anchorage digital

Post-Quantum Key Migration via Zero-Knowledge Proofs of Hash Preimage Knowledge

EK2022

Executive Summary

Authors:
Dustin Ray
Prasanna Gautam
Sean Ryan

Published:
March 2026

The Problem

Many cryptographic systems, Bitcoin chief among them, identify users not by their public key directly but by a *hash* of that public key. Today this is a mundane implementation detail. In a future where large-scale quantum computers exist, it becomes a critical security property.

A sufficiently powerful quantum computer running Shor's algorithm can reverse-engineer a private key from a public key. But it cannot reverse-engineer a public key from its hash. That means any account whose public key has never been exposed on the blockchain is, in a narrow but important sense, already quantum-safe at rest. The catch: if the system disables classical signature schemes to prevent quantum theft, those protected accounts are frozen too. Their owners still hold their private keys, but the system no longer accepts the old type of signature. The funds are safe but inaccessible.

This paper asks: how do you let legitimate owners reclaim frozen accounts and migrate them to a post-quantum signature scheme, without ever revealing the information a quantum attacker would need?

The Proposed Solution: A "Turnstile"

The authors define a one-way migration mechanism they call a **turnstile**. The idea has three steps, all performed in a single transaction:

1. **Prove ownership.** The account owner generates a cryptographic proof (a STARK, a type of zero-knowledge proof) demonstrating that they know the private key behind the frozen account. Crucially, this proof reveals nothing about the private key or the public key to anyone else.
2. **Derive a new post-quantum key.** Inside the same proof, the owner deterministically derives a new key pair compatible with a post-quantum signature scheme. The derivation uses a standard key derivation function (HKDF) applied to the original private key. This means the new key is uniquely and automatically determined by the old one. No new secret material needs to be generated or backed up.
3. **Atomic re-registration.** The system verifies the proof and, in a single step, transfers the account from the old (frozen) signing rules to the new post-quantum signing rules. The account value is unchanged; only the authentication method changes.

After migration, the owner signs future transactions with their new post-quantum key. The original private key (or the seed phrase it was derived from) remains the sole backup secret, since the new key can always be re-derived from it.

What Makes This Different from Prior Work

Several researchers have independently recognized the core insight here: hash commitments are a quantum-safe anchor, and zero-knowledge proofs can exploit that anchor for migration. The paper cites prior work by Sattath and Wyborski (2023), Buterin (2024), Kiraz and Kardas (2026), and others. What this paper adds is:

- A complete formal construction. The authors define the exact mathematical relation being proven, state correctness and soundness theorems, and provide a full security reduction. Prior proposals were either informal, partial, or focused on different goals (e.g., building new signature schemes rather than a migration protocol).
- Deterministic key binding via HKDF. The new post-quantum key is derived inside the zero-knowledge proof from the original private key through a domain-separated key derivation function. This eliminates "key substitution" attacks, where an attacker who somehow obtained the private key could race the legitimate owner and register a different post-quantum key. With deterministic binding, the attacker and the legitimate owner would derive the same new key, reducing the attack to a standard race condition that existing consensus mechanisms already handle.
- Generality. The construction is parameterized over any hash function, elliptic curve, and post-quantum signature scheme. It is not tied to a specific blockchain, though Bitcoin is the primary application.

Security Argument in Plain Terms

The paper's security case rests on two pillars:

Hash preimage resistance. A quantum attacker who sees only the hash of a public key cannot recover the public key. Grover's algorithm (the best known quantum approach for this kind of search) provides a quadratic speedup, reducing an n-bit hash to roughly n/2 bits of effective security. For Bitcoin's 160-bit HASH160, that gives approximately 80 bits of quantum security. The authors show that even at 80 bits, the concrete cost of each step in the quantum search (which must simulate elliptic curve math as a reversible quantum circuit) makes the attack astronomically impractical: on the order of 10^25 seconds even with generous assumptions about future quantum hardware.

Zero-knowledge proof soundness. No one who does not know the private key can fabricate a valid proof. The paper formally reduces this to the preimage resistance of the hash function: if an attacker could produce a fake proof, they could be used as a subroutine to break the hash, which is assumed to be infeasible.

The paper is transparent about a formal gap in the literature: the quantum knowledge-soundness of non-interactive STARK proofs (compiled via the Fiat-Shamir heuristic) has not been fully proven. The authors state this explicitly, note that no quantum attack on any hash-based proof system is known, and observe that even NIST-standardized post-quantum schemes rest on analogous conjectured-but-unproven hardness assumptions.

Scope and Limitations

The mechanism applies only to accounts whose public keys have never been revealed. For Bitcoin, the authors estimate this covers roughly two-thirds of the circulating supply (approximately 13.5 million BTC). The remaining one-third, including P2PK outputs, Taproot key-path outputs, and any address that has been spent from, has exposed public keys and falls outside the turnstile's protection. Those accounts must migrate proactively before a quantum computer exists.

The paper also does not protect against classical compromise of the private key (e.g., a stolen seed phrase). If an attacker obtains the key through non-quantum means, they can produce the same proof and derive the same post-quantum key as the legitimate owner. This is inherent to any scheme that derives the new key from the old secret.

Practical Feasibility

Proof generation is estimated at roughly 5 to 30 seconds on consumer hardware with GPU acceleration, based on existing benchmarks for similar zero-knowledge circuits. This is a one-time cost per account.

Proof size is the primary practical challenge. Current benchmarks for ECDSA-scale STARK proofs range from 2 to 10 MB, which is large relative to typical blockchain transactions. The authors discuss proof aggregation (batching many proofs into one per-block proof) and recursive compression as paths to reducing this overhead, but acknowledge these techniques have not yet been demonstrated at this scale.

Verification cost is modest: single-digit milliseconds per proof, which is within the per-transaction budget of existing full nodes.

Relationship to Bitcoin's Migration Roadmap

The paper positions the turnstile as a concrete implementation of "Phase C" in the QBIP migration framework proposed by Lopp and Papathanasiou (2025). QBIP envisions three phases: Phase A restricts new spending to quantum-vulnerable addresses, Phase B freezes remaining legacy outputs, and Phase C (previously a placeholder) recovers frozen outputs via zero-knowledge proofs. The turnstile fills the Phase C gap with a formal specification and security analysis. It is also complementary to BIP-360, which defines the post-quantum output type that serves as the migration destination.

TL;DR

The paper formalizes and analyzes a mechanism for migrating cryptographic accounts from classical to post-quantum security without revealing the secrets that quantum computers could exploit. It addresses a real and growing concern, provides rigorous security arguments, is honest about its assumptions and limitations, and situates itself clearly within the broader ecosystem of related proposals. The core contribution is taking an idea that multiple researchers had identified informally and turning it into a fully specified protocol with formal security guarantees.

Post-Quantum Key Migration via Zero-Knowledge Proofs of Hash Preimage Knowledge

Dustin Ray, Prasanna Gautam, Sean Ryan
Anchorage Digital
dustin.ray@anchorlabs.com, prasanna.gautam@anchorlabs.com, sean.ryan@anchorlabs.com

March 2026

Abstract

Many deployed cryptographic systems bind signing authority to a hash commitment $H(P)$ rather than to the public key P itself. When classical signing paths are disabled to prevent quantum key extraction, records whose P has never been exposed are quantum-safe at rest, since $H(P)$ resists preimage attacks even from a quantum adversary, but their owners lose the ability to authenticate. The observation that hash-based key derivation provides a post-quantum anchor, enabling migration via a zero-knowledge proof of knowledge of the underlying secret, has been independently identified in multiple prior works.

This paper provides the first formal construction and security analysis of a complete migration protocol for such systems. We define a *turnstile*: a one-way mechanism in which the owner of a frozen record constructs a STARK proof demonstrating (1) knowledge of a private key sk whose corresponding public key hashes to the on-chain commitment, and (2) that a post-quantum keypair derived deterministically from sk via HKDF produces a specific PQ public key $\mathsf{pk}_{\mathsf{pq}}$. The system verifies this proof and atomically re-registers the record under $\mathsf{pk}_{\mathsf{pq}}$. The deterministic key binding via domain-separated HKDF, absent from all prior proposals, eliminates key substitution attacks and preserves the user's existing backup as the sole secret required for migration. We present the construction as a generic cryptographic primitive parameterized over any hash function, elliptic curve, and PQ signature scheme; provide correctness and soundness theorems; analyze zero-knowledge, non-malleability, and front-running resistance; and estimate proof generation and verification costs. Bitcoin is the primary motivating application; Section C provides the detailed instantiation.

1 Introduction

Many deployed cryptographic systems bind a user's signing authority to a hash commitment $H(P)$ rather than to the public key P itself. In such systems, the public key is revealed only at the moment of authentication, and the hash commitment serves as the persistent identifier. This design pattern appears in blockchain networks (e.g., Bitcoin's P2PKH and P2WPKH address types), certificate transparency systems, and various credentialing frameworks. As long as P has never been exposed, $H(P)$ is quantum-safe by virtue of hash preimage resistance: Shor's algorithm can derive a private key from P, but cannot obtain P from $H(P)$.

However, quantum safety at rest does not solve the migration problem. If the system disables classical (ECDSA/EdDSA) signing paths to prevent quantum theft of exposed credentials, the hash-protected records are frozen too: their owners retain their private keys but the system no longer accepts classical signatures. What is needed is a mechanism that allows the owner of a frozen record to demonstrate possession of the underlying private key and transition signing authority to a post-quantum (PQ) signature scheme, without revealing the private key or public key at any point during the process.

The quantum threat to elliptic curve cryptography is well established. Shor's algorithm [1] solves the elliptic curve discrete logarithm problem (ECDLP) in polynomial time on a quantum computer, requiring approximately $6\lceil \log_2 q \rceil$ logical qubits for a curve of group order q [1] (approximately 2,124 for secp256k1 [2]). While no quantum computer of this scale exists today, expert surveys place the probability of a cryptographically relevant quantum computer (CRQC) within a decade at 30 to 50% [3], and IBM's hardware roadmap projects approximately 2,000 logical qubits by 2033 [4]. NIST's transition guidance [5] targets deprecation of all quantum-vulnerable asymmetric cryptography by 2035. The Federal Reserve has warned of "harvest now, decrypt later" risks to blockchain networks [6].

