Post-Quantum Migration for Bitcoin and Ethereum: Hidden-Key Proofs, Finalization, and Enforcement

Mehmet Sabir Kiraz[1] and Suleyman Kardas[2]
mehmet.kiraz@dmu.ac.uk suleyman.kardas@batman.edu.tr

[1] De Montfort University, Leicester, UK
[2] Batman University, Batman, Turkiye

Abstract. Quantum computers threaten the elliptic-curve signatures used by Bitcoin and Ethereum once a public key appears on-chain. Hidden-key credentials, including those controlling Bitcoin P2PKH/P2WPKH outputs and unused Ethereum EOAs, have a different risk profile: their public keys are hidden behind hash-derived addresses, but classical spending reveals them. We propose a migration design that separates revealed-key and hidden-key credentials. Revealed-key credentials may use hybrid classical/post-quantum authorization only while the legacy signing secret remains uncompromised; after quantum recovery of that secret, an independent rescue policy or pre-committed recovery credential is required to distinguish the owner from an attacker. Hidden-key credentials use STARK-verifiable ownership relations that bind a legacy identifier to a commitment to a post-quantum public key without explicitly publishing the legacy elliptic-curve public key as a public value. Complete enrollment opens the commitment and verifies post-quantum key possession. The design adds one-way migration-registry semantics, quantum hash-security analysis, scenario-based network-capacity evaluation, and an SP1-based prototype. We distinguish Groth16/PLONK wrapping, which is only a current-EVM compatibility path, from an intended quantum-era transparent verification path; the exact post-quantum security of a concrete SP1 deployment depends on the pinned SP1 version, proof mode, and all cryptographic subarguments used by that configuration.

Keywords: Quantum Computing · Post-Quantum Cryptography (PQC) · Blockchain · Bitcoin · Ethereum · STARK proofs.

1 Introduction

Recent progress in quantum computing raises a significant threat to current cryptographic methods, especially elliptic-curve cryptography (ECC), which secures networks such as Bitcoin and Ethereum [41,10,33,12]. Algorithms such as Shor's algorithm could efficiently break these widely used cryptographic primitives by solving discrete logarithm and factoring problems, weakening the fundamental security of cryptocurrencies [42,16,34]. Since blockchain networks

like Bitcoin and Ethereum protect billions in funds and support widely used decentralized applications, the emergence of functional quantum computers could expose these funds to theft, leading to theft and loss risks to the financial system [6,22,19]. Hence, ensuring long-term security in the blockchain domain requires early migration to quantum-resistant cryptographic standards.

Consider a long-unspent P2PKH or P2WPKH output whose public key has never appeared on-chain. Such an output is protected by a hash-derived address until it is spent. If the owner spends it in a future quantum-capable era, the spending transaction reveals the public key and may create a race window before confirmation. This motivates a hidden-key migration mechanism that proves ownership and binds the asset to a post-quantum public key without publishing the legacy public key. This differs from early P2PK outputs, including many Satoshi-era coinbase outputs, where the public key may already be visible on-chain and the asset belongs to the revealed-key category.

This paper studies this migration problem as a systems-and-cryptography question rather than as a single opcode or wallet upgrade. The key design distinction is whether the legacy public key has already been revealed. If it has, a hybrid classical/post-quantum authorization step is meaningful only before the legacy private key has become recoverable by the adversary; after such compromise, the classical credential no longer distinguishes the legitimate owner from an attacker. If the public key has not been revealed, the migration mechanism should avoid making legacy public-key publication a prerequisite for ownership proof. Our construction therefore combines post-quantum signatures with STARK-verifiable ownership relations and a one-way migration registry. We use the term *hidden-key migration* for the latter path: the proof binds a legacy hash-derived address to a fresh post-quantum public key while keeping the legacy public key out of the public statement and out of the on-chain transaction data.

Scope. This paper addresses the enrollment and key-binding layer of post-quantum migration. A registry alone cannot disable the existing authorization rules of Bitcoin or Ethereum. Enforcing post-quantum-only future spending requires a separate deployment mechanism, such as a consensus change, a contract-enforced wrapper, a Layer 2 registry, or a future account/UTXO migration rule. Our registry semantics are intended to serve as the interface between the enrollment proof and such enforcement mechanisms. In the Bitcoin case, the migrated identifier is normally a UTXO or key-hash commitment; in the Ethereum case, it is an account-authority binding rather than the account's complete off-chain and cross-domain authority surface. The protocol also does not solve post-compromise attribution for revealed keys: once an adversary can derive a legacy private key, signatures under that key can no longer identify the historical owner.

2 Legacy-Key Exposure Model in Bitcoin and Ethereum

The migration path depends on whether the legacy elliptic-curve public key is still hidden. In Bitcoin, P2PKH and P2WPKH outputs expose only $h_{160} =$

RIPEMD160(SHA256(Q)) until spending, whereas P2PK contains the public key directly and P2TR commits to a 32-byte x-only Taproot output key in the UTXO. Thus P2TR is a revealed-key case for the Shor-style threat model even though optional script paths are committed separately. In Ethereum, an EOA address is the low 20 bytes of the Keccak-256 hash of the uncompressed secp256k1 public key; an unused EOA can therefore remain a hidden-key credential until it originates a signed transaction, after which the public key is recoverable from the transaction signature. These hash-hiding properties apply only provided that the legacy public key has not been disclosed elsewhere.

Table 1. Bitcoin output type and legacy-key exposure at output creation

Output type	**Key visibility at creation**
P2PK	Public key visible in the locking script
P2PKH/P2WPKH	Public key hidden behind HASH160 until spending
P2TR	32-byte x-only Taproot output key visible

This classification applies to ordinary and coinbase outputs alike: the fact that an output originates from a coinbase transaction does not determine whether its key is hidden; the locking-script type does. When a P2PKH/P2WPKH output is spent, the public key is supplied in the script/witness and becomes observable. For Ethereum, the analogous transition occurs when an EOA originates a signed transaction and sender recovery makes its public key derivable from public transaction data; the EVM's `ecrecover` precompile provides the same recovery primitive to contracts. Consequently, a hidden-key enrollment must avoid using the target credential itself merely to submit or fund migration.

Exposure classification cannot rely on chain history alone. Exported public keys or extended-public-key material, multisig/co-signer workflows, signed messages, EIP-712/permit-style signatures, custody or service integrations, or other key-sharing interfaces may disclose a public key without an on-chain revealing transaction. A wallet or migration service should therefore treat a credential as revealed-key whenever the public key is known outside the owner's protected environment.

Once a secp256k1 public key is available, Shor's algorithm threatens the corresponding discrete-log secret. Babbush et al. report 256-bit ECDLP circuits using fewer than approximately 1,200 logical qubits and 90 million Toffoli gates, or fewer than approximately 1,450 logical qubits and 70 million Toffoli gates; under their stated superconducting architecture, error-rate, and connectivity assumptions, they estimate execution in minutes with fewer than roughly half a million physical qubits [4]. These are architecture-dependent estimates, not evidence that such a machine exists today, but they strengthen the case for migration before the threat becomes operational. Hidden-key credentials are comparatively better protected because Shor cannot be applied directly to an

unknown public key; they retain only a hash barrier, whose single- and multi-target quantum margins are analyzed in Section 8.

For Ethereum, migration of an EOA's L1 account authority does not automatically revoke every authority associated with the same ECDSA key. Token approvals, permit-based flows, bridges, Layer 2 accounts, `ecrecover`-based applications, and administrative roles may require application-specific migration or explicit registry integration. Our Ethereum scope is therefore execution-layer EOA authority; validator BLS keys, consensus-layer authorization, and KZG-related cryptography are outside scope.

3 Related Work

3.1 Recent Bitcoin and Ethereum Migration Proposals

Several recent proposals directly address the migration problem and define the closest comparison points. BIP-360 proposes Pay-to-Merkle-Root (P2MR), removing the Taproot key-path spend and preparing script-tree outputs for future post-quantum integration [7]. BIP-361 proposes staged Bitcoin migration and a legacy-signature sunset [29]. Buterin outlines an Ethereum emergency hard fork with STARK-based recovery [15]. Chaincode Labs survey Bitcoin's quantum threat and migration design space [30]; Babbush et al. provide updated secp256k1 resource estimates and on-spend attack analysis [4]; Coinbase's advisory work considers blockchain migration policy [18]; and Fan et al. propose Ethereum migration using quantum-safe ZKPs and proof aggregation [21]. Our distinction is not the observation that migration is needed, but the composition of hidden-key proof relations with explicit state-binding, post-quantum possession, non-revealing submission, one-way registry semantics, and deployment/capacity analysis across Bitcoin- and Ethereum-style systems.

Broader literature establishes the surrounding PQC-migration and blockchain context without providing the same composed hidden-key finalization mechanism. Alghamdi and Almuhammadi survey cryptocurrency exposure and candidate PQ signatures [1]; Alnahawi et al. and Wiesmaier et al. study PQC migration and crypto-agility [3,39]; Tan and Zhou classify blockchain migration strategies and deployment impact [37]; Gomes et al. and Saha et al. focus primarily on post-quantum consensus or new PQ blockchain constructions [25,35]; Thakur et al. address enterprise PQC inventory and migration tooling [38]; Gharavi et al. survey PQ blockchain security for IoT [23]; Baseri et al. provide blockchain quantum-threat modelling and migration architectures [6]; and Sezer et al. systematize privacy-preserving PQ blockchain designs [14]. Sultan et al. propose a grace-period transition followed by invalidation of unconverted legacy coins [2], illustrating the sunset/non-migrator policy dimension that we analyze separately in Section 10.

