post-quantum key at deposit time. An EIP-8394-style multi-scheme deposit contract, or an equivalent, is required so that a validator can supply either a classical or a post-quantum public key when it enters the set.
- **Existing validators.** A registration mechanism is needed so validators already in the set can add a post-quantum key without exiting and re-depositing. EIP-8450 is the current leading design, but it is written for leanXMSS. A lattice-based consensus path would require additional design work to define the key format, the verification method inside consensus clients, and the aggregation or reduced-check strategy.
- **Account model.** No generic multi-authenticator validator account comparable to EIP-8130 exists today. Consensus keys are still bound to a single scheme chosen at registration time. Extending the Keystore model to validators would simplify rotation and hybrid operation, but that extension has not been proposed.

Until those pieces are in place, a chain that operates its own validator set cannot offer post-quantum consensus signatures, even if its execution layer already supports them.

5.3 Later steps

The execution-layer path is the shortest way to offer post-quantum authentication. It is the start of the migration. Three further pieces of work sit outside that first step, for different reasons.

- **Signature aggregation and leanVM.** These cut the on-chain cost of large or frequent post-quantum signatures. A single account can authenticate with a lattice signature once the Keystore and precompiles exist. Aggregation matters when a chain must verify many such signatures in one block, including a chain that moves its validators onto lattice signatures. That is a cost of scale, and it can be solved after the account path is live.
- **Retiring ECDSA.** Opt-in agility leaves the old scheme in place, so an exposed key stays forgeable until its assets move. A later hard fork can stop accepting ECDSA and give a recovery path to accounts that have not migrated. That fork should wait until adoption is high enough that a deadline will not strand the accounts still on the old scheme. The fork is how the account migration ends.
- **Replacing KZG and on-chain SNARK systems.** Blob commitments and pairing-based verifiers remain forgeable after a quantum break of the curves. Replacing them, with hash-based STARKs, lattice commitments, or new precompiles, is what keeps proofs and data commitments sound, and the work is independent of account authentication. For any check

that occurs on Ethereum L1, the replacement requires an Ethereum upgrade. An L2 cannot complete it unilaterally. It should be scheduled once accounts can already authenticate with lattice signatures.

Sequencing this work after the account path keeps the first step to two protocol features. Aggregation follows as load requires it. Retiring ECDSA, and replacing pairing-based proofs, are how the migration is completed.

5.4 Scope and adoption

The changes above are ones a chain's maintainers can make themselves, without waiting for Ethereum to finish its quantum-security roadmap. A Keystore (EIP-8130 or an equivalent) together with the ML-DSA verification precompiles (EIP-8355) protects transaction authorization: an account can register a post-quantum authenticator, and the chain can verify it. Consensus is covered only for a chain that also adopts the deposit and registration changes in Section 5.2. Validators that have not registered a post-quantum key remain exposed.

Applications built on the chain are out of scope. The fork does not upgrade bridges or existing smart contracts. It gives users and builders the option to use post-quantum signatures. It does not replace `ecrecover` in contracts that already rely on it, and it does not replace pairing-based checks. Those remain the longer-term work described above.

This is a design recommendation. The combination has not been implemented or tested. Only EIP-7702 is live on a public network. EIP-8130 and EIP-8141 have run on devnets, and the other proposals have not been activated on a network.

The report covers the chain, not the application layer. Post-quantum signatures are optional, so a chain can ship them without waiting for wallets, exchanges, and custodians. Those applications can accept both signature types at first and default to post-quantum. Funds are protected as applications and holders adopt the new signatures, not at the moment the chain ships.

An account whose public key is already exposed remains forgeable by a quantum adversary until its assets leave that key. Most holders will have that move done for them by an exchange or a custodian. A wallet can instead prompt the holder to generate a new seed phrase and transfer the assets to new post-quantum addresses derived from that seed. The Keystore path can keep an existing address, but it does not by itself move assets off a key that is already exposed.

6 Conclusion

An EVM can offer post-quantum authentication with two execution-layer changes: a Keystore that lets accounts register and rotate authenticators, and ML-DSA precompiles that support the higher NIST parameter sets. These are enough for the chain to let user accounts and privileged keys use lattice signatures without a new consensus layer or a forced cut-over. They do not by themselves protect validators, bridges, existing contracts, or assets held under keys that are already exposed. Wallets, exchanges, and custodians can move those assets once the chain has shipped. Aggregation and leanVM follow as the cost of many signatures becomes the constraint. Retiring ECDSA removes the residual account exposure, and replacing pairing-based proofs is what keeps proofs and blob commitments sound. A chain that runs its own validators also has

a consensus-key migration to do. The path to offering the signatures is available to any EVM today.