Three developments have converged to make a migration mechanism feasible. First, NIST finalized three post-quantum signature standards in August 2024 [7], with additional candidates including SQIsign [8] under active evaluation, providing mature PQ signature schemes whose key generation accepts a pseudorandom seed as input. Second, zero-knowledge proof systems, specifically STARKs [9], have reached practical maturity through general-purpose zero-knowledge virtual machines (zkVMs) such as RISC Zero [10] and Stwo [11], which can prove arbitrary computations (including elliptic curve operations and hash functions) on consumer hardware [12]. Third, multiple communities have produced concrete proposals for phased PQ migration, establishing the system-level scaffolding within which a migration mechanism can operate (see Section C for Bitcoin-specific proposals). The urgency of this work is underscored by recent milestones: BIP-360 (Pay-to-Merkle-Root), the first Bitcoin Improvement Proposal for post-quantum output types, was merged into the Bitcoin BIPs repository in February 2026 [13]; Blockstream deployed the first post-quantum signed transactions on the Liquid sidechain in March 2026 using the SHRINCS hash-based signature scheme [14, 15]; and the Ethereum Foundation launched a dedicated post-quantum security hub (`pq.ethereum.org`) consolidating its multi-year research roadmap [16, 17].

Prior work. The observation that hash-based key derivation provides a post-quantum anchor for classical signature schemes, and that zero-knowledge proofs can exploit this anchor for migration, has been independently identified in multiple contexts. Sattath and Wyborski [18] formalized "signature lifting" in 2023, showing that a post-quantum one-way function (the hash) sitting in the key derivation chain between secret and public key can be exploited to lift a pre-quantum signature scheme to a post-quantum one, using Picnic signatures as the ZK mechanism. Buterin [19] proposed STARK-based emergency migration for Ethereum in 2024, freezing classical spends and allowing users to prove knowledge of a private preimage to transition accounts to post-quantum validation. Kiraz and Kardas [21] concurrently propose STARK circuits for Bitcoin and Ethereum address migration. Baldimtsi, Chalkias, and Roy [22] analyze post-quantum readiness of EdDSA seed derivation chains, showing that SLIP-0010 hash-chain-based private key derivation supports ZK proofs of possession while BIP-32 non-hardened ECDSA derivation does not. Earlier, Tan and Zhou [23] studied the impact of migrating blockchains away from ECDSA on users and applications.

The gap. Despite this convergent interest, no prior work provides a complete formal construction with security analysis for a migration protocol in hash-committed public key systems. Sattath and Wyborski [18] build a new signature scheme (signature lifting) rather than a migration protocol, and do not address deterministic PQ key binding, consensus-level transaction formats, or eligible key class analysis. Buterin [19] proposes an emergency response for Ethereum's account model without formal definitions, security theorems, or deterministic key binding (users choose arbitrary new validation code). Ray's mailing list post [20] and the QBIP Phase C placeholder [24] lack technical specification. Kiraz and Kardas [21] provide STARK circuit designs and SP1 benchmarks but no formal NP relation definition, correctness or soundness theorems, security reductions, or deterministic key binding via HKDF; their contribution is the circuit

construction and address-format specifics, not the protocol-level security analysis.

Contribution. This paper provides the first formal construction and security analysis of a complete migration protocol for hash-committed public key systems. We define a generic cryptographic primitive, which we call a *turnstile*, that migrates signing authority from ECDSA to a post-quantum signature scheme in a single non-interactive step. The owner of a frozen record constructs a STARK proof demonstrating knowledge of the ECDSA private key sk such that (1) the corresponding public key hashes to the on-chain commitment, and (2) a PQ keypair derived deterministically from sk via domain-separated HKDF [25] matches a claimed PQ public key. The system verifies this proof and atomically re-registers the record under the PQ public key. The private key and ECDSA public key are never revealed. The construction is parameterized over any hash function, elliptic curve, and PQ signature scheme.

Our specific contributions are:

1. A formal definition of the turnstile NP relation (Definition 1) with correctness and soundness theorems (Theorem 1, Theorem 2).
2. Deterministic PQ key binding via HKDF inside the ZK proof, eliminating key substitution attacks, a property absent from all prior proposals.
3. Corrected quantum security analysis of n-bit commitment hash functions ($n/2$-bit quantum security via multi-solution Grover search), with $\mathsf{HASH160}$ ($n = 160$, 80-bit security) as the primary instantiation and concrete Grover oracle cost estimates.
4. A comprehensive comparison of migration paradigms: ZK turnstile vs. commit-delay-reveal vs. signature-based ownership proofs (Section 7).
5. A detailed Bitcoin instantiation (Section C), including eligible UTXO classes, transaction formats, and the relationship to BIP-360 and the QBIP framework.

Organization. Section 2 defines the assumptions and scope of the mechanism, including eligible and excluded record classes and the threat model. Section 3 provides background on ECDSA, hash-committed public key systems, post-quantum signature schemes, and STARKs. Section 4 presents the turnstile construction. Section 5 analyzes its security properties. Section 6 discusses practical considerations including proof generation cost, on-chain overhead, and verification cost. Section 7 situates the construction within the broader landscape of PQ migration approaches. Section 8 identifies open questions and directions for future work. Section A contains a note on the post-quantum security of FRI-based STARKs. Section B discusses the mechanism's applicability to other blockchain systems. Section C provides the detailed Bitcoin instantiation.

2 Assumptions and Scope

This section formally defines the preconditions, eligible record classes, threat model, and explicit non-assumptions under which the proposed turnstile mechanism operates. Every claim in subsequent sections is conditioned on the assumptions stated here. We begin with the system-level preconditions that must hold before the mechanism can be deployed, then characterize the record set to which it applies, and finally specify the adversarial model.

We do not advocate for any particular system adopting the changes described below. The turnstile mechanism is designed to be available *if* the necessary preconditions are met; it is a supplement to broader migration efforts, not a driver of them.

2.1 System Model and Preconditions

The turnstile mechanism does not operate in isolation. It presupposes several system-level capabilities, each of which is a substantial undertaking in its own right. We state these as formal assumptions so that the security analysis in Section 5 can reference them precisely.

Assumption 1 (PQ signature verification support)**.** The system supports verification of post-quantum signatures: a post-quantum signature scheme PQ has been adopted such that the system can verify PQ signatures and recognize PQ-secured records. The turnstile mechanism is agnostic to the specific PQ scheme chosen; it requires only that the scheme's key generation function accepts a pseudorandom seed as input (see Assumption 4). See Section C for the Bitcoin instantiation via BIP-360 [13].

Assumption 2 (Classical signing paths disabled)**.** Classical (ECDSA/EdDSA) signing paths have been disabled: the system no longer accepts classical signatures for authentication. Records locked to classical public key commitments can no longer be spent via their original signing paths. The turnstile mechanism is designed to mitigate the collateral damage of such a freeze by providing a recovery path for hash-protected records whose owners retain their keys. It is compatible with any activation strategy; we require only that classical signing has been disabled. See Section C for Bitcoin-specific freeze proposals including QBIP Phase B [24] and QRAMP [26].

Assumption 3 (STARK proof verification)**.** The system can verify STARK proofs as part of its validation rules. STARK verification can be realized as a validation rule that invokes an external STARK verification library when a migration request is encountered, rejecting the request if proof verification fails. The specific integration mechanism is outside the scope of this paper; we require only that the verification capability exists and that the STARK proof system satisfies completeness, computational soundness, and zero-knowledge (see Section 3.5 and Section A for a discussion of post-quantum security). See Section C for Bitcoin-specific integration paths.

Assumption 4 (PQ keygen from pseudorandom seed)**.** The adopted PQ signature scheme PQ accepts a pseudorandom seed as input to its key generation function:

$$(\mathsf{pk}_{\mathsf{pq}}, \mathsf{sk}_{\mathsf{pq}}) \leftarrow \mathsf{PQ.Keygen}(\mathsf{seed})$$

This is a standard interface for all NIST-approved PQ signature schemes. ML-DSA (FIPS 204 [27]) generates keys from a 32-byte seed; SLH-DSA (FIPS 205 [28]) uses a similar seed-based generation; and hash-based schemes such as XMSS and SHRINCS [15] derive their entire key tree from a short seed. The ECDSA private key $\mathsf{sk} \in \mathbb{F}_q$ is a 256-bit (32-byte) scalar sampled uniformly at random during wallet creation. Rather than using sk directly as the PQ seed, we derive the seed via HKDF [25]:

$$\mathsf{prk} = \text{HKDF-Extract}_{\text{SHA-256}}(\mathsf{salt} = \texttt{"PQ-TURNSTILE-v1"},\ \mathsf{IKM} = \mathsf{sk}) \tag{1}$$

$$\mathsf{seed}_{\mathsf{pq}} = \text{HKDF-Expand}_{\text{SHA-256}}(\mathsf{prk},\ \mathsf{info} = \texttt{"PQ-TURNSTILE-KEYGEN"},\ L = 32) \tag{2}$$

where HKDF-Extract$(\mathsf{salt}, \mathsf{IKM})$ = HMAC-SHA256$(\mathsf{salt}, \mathsf{IKM})$ and HKDF-Expand produces L bytes of output keying material via iterated HMAC [29].

This extract-then-expand construction serves four purposes. First, the extraction step is a provable randomness extractor [25, 29]: it produces a pseudorandom key prk that is cryptographically independent of sk under the PRF assumption on HMAC, even if sk has non-uniform entropy distribution. Second, the two-stage structure provides stronger domain separation than a single tagged hash: the salt in the extract step and the info string in the expand step bind the derivation to a specific protocol context, following the key separation principles of NIST SP 800-57 [30]. Third, this mirrors the construction used in RFC 6979 [31] for deterministic

ECDSA nonce generation, a directly analogous prior art for deriving secondary secrets from sk. Fourth, the expand step supports extensibility: if the protocol later requires additional derived keys (e.g., for PQ scheme agility), HKDF-Expand can produce multiple independent outputs from the same prk by varying the info parameter, without re-extracting from sk.