4 Our Contributions

This work does not claim novelty for the observation that revealed-key and hidden-key credentials have different risk profiles, nor for the standard distinction

Table 2. Positioning Against Recent Post-Quantum Blockchain Migration Proposals

Work	**BTC**	**ETH**	**Hidden-key**	**Impl.**	**Main focus**
BIP-360/P2MR [7]	✓	–	Partial	Draft/spec	New Bitcoin output type with Taproot key-path removed; prepares script-tree outputs for future PQ signatures.
BIP-361 [29]	✓	–	Rescue-oriented	Draft/spec	Legacy-signature sunset and staged migration incentives after a PQ output type exists.
Buterin [15]	–	✓	Recovery-oriented	Proposal	Emergency Ethereum hard fork, wallet migration, and STARK-based recovery after a quantum incident.
Fan et al. [21]	–	✓	Limited	Prototype	Ethereum-focused PQ migration using ZK proofs, proof URIs, and aggregation.
Milton–Shikhelman [30]	✓	–	Discussed	Report	Bitcoin threat model, exposure categories, and migration design space.
Babbush et al. [4]	✓	✓	Discussed	Analysis	Quantum resource estimates, public-mempool/on-spend attacks, and mitigations.
This work	✓	✓	✓	Prototype	Hidden-key relations plus composed finalization obligations, one-way registry semantics, and capacity analysis.

between collision and preimage resistance. Our contribution is instead to identify and compose the checks required to turn a hidden-key ownership proof into a deployable post-quantum migration mechanism across Bitcoin- and Ethereum-style systems. The main contributions are:

1. **Formal hidden-key enrollment relations and composed finalization conditions.** We specify Bitcoin and Ethereum relations that bind a legacy identifier to a post-quantum key commitment without publishing the legacy elliptic-curve public key, and we make explicit that proof validity alone is insufficient for production finalization. In particular, Bitcoin finalization additionally requires authenticated UTXO-to-key-hash state binding, both systems require post-quantum proof of possession, and hidden-key enrollment requires non-revealing transaction submission.
2. **Provisional-to-migrated registry semantics.** We distinguish proof/evidence recording from irreversible migration: a registry entry remains *provisional* until the applicable finalization obligations succeed, after which a one-way migrated status/nullifier can be consumed by a consensus-, contract-, or Layer 2 enforcement mechanism.
3. **Crypto-agile, role-separated statement design.** The migration message binds the legacy identifier, destination commitment, suite, epoch, migration mode, and chain context. We separate message-hash and binding-hash roles, enabling suite-specific security targets while preventing replay and destination-key substitution.
4. **Deployment and ecosystem-scale evaluation.** We analyze enforcement choices, non-migrating users, enrollment granularity, and Bitcoin/Ethereum capacity under explicit assumptions, including the distinction between per-UTXO and grouped enrollment and between coordination-only and complete PQ-authorization paths.
5. **Prototype and verification-path analysis.** We implement the migration logic using SP1 guest programs, a Rust SDK, and an EVM verification contract. We separate the current Groth16/PLONK compatibility wrappers from an intended quantum-era transparent verification path and state that post-quantum security must be established for the exact pinned proving backend and all cryptographic subarguments.

5 End-to-End Migration Lifecycle

A complete migration has six stages. First, wallets, custodians, and indexers classify each credential or account-authority binding as revealed-key or hidden-key using both on-chain history and off-chain exposure information (Section 2). Second, the owner generates a post-quantum key pair under the selected suite. Third, before legacy-key compromise the owner produces either a hybrid revealed-key enrollment object or a hidden-key proof; if a revealed legacy key is already compromised, an independent rescue policy is required rather than dual signatures alone. Fourth, a valid proof may create a *provisional* record, but production

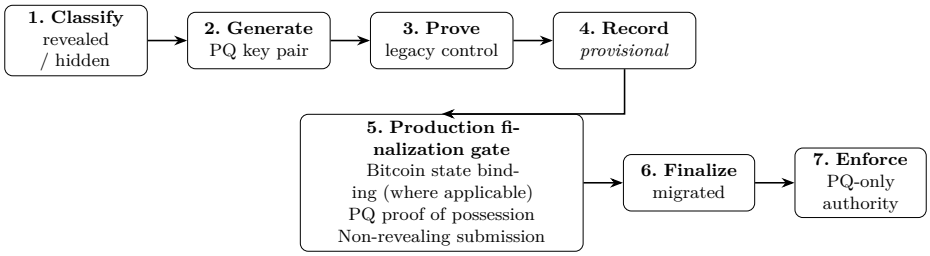

Fig. 1. End-to-end migration lifecycle and production finalization gate. A valid ownership proof may create only a provisional record; the transition to an enforcement-recognized migrated state occurs only after every applicable finalization obligation succeeds.

finalization is governed by the three obligations in Section 7.3: Bitcoin state binding where applicable, PQ proof of possession, and non-revealing submission for hidden credentials. Fifth, the deployment's consensus, contract, account, or Layer 2 mechanism activates the new authority. Sixth, post-migration spending or key rotation proceeds through the post-quantum path defined by that enforcement mechanism.

Operationally, a Bitcoin `OP_RETURN` commitment is only an audit/coordination record unless an enforcement rule recognizes it, and a hidden-key target UTXO must not fund its own anchor. Likewise, a hidden-key Ethereum EOA must not originate its own enrollment transaction; a relayer, sponsor, or separate funded account submits the proof. The relayer is not trusted for authorization because the signed/proved message binds the destination and context, but it remains part of the liveness and ordering threat model.

Figure 1 summarizes this end-to-end flow and highlights the distinction between proof acceptance, provisional recording, and production finalization.

6 Post-Quantum Migration Framework for Legacy Bitcoin and Ethereum Authorities

We introduce a framework that combines STARK-verifiable relations [8] with a lattice-based signature scheme (ML-DSA, standardized in FIPS 204 [31]) to enable blockchain address owners to bind legacy authority to quantum-resistant key material without intentionally including a hidden legacy elliptic-curve public key as a declared public input or transaction field. This is a public-key non-publication property, not by itself a zero-knowledge guarantee for the proof transcript. The design covers two distinct threat models—revealed-key and hidden-key credentials—and provides a graduated migration path for each.

We instantiate the prototype with ML-DSA because it is a NIST-standardized post-quantum signature scheme with relatively compact signatures compared with hash-based alternatives. The construction is not tied to ML-DSA: the `suiteID` identifies a complete cryptographic suite, including the exact signature parameter

set, public-key encoding, signature-verification rule, and role-separated migration hashes. We write $H_{\mathsf{msg,suite}}$ for message/transcript hashing and $H_{\mathsf{bind,suite}}$ for collision-sensitive binding commitments such as c_{pq} and canonical registry leaves. Hash-based signatures such as SLH-DSA may be preferable when conservative hash-based security assumptions are prioritized, while ML-DSA offers smaller signatures and practical implementation costs. Because FIPS 205 defines multiple SLH-DSA parameter sets, a production `suiteID` must name the exact set rather than the generic family [32].

Table 3. Example production-oriented crypto-agile suite identifiers with role-separated hashes

suiteID	**PQ signature**	H_{msg}	H_{bind}	**Verification rule**
`0x01`	ML-DSA-65	SHA-512/256	SHA-384	Verify ML-DSA-65 signature
`0x02`	ML-DSA-87	SHA-512/256	SHA-512	Verify ML-DSA-87 signature
`0x03`	SLH-DSA-SHA2-128s	SHA3-256	SHA3-384	Verify exact FIPS-205 parameter set

Security note. The table separates message hashing from binding commitments so that the suite is consistent with the hash-role analysis below. A 256-bit message hash gives an approximately 2^{128} generic quantum targeted-preimage bound under Grover search, whereas a binding commitment whose conservative design target is approximately 128-bit generic quantum collision *query complexity* can select a 384-bit-or-wider output. These rows are production-oriented examples; the present prototype uses narrower 256-bit Keccak/SHA-family commitments in some paths and is therefore not claimed to instantiate the full 128-bit quantum-collision target for H_{bind}.

6.1 Circuit Design Overview and Commitment Rationale

The hidden-key circuits implement the formal relations of Section 7. For Bitcoin, the witness contains the exact encoded secp256k1 public key, a legacy ECDSA signature, and the candidate pk_{pq}; the public statement contains the chain-bound output identifier, h_{160}, c_{pq}, keyEncoding, suiteID, epochID, and mode. The circuit rederives HASH160, verifies the legacy signature on the canonical migration message, validates the suite-specific PQ-key encoding, and checks $H_{\mathsf{bind,suite}}(pk_{pq}) = c_{pq}$. The Ethereum circuit is analogous, replacing HASH160/outpoint semantics with Ethereum address derivation and chainID. The normative statements, witnesses, message layouts, and complete conditions are given once in Sections 7.1–7.2.

Committing to pk_{pq} is not intended to keep the PQ public key secret: complete enrollment opens it. The commitment c_{pq} instead gives the proof and registry a fixed-size, suite-defined binding value, avoids placing a multi-kilobyte PQ key directly in the STARK public values, and decouples proof generation from the later PQ-possession gate.

6.2 Proof Generation and Non-Revealing Submission

Users generate PQ keys offline, prove the relevant legacy-to-c_{pq} relation, and submit the proof together with the public statement. The legacy elliptic-curve public key remains witness data rather than an explicit public statement or transaction field. Operational preservation of this property requires separate transaction funding: an unused Ethereum EOA uses a relayer/sponsor/separate funded account, and a target Bitcoin P2PKH/P2WPKH UTXO is not spent merely to pay for its own anchor. The relayer may censor, delay, copy, or reorder a submission but cannot substitute a different destination without violating the signed/proved binding. Non-publication is not itself a zero-knowledge claim; witness privacy depends on the concrete proving configuration.