The derived seed $\mathsf{seed}_{\mathsf{pq}}$ provides at least 256 bits of entropy (as the output of HMAC-SHA256 applied to a uniformly random 256-bit input), which meets or exceeds the seed requirements of all candidate PQ schemes at NIST Security Level I through V.

Remark 1 (Circuit cost of HKDF inside the STARK)**.** HKDF-Extract is a single HMAC-SHA256 evaluation (two SHA-256 compressions: one for the inner hash, one for the outer hash). HKDF-Expand with $L = 32$ requires one additional HMAC-SHA256 evaluation. The total overhead relative to a single raw SHA-256 call is therefore three additional SHA-256 compression function evaluations, which is negligible compared to the ~ 256 elliptic curve point operations required for $\mathsf{sk} \cdot G$. In zkVM environments with SHA-256 precompiles [10, 32], this adds single-digit milliseconds to proving time.

Remark 2. Assumption 4 captures the key property exploited by the turnstile (and by prior proposals such as signature lifting [18]): the same secret that controls a legacy record can deterministically generate a PQ keypair through a domain-separated derivation, enabling a seamless transition of spending authority without introducing any new secret material. The user's existing backup (whether a hierarchical deterministic seed, a raw private key, or a hardware wallet seed) remains the sole secret required for migration. After migration, the ECDSA private key sk is no longer used for signing; only the derived $\mathsf{sk}_{\mathsf{pq}}$ is used going forward. The domain separation ensures that the retirement of the ECDSA spending path does not create any cryptographic dependency between the old and new schemes.

2.2 Eligible Record Classes

The turnstile mechanism applies to records in a hash-committed public key system where the public key P has never been revealed on any publicly observable medium.

Assumption 5 (Unexposed public key)**.** The mechanism applies exclusively to records whose ECDSA public key P has never been revealed on the system's ledger or any publicly observable medium. The on-chain representation is a hash commitment $H(P)$ (or a derived encoding), and P itself does not appear in any transaction input, output, or witness data on any chain, including forks that share the same key derivation hierarchy.

This assumption is load-bearing for the entire construction. If P is known to an adversary with access to a cryptographically relevant quantum computer (CRQC), the adversary can compute $\mathsf{sk} = \log_G P$ via Shor's algorithm [1] and produce a valid STARK proof, making the migration indistinguishable from a legitimate one. The hash commitment $H(P)$ is the sole barrier protecting sk from quantum extraction, and it holds because hash preimage resistance is not weakened by quantum computation beyond Grover's quadratic speedup. For an n-bit hash function H, this yields approximately $n/2$-bit formal quantum security (e.g., $\sim$80-bit for $n = 160$; $\sim$128-bit for $n = 256$). The concrete cost of each Grover oracle call (which must evaluate elliptic curve scalar multiplication over E as a reversible quantum circuit) renders the attack intractable in practice (see Remark 6 for the detailed analysis) [33, 34].

See Section C.2 for the specific Bitcoin UTXO classes that satisfy this assumption and a quantitative estimate of the eligible set.

2.3 Excluded Record Classes

Records whose public key P is already exposed (whether embedded directly in the on-chain representation, revealed by a prior authentication operation, or disclosed through a related

system) are *not* eligible for the turnstile mechanism. A quantum adversary who knows P can derive sk via Shor's algorithm and produce a valid proof indistinguishable from the legitimate owner's. See Section C.3 for a concrete enumeration in the Bitcoin setting.

2.4 Threat Model

We consider an adversary $\mathcal{A}$ with the following capabilities:

1. **Quantum computation.** $\mathcal{A}$ has access to a cryptographically relevant quantum computer (CRQC) capable of running Shor's algorithm on the elliptic curve group (E, G, q). The best known resource estimate is approximately $6\lceil \log_2 q \rceil$ logical qubits [1] (2,124 for secp256k1 [2]). Physical qubit requirements depend on error correction overhead but are estimated at 13 million or fewer for a 24-hour attack window [35]. Gidney [36] has demonstrated continued downward pressure on these estimates, achieving RSA-2048 factorization with fewer than one million noisy qubits.

2. **Full ledger observation.** $\mathcal{A}$ can observe all data on the system's ledger, the peer-to-peer network (including pending broadcasts), and all public fork chains. Any public key P that has ever appeared in any transaction on any of these networks is assumed to be known to $\mathcal{A}$.

3. **Classical computation.** $\mathcal{A}$ has polynomially bounded classical computational resources. In particular, $\mathcal{A}$ cannot break collision resistance or preimage resistance of H (or its constituent primitives) by classical means.

The adversary *cannot*:

1. **Break hash preimage resistance.** Grover's algorithm provides a quadratic speedup for unstructured search, reducing the effective security of an n-bit hash to approximately $n/2$ bits [33]. At 10^9 quantum operations per second, a Grover search over 128-bit space would require approximately 10^{19} seconds (over 300 billion years). Furthermore, Grover's algorithm does not parallelize effectively [4]. We therefore treat H and its constituent primitives (e.g., SHA-256, RIPEMD-160) as preimage-resistant against $\mathcal{A}$. This is consistent with NIST's position that symmetric and hash-based primitives do not require migration as part of post-quantum standardization [5].

2. **Recover P from $H(P)$ for unexposed keys.** This follows directly from hash preimage resistance. If P has never appeared on chain or in any observable medium, $\mathcal{A}$ possesses only $H(P)$ and cannot recover P to apply Shor's algorithm. This is the core security property on which the turnstile relies.

3. **Forge STARK proofs.** By the computational soundness of the STARK proof system (see Section 3.5 and Section A), $\mathcal{A}$ cannot produce a valid proof for a false statement. Concretely, $\mathcal{A}$ cannot produce a proof that a given sk' satisfies the turnstile relation (Section 4.2) unless sk' is in fact the correct private key for the target record.

2.5 Non-Assumptions

To clarify the scope of the proposal and prevent misreading, we explicitly state several things the mechanism does *not* assume:

- **No specific PQ signature scheme.** The construction is parameterized over any PQ signature scheme satisfying Assumption 4. We do not require ML-DSA, SLH-DSA, FN-DSA, or any particular algorithm. The choice of scheme affects proof size and on-chain costs (Section 6) but not the security argument.

- **No specific STARK implementation or zkVM.** We require only a proof system satisfying completeness, computational soundness, and zero-knowledge, with no trusted setup. Care must be taken in selecting a proving backend that maintains post-quantum security end-to-end. Some production zkVMs incorporate non-PQ cryptographic assumptions in their core proof systems. Additionally, most zkVMs offer an optional STARK-to-SNARK wrapping layer (e.g., Groth16 or PLONK over BN254) for compact on-chain verification in EVM environments; this wrapping is not required but, if used, introduces pairing-based assumptions that are also quantum-vulnerable. For the turnstile mechanism, the proving backend must avoid discrete-log-based or pairing-based components in its core proof system, as these are known to be quantum-vulnerable. Lattice-based assumptions (e.g., Module-LWE) are believed to be post-quantum secure and are the foundation of NIST-standardized schemes such as ML-DSA [27]; however, they introduce additional cryptographic assumptions beyond hash function security. A purely hash-based proving system minimizes the assumption surface: its post-quantum security reduces entirely to the collapsing property of the hash function [37, 38]. RISC Zero [10] uses FRI-based STARKs with hash-based commitments and no elliptic curve assumptions in its core proof system; StarkWare's Stwo [11] similarly relies only on hash-based cryptography. These or comparable hash-only proving systems are suitable, provided any optional SNARK wrapping is disabled. A lattice-based proof system would also be PQ-safe under current understanding, but would carry a broader assumption set. The choice of proving system affects proof generation time and proof size but, provided it avoids quantum-vulnerable assumptions, does not affect the security argument.

- **No specific seed derivation standard.** The witness in the STARK proof is the ECDSA private key sk itself, not a seed phrase or mnemonic. Our formulation covers any user who possesses sk, regardless of how it was derived: from a hierarchical deterministic seed, from a raw private key, from a brainwallet, or from any other source. A user with a seed phrase can derive sk via the standard derivation path and then use it as the witness; no additional assumptions about the derivation mechanism are required.

- **No trusted setup.** STARKs, unlike many SNARK constructions (e.g., Groth16, PLONK with KZG commitments), require no trusted setup ceremony and no structured reference string. The only public parameters are the hash function and the agreed-upon proof statement. This property is essential for a decentralized consensus mechanism, where any form of trusted setup would be incompatible with the system's trust model [39].

- **No assumption about quantum timeline.** The mechanism is designed to be deployed *before* a CRQC exists, as a component of a proactive migration strategy. It does not assume that quantum computers have already broken ECDSA, nor does it assume any particular timeline for when they will. The Global Risk Institute survey [3] places the probability of a CRQC within ten years at 30–50%; IBM's hardware roadmap projects approximately 2,000 logical qubits (near the threshold for 256-bit elliptic curves) by 2033 [4]. The mechanism remains valid regardless of when or whether these projections materialize.

3 Preliminaries

This section establishes the notation, definitions, and background material referenced throughout the paper. Readers familiar with elliptic curve cryptography, hash-committed public key systems, post-quantum signature schemes, and zero-knowledge proof systems may wish to skim this section and refer back as needed.

3.1 Notation

We use the following notation throughout the paper.