6.3 Potential Migration Scenarios

We distinguish two scenarios based on the exposure status of the classical public key, as each entails a different threat model and requires a different enrollment path.

Scenario A: Revealed Public Keys – Pre-Compromise Hybrid Migration. Once a user's elliptic-curve public key appears on-chain, a sufficiently capable quantum computer can in principle recover the corresponding private key with Shor's algorithm [36]. A crucial security boundary follows: a classical signature is useful as evidence of continuity only *before* that legacy private key has become recoverable by the adversary. If a cryptographically relevant quantum computer has already recovered the legacy secret, the attacker can generate its own post-quantum key pair and produce both the legacy and post-quantum attestations accepted by a naive dual-signature enrollment. The protocol then has no purely cryptographic way to distinguish the historical owner from the attacker.

Accordingly, the revealed-key path is a *pre-compromise migration mechanism*. Post-compromise rescue requires an independent signal that the attacker cannot derive from the exposed legacy key, for example a pre-committed recovery credential, a previously registered recovery policy, an institutional/custodial recovery process, or an explicitly governed freeze-and-recovery mechanism. A "first valid enrollment wins" rule is not sufficient after legacy-key compromise because it merely turns recovery into a race.

Before compromise, the hybrid enrollment proceeds as follows:

1. **Post-quantum key generation.** The user generates a post-quantum key pair (sk_{pq}, pk_{pq}) under the selected $\mathsf{suiteID}$.
2. **Legacy-scheme dispatch.** The migration object includes a $\mathsf{legacySchemeID}$ that identifies the exact legacy verification rule. Bitcoin P2PK/P2PKH/P2WPKH credentials use secp256k1 ECDSA, Ethereum EOAs use secp256k1 ECDSA, while Taproot key-path credentials use BIP 340 Schnorr signatures over

secp256k1.[3] The current SP1 guest programs implement the ECDSA path; P2TR is therefore classified by the framework but BIP 340 verification is not claimed as implemented by the present guest artifact.

3. **Classical attestation.** The user signs the canonical domain-separated migration message with the legacy private key according to $\mathsf{legacySchemeID}$, proving continuity with the existing identifier only while that credential remains unforgeable by the adversary.
4. **Post-quantum attestation.** The user signs the same migration message with sk_{pq}, producing σ_{pq}. This proves possession of the proposed post-quantum key and prevents accidental or malicious binding to an unrelated PQ public key in the pre-compromise model.
5. **Registration.** A complete enforcement path verifies both attestations and records the migrated status or nullifier. In the present EVM prototype, full on-chain ML-DSA verification is deferred and a hash of σ_{pq} is recorded as an audit anchor; this is a compatibility mechanism, not a complete irreversible authority update by itself.
6. **Post-quantum continuation.** Once the relevant enforcement mechanism is active, future transactions or account-authority updates for the migrated identifier are authorized under pk_{pq} using post-quantum signatures.

Scenario B: Hidden Public Keys - Hidden-Key Migration via STARK Proofs. For a long-unspent P2PKH/P2WPKH UTXO or an unused Ethereum EOA whose public key has not been disclosed on- or off-chain, the hash-derived identifier prevents a direct Shor attack until the public key becomes available. The migration path is:

1. **Post-quantum key generation.** Generate (sk_{pq}, pk_{pq}) under the selected $\mathsf{suiteID}$.
2. **Proof generation.** Prove the Bitcoin or Ethereum hidden-key relation with the legacy public key, ECDSA signature, and candidate pk_{pq} as witness/auxiliary inputs and $c_{pq} = H_{\mathsf{bind,suite}}(pk_{pq})$ in the public statement.
3. **PQ possession.** Sign the applicable canonical message—m_{BTC} or m_{ETH}—with sk_{pq}. A proof or signature hash may be recorded provisionally, but it is not a finalized migration.
4. **State binding and submission.** Bitcoin additionally requires authenticated UTXO-to-h_{160} state binding; hidden-key submissions use a relayer or separate funding so that enrollment itself does not reveal the legacy key.
5. **Finalization.** The deployment sets the irreversible migrated status/nullifier only after the applicable obligations of Section 7.3 succeed. The current EVM leaf insertion remains provisional because the prototype does not verify ML-DSA on chain.
6. **Post-quantum continuation.** Once the enforcement path is active, the migrated UTXO, wrapped asset, or account-authority binding is controlled through pk_{pq} or an explicitly defined PQ key-rotation policy.

[3] BIP 340: `https://bips.dev/340/`.

Remark 1. The STARK relation proves a legacy-to-PQ binding; it does not itself disable Bitcoin's or Ethereum's legacy authorization rule. Moreover, if a hidden-key proof is used after a cryptographically relevant quantum computer exists, the selected proving mode must provide the witness privacy required by that threat model; supplying a value as witness data alone does not guarantee zero knowledge.

Ethereum delegation via EIP-7702. EIP-7702 provides a concrete present-day mechanism by which an Ethereum EOA can delegate execution to registry-aware or account-abstraction-style code. It is therefore relevant to the enforcement layer, especially for pre-compromise/revealed-key migration. However, EIP-7702 does not by itself create an irreversible post-quantum authority switch: its delegation authorization is still produced under the legacy secp256k1 ECDSA key, and that legacy authority can later authorize a different delegation. Moreover, an EIP-7702 authorization contains an ECDSA signature from which the legacy public key is recoverable, so using it for a previously unused EOA would defeat the hidden-key non-publication objective. We therefore treat EIP-7702 as a useful enforcement hook for already-revealed or deliberately transitioned EOAs, not as a standalone hidden-key-preserving PQ migration mechanism.[4]

7 Formal Migration Relations and Critical Security Claims

This section makes explicit the migration relations proved by the STARK circuits. The goal is not merely to verify a computation, but to bind a legacy asset identifier to a post-quantum public key while keeping the legacy elliptic-curve public key outside the public statement.

7.1 Bitcoin Hidden-Key Migration Relation

For a Bitcoin P2PKH or P2WPKH output, let the public statement be

$$x_{\mathsf{BTC}} = (\mathsf{assetID}, h_{160}, c_{pq}, \mathsf{keyEncoding}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{mode}),$$

where $\mathsf{assetID}$ is a chain-bound legacy output identifier. Canonically, $\mathsf{assetID} = \mathrm{Enc}(\mathsf{chainDomain}_{\mathsf{BTC}}, \mathsf{txid}, \mathsf{vout})$, where $\mathsf{chainDomain}_{\mathsf{BTC}}$ is a deployment-specific identifier that distinguishes the intended Bitcoin migration domain even from descendant chains that share the same historical genesis block. The value h_{160} is the visible public-key hash, c_{pq} is the public commitment to the candidate post-quantum key, $\mathsf{keyEncoding}$ records the exact legacy encoding rule, and mode distinguishes hidden-key migration from revealed-key migration. The STARK witness is

$$w_{\mathsf{BTC}} = (Q_{enc}, \sigma_{\mathsf{ecdsa}}, pk_{pq}),$$

[4] Ethereum Improvement Proposal 7702, "Set EOA account code," `https://eips.ethereum.org/EIPS/eip-7702`.

where Q_{enc} is the exact secp256k1 public-key encoding used to derive the legacy address, σ_{ecdsa} is an ECDSA signature under Q_{enc}, and pk_{pq} is the candidate post-quantum public key. The relation $R_{\mathsf{BTC-hidden}}(x_{\mathsf{BTC}}, w_{\mathsf{BTC}}) = 1$ iff:

1. $h_{160} = \mathrm{RIPEMD160}(\mathrm{SHA256}(Q_{enc}))$ and Q_{enc} is consistent with $\mathsf{keyEncoding}$. P2PKH may bind a compressed or uncompressed public-key encoding according to the legacy output; standard P2WPKH requires a compressed 33-byte key and therefore rejects an uncompressed witness key.[5]
2. $\mathrm{ValidPQKey}(pk_{pq}, \mathsf{suiteID}) = 1$ and $H_{\mathsf{bind,suite}}(pk_{pq}) = c_{pq}$;
3. the migration message is domain-separated and binds the destination commitment:

$$m_{\mathsf{BTC}} = H_{\mathsf{msg,suite}}\big(\mathrm{Enc}(\texttt{"PQ-MIG-BTC-v1"}, \mathsf{assetID}, h_{160}, c_{pq}, \mathsf{keyEncoding}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{mode})\big);$$

4. $\mathrm{ECDSA.Verify}(Q_{enc}, m_{\mathsf{BTC}}, \sigma_{\mathsf{ecdsa}}) = 1$.

The STARK proves the cryptographic relation but does not query Bitcoin chain state. Production finalization therefore composes it with

$$\mathrm{CheckUTXOState}(\mathsf{assetID}, h_{160}, \mathsf{keyEncoding}) = 1$$

and the PQ-possession gate. The exact semantics and deployment requirements of these checks are defined once in Section 7.3.

7.2 Ethereum Hidden-Key Migration Relation

For an Ethereum externally owned account, let

$$x_{\mathsf{ETH}} = (a_{\mathsf{eth}}, c_{pq}, \mathsf{chainID}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{mode})$$

and let the STARK witness be

$$w_{\mathsf{ETH}} = (Q_u, \sigma_{\mathsf{ecdsa}}, pk_{pq}),$$

where Q_u is the 65-byte uncompressed secp256k1 public key and pk_{pq} is the candidate post-quantum public key. The relation $R_{\mathsf{ETH-hidden}}(x_{\mathsf{ETH}}, w_{\mathsf{ETH}}) = 1$ iff:

1. $a_{\mathsf{eth}} = \mathrm{Keccak256}(Q_u[1{:}65])[12{:}32]$;
2. $\mathrm{ValidPQKey}(pk_{pq}, \mathsf{suiteID}) = 1$ and $H_{\mathsf{bind,suite}}(pk_{pq}) = c_{pq}$;
3. $$m_{\mathsf{ETH}} = H_{\mathsf{msg,suite}}\big(\mathrm{Enc}(\texttt{"PQ-MIG-ETH-v1"}, a_{\mathsf{eth}}, \mathsf{chainID}, c_{pq}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{mode})\big);$$
4. $\mathrm{ECDSA.Verify}(Q_u, m_{\mathsf{ETH}}, \sigma_{\mathsf{ecdsa}}) = 1$.