Symbol	Description
(E, G, q)	Elliptic curve group of prime order q with generator G
sk	Private key; a scalar in $[1, q-1]$
P	Public key; $P = \mathsf{sk} \cdot G$
$\mathsf{encode}(P)$	Deterministic injective encoding of P to a byte string
$H(\cdot)$	Commitment hash function; $H : \{0,1\}^* \to \{0,1\}^n$
n	Output length of H in bits
H_{chain}	On-chain hash commitment; $H_{\text{chain}} = H(\mathsf{encode}(P))$
prk	Pseudorandom key; prk = HMAC-SHA256("PQ-TURNSTILE-v1", sk)
$\mathsf{seed}_{\mathsf{pq}}$	PQ keygen seed; derived via HKDF (Eqs. 1–2)
PQ	An adopted post-quantum signature scheme
$\mathsf{PQ.Keygen}(\cdot)$	PQ key generation from a pseudorandom seed
$\mathsf{pk}_{\mathsf{pq}}$	PQ public key; output of $\mathsf{PQ.Keygen}(\mathsf{seed}_{\mathsf{pq}})$
$\mathsf{sk}_{\mathsf{pq}}$	PQ private (signing) key; output of $\mathsf{PQ.Keygen}(\mathsf{seed}_{\mathsf{pq}})$
π	STARK proof for the turnstile relation $\mathcal{R}$
$\mathcal{R}$	The turnstile NP relation (Definition 1)
λ	Security parameter
$\mathsf{negl}(\lambda)$	A function negligible in λ
$\mathcal{A}$	Adversary
CRQC	Cryptographically relevant quantum computer

Table 1: Summary of notation.

The construction is parameterized over (E, G, q), encode, H, and PQ. Section C instantiates with E = secp256k1, encode = SEC1 compressed encoding (33 bytes), and $H = \mathsf{HASH160} = \mathsf{RIPEMD160} \circ \mathsf{SHA256}$ ($n = 160$).

3.2 Elliptic Curve Key Generation

The construction is defined over an elliptic curve group (E, G, q) of prime order q with standard generator G, equipped with a deterministic injective encoding $\mathsf{encode} : E \to \{0,1\}^*$ that maps curve points to byte strings. Key generation proceeds as follows:

1. Sample $\mathsf{sk} \xleftarrow{\$} [1, q-1]$ uniformly at random. In practice, sk is often derived deterministically from a seed via hierarchical deterministic derivation, but the cryptographic properties depend only on sk being a uniformly random element of $[1, q-1]$.

2. Compute $P = \mathsf{sk} \cdot G$, where G is the standard generator point and $\cdot$ denotes elliptic curve scalar multiplication (repeated point addition).

3. The public key P is an elliptic curve point. Its encoding $\mathsf{encode}(P)$ is a deterministic byte string representation (e.g., SEC1 compressed encoding for secp256k1, yielding 33 bytes).

The security of ECDSA (and Schnorr-like schemes) rests on the elliptic curve discrete logarithm problem (ECDLP): given P and G, computing sk such that $P = \mathsf{sk} \cdot G$ is believed to be computationally infeasible for classical adversaries. Shor's algorithm [1] solves the ECDLP in polynomial time on a quantum computer, requiring approximately $6\lceil \log_2 q \rceil$ logical qubits [1] (approximately 2,124 for secp256k1 [2]). This is the fundamental vulnerability that motivates post-quantum migration.

3.3 Hash-Committed Public Key Systems

Many deployed systems store a *hash commitment* $H(P)$ to the user's public key P rather than P itself. In such systems, P is revealed only when the user exercises the corresponding spending or authentication operation. Because hash preimage resistance is not meaningfully weakened by quantum computation (Grover's algorithm provides at most a quadratic speedup, and NIST does not consider hash functions to require post-quantum migration [5, 33]), $H(P)$ serves as a *quantum-safe anchor*: an adversary with a CRQC can derive sk from P via Shor's algorithm, but cannot obtain P from $H(P)$. The turnstile mechanism exploits precisely this property. See Section C.1 for the specific Bitcoin output types that instantiate this pattern.

3.4 Post-Quantum Signature Schemes

A post-quantum (PQ) signature scheme is a digital signature scheme believed to be secure against adversaries with access to quantum computers. NIST finalized three PQ standards in August 2024 [7]: FIPS 203 (ML-KEM, a key encapsulation mechanism), FIPS 204 (ML-DSA, a lattice-based signature scheme formerly known as Dilithium) [27], and FIPS 205 (SLH-DSA, a stateless hash-based signature scheme formerly known as SPHINCS+) [28]. FIPS 206 (FN-DSA, based on FALCON) remains in draft. Additional relevant schemes include XMSS and LMS (stateful hash-based schemes standardized in RFC 8391 and RFC 8554), SHRINCS [15, 40], a novel hybrid combining stateful XMSS with a stateless SPHINCS+ fallback to achieve 324-byte first signatures, and SQIsign, an isogeny-based scheme currently under NIST evaluation as a Round 2 additional signature candidate [8].

For the purposes of this paper, we require only the standard interface shared by all these schemes:

- $\mathsf{PQ.Keygen}(\mathsf{seed}) \to (\mathsf{pk_{pq}}, \mathsf{sk_{pq}})$: Deterministic key generation from a pseudorandom seed. ML-DSA generates keys from a 32-byte seed [27]; SLH-DSA uses a similar seed-based construction [28]; hash-based schemes derive their key trees from a short seed [15, 41].
- $\mathsf{PQ.Sign}(\mathsf{sk_{pq}}, m) \to \sigma$: Signature generation.
- $\mathsf{PQ.Verify}(\mathsf{pk_{pq}}, m, \sigma) \to \{0, 1\}$: Signature verification.

The critical property for the turnstile mechanism is that $\mathsf{PQ.Keygen}$ is deterministic and accepts a 256-bit seed. This allows the ECDSA private key sk to serve directly as the seed input (Assumption 4), establishing the link between legacy and PQ spending authority.

PQ signature schemes produce substantially larger signatures and public keys than ECDSA, with the notable exception of isogeny-based constructions. Campbell [42] provides detailed size comparisons: ML-DSA-44 produces 2,420-byte signatures with 1,312-byte public keys; FN-DSA-512 produces 666-byte signatures with 897-byte public keys; SLH-DSA-128s produces 7,856-byte signatures with 32-byte public keys. SQIsign occupies a distinct point in the design space, producing 148-byte signatures with 65-byte public keys [8, 43], the most compact post-quantum signature profile known. For comparison, ECDSA on a 256-bit curve (e.g., secp256k1) produces 71-byte signatures with 33-byte (compressed) public keys. The size increase for lattice- and hash-based schemes is the primary practical challenge for PQ adoption in space-constrained systems [44, 4] and motivates witness aggregation approaches such as ZK-based compression [44].

3.5 STARKs and Zero-Knowledge Proofs

A *zero-knowledge proof* (ZKP) is a protocol in which a prover convinces a verifier that a statement is true without revealing any information beyond the truth of the statement. A *succinct* ZKP (also called a succinct argument) has the additional property that the proof is short and the verifier's running time is small relative to the computation being proven [38].

STARKs (Scalable Transparent Arguments of Knowledge) are a family of succinct ZKPs introduced by Ben-Sasson et al. that satisfy several properties critical for consensus applications [38, 11]:

- **Completeness.** If the statement is true, the honest prover can produce a proof that the verifier accepts.
- **Computational soundness.** If the statement is false, no computationally bounded prover can produce a proof that the verifier accepts, except with negligible probability in the security parameter λ.
- **Zero-knowledge.** The proof reveals nothing about the private witness beyond what is implied by the truth of the statement.
- **Transparency (no trusted setup).** The proof system requires no trusted setup ceremony, structured reference string, or secret parameters. The only public parameters are the hash function and the agreed-upon statement format. This property distinguishes STARKs from pairing-based SNARKs (e.g., Groth16, PLONK with KZG commitments), which require a trusted setup [39].
- **Succinctness.** The proof size is polylogarithmic in the computation size, and verification time is polylogarithmic in the computation size [38]. In practice, STARK proofs range from 45 to 200 KB, and verification completes in single-digit milliseconds [11, 12].
- **Post-quantum security (conjectured).** STARKs rely on collision-resistant hash functions (specifically, the *collapsing* property [37]) rather than elliptic curve or lattice assumptions. They are therefore believed to be secure against quantum adversaries. Chiesa et al. [38] proved post-quantum security for PCP-based succinct arguments; the extension to IOP-based constructions (which underlies FRI, the commitment scheme used in modern STARKs) remains formally open. See Section A for a detailed discussion.

Practical realization: zkVMs. A zero-knowledge virtual machine (zkVM) is a general-purpose system that generates STARK (or SNARK) proofs for arbitrary computations expressed as programs in a standard instruction set (typically RISC-V). The user writes a program encoding the relation to be proven, and the zkVM handles arithmetization, constraint generation, and proof construction automatically. Major zkVM platforms include RISC Zero [10], SP1 [32, 45], and StarkWare's Stwo [11]. For the turnstile mechanism, the user writes a program that takes sk as private input, computes $P = \mathsf{sk} \cdot G$, $H(\mathsf{encode}(P))$, and $\mathsf{PQ.Keygen}(\mathsf{sk})$, and outputs the public values $(H_{\text{chain}}, \mathsf{pk}_{\mathsf{pq}})$ for verification. The zkVM proves correct execution of this program without revealing sk or any intermediate values. As noted in Remark 9, care must be taken to select a zkVM whose proving backend relies exclusively on hash-based cryptographic assumptions to maintain post-quantum security end-to-end.

4 The Turnstile Construction

4.1 Overview

We describe a protocol by which the owner of a frozen legacy record locked to $H(P)$ can migrate signing authority to a post-quantum signature scheme in a single non-interactive step. The protocol produces a *migration request* that is verified by the system and atomically re-registers the record under a new PQ public key. We call this a *turnstile* because the transition is one-directional: credentials pass from the legacy ECDSA domain into the PQ domain, gated by a zero-knowledge proof of ownership, and cannot return. This directionality is enforced by the classical signing freeze (Assumption 2), which prevents any subsequent ECDSA authentication.

The protocol proceeds as follows. Let sk be the ECDSA private key controlling a frozen record, and let H_{chain} denote the on-chain hash commitment $H(\mathsf{encode}(P))$.