[5] BIP 143, "Restrictions on public key type": `https://bips.dev/143/`.

The inclusion of chainID prevents cross-chain replay, while epochID and suiteID support migration windows and agility. Ethereum needs no separate UTXO state-binding predicate because the address is itself the legacy account identifier, but production finalization still requires the PQ-possession gate of Section 7.3. This binds an account-authority identifier; it does not automatically migrate token approvals, contract roles, bridge permissions, or other external authority associated with the same ECDSA key.

7.3 Production Finalization Obligations

The proof relations are necessary but not sufficient to finalize an irreversible production migration. The enforcement layer must satisfy three additional obligations:

1. **Bitcoin legacy-state binding.** For Bitcoin hidden-key enrollment, verify $\mathrm{CheckUTXOState}(\mathsf{assetID}, h_{160}, \mathsf{keyEncoding}) = 1$ before consuming the nullifier. This prevents a prover who controls some unrelated hidden key from naming a victim's assetID together with the prover's own h_{160}. Consensus-native enforcement can use the UTXO set directly; external registries require an authenticated Bitcoin state mechanism, and transaction inclusion alone does not establish current unspentness.
2. **Post-quantum possession gate.** Verify the opening $H_{\mathsf{bind},\mathsf{suite}}(pk_{pq}) = c_{pq}$ and a valid σ_{pq} over the canonical migration message before setting `status=migrated` or irreversibly consuming the production nullifier. A STARK proof, a PQ-key length check, or a stored hash of σ_{pq} is insufficient by itself.
3. **Non-revealing submission for hidden credentials.** The enrollment transport must not exercise the hidden legacy credential merely to pay fees. An unused Ethereum EOA uses a relayer/sponsor/separate funded account; a Bitcoin target P2PKH/P2WPKH UTXO is not spent to fund its own anchor. The relayer is untrusted for authorization but is included in the liveness and ordering threat model.

A prototype may record proof evidence before these obligations are all satisfied, but such a record is *provisional*. In this paper, *finalization* means the transition from provisional evidence to an enforcement-recognized migrated state after every applicable obligation above has succeeded.

7.4 Chain Domain Separation and Canonical Serialization

The concatenation symbol used informally above is not a protocol encoding. For deployment, every signed or proved migration message is constructed from a canonical, injective byte encoding Enc. We define $\mathrm{Enc}(F_1, \ldots, F_k)$ as a versioned typed sequence in which each field has an unambiguous type and canonical byte representation and every variable-length field is length-prefixed. Integer widths and endianness are fixed by the suite specification; non-canonical encodings are rejected. The same encoder must be used by wallets, the zkVM guest, off-chain verifiers, and any consensus or contract enforcement layer.

For Bitcoin, the chain-bound asset identifier is

$$\mathsf{assetID} = \mathrm{Enc}(\mathsf{chainDomain}_{\mathsf{BTC}}, \mathsf{txid}, \mathsf{vout}),$$

where $\mathsf{chainDomain}_{\mathsf{BTC}}$ is a migration-domain identifier defined by the deployment specification. It should commit to sufficient protocol/deployment context (for example, a canonical network descriptor plus the migration deployment/activation identifier) so that two chains sharing historical transaction identifiers cannot accept the same enrollment unintentionally. For Ethereum, $\mathsf{chainID}$ already provides the primary chain-domain field, and the deployment may additionally bind a registry or protocol-version identifier when multiple migration registries coexist on the same chain.

Accordingly, the normative message definitions are interpreted as

$$m_{\mathsf{BTC}} = H_{\mathsf{msg,suite}}\big(\mathrm{Enc}(\texttt{"PQ-MIG-BTC-v1"}, \mathsf{assetID}, h_{160}, c_{pq}, \mathsf{keyEncoding}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{mode})\big)$$

and

$$m_{\mathsf{ETH}} = H_{\mathsf{msg,suite}}\big(\mathrm{Enc}(\texttt{"PQ-MIG-ETH-v1"}, a_{\mathsf{eth}}, \mathsf{chainID}, c_{pq}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{mode})\big).$$

The displayed $\|$ notation elsewhere in the paper should therefore be read as shorthand for this canonical encoding, not as ambiguous raw concatenation. Because $\mathsf{assetID}$ is already committed as a public value by the Bitcoin guest, chain binding can be introduced by constructing $\mathsf{assetID}$ with this canonical domain component without adding a new independent public field to the relation.

7.5 Completeness, Soundness, and Privacy Scope

Completeness. An honest owner who knows the legacy signing key can compute the corresponding public key, sign the domain-separated migration message, and produce an accepting proof for the appropriate relation.

Knowledge soundness and destination binding. Assuming the knowledge soundness of the underlying STARK system and the stated signature/hash assumptions, any accepting proof implies knowledge of witness values satisfying the relevant relation, including a legacy ownership witness and a suite-valid pk_{pq} that opens c_{pq}. Destination substitution has two distinct cases. If an adversary changes c_{pq} to a commitment to a different PQ key, the canonical migration message changes, so the adversary must obtain a valid legacy signature on that different message (or otherwise break the legacy-signature assumption). If the adversary keeps the signed c_{pq} fixed but attempts to open it with a different PQ key, the adversary must find an alternative valid opening of the binding commitment. In either case, production finalization additionally requires a valid post-quantum possession signature for the opened key.

Public-key non-publication. The legacy elliptic-curve public key is not intentionally included in the public statement, registry leaf, or enrollment transaction fields defined by the protocol. This removes the explicit design requirement to

reveal the legacy public key merely to enroll. It does not, by itself, establish a zero-knowledge guarantee for the proof transcript.

Zero-knowledge caveat. The implementation-level claim is therefore limited to non-publication of Q as a declared public value. A formal witness-privacy or zero-knowledge claim additionally depends on the concrete proving backend and configuration. Transparent STARK systems and zkVM deployments are not automatically zero-knowledge merely because a value is supplied as witness data. Consequently, unless the selected SP1 proof mode is configured and documented to provide zero knowledge, this paper does not claim that the proof transcript leaks no information about Q beyond the public statement.

7.6 Mempool, Replay, and Front-Running Resistance

A public mempool adversary or untrusted relayer may copy, censor, delay, replay, reorder, or race an enrollment. The canonical messages therefore bind the chain context, c_{pq}, suiteID, epochID, and mode, while registry/nullifier state makes accepted migrations one-shot. Replacing c_{pq} changes the legacy-signed message; keeping c_{pq} fixed while substituting another PQ key requires an alternative valid opening. Production acceptance is additionally subject to the finalization obligations of Section 7.3. A first-valid-finalized rule is meaningful only while the legacy credential remains unforgeable; after quantum recovery of a revealed key it cannot distinguish the historical owner from an attacker. Relayers remain untrusted for authorization but may affect liveness and ordering; deployments may additionally use private relay, commit–reveal, encrypted mempools, or batching.

8 Improving the Security of the Hash Functions through a One-Way Post-Quantum Transition

A common misconception in discussions of quantum risk for Bitcoin and Ethereum is to consider all hash-related attacks as equivalent. In practice, the relevant quantum attack complexity depends on the security property being targeted (e.g., collision vs. preimage), and this distinction is essential for correctly modelling the security of hidden-key addresses (such as Bitcoin P2PKH/P2WPKH and Ethereum EOAs before first use). The authors in [27] emphasize that quantum attacks affect hash security margins differently depending on the attack objective and construction details, without implying that SHA-2 is "completely broken" in practice today.

8.1 Collision vs. Preimage Security in the Quantum Setting

Let $H : \{0,1\}^\star \to \{0,1\}^n$ be an n-bit hash function. In the generic (black-box) setting:

- **Collision resistance** asks for two distinct inputs $x \neq x'$ such that $H(x) = H(x')$.

– **Preimage resistance** asks, given a target digest y, to find x such that $H(x) = y$.

These are different problems and lead to different generic quantum attack complexities:

– **Quantum collision attacks (BHT-style):** approximately $2^{n/3}$ queries in the generic query model [13].
– **Quantum preimage attacks (Grover-style):** approximately $2^{n/2}$ queries in the generic model [26].

The BHT cube-root figure is a *query-complexity* bound and should not be read directly as a realistic physical attack cost. Quantum-accessible memory, lookup and communication costs, circuit area, and parallelization materially affect end-to-end collision-search cost; Bernstein's cost analysis shows that the asymptotic query advantage need not translate into the same advantage under realistic hardware cost metrics [9]. Accordingly, the wider binding-hash choices below are conservative query-complexity design targets rather than claims that a 256-bit hash offers only about 2^{85} units of practical physical security.

Therefore, statements such as "a 160-bit hash gives only ~ 53 bits of security" are correct for collision resistance (since $2^{160/3} \approx 2^{53.3}$), but do not directly imply a ~ 53-bit preimage attack against a hidden Bitcoin address. For targeted preimage search, the generic quantum baseline remains approximately $2^{n/2}$ (e.g., $\sim 2^{80}$ for a 160-bit hash image) [26]. This distinction is particularly important for Bitcoin P2PKH/P2WPKH outputs, where the public key is hidden behind

$$h_{160} = \text{RIPEMD-160}(\text{SHA-256}(Q)).$$

The security question for theft of an unspent output is not a generic collision attack but rather a targeted attack path (e.g., constrained preimage search over valid public-key or private-key domains).