1. **Key derivation.** The user computes $P = \mathsf{sk} \cdot G$ (the public key on E) and verifies locally that $H(P) = H_{\text{chain}}$. The user then derives the domain-separated PQ seed and computes the PQ keypair:

$$\mathsf{seed}_{\mathsf{pq}} \leftarrow \text{HKDF}(\mathsf{sk}), \quad (\mathsf{pk}_{\mathsf{pq}}, \mathsf{sk}_{\mathsf{pq}}) \leftarrow \mathsf{PQ.Keygen}(\mathsf{seed}_{\mathsf{pq}})$$

   per Assumption 4 (Eqs. 1–2). The two-stage HKDF extract-then-expand construction ensures cryptographic independence between the ECDSA and PQ key material. This step is performed entirely offline and reveals nothing.

2. **Proof generation.** The user generates a STARK proof π attesting to the relation defined in Section 4.2. The proof binds three facts: (a) the user knows a secret sk whose corresponding public key hashes to H_{chain}, (b) the PQ public key $\mathsf{pk}_{\mathsf{pq}}$ is deterministically derived from that same sk, and (c) neither sk nor P is revealed. Proof generation can be performed on consumer hardware using a general-purpose zkVM (see Section 6.1 for benchmarks from RISC Zero [10], SP1 [32], and Stwo [11]).

3. **Request construction and broadcast.** The user constructs a migration request referencing the frozen record as input and specifying $\mathsf{pk}_{\mathsf{pq}}$ as the signing authority for the output (see Section 4.3 for the request format). The witness contains the STARK proof π and the public value $\mathsf{pk}_{\mathsf{pq}}$. The request is broadcast to the peer-to-peer network.

4. **Network verification.** Full nodes verify π against the public inputs $(H_{\text{chain}}, \mathsf{pk}_{\mathsf{pq}})$. STARK verification is succinct [38]: polylogarithmic in the size of the proven computation. If verification succeeds, the node accepts the transaction as valid.

5. **State transition.** Upon inclusion in a block, the frozen record is consumed and a new record is created, locked to $\mathsf{pk}_{\mathsf{pq}}$ under the PQ signing rules established by Assumption 1. All subsequent spends from this output require a valid PQ signature $\sigma \leftarrow \mathsf{PQ.Sign}(\mathsf{sk}_{\mathsf{pq}}, m)$, verified by the network via $\mathsf{PQ.Verify}(\mathsf{pk}_{\mathsf{pq}}, m, \sigma)$.

After step 5, the migration is complete. The user's funds are now secured by the PQ scheme, and the original ECDSA spending path no longer exists. The user's secret material has not changed: the same sk (or the seed from which it was derived) remains the root of spending authority, now expressed through the PQ keypair rather than the ECDSA keypair.

4.2 The Proof Statement

The STARK proof establishes knowledge of a witness satisfying an NP relation $\mathcal{R}$. We define $\mathcal{R}$ formally as follows.

Definition 1 (Turnstile relation $\mathcal{R}$)**.** Let (E, G, q) be an elliptic curve group of prime order q with generator G, let $\mathsf{encode} : E \to \{0,1\}^*$ be a deterministic point encoding, let $H : \{0,1\}^* \to \{0,1\}^n$ be the commitment hash function, and let $\mathsf{PQ.Keygen} : \{0,1\}^{256} \to \mathcal{K}$ be the key generation function of the adopted PQ signature scheme (Assumption 4). The turnstile relation is:

$$\mathcal{R} = \left\{ ((H_{\text{chain}}, \mathsf{pk}_{\mathsf{pq}});\ \mathsf{sk}) \;\middle|\; \begin{array}{l} \mathsf{sk} \in [1, q-1] \\ P = \mathsf{sk} \cdot G \\ H(\mathsf{encode}(P)) = H_{\text{chain}} \\ \mathsf{prk} = \text{HMAC-SHA256}(\texttt{"PQ-TURNSTILE-v1"},\ \mathsf{sk}) \\ \mathsf{seed}_{\mathsf{pq}} = \text{HKDF-Expand}(\mathsf{prk},\ \texttt{"PQ-TURNSTILE-KEYGEN"},\ 32) \\ \mathsf{PQ.Keygen}(\mathsf{seed}_{\mathsf{pq}}) = (\mathsf{pk}_{\mathsf{pq}}, \cdot) \end{array} \right\}$$

where $\mathsf{encode}(P)$ denotes the deterministic encoding of the elliptic curve point P (see Section 3.1; e.g., SEC1 compressed encoding for secp256k1).

The public inputs (also called the *instance*) are the pair $(H_{\text{chain}}, \mathsf{pk}_{\mathsf{pq}})$. The private witness is sk. The proof π demonstrates that there exists a witness sk satisfying all five conditions simultaneously, without revealing sk, the intermediate value P, or $\mathsf{seed}_{\mathsf{pq}}$.

Remark 3 (Role of each condition)**.** Each condition in $\mathcal{R}$ serves a distinct security purpose:

- $\mathsf{sk} \in [1, q-1]$ ensures the witness is a valid scalar in the curve's scalar field (i.e., an integer in $[1, q-1]$ where q is the group order).
- $P = \mathsf{sk} \cdot G$ performs the elliptic curve key derivation, computing the public key from the private key via scalar multiplication.
- $H(\mathsf{encode}(P)) = H_{\text{chain}}$ binds the proof to a specific on-chain record. Without this condition, a prover could use an arbitrary sk unrelated to any existing commitment.
- The HKDF extract and expand steps (Eqs. 1–2) apply domain separation, deriving the PQ seed from sk through a two-stage HMAC-based construction. This ensures cryptographic independence between the ECDSA and PQ key domains: knowledge of $\mathsf{pk}_{\mathsf{pq}}$ does not leak information about sk beyond what is implied by the HMAC-SHA256 preimage barrier, and any algebraic structure in sk is destroyed by the extraction step before it enters the PQ key generation process.
- $\mathsf{PQ.Keygen}(\mathsf{seed}_{\mathsf{pq}}) = (\mathsf{pk}_{\mathsf{pq}}, \cdot)$ ensures the PQ public key is deterministically derived from the domain-separated seed. Without this condition, an adversary who obtained sk (e.g., from a compromised backup) could race the legitimate owner by registering a different PQ public key. The deterministic binding eliminates this degree of freedom: given sk, there is exactly one valid $\mathsf{pk}_{\mathsf{pq}}$.

Remark 4 (Computational cost within the proof)**.** The dominant computational operation within the proof is the elliptic curve scalar multiplication $P = \mathsf{sk} \cdot G$. This is a well-studied operation in the ZK literature. Early circom implementations required approximately 1.5 million R1CS constraints for full ECDSA verification on secp256k1 [46]; Tehrani [47] reduced this to approximately 200,000 constraints with a proving time of roughly 15 seconds on consumer hardware. Spartan-ECDSA [48] achieves approximately 4 seconds for secp256k1 verification in a web browser. Modern zkVM platforms (RISC Zero [10], SP1 [32]) provide hardware-accelerated secp256k1 precompiles, and the Fenbushi zkVM benchmarking report [12] measures full ECDSA proof generation at approximately 7 seconds on desktop hardware. The H and $\mathsf{PQ.Keygen}$ computations are comparatively inexpensive, as hash functions and PQ key derivation are native operations in most zkVMs.

4.3 Migration Request Format

At a generic level, a migration request references one or more frozen records by their identifier, and carries two pieces of data: the STARK proof π and the PQ public key $\mathsf{pk}_{\mathsf{pq}}$ (a public input to π). The system creates new records locked to $\mathsf{pk}_{\mathsf{pq}}$, preserving the original value (minus any fees). See Section C.4 for the concrete Bitcoin transaction format.

4.4 Binding Properties

The turnstile construction achieves three binding properties that together ensure only the legitimate owner can migrate a given record, and that the resulting PQ key is uniquely determined. We state these informally here; the formal security analysis follows in Section 5.

Commitment binding. The proof π is verified against a specific H_{chain} extracted from the referenced record's commitment. A proof generated for one record cannot be replayed against a different record (with a different H_{chain}) because the hash check in $\mathcal{R}$ would fail. This prevents cross-record replay.

Key binding. The deterministic derivation $\mathsf{PQ.Keygen}(\mathsf{sk}) = (\mathsf{pk_{pq}}, \cdot)$ within $\mathcal{R}$ ensures that for any given sk, there is exactly one valid $\mathsf{pk_{pq}}$. An adversary who somehow obtains sk (which would require breaking hash preimage resistance per Section 2.4) cannot register an alternative PQ key. The legitimate owner and the adversary would derive the same $\mathsf{pk_{pq}}$, reducing the attack to a race condition on request inclusion rather than a key substitution attack. This race is no different from the existing double-spend problem, which the system's consensus mechanism already handles.

Ownership binding. The STARK proof demonstrates knowledge of sk without revealing it. By the soundness of the proof system ([38]; see also Section A), no entity that does not know sk can produce a valid proof. Combined with Assumption 5 (the public key P has never been revealed), this ensures that only the holder of sk can initiate the migration. The zero-knowledge property ensures that the proof itself does not leak information about sk or P that could be used in a subsequent attack on other records controlled by the same key hierarchy.

4.5 Consensus Validation Rules

Beyond STARK proof verification, the system must enforce eligibility checks to ensure that the turnstile is applied only to records whose public key has never been revealed. The security of the mechanism depends not only on the cryptographic properties of the STARK but also on the network's ability to reject migration attempts for records that fall outside the eligible set (Section 2.2). See Section C.5 for the specific Bitcoin consensus rules (output type check, address reuse check) and a discussion of the limits of on-chain enforcement.

5 Security Analysis

5.1 Correctness

We show that a legitimate owner holding sk can always produce a valid migration request that the system will accept.