Multi-target hidden-key search. The single-target $2^{n/2}$ Grover estimate should not be interpreted as the ecosystem-wide cost of finding *some* vulnerable hidden address. If an idealized attacker has M independent target digests and succeeds by finding a preimage for any one of them, generic amplitude amplification gives an oracle-query scaling of approximately

$$O\left(\sqrt{\frac{2^n}{M}}\right) = 2^{(n-\log_2 M)/2},$$

with related multi-target quantum analyses making the same asymptotic point [5]. Thus, a large population of hidden addresses reduces the generic search exponent relative to a single fixed target. This is a population-level risk adjustment, not a direct wall-clock theft estimate: a concrete attack must search a valid secp256k1-derived domain, implement SHA-256/RIPEMD-160 or Keccak membership tests as reversible circuits, handle the target set, and pay the corresponding gate,

memory, communication, and error-correction costs. We therefore retain the single-target bound when discussing one identified address, but do not treat it as the effective security margin against an attacker willing to steal from any member of a large hidden-key population.

8.2 Our Proposal: One-Way Post-Quantum Transition

For each migrated identifier, legacy authorization is used only for enrollment; after the applicable obligations of Section 7.3 succeed, the enforcement domain switches future authorization to the PQ key (or a defined PQ key-rotation policy). The historical chain remains valid, but the migrated-state root becomes the new authorization reference. The legacy hash layer—for example RIPEMD160 in an already-created P2PKH/P2WPKH output—cannot be retroactively upgraded, so it remains an immutable compatibility constraint while the new migration layer uses crypto-agile hashes.

Based on the collision/preimage analysis above, hashes are assigned by role:

- **Targeted-preimage/second-preimage roles** such as message digests, transcript hashes, or proof anchors may use a 256-bit output when an approximately 128-bit generic quantum search target is sufficient.
- **Collision-sensitive binding roles** such as adversarially generated PQ-key or registry commitments may conservatively use a 384-bit-or-wider output when the design target is approximately 128-bit generic quantum *query complexity*; this is a design target, not a physical-cost estimate.
- **Prototype specializations** using 256-bit Keccak/SHA-family commitments remain implementation demonstrations but are not claimed to meet the conservative 128-bit quantum-collision query target.

The canonical message field order remains the one defined in Section 7.4; $H_{\mathsf{msg,suite}}$ is used for message/transcript hashing and $H_{\mathsf{bind,suite}}$ for c_{pq} and canonical registry commitments.

8.3 Migration Registry Semantics and Irreversibility

The registry distinguishes evidence recording from enforcement finalization. A record may be represented as

$$(\mathsf{assetID}, c_{pq}, pk_{pq}, \mathsf{suiteID}, \mathsf{epochID}, \mathsf{status} \in \{\texttt{provisional}, \texttt{migrated}\}),$$

where `migrated` is reachable only after Section 7.3. A canonical authenticated leaf is

$$\ell = H_{\mathsf{bind,suite}}(\mathrm{Enc}(\mathsf{assetID}, c_{pq}, \mathsf{suiteID}, \mathsf{epochID}))\,.$$

Leaf insertion may record provisional evidence; it is not by itself migration finalization. Once the enforcement domain marks the identifier `migrated`, legacy authorization is accepted only for the enrollment transition and future authorization proceeds through the PQ path. Irreversibility is therefore a property of the enforcement semantics that recognize the registry, not of the hash commitment alone.

9 Implementation and Benchmarking

9.1 Implementation in SP1

Our implementation uses separate Bitcoin and Ethereum SP1 guest programs plus a Rust SDK (`pq-bitcoin`, ∼1,760 lines). Shared code implements P2PKH/P2WPKH address derivation, Ethereum Keccak-256 address derivation, ML-DSA key parsing, canonical migration-message construction, and ABI-compatible data structures. SP1 compiles the guest programs to RISC-V and proves their execution; the exact post-quantum status of a concrete proving path is treated separately in Section 9.4 rather than inferred from the "STARK" label.

9.2 STARK Circuit Design

The guest programs implement the formal relations of Sections 7.1 and 7.2; this subsection records only implementation-specific details. For Bitcoin, the host writes Q_{enc}, a compact ECDSA signature, pk_{pq}, and the statement fields to SP1 input. It constructs the chain-bound $\mathsf{assetID}$ and the signed message using the canonical Enc of Section 7.4. The guest commits the public tuple as ABI-encoded values using `sp1_zkvm::io::commit_slice`, including c_{pq} rather than raw pk_{pq}. It does not query Bitcoin chain state, so CheckUTXOState remains an enforcement-layer obligation. The Ethereum guest uses the same ABI/serialization discipline with the Ethereum relation, Keccak-derived address, and m_{ETH}.

9.3 Full STARK/SNARK Integration Flow

The proof pipeline proceeds through five stages:

1. **Key generation.** The host script (`script/src/bin/main.rs`) generates a fresh secp256k1 keypair (d, Q) and an ML-DSA-65 keypair (sk_{pq}, pk_{pq}) using `OsRng`. The ECDSA private key signs the domain-separated migration message m_{BTC} from Section 7.1, and the ML-DSA signing key signs the same message to prove PQ key liveness.
2. **zkVM execution.** The host writes the encoded public key Q (compressed in the reference run), the ECDSA signature σ, pk_{pq}, and the statement fields including h_{160} to `SP1Stdin`. The guest computes and commits $c_{pq} = H_{\mathsf{bind},\mathsf{suite}}(pk_{pq})$ with the remaining public values. In the current execute-only artifact this completes in approximately 541,000 RISC-V cycles and approximately 90 ms on the reported M-series Mac environment.
3. **Proof generation.** The pinned SP1 prover generates a proof for the guest execution using the backend selected by that SP1 release and proof mode. The artifact must record the exact SP1 version and configuration; this paper does not assume that every SP1 backend has identical FRI, memory-argument, recursion, zero-knowledge, or post-quantum security properties.

4. **SNARK wrapping for current EVM compatibility.** For EVM verification, the STARK proof can be recursively wrapped into either a Groth16 or PLONK proof. This reduces the on-chain verification cost to $O(1)$ gas ($\sim$250K gas for Groth16, $\sim$300K for PLONK), but this wrapped path is not itself fully post-quantum because the wrapper relies on pairing-friendly elliptic-curve assumptions.
5. **Fixture generation.** The `evm.rs` binary serializes the wrapped proof, public values, and verification key into a JSON fixture file that is consumed by the Foundry test suite for end-to-end Solidity verification.

9.4 Post-Quantum Status of Verification Paths

The verification path must be evaluated separately from the migration relation. A transparent hash-based STARK/FRI construction can support post-quantum verification only when every component of the pinned configuration has an adequate post-quantum argument. This cannot be assumed for every zkVM or SP1 release: the SP1 V4 Turbo memory-argument specification, for example, documents an elliptic-curve-based multiset-hashing component and a targeted classical security level.[6] A deployment claiming end-to-end PQ security must therefore pin and audit the SP1 version, proof mode, recursion path, memory/lookup arguments, hashes, challenge derivation, security parameters, and verifier. Groth16 and pairing-based PLONK wrappers are explicitly current-EVM compatibility mechanisms rather than fully PQ verification paths.

9.5 On-Chain Verification: `PQBitcoin.sol`

The Solidity smart contract (304 lines, Solidity $\geq$ 0.8.20) verifies wrapped proofs on-chain by delegating to the `SP1VerifierGateway`. The contract implements four enhancement layers:

Access Control (Ownable2Step). Administrative functions (`updateVerifier`, `updateVKey`) are restricted to the contract owner via a two-step ownership transfer pattern:

1. The current owner calls `transferOwnership(newOwner)`, setting `pendingOwner`.
2. The pending owner calls `acceptOwnership()`, completing the transfer.

This prevents accidental transfers to incorrect addresses (e.g., due to typos) and uses custom Solidity errors (`Unauthorized`, `NotPendingOwner`) for gas efficiency.

[6] Succinct, "SP1 V4 Turbo: Memory Argument via Elliptic Curve based Multiset Hashing," `https://docs.succinct.xyz/assets/files/SP1_Turbo_Memory_Argument-b042ba18b58c4add20a8370f4802f077.pdf`.

ML-DSA Key Size Validation. The internal `_validatePQKeySize` function takes a `uint256` length and enforces that the post-quantum public key matches one of the three valid ML-DSA sizes:

$$|pk_{pq}| \in \{1312, 1952, 2592\} \text{ bytes}$$

This validation is applied in both state-changing (`verifyMigrationProof`) and view-only (`verifyMigrationProofView`) functions, rejecting malformed keys with a custom `InvalidPQKeySize(uint256)` error before any state modification. It is only a parsing/suite-consistency check; it must be combined with post-quantum signature verification for a complete authority transfer.

Hybrid Dual-Signature Migration. The `verifyDualSigMigration` function extends the basic migration with an ML-DSA signature commitment:

1. The STARK proof is verified on-chain, proving legacy ECDSA ownership and binding it to the committed PQ-key digest c_{pq}.
2. The ML-DSA signature σ_{pq} over the migration message is hashed: $h_\sigma = \text{keccak256}(\sigma_{pq})$.
3. The hash h_σ is stored in the `pqSignatures` mapping, enabling off-chain verifiers to confirm that the PQ key holder explicitly authorized the migration.

Full on-chain ML-DSA verification is deferred in the present EVM prototype; the tested Solidity configuration exhibited an order-of-magnitude cost of approximately 5M gas for ML-DSA-65 verification, while storing the signature hash provides a much cheaper audit anchor (approximately 25K gas in the tested contract path). This audit-anchor path does not satisfy the production finalization obligations of Section 7.3.