Theorem 1 (Correctness)**.** *Let $\mathsf{sk} \in \mathbb{F}_q$ be the private key controlling a frozen record with on-chain commitment H_{chain}, where the corresponding public key $P = \mathsf{sk} \cdot G$ has never been revealed (Assumption 5). If the STARK proof system satisfies completeness, then the legitimate owner can produce a proof π that the system will accept, thereby migrating the record to a PQ-secured output locked to $\mathsf{pk_{pq}}$.*

Proof. The owner computes $P = \mathsf{sk} \cdot G$ via elliptic curve scalar multiplication and verifies $H(\mathsf{encode}(P)) = H_{\text{chain}}$. This holds by construction, since the address was originally derived from sk via the same procedure. The owner then computes $(\mathsf{pk_{pq}}, \mathsf{sk_{pq}}) \leftarrow \mathsf{PQ.Keygen}(\mathsf{sk})$, which is a deterministic function (Assumption 4). All four conditions of $\mathcal{R}$ (Definition 1) are satisfied. By the completeness property of the STARK proof system [38], the honest prover can generate a proof π that the verifier accepts with probability 1 (or $1 - \mathsf{negl}(\lambda)$ for computational completeness, where λ is the security parameter). The migration transaction containing π and $\mathsf{pk_{pq}}$ passes consensus validation and is included in a block. □

Remark 5. Correctness holds regardless of the specific PQ scheme or STARK implementation, provided they satisfy their respective standard definitions. The owner requires only sk (or a seed

from which sk can be derived via hierarchical deterministic derivation) and access to a STARK prover. No interaction with any third party is required.

5.2 Soundness: Resistance to Unauthorized Migration

The central security property is that no entity other than the legitimate owner of sk can produce a valid migration request for a record locked to H_{chain}. We first analyze the three adversarial scenarios informally to build intuition, then state and prove the formal theorem.

Scenario 1: Classical adversary. A classical adversary $\mathcal{A}$ who does not know sk and does not possess a quantum computer must either (a) find sk such that $H(\mathsf{encode}(\mathsf{sk} \cdot G)) = H_{\text{chain}}$, which requires breaking the preimage resistance of H composed with elliptic curve scalar multiplication, or (b) forge a STARK proof for a false statement, which requires breaking the computational soundness of the proof system. Under standard cryptographic assumptions (H and its constituent primitives are preimage-resistant; the proof system is sound [38]), both approaches are computationally infeasible.

Scenario 2: Quantum adversary with H_{chain} only. This is the primary threat scenario. $\mathcal{A}$ possesses a CRQC and knows H_{chain} (which is public on chain) but does not know P (Assumption 5). Shor's algorithm [1] computes discrete logarithms on elliptic curves, but it requires the public key P as input. To obtain P from H_{chain}, $\mathcal{A}$ must invert H, which is a hash preimage problem.

We analyze this carefully. The attacker seeks any $\mathsf{sk}' \in [1, q-1]$ such that

$$H(\mathsf{encode}(\mathsf{sk}' \cdot G)) = H_{\text{chain}}.$$

The search space is the scalar field $\mathbb{F}_q$ with $|\mathbb{F}_q| \approx 2^{\lceil \log_2 q \rceil}$. When $\lceil \log_2 q \rceil > n$ (i.e., the scalar field is larger than the hash output space), the pigeonhole principle implies that the expected number of preimages for any given H_{chain} value is approximately

$$M \;\approx\; \frac{|\mathbb{F}_q|}{2^n}. \tag{3}$$

Grover's algorithm [49, 33] with M solutions in a search space of size N achieves the optimal query complexity

$$O\left(\sqrt{\frac{N}{M}}\right) \;=\; O\left(\sqrt{2^n}\right) \;=\; O\left(2^{n/2}\right) \tag{4}$$

quantum oracle evaluations [34]. The formal quantum security of H as a preimage-resistant function in this context is therefore **approximately $n/2$ bits**. For Bitcoin's $\mathsf{HASH160}$ ($n = 160$) this yields $\sim$80-bit security; for SHA-256 ($n = 256$), $\sim$128-bit security.

Remark 6 (Concrete security margin)**.** The $n/2$-bit bound counts *oracle queries*, not wall-clock time. Each Grover oracle call for this problem must implement $f(\mathsf{sk}) = H(\mathsf{encode}(\mathsf{sk} \cdot G))$ as a *reversible quantum circuit*. This requires: (a) a full elliptic curve scalar multiplication over E ($\approx \lceil \log_2 q \rceil$ double-and-add steps over a prime field), estimated at 1.26×10^{10} Toffoli gates for 256-bit curves [2]; (b) the point encoding encode; and (c) evaluation of H (which may involve multiple chained hash primitives, e.g., SHA-256 at $\approx 2.1 \times 10^5$ Toffoli gates per evaluation [50]). The dominant cost is the elliptic curve scalar multiplication. We illustrate with Bitcoin's parameters ($n = 160$, $q \approx 2^{256}$): at a generous estimate of 10^9 logical operations per second on a future CRQC, 2^{80} oracle evaluations each requiring $\sim 10^{10}$ gates would take approximately $2^{80} \times 10^{10}/10^9 \approx 10^{25}$ seconds ($\approx 3 \times 10^{17}$ years). Even accounting for architectural improvements, this exceeds any plausible quantum computing timeline by many orders of magnitude.

Furthermore, Grover's algorithm does not parallelize effectively: k quantum computers provide only a $\sqrt{k}$ speedup rather than a linear one [51, 4]. A fleet of 10^6 quantum computers would reduce the effective query count by only a factor of 10^3, leaving $\sim 10^{22}$ seconds of computation.

We therefore conclude that while the formal quantum security of an n-bit commitment hash is approximately $n/2$ bits (which, for $n < 256$, falls below NIST's Category I threshold of 128-bit equivalent security [5]), the *concrete* cost of mounting a Grover preimage attack against the composite function f is computationally intractable for any foreseeable quantum technology.

Scenario 3: Quantum adversary with knowledge of P. If the public key P has been revealed (violating Assumption 5), Shor's algorithm [1] recovers sk from P in quantum polynomial time with $\sim$2,124 logical qubits [2]. The adversary then holds the same witness as the legitimate owner and can produce a valid migration request. The turnstile cannot protect against this scenario; it is excluded by assumption. Records with exposed public keys require proactive migration before a CRQC exists (see Section 2.3).

We now state and prove the formal soundness theorem.

Theorem 2 (Soundness)**.** *Let $f : [1, q-1] \to \{0,1\}^n$ denote the composite function $f(\mathsf{sk}) = H(\mathsf{encode}(\mathsf{sk} \cdot G))$. Assume:*

(i) Assumption 5: the public key P corresponding to H_{chain} has not been revealed.

(ii) The STARK proof system is a quantum argument of knowledge*: there exists a quantum polynomial-time (QPT) extractor $\mathcal{E}$ such that for any QPT prover $\mathcal{A}^*$ that causes the verifier to accept on instance $(H_{\text{chain}}, \mathsf{pk_{pq}}^*)$ with probability ε, $\mathcal{E}$ extracts a witness sk^* satisfying $\mathcal{R}$ (Definition 1) with probability at least $\varepsilon - \mathsf{negl}(\lambda)$. See Remark 7 and Section A.*

(iii) f is quantum preimage-resistant: no QPT algorithm can find a preimage of f with non-negligible probability. For an n-bit hash output, the optimal quantum query complexity is $O(2^{n/2})$ (Equation (4)).

Then no QPT adversary $\mathcal{A}$ can produce a valid migration request for a record locked to H_{chain} except with negligible probability in the security parameter λ.

Proof. We proceed by reduction to the quantum preimage resistance of f. Assume for contradiction that there exists a QPT adversary $\mathcal{A}$ that, given a target commitment H_{chain} on the ledger, outputs a pair $(\pi^*, \mathsf{pk_{pq}}^*)$ such that the STARK verifier accepts on instance $(H_{\text{chain}}, \mathsf{pk_{pq}}^*)$ with non-negligible probability $\varepsilon(\lambda)$.

Step 1 (Extraction). By assumption (ii), the STARK proof system is a quantum argument of knowledge. Applying the QPT extractor $\mathcal{E}$ to $\mathcal{A}$, we obtain a witness $\mathsf{sk}^* \in [1, q-1]$ satisfying all conditions of the turnstile relation $\mathcal{R}$ (Definition 1) with probability at least $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$. In particular, the extracted sk^* satisfies:

$$f(\mathsf{sk}^*) = H(\mathsf{encode}(\mathsf{sk}^* \cdot G)) = H_{\text{chain}}, \tag{5}$$

$$\mathsf{PQ.Keygen}\big(\mathrm{HKDF}(\mathsf{sk}^*)\big) = (\mathsf{pk_{pq}}^*, \cdot). \tag{6}$$

Equation (5) establishes that sk^* is a preimage of H_{chain} under f.

Step 2 (Reduction to preimage resistance). We construct a QPT algorithm $\mathcal{B}$ that finds preimages of f, contradicting assumption (iii).

$\mathcal{B}$ receives a preimage challenge: a value $y \in \{0,1\}^n$ sampled as $y = f(\hat{\mathsf{sk}})$ for $\hat{\mathsf{sk}} \xleftarrow{\$} [1, q-1]$, and must output any $\mathsf{sk}' \in [1, q-1]$ with $f(\mathsf{sk}') = y$. $\mathcal{B}$ proceeds as follows:

1. $\mathcal{B}$ constructs a simulated ledger containing a frozen record whose on-chain commitment is $H_{\text{chain}} := y$. This simulation is *perfect*: in the real system, a commitment H_{chain} is of the form $f(\mathsf{sk})$ for a uniformly random $\mathsf{sk} \xleftarrow{\$} [1, q-1]$, which is identically distributed to the challenge $y = f(\hat{\mathsf{sk}})$. The adversary $\mathcal{A}$ cannot distinguish the simulated environment from a real ledger.

2. $\mathcal{B}$ executes $\mathcal{A}$ in the simulated environment. If $\mathcal{A}$ outputs a pair $(\pi^*, \mathsf{pk_{pq}}^*)$ that the verifier accepts, $\mathcal{B}$ invokes the extractor $\mathcal{E}$ to obtain sk^*.

3. $\mathcal{B}$ outputs sk^*.