Incremental Merkle Tree Registry. An append-only incremental Merkle tree (`MerkleTree.sol`, depth 20, capacity $2^{20} \approx 1$M leaves) records prototype enrollment evidence. Because the current contract does not perform the production PQ-possession gate (and does not itself authenticate Bitcoin UTXO state), these leaves are provisional records rather than finalized migrations. The current Solidity prototype uses the specialized leaf

$$\ell_{\text{proto}} = \text{keccak256}(h_{160} \parallel \text{keccak256}(pk_{pq})).$$

This prototype leaf should be distinguished from the canonical crypto-agile registry leaf defined above: it effectively uses $c_{pq} = \text{keccak256}(pk_{pq})$ and does not yet include all suite/epoch/domain fields. A production implementation should use the canonical leaf or an equivalently domain-separated encoding so that the implemented state commitment matches the formal registry semantics. The tree uses pre-computed zero hashes for empty subtrees, achieving $O(\log N)$ insert complexity at ∼60K gas per migration. The `getMerkleRoot()` view function reveals the current root, enabling light clients to verify migration inclusion via standard Merkle proofs without downloading the full contract

state. Events `AddressMigrated` and `DualSigMigrated` include the `leafIndex` and `newMerkleRoot`, allowing off-chain indexers to reconstruct proofs from event logs.

9.6 Bitcoin Integration: Multi-Path Strategy

Since Bitcoin lacks smart contract capability, we implement four complementary on-chain strategies:

Phase 1: OP_RETURN Commitments. The SDK generates compact 71-byte payloads containing SHA256 hashes of both the PQ public key and the STARK proof and embeds them in `OP_RETURN` outputs. This size remains compatible with the historically common 80-byte data-carrier policy, while Bitcoin Core 30.0 increased the default `-datacarriersize` to 100,000 bytes and permits multiple data-carrier outputs [11]. The 71-byte format is therefore a deliberate compact encoding rather than a claim about a current consensus-level 80-byte limit. For hidden-key migration, the target P2PKH/P2WPKH UTXO must not be used as an input merely to fund this anchor, since spending it reveals Q_{enc}; a relayer or separate funded input pays the transaction fee.

The flags byte encodes proof type (Groth16 `0x01`, PLONK `0x02`) and migration mode (dual-sig `0x04`). A companion REST API indexer (808 lines, Axum/Tokio) scans the Bitcoin blockchain for `PQMG`-prefixed OP_RETURN outputs and maintains a queryable migration registry.

Phase 2: Experimental Taproot Commitment Carrier (BIP-341; not PQ authorization).

The SDK also demonstrates Taproot script-tree construction for committing a PQ-key hash. This phase is *not* a hidden-key asset-protection mechanism and must not be used to custody economically meaningful value. P2TR exposes an x-only output key in the UTXO and is therefore already in the revealed-key category of our threat model. Moreover, the demonstration leaves below are intentionally non-authorizing commitment/recovery placeholders:

- **Leaf 0 (PQ Commitment Demonstration):** The script is `OP_SHA256 <pk_hash>` followed by `OP_EQUALVERIFY OP_TRUE`. Once pk_{pq} is opened during enrollment, anyone who knows that public key can satisfy this leaf. It proves only knowledge of a public preimage, not possession of sk_{pq}.
- **Leaf 1 (Timelock Demonstration):** `<N> OP_CSV OP_DROP OP_TRUE`. After the relative timelock, this branch is likewise satisfiable without a secret key.

Accordingly, these scripts are retained only as implementation examples of commitment placement and Taproot control-block construction. A production value-carrying output would require an actual authorization predicate—for example a future consensus-recognized PQ signature/STARK verification rule—rather than either of the two demonstration leaves above. The implementation includes full Bech32m (BIP-350) encoding for P2TR address generation, tagged hashes (BIP-340/341), and control block construction.

Phase 3: Bitcoin L2 Deployment

EVM-compatible Bitcoin L2s (BOB, Citrea, Botanix) may allow deploying `PQBitcoin.sol` as a smart-contract registry for replay protection, Merkle proofs, and event emission. However, a registry contract cannot safely finalize a Bitcoin UTXO migration merely from the STARK statement: it also needs an authenticated mechanism establishing $\mathrm{CheckUTXOState}(\mathsf{assetID}, h_{160}, \mathsf{keyEncoding}) = 1$. A transaction-inclusion proof alone does not prove current unspentness. The exact state-proof, bridge, oracle, or spend-tracking mechanism and its trust assumptions are therefore part of the L2 deployment model.

Phase 4: Soft Fork - `OP_CHECKQUANTUMSIG`.

We propose the semantics of two new Bitcoin script operations as a long-term design direction, while intentionally leaving exact opcode numbers unspecified:

- **`OP_CHECKQUANTUMSIG`:** would verify a suite-selected post-quantum signature natively in Bitcoin Script.
- **`OP_CHECKSTARKPROOF`:** would verify a STARK proof binding a Bitcoin identifier to a post-quantum key commitment.

In Tapscript, opcode values 187–254 are currently designated `OP_SUCCESSx`; using one before a soft fork assigns semantics causes unconditional script success and is unsafe. A future soft fork could repurpose an appropriate `OP_SUCCESSx` value, but the concrete assignment must be specified by a BIP rather than assumed here [40].

Example Locking Script (scriptPubKey):

```
<legacy_id> <pq_commitment> <vkey> OP_CHECKSTARKPROOF OP_VERIFY
```

Example Unlocking Script (witness):

```
<stark_proof>
```

A future BIP-style specification would need to define resource limits, activation rules, and sigops-equivalent accounting for post-quantum signature verification and STARK proof verification. Such a specification could use a version-bits activation path and Tapscript leaf-version separation, but the exact parameters should be treated as deployment choices rather than assumptions of this paper.

9.7 Benchmarking

zkVM Execution. Table 4 summarizes the SP1 execution benchmarks for the Bitcoin circuit with ML-DSA-65 keys.

Table 4. SP1 Benchmark and Artifact Status for the Bitcoin Circuit

Category	Metric	Result / status
zkVM execution	RISC-V cycles (execute-only)	541,000
	Wall-clock time (execute-only)	∼90 ms
Raw transparent-STARK artifact	Raw STARK proving time	not separately instrumented in the current artifact
	Raw STARK proof size	not separately exported in the current artifact
	Trace rows / constraint count	not available in the current artifact
Proof wrapping (STARK → SNARK)	Groth16 wrapping time (includes recursion)	∼5 min
	PLONK wrapping time (includes recursion)	∼5 min
Wrapped proof size	Groth16 proof size (constant)	∼260 B
	PLONK proof size (constant)	∼800 B

Note. The figures in this table are approximate artifact-level observations from the reported development environment, not hardware-independent bounds or a statistically characterized performance study. Reproducibility requires pinning the repository commit, SP1/Rust toolchain, proving mode, benchmark command, and hardware configuration in the archived artifact. The Groth16/PLONK rows are reported only as current-EVM compatibility measurements and are not part of the fully post-quantum verification claim, because the wrappers rely on pairing-friendly elliptic-curve assumptions.

Table 5. Solidity Contract Gas Costs (Merkle insertion reported separately)

Operation	Gas
`verifyMigrationProof`	∼390K
`verifyDualSigMigration`	∼670K
Merkle tree insert (per leaf)	∼60K
`updateVerifier`	∼21K
`transferOwnership`	∼41K
`acceptOwnership`	∼37K

On-Chain Gas Costs. Table 5 presents the Ethereum gas costs for the main contract operations. The Merkle-tree insertion cost is shown as a separate component to make the accounting explicit.

Deployments that insert a registry leaf inside the same migration transaction should add this component and event-emission overhead to the base proof-verification cost. These gas figures should likewise be interpreted as approximate measurements of the tested contract paths under the artifact's EVM/toolchain configuration; the archived artifact should pin the exact commit, Foundry/Solidity versions, fork rules, commands, and repeated-run methodology used to reproduce them. The complete implementation is available at `https://github.com/skardas/pq_bitcoin`.

10 Non-Migrating Users and Ecosystem-Scale Cost

A complete ecosystem migration must define what happens to credentials that never enroll. Staged/sunset proposals such as BIP-361 and Sultan et al. illustrate this policy dimension [29,2]. We distinguish four policy choices. Voluntary migration preserves compatibility but leaves legacy funds or account authorities exposed. Incentive migration can use wallet warnings, exchange policy, fee policy, or custody requirements to encourage enrollment. Sunset migration disables or restricts legacy authorization after a deadline, but it requires strong social consensus and careful treatment of lost keys. Emergency freeze or recovery may protect funds after a quantum incident, but it creates governance and ownership disputes. Our key-binding layer is compatible with these policies, but it does not choose one for the ecosystem.

Migration cost also depends on the unit of enrollment. A per-UTXO Bitcoin design is simple but may be expensive when many outputs are controlled by the same key. A per-key-hash variant can amortize enrollment across multiple eligible outputs sharing the same HASH160 value. In that mode, the registry identifier is the chain-domain-separated key hash (optionally bound to a snapshot/epoch) rather than an individual outpoint; the production state predicate verifies the eligible unspent outputs covered by that group as being locked to the declared h_{160}; and the nullifier is consumed per key-hash group. This reduces the number of registrations, but deliberately links outputs that might otherwise remain unlinkable and removes selective per-output migration within the grouped scope, so the deployment must fix the snapshot and eligibility rule. For Ethereum, a per-EOA registry entry is not the same as migrating the account's complete authority surface, because tokens, approvals, bridges, Layer 2 systems, and administrative roles may require application-specific migration logic.