Step 3 (Success analysis). $\mathcal{B}$ succeeds whenever both events occur: (a) $\mathcal{A}$ produces an accepted proof, and (b) the extractor $\mathcal{E}$ recovers a valid witness. By the argument-of-knowledge property, the joint probability of both events is at least $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$. When both events occur, Equation (5) guarantees that $f(\mathsf{sk}^*) = y$, so $\mathcal{B}$'s output is a valid preimage.

The overall success probability of $\mathcal{B}$ is therefore:

$$\Pr[\mathcal{B} \text{ finds preimage}] \;\geq\; \varepsilon(\lambda) - \mathsf{negl}(\lambda).$$

Step 4 (Contradiction). $\mathcal{B}$ is QPT: it runs $\mathcal{A}$ (QPT by assumption) and $\mathcal{E}$ (QPT by assumption (ii)) each a single time, with polynomial overhead for the simulation. By assumption (iii), no QPT algorithm finds a preimage of f with non-negligible probability. Since $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$ is non-negligible, we have a contradiction. Therefore $\varepsilon(\lambda)$ must be negligible in λ. □

Remark 7 (On the quantum knowledge soundness assumption)**.** Assumption (ii) of Theorem 2 requires the STARK proof system to be a quantum argument of knowledge (i.e., to admit a QPT extractor against quantum provers). This is the quantum analogue of the standard argument-of-knowledge property that holds classically for all deployed STARK systems.

In the quantum setting, the strongest formal result to date is due to Chiesa, Dall'Agnol, Di, Guan, and Spooner [52], who proved quantum knowledge soundness for *interactive* IOP-based succinct arguments when the vector commitment scheme is collapsing [37]. Their result covers the interactive BCS transformation [38] and applies to multi-round IOPs, which is the model underlying FRI-based STARKs.

The extension to the *non-interactive* setting (i.e., STARKs compiled via the Fiat-Shamir heuristic in the quantum random oracle model) remains formally open as of this writing. In particular, the quantum extractability of Fiat-Shamir-compiled IOPs is not covered by the Di et al. result, and we are not aware of a published proof establishing this property.

We therefore state assumption (ii) explicitly and note three mitigating factors. First, the underlying cryptographic assumption is conservative: STARKs rely only on collision-resistant (more precisely, collapsing [37]) hash functions, and no quantum attack on any hash-based proof system is known. Second, the gap between interactive and non-interactive quantum knowledge soundness is a limitation of current proof techniques (specifically, the difficulty of quantum rewinding across Fiat-Shamir compilation), not evidence of an actual vulnerability. Third, NIST-standardized post-quantum signature schemes (ML-DSA [27], SLH-DSA [28]) are similarly "believed" secure against quantum adversaries based on conjectured hardness of their underlying problems, without proofs of security in the standard model. The assumption surface of hash-based STARKs is strictly more conservative than that of any lattice-based or isogeny-based construction. A formal proof of quantum knowledge soundness for Fiat-Shamir-compiled IOPs would remove this assumption entirely and is an important open problem in the foundations of succinct arguments.

Remark 8 (Why knowledge soundness, not plain soundness)**.** The proof requires the argument-of-knowledge property (extraction), not merely plain computational soundness. The distinction

is essential. Plain soundness guarantees only that no adversary can produce a proof for a *false* statement. However, the turnstile relation $\mathcal{R}$ *does* have a valid witness (the legitimate owner's sk), so the statement $(H_{\text{chain}}, \mathsf{pk_{pq}})$ is true for the correct $\mathsf{pk_{pq}}$. An adversary who finds any $\mathsf{sk}' \neq \mathsf{sk}$ with $f(\mathsf{sk}') = H_{\text{chain}}$ can compute the corresponding $\mathsf{pk_{pq}}'$ via the deterministic HKDF derivation, and the statement $(H_{\text{chain}}, \mathsf{pk_{pq}}')$ is also true. Plain soundness does not prevent a proof for this true statement.

The argument-of-knowledge property closes this gap: it guarantees that any prover who produces an accepted proof must "know" a valid witness, in the formal sense that the extractor can recover it. The reduction in Theorem 2 then shows that knowing a valid witness is equivalent to having found a preimage of H_{chain} under f, which is hard by assumption (iii).

Note also that the deterministic HKDF key binding (Assumption 4) plays a role here. If the PQ public key were a free parameter (not derived from sk inside the proof), an adversary who found sk' with $f(\mathsf{sk}') = H_{\text{chain}}$ could register any PQ key of their choosing, turning a preimage attack into a key substitution attack. The deterministic binding eliminates this degree of freedom: the adversary's $\mathsf{pk_{pq}}'$ is uniquely determined by their sk', and whether $\mathsf{pk_{pq}}' = \mathsf{pk_{pq}}$ or not depends on whether $\mathsf{sk}' = \mathsf{sk}$. In the former case, the adversary cannot spend the migrated funds (they lack $\mathsf{sk_{pq}}$ unless they also possess sk); in the latter case, they have found a *second preimage* of H_{chain} under f, which is at least as hard as the first-preimage problem analyzed above.

5.3 Zero-Knowledge: No Information Leakage

The zero-knowledge property ensures that the migration request does not reveal information about sk or the intermediate value P beyond what can be deduced from the public inputs $(H_{\text{chain}}, \mathsf{pk_{pq}})$.

What the proof reveals. The public inputs $(H_{\text{chain}}, \mathsf{pk_{pq}})$ are both visible on chain after the migration request is confirmed. H_{chain} was already public (it is the record's on-chain commitment). $\mathsf{pk_{pq}}$ is newly revealed but is necessary for future PQ-authenticated spends. Given these public values, any observer can deduce that there exists some sk satisfying $\mathcal{R}$, but this is a tautology: the existence of such an sk is precisely what makes the record spendable. The zero-knowledge property of the STARK proof ensures that π reveals nothing beyond this existential statement [38].

What the proof does not reveal. The private key sk, the ECDSA public key P, and any intermediate values in the computation (partial products in the scalar multiplication, intermediate hash states) are not extractable from π by any polynomial-time adversary, classical or quantum. This is critical because revealing P would immediately enable a quantum adversary to derive sk via Shor's algorithm [1], compromising not only the migrated record but potentially other records derived from the same key hierarchy.

Post-migration security. After migration, the user's funds are secured by the PQ signature scheme. The private key $\mathsf{sk_{pq}}$ (derived deterministically from $\mathsf{seed_{pq}}$ via HKDF, per Assumption 4) is never transmitted or revealed. The security of future spends depends solely on the unforgeability of the PQ scheme under quantum attack [27, 28], which is the standard assumption underlying all PQ-secured outputs (Assumption 1).

Cross-scheme domain separation. A natural concern is whether the reuse of sk across two cryptographic schemes (ECDSA key derivation and PQ seed derivation) creates a correlation that could be exploited. Specifically: does the PQ public key $\mathsf{pk_{pq}}$, which is public after migration,

leak information about sk that could weaken the hash preimage barrier protecting other records controlled by the same key hierarchy?

The HKDF-based seed derivation (Assumption 4) addresses this risk. The extraction step produces prk via HMAC-SHA256 keyed by a protocol-specific salt, and the expansion step derives $\mathsf{seed_{pq}}$ bound to a distinct info string. Under the PRF assumption on HMAC, these outputs are cryptographically independent of sk and of the hash functions used in ECDSA and the on-chain address commitment. The PQ keypair is derived from $\mathsf{seed_{pq}}$, not from sk directly, and recovering sk from $\mathsf{seed_{pq}}$ requires inverting both HKDF-Expand and HKDF-Extract (preimage attacks on HMAC-SHA256). The derivation chain is:

$$\mathsf{sk} \xrightarrow{\text{HKDF-Extract}} \mathsf{prk} \xrightarrow{\text{HKDF-Expand}} \mathsf{seed_{pq}} \xrightarrow{\mathsf{PQ.Keygen}} \mathsf{pk_{pq}}$$

An adversary who observes $\mathsf{pk_{pq}}$ must invert $\mathsf{PQ.Keygen}$ (breaking the PQ scheme), then HKDF-Expand and HKDF-Extract (breaking HMAC-SHA256 preimage resistance) to recover sk.

These are independent barriers under standard assumptions. The domain separation tag `"PQ-TURNSTILE-KEYGEN"` further ensures that even if sk is used as input to other tagged hash functions in different contexts (e.g., hierarchical key derivation, message signing), the outputs are cryptographically independent. This is standard practice in modern protocol design and follows the domain separation principles established in NIST SP 800-185 and applied throughout the NIST PQ standards [27, 28].

After migration, sk is no longer used for ECDSA signing (the ECDSA spending path is frozen by Assumption 2), and the derived $\mathsf{sk_{pq}}$ serves as the sole signing key. Once migration is confirmed, sk can be securely forgotten for spending purposes: retaining it provides no additional capability beyond what $\mathsf{sk_{pq}}$ already provides, and discarding it eliminates a potential attack surface. Users who retain their seed phrase for backup purposes should understand that the seed phrase remains the root of $\mathsf{sk_{pq}}$ through the deterministic derivation chain; compromise of the seed phrase after migration would allow an attacker to rederive sk, then $\mathsf{seed_{pq}}$, then $\mathsf{sk_{pq}}$. The domain separation ensures that no information flows backward from the PQ domain to the ECDSA domain, but the forward derivation path from sk to $\mathsf{sk_{pq}}$ is deterministic by design.

Threat model boundary: direct compromise of sk. The turnstile mechanism protects against a specific threat: quantum extraction of sk from an exposed public key P via Shor's algorithm. It achieves this by ensuring P is never revealed. It does *not* protect against direct, classical compromise of sk (e.g., stolen seed phrase, compromised backup, side-channel extraction from a signing device). If an attacker obtains sk through any classical means, they can derive $\mathsf{seed_{pq}}$ via HKDF and compute the same $\mathsf{pk_{pq}}$ as the legitimate owner. This reduces the attack to a race condition on migration request inclusion, identical to the threat model for any deterministic key derivation system. This limitation is inherent to any migration mechanism that derives the new key from the old secret, including QBIP Phase C [24] and all commit-reveal schemes [53, 54, 55].

An optional defense-in-depth measure for custodians or sophisticated users would be to include an additional random nonce r as a private witness in the STARK proof, modifying the seed derivation to $\mathsf{seed_{pq}} = \text{HKDF}(\mathsf{sk} \| r)$, with the nonce r concatenated to the input keying material. This ensures that even direct compromise of sk does not yield $\mathsf{seed_{pq}}$ without knowledge of r. However, this introduces r as new secret material that must be independently generated, stored, and backed up. For institutional custodians with secure key management infrastructure, this trade-off may be acceptable. For retail users, it sacrifices the turnstile's core property that the existing backup is the sole secret required for migration. We therefore treat the nonce as an optional extension rather than a requirement of the base protocol, and note that the base protocol's threat model (protection against quantum key extraction, not against classical secret compromise) is explicitly and deliberately scoped.

5.4 Non-Malleability and Replay Protection

We require that a valid migration request cannot be modified by a third party to alter its effect, and that a proof generated for one migration cannot be reused for another.

Cross-record replay. The proof π is verified against H_{chain}, which is extracted from the specific record referenced by the request input. A proof generated for a record with commitment H_1 cannot satisfy the relation $\mathcal{R}$ for a different record with commitment $H_2 \neq H_1$, because the hash check would fail. This holds unconditionally (not depending on any computational assumption) as long as the verifier correctly extracts H_{chain} from the referenced output.

Key substitution. The deterministic derivation $\mathsf{PQ.Keygen}(\mathsf{sk}) = (\mathsf{pk_{pq}}, \cdot)$ within $\mathcal{R}$ prevents an adversary from modifying the migration request to substitute a different PQ key. If the adversary changes $\mathsf{pk_{pq}}$ to $\mathsf{pk_{pq}}'$, the proof π will fail verification because $\mathsf{pk_{pq}}'$ does not match the value committed inside the proof. Generating a new valid proof for $\mathsf{pk_{pq}}'$ requires knowledge of an sk' such that both $H(\mathsf{encode}(\mathsf{sk}' \cdot G)) = H_{\text{chain}}$ and $\mathsf{PQ.Keygen}(\mathsf{sk}') = (\mathsf{pk_{pq}}', \cdot)$. If $\mathsf{sk}' \neq \mathsf{sk}$, the hash check fails (assuming H is collision-resistant on the image of the EC group, which holds under standard assumptions [54]). If $\mathsf{sk}' = \mathsf{sk}$, then $\mathsf{pk_{pq}}' = \mathsf{pk_{pq}}$ by determinism, contradicting the assumption that the key was substituted.

Request-level binding. To prevent a network-level adversary from extracting π from a broadcast migration request and including it in a different request (e.g., one that sends the migrated funds to a different output), the proof should additionally commit to a request identifier or a nonce specific to the migration request. This can be achieved by including a request commitment as an additional public input to $\mathcal{R}$, extending the relation to:

$$\mathcal{R}' = \mathcal{R} \ \wedge \ \mathsf{req\_commit} = \mathsf{Hash}(\text{migration request fields})$$

This is a standard technique in ZK-based transaction systems [54] and adds negligible overhead to the proof circuit.

5.5 Front-Running and Broadcast Considerations

When a migration request is broadcast to the peer-to-peer network, it enters the pending pool before being included in a block. During this interval, the request's contents are visible to all network participants, including block producers and any adversary monitoring pending broadcasts. We analyze whether this creates exploitable attack vectors.

What is visible in the pending pool. The migration request contains: the identifier referencing the frozen record, the STARK proof π, and the PQ public key $\mathsf{pk_{pq}}$. The on-chain commitment H_{chain} is not transmitted as a separate field in the request; it is already public, having been embedded in the referenced record's commitment since the record was created. The verifier extracts H_{chain} from the referenced record and uses it as a public input when checking π. Crucially, H_{chain} is a public input *inside* the proof: π is bound to the specific H_{chain} at proof generation time, and verification fails if the proof is checked against a different commitment. The secret value P (the preimage of H_{chain}) is never revealed, neither in the transaction nor inside π.

Can an adversary front-run the migration? The migration request reveals $\mathsf{pk_{pq}}$ and π in the clear. An adversary who observes this transaction cannot:

1. **Produce an alternative valid proof.** This would require knowledge of sk, which the adversary does not possess (by Assumption 5 and the analysis in Section 5.2).

2. **Modify the proof to register a different PQ key.** The key binding property (Section 4.4) prevents key substitution without producing a new valid proof.

3. **Extract sk from π.** The zero-knowledge property (Section 5.3) ensures π reveals nothing about sk beyond the existential statement implied by the public inputs.

The adversary *can* observe that a migration is occurring for a specific H_{chain} and learn the resulting $\mathsf{pk}_{\mathsf{pq}}$. This is a privacy concern (the link between the legacy commitment and the new PQ key is public) but not a security concern in the sense of credential theft. Privacy implications are discussed in Section 8.

Block producer censorship. A block producer could refuse to include migration requests, delaying or preventing migration. This is the standard censorship concern for any transaction and is not specific to the turnstile mechanism. Decentralized consensus and fee markets provide the same censorship resistance properties for migration requests as for ordinary transactions. If migration requests carry competitive fees, rational block producers are incentivized to include them.

Denial-of-service via invalid proofs. An adversary could broadcast requests containing invalid STARK proofs, forcing nodes to expend verification resources before rejecting them. This is a meaningful DoS concern given that STARK verification for ECDSA-scale circuits takes 50 ms to 2 s per proof [56], substantially more than standard signature verification. Mitigations include relay-level proof validation (rejecting invalid proofs before forwarding), fee-based rate limiting, requiring a minimum record value for turnstile eligibility, and the observation that migration requests are expected to be infrequent relative to standard transaction volume.

6 Practical Considerations

6.1 Proof Generation Cost

The dominant cost of the turnstile mechanism for individual users is the generation of the STARK proof π. This is a one-time computation performed locally by the user (or delegated to a trusted proving service), and its feasibility on consumer hardware is a practical prerequisite for broad adoption.

We illustrate costs using secp256k1 as the instantiation curve, as it has the most extensive ZK benchmarking literature. The most expensive operation inside the proof circuit is the elliptic curve scalar multiplication $P = \mathsf{sk} \cdot G$. For a 256-bit curve, this involves approximately 256 double-and-add steps over a 256-bit prime field, each requiring multiple field multiplications. In early ZK implementations, full ECDSA verification on secp256k1 required approximately 1.5 million R1CS constraints and a proving key exceeding 1 GB [46]. Tehrani [47] reduced this to approximately 200,000 constraints with a proving time of roughly 15 seconds and a 134 MB proving key on a MacBook Pro. Spartan-ECDSA [48] achieves approximately 4 seconds for secp256k1 verification in a web browser on an M1 MacBook Pro, using the secq256k1 curve for right-field arithmetic to avoid the field mismatch penalty [39].

Modern general-purpose zkVMs have further reduced proving costs through hardware acceleration and dedicated precompiles. RISC Zero [10] provides a secp256k1 ECDSA verification precompile that reduces the operation to approximately 10 cycles for a 256-bit modular multiply. SP1 [32, 45] includes precompiles for secp256k1, ed25519, SHA-256, and Keccak-256, with GPU-accelerated proving that achieves sub-12-second Ethereum block proofs on 16 NVIDIA RTX 5090 GPUs [45]. The Fenbushi Capital zkVM benchmarking report [12], which tested eight zkVM platforms on ECDSA k256 (secp256k1) specifically, found that RISC Zero (GPU),

SP1 (GPU), and OpenVM (CPU) demonstrated the best time efficiency, with ECDSA proofs completing in approximately 7 seconds for simulated EVM execution on desktop-class hardware.

Remark 9 (Post-quantum safety of the proving backend)**.** Not all zkVM platforms are post-quantum secure. SP1, for example, assumes the hardness of the discrete logarithm problem over an extension of the BabyBear field [45] and employs a STARK-to-SNARK wrapping layer (Groth16 or PLONK over BN254) for proof compression, both of which are vulnerable to quantum attack. For the turnstile mechanism, where post-quantum security is the entire purpose, the proving backend must rely exclusively on hash-based cryptographic assumptions. RISC Zero [10] uses FRI-based STARKs with hash-based commitments and no elliptic curve assumptions in its core proof system, making it suitable for this application. StarkWare's Stwo prover [11] similarly relies only on hash-based cryptography. The Fenbushi benchmarks cited above remain informative for estimating proving time (the computational cost of evaluating secp256k1 inside a zkVM is similar regardless of the backend), but any deployment of the turnstile must use a PQ-safe proving system that does not introduce elliptic curve or pairing-based assumptions at any layer.

A secondary consideration is the choice of hash function used internally by the proving system. Many modern STARK implementations use Poseidon or Poseidon2 as their Merkle commitment hash due to its arithmetic-circuit efficiency [11, 10]. Poseidon's security rests on the hardness of solving systems of multivariate polynomial equations (the MQ problem), which is believed to be quantum-resistant in the same sense as any symmetric primitive: Grover's algorithm provides a quadratic speedup, reducing M-bit classical security to approximately $M/2$-bit quantum security. At 256-bit security parameters, this yields 128-bit post-quantum security, which is adequate. However, Poseidon is substantially younger than SHA-256 and has received less cryptanalytic scrutiny; the Ethereum Foundation launched a dedicated Poseidon cryptanalysis initiative and a $1M bounty program in 2025 to accelerate security evaluation [57]. For maximum conservatism in a high-assurance consensus application, a proving system using SHA-256 or another NIST-approved hash for its internal commitments would carry lower cryptographic risk, at the cost of increased proving time. This is an implementation-level tradeoff that does not affect the construction's security argument, which is parameterized over any collision-resistant hash function.

The hash computation H (e.g., SHA-256 followed by RIPEMD-160 for $\mathsf{HASH160