A deployment study should therefore start from a chain snapshot, estimate the number of enrollments under each accounting model, and compare the required transactions with available block capacity over the intended migration window. Our prototype reports per-enrollment proving and gas costs; the following scenario analysis makes the corresponding ecosystem-scale capacity explicit without treating every UTXO or EOA as equivalent.

Table 6. Enforcement alternatives and deployment consequences

Mechanism	**Finalization obligations**	**Non-migrators**	**Main deployment challenge**
Bitcoin consensus soft fork	Can enforce UTXO state binding, PQ possession, and migrated-state rules natively	Legacy path may remain available until a later sunset	Consensus activation, wallet/miner adoption, and resource accounting
Hard-fork / legacy sunset	Can make the PQ authority switch mandatory at protocol level	Legacy credentials may become frozen or invalid after the deadline	Governance, lost-key treatment, chain-split risk, and migration-window design
Contract / wrapper registry	Can enforce PQ possession and registry semantics for opted-in assets; Bitcoin state binding needs an authenticated external state source	Unwrapped assets remain under legacy authorization	Limited coverage and dependence on bridge/oracle/state-proof assumptions
Layer 2 registry	Can enforce the three obligations if it maintains authenticated underlying state and routes authorization through the registry	Users outside the Layer 2 remain unaffected	Settlement/trust assumptions, data availability, and cross-domain authority coverage
EIP-7702 delegation	Can route an already-revealed EOA through registry-aware code, but cannot remove the underlying ECDSA delegation authority by itself	Non-delegating EOAs remain unchanged	Re-delegation under the legacy key and public-key exposure make it unsuitable for hidden-key preservation

Table 7. Migration-cost dimensions

System	Unit of enrollment	Main cost driver
Bitcoin P2PKH/P2WPKH	UTXO or key-hash group	Proof size, anchor transaction weight, registry entry, and block-space availability
Bitcoin P2PK/P2TR	Revealed key or output	Hybrid signature path or protocol-level spend rule
Ethereum EOA	Account-authority binding	Verifier gas, registry update cost, and application integration
Ethereum tokens/roles	Contract-specific authority	Whether the contract consults the registry or supports its own migration path

10.1 Scenario-Based Capacity Analysis

Bitcoin. A public `gettxoutsetinfo` snapshot at block height 964,695 on 30 August 2026 reports 165,765,894 unspent outputs [17]. This number is *not* the number of hidden-key outputs requiring migration: it includes heterogeneous script types, spent-key reuse patterns, dust, shared key hashes, and outputs that could be grouped under a per-key-hash policy. We therefore use it only as a conservative scale reference for a worst-case per-UTXO accounting model. Bitcoin's consensus block-weight limit is 4,000,000 weight units, corresponding to at most approximately 1,000,000 virtual bytes per block [28]. With a ten-minute target interval, the theoretical daily capacity is approximately 144×10^6 vbytes. If an enrollment transaction consumes an average of w vbytes and a fraction η of block space is allocated to migration, the approximate enrollment capacity is

$$C_{\mathsf{BTC}}(\eta, w) = \frac{\eta \cdot 144 \times 10^6}{w} \quad \text{enrollments/day.}$$

It is important to distinguish a compact *anchor/coordination transaction* from a *complete PQ enrollment*. The $w = 300$ vbyte scenario used below is representative only of a compact coordination/anchor assumption. A complete ML-DSA-65 enrollment that opens c_{pq} and carries an ML-DSA-65 public key and signature has $1{,}952 + 3{,}309 = 5{,}261$ raw credential bytes before any proof material, script/witness framing, transaction overhead, or batching [31]. Future Bitcoin designs could place some data in witness, use aggregation, or keep bulky proof material off chain, so there is no single universal full-enrollment vsize. To make the capacity consequence explicit without presenting a fabricated exact transaction size, Table 8 shows three clearly labelled scenario parameters: a 300-vbyte anchor, an aggressive 1,500-vbyte witness-heavy complete-enrollment scenario, and a 5,500-vbyte approximately all-inline/non-witness-equivalent scenario. The latter two are sensitivity-analysis inputs rather than measured transaction sizes.

These values show why enrollment granularity and data placement matter. A literal per-UTXO complete migration can consume many years of block space

Table 8. Illustrative Bitcoin migration sensitivity using 165,765,894 UTXOs as a scale reference

Scenario	w (vB)	**Enroll./day**	**Days 25%**	**Days 100%**
Anchor/coordination only	300	120,000	1,381	345
Complete PQ, witness-heavy sensitivity	1,500	24,000	6,907	1,727
Complete PQ, all-inline sensitivity	5,500	6,545	25,325	6,331

even under aggressive capacity allocation. Grouping outputs that share an authorization key, batching/aggregation, or a protocol design in which only a compact commitment is placed on L1 can reduce the load substantially. A production study should therefore classify a pinned UTXO snapshot by script type, exposure status, key-hash reuse, and chosen enforcement path before treating the UTXO count as the migration count.

Ethereum. Ethereum currently uses 12-second slots and a 60M block gas limit, with 30M gas as the target size under the present configuration [20,24]. Let g denote gas consumed by the relevant enrollment path and η the fraction of the 60M block limit hypothetically allocated to migration. With approximately 7,200 slots per day,

$$C_{\mathsf{ETH}}(\eta, g) = \frac{\eta \cdot 60 \times 10^6 \cdot 7200}{g} \quad \text{enrollments/day.}$$

The prototype's $\sim$ 390K and $\sim$ 670K paths are proof/registry or audit-anchor measurements; they are not complete on-chain PQ-authorization costs because the current contract does not verify ML-DSA on chain. The same artifact reports an order-of-magnitude value of approximately 5M gas for ML-DSA-65 verification. To expose the difference, Table 9 reports both prototype coordination paths and a simple additive complete-enrollment sensitivity in which the 5M-gas PQ verification cost is added to the corresponding proof path. The additive rows are scenario calculations, not measurements of a single optimized combined Solidity transaction.

Table 9. Illustrative Ethereum capacity at 25% of a 60M block gas limit

Path	g	**Enrollments/day**	**Days per 1M**
Proof/registry path only	$\sim$ 390K	276,923	3.6
Dual-signature audit-anchor path	$\sim$ 670K	161,194	6.2
Complete PQ sensitivity: 390K+5M	$\sim$ 5.39M	20,037	49.9
Complete PQ sensitivity: 670K+5M	$\sim$ 5.67M	19,048	52.5

These figures are capacity scenarios rather than forecasts: normal application traffic, fee-market effects, calldata, verifier changes, and contract-specific authority

migrations reduce practical throughput. Moreover, there is no single canonical "Ethereum asset count" corresponding to the Bitcoin UTXO set, because an EOA may control ETH, tokens, roles, approvals, bridges, and Layer 2 authorities. The relevant Ethereum migration population must therefore be defined by a chain-state and application-authority snapshot rather than by address count alone.

This scenario analysis answers the network-capacity question at two distinct levels—coordination/enrollment and complete PQ authorization—while avoiding a fabricated claim that every current UTXO or Ethereum address requires an identical enrollment. A deployment-grade evaluation should repeat the calculation on a pinned chain snapshot and classify the actually eligible Bitcoin outputs and Ethereum authority bindings.

11 Conclusion and Future Work

We presented a unified post-quantum migration framework for Bitcoin outputs and Ethereum account authorities. Revealed-key credentials admit hybrid continuity only before legacy-key compromise; hidden-key credentials instead use STARK-verifiable ownership relations that bind the legacy identifier to c_{pq} without explicitly publishing the legacy public key. The central systems result is that proof validity is not itself migration finalization: production deployment must satisfy the state-binding, PQ-possession, and non-revealing-submission obligations of Section 7.3, while the registry distinguishes `provisional` evidence from an enforcement-recognized `migrated` state.

The framework also separates message and binding hash roles, accounts for single- and multi-target quantum hash search, and quantifies migration capacity under explicit Bitcoin and Ethereum scenarios. The SP1 prototype demonstrates implementation feasibility, while current Groth16/PLONK wrapping and the prototype Merkle leaf remain compatibility/specialization paths rather than an end-to-end PQ deployment. Future work should implement authenticated Bitcoin state checks and full PQ-signature enforcement, evaluate exact alternative suites such as FIPS-205 SLH-DSA parameter sets, repeat the capacity study on classified chain snapshots, and study governance for non-migrating users.

Acknowledgments

The authors thank Prof. Dr. Osmanbey Uzunkol for his valuable feedback on the quantum hash-security and post-quantum transition discussion in this part of the work, which helped improve the security analysis and strengthen the proposed framework. The authors used ChatGPT, an AI-based language model developed by OpenAI, for grammar checking, language polishing, formatting assistance, and critical review of manuscript clarity and internal consistency. All scientific contributions, security arguments, experimental results, references, and final technical decisions were reviewed and verified by the authors, who remain fully responsible for the correctness, originality, and integrity of the manuscript.

References

1. Alghamdi, S., Almuhammadi, S.: The future of cryptocurrency blockchains in the quantum era. In: 2021 IEEE International Conference on Blockchain (Blockchain). pp. 544–551 (2021). https://doi.org/10.1109/Blockchain53845.2021.00082
2. Almuhammadi, S., Alghamdi, S.: A novel transition protocol to post-quantum cryptocurrency blockchains. https://doi.org/10.3389/fcomp.2025.1457000 (2025), accessed: 12-04-2026
3. Alnahawi, N., Wiesmaier, A., Grasmeyer, T., Geißler, J., Zeier, A., Bauspieß, P., Heinemann, A.: On the state of post-quantum cryptography migration. INFORMATIK 2021 (2021). https://doi.org/10.18420/informatik2021-078
4. Babbush, R., Zalcman, A., Gidney, C., Broughton, M., Khattar, T., Neven, H., Bergamaschi, T., Drake, J., Boneh, D.: Securing elliptic curve cryptocurrencies against quantum vulnerabilities: Resource estimates and mitigations. Tech. rep., Google Quantum AI (2026), white paper and arXiv preprint. Available at https://arxiv.org/abs/2603.28846; accessed 2026-09-01
5. Banegas, G.S., Bernstein, D.J.: Low-communication parallel quantum multi-target preimage search. In: Selected Areas in Cryptography – SAC 2017. Lecture Notes in Computer Science, vol. 10719, pp. 325–335. Springer (2017). https://doi.org/10.1007/978-3-319-72565-9_16
6. Baseri, Y., Hafid, A., Shahsavari, Y., Makrakis, D., Khodaiemehr, H.: Blockchain security risk assessment in quantum era, migration strategies, and proactive defense. IEEE Communications Surveys & Tutorials **28**, 2925–2964 (2026). https://doi.org/10.1109/COMST.2025.3621113
7. Beast, H., Heilman, E., Foxen Duke, I.: BIP 360: Pay-to-Merkle-Root (P2MR). Bitcoin Improvement Proposal, Draft (2026), available at https://bips.dev/360/; accessed 2026-07-02
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive **Report 2018/046** (2018)
9. Bernstein, D.J.: Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? Presented at SHARCS (2009), workshop paper / talk manuscript
10. Bernstein, D.J.: Post-quantum Cryptography, pp. 1846–1847. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-030-71522-9_386
11. Bitcoin Core Developers: Bitcoin core 30.0 release notes. Bitcoin Core release notes (2025), available at https://bitcoincore.org/en/releases/30.0/; accessed 2026-09-01
12. Bizri, M.E., El-Hajj, A.M., Sliman, L., Haidar, A.M.: Institutional approaches to post-quantum cryptography: A comparative analysis of migration frameworks. IEEE Access **14**, 3259–3283 (2026). https://doi.org/10.1109/ACCESS.2025.3650465
13. Brassard, G., HØyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN'98: Theoretical Informatics. pp. 163–169. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
14. Bugra Sezer, B., Akleylek, S., Nuriyev, U.: PP-PQB: privacy-preserving in post-quantum blockchain-based systems: A systematization of knowledge. IEEE Access **13**, 41382–41405 (2025). https://doi.org/10.1109/ACCESS.2025.3545943
15. Buterin, V.: How to hard-fork to save most users' funds in a quantum emergency. Ethereum Research post (2024), available at https://ethresear.ch/t/how-to-hard-fork-to-save-most-users-funds-in-a-quantum-emergency/18901; accessed 2026-07-02

16. Campbell, R.: Hybrid post-quantum signatures for bitcoin and ethereum: A protocol-level integration strategy. Preprints (September 2025). https://doi.org/10.20944/preprints202509.2079.v1, https://doi.org/10.20944/preprints202509.2079.v1
17. ChainQuery: Bitcoin core gettxoutsetinfo snapshot at height 964695. Public Bitcoin Core RPC snapshot (Aug 2026), snapshot captured 2026-08-30 06:07 UTC; txouts=165765894. Available at https://chainquery.com/rpc/gettxoutsetinfo; accessed 2026-09-01
18. Coinbase Independent Advisory Board on Quantum Computing and Blockchain: Quantum computing and blockchain. Tech. rep., Coinbase Independent Advisory Board on Quantum Computing and Blockchain (Apr 2026), position paper. Available at https://assets.ctfassets.net/sygt3q11s4a9/6EjYavuGdtJDYCqaJrASj9/9f464a8bf26f44bd6c85710fe7e4a29f/Quantum_Computing_and_Blockchain_v10.3_15April2026.pdf; accessed 2026-07-02
19. Dalal, A.: Quantum computing and blockchain: Advanced security in the Metaverse. In: Kumar, A., Rathore, P.S., Moreira, F. (eds.) Blockchain Enabled Metaverse for Smart Wireless Sensor Networks, pp. 81–105. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-07305-1_5
20. Ethereum.org: Blocks. Ethereum.org developer documentation (2026), documents 12-second slots and the current 60 million block gas limit with a 30 million gas target. Available at https://ethereum.org/developers/docs/blocks/; accessed 2026-09-01
21. Fan, X., Tan, T.G., Ho, N., Choy, S.H.: Enabling a smooth migration towards post-quantum security for ethereum. In: Blockchain – ICBC 2024. pp. 3–15. Springer Nature Switzerland, Cham (2025)
22. Fukuda, K., Matsuo, S., Suga, Y., Ito, T.: The grand challenge of PQC migration: Analysis of modern blockchain and intertwined human egoisms. Cryptology ePrint Archive, Paper 2025/1626 (2025), https://eprint.iacr.org/2025/1626
23. Gharavi, H., Granjal, J., Monteiro, E.: Post-quantum blockchain security for the internet of things: Survey and research directions. IEEE Communications Surveys & Tutorials **26**(3), 1748–1774 (2024). https://doi.org/10.1109/COMST.2024.3355222
24. Gold, S., Jayanthi, P., Wahrstätter, T., Beekhuizen, C., Dietrichs, A., Feist, D., Stokes, A., Rudolph, J., Rebuffo, G., Slivkoff, S., Chodoła, K.: EIP-7935: Set default gas limit to 60m. Ethereum Improvement Proposal (2025), available at https://eips.ethereum.org/EIPS/eip-7935; accessed 2026-09-01
25. Gomes, J., Khan, S., Svetinovic, D.: Fortifying the blockchain: A systematic review and classification of post-quantum consensus solutions for enhanced security and resilience. IEEE Access **11**, 74088–74100 (2023). https://doi.org/10.1109/ACCESS.2023.3296559
26. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC '96). pp. 212–219. ACM (1996). https://doi.org/10.1145/237814.237866
27. Hosoyamada, A.: Security of hash functions against attacks using quantum computers. NTT Technical Review **21**(7), 43–47 (07 2023). https://doi.org/10.53829/ntr202307fa4, https://cir.nii.ac.jp/crid/1390578360713619840
28. Lombrozo, E., Lau, J., Wuille, P.: BIP 141: Segregated witness (consensus layer). Bitcoin Improvement Proposal (2015), available at https://bips.dev/141/; accessed 2026-09-01

29. Lopp, J., Papathanasiou, C., Smith, I., Ross, J., Vaile, S., Dallaire-Demers, P.L.: BIP 361: Post quantum migration and legacy signature sunset. Bitcoin Improvement Proposal, Draft (2026), available at https://bips.dev/361/; accessed 2026-07-02
30. Milton, A., Shikhelman, C.: Bitcoin and quantum computing: Current status and future directions. Tech. rep., Chaincode Labs (May 2025), available at https://chaincode.com/bitcoin-post-quantum.pdf; accessed 2026-07-02
31. National Institute of Standards and Technology: Module-lattice-based digital signature standard. Tech. Rep. FIPS 204, U.S. Department of Commerce (Aug 2024). https://doi.org/10.6028/NIST.FIPS.204
32. National Institute of Standards and Technology: Stateless hash-based digital signature standard. Tech. Rep. FIPS 205, U.S. Department of Commerce (Aug 2024). https://doi.org/10.6028/NIST.FIPS.205
33. Nguyen, H., Huda, S., Nogami, Y., Nguyen, T.T.: Security in post-quantum era: A comprehensive survey on lattice-based algorithms. IEEE Access **13**, 89003–89024 (2025). https://doi.org/10.1109/ACCESS.2025.3571307
34. Ravi, C.: Quantum computing and cybersecurity: Systematic review of algorithms, challenges, and emerging solutions. In: Goyal, S.B., Kumar, V., Islam, S.M.N., Ghai, D. (eds.) Quantum Computing, Cyber Security and Cryptography: Issues, Technologies, Algorithms, Programming and Strategies, pp. 407–440. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-96-4948-8_16
35. Saha, R., Kumar, G., Devgun, T., Buchanan, W.J., Thomas, R., Alazab, M., Hoon-Kim, T., Rodrigues, J.J.P.C.: A blockchain framework in post-quantum decentralization. IEEE Transactions on Services Computing **16**(1), 1–12 (2023). https://doi.org/10.1109/TSC.2021.3116896
36. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE (1994)
37. Tan, T.G., Zhou, J.: Migrating blockchains away from ecdsa for post-quantum security: A study of impact on users and applications. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology. pp. 308–316. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-25734-6_19
38. Thakur, M.S.D., Vidhani, K., Syed, H.B., M.A., R.: Enterprise post quantum cryptography migration tools. In: 2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS). pp. 327–329 (2024). https://doi.org/10.1109/COMSNETS59351.2024.10427442
39. Wiesmaier, A., Alnahawi, N., Grasmeyer, T., Geißler, J., Zeier, A., Bauspieß, P., Heinemann, A.: On PQC migration and crypto-agility (2021), https://arxiv.org/abs/2106.09599
40. Wuille, P., Nick, J., Towns, A.: BIP 342: Validation of taproot scripts. Bitcoin Improvement Proposal (2020), available at https://bips.dev/342/; accessed 2026-09-01
41. Yalamuri, G., Honnavalli, P., Eswaran, S.: A review of the present cryptographic arsenal to deal with post-quantum threats. Procedia Computer Science **215**, 834–845 (2022). https://doi.org/https://doi.org/10.1016/j.procs.2022.12.086, https://www.sciencedirect.com/science/article/pii/S1877050922021573, 4th International Conference on Innovative Data Communication Technology and Application
42. Yang, Z., Alfauri, H., Farkiani, B., Jain, R., Pietro, R.D., Erbad, A.: A survey and comparison of post-quantum and quantum blockchains. IEEE Communications

Surveys & Tutorials **26**(2), 967–1002 (2024). https://doi.org/10.1109/COMST.2023.3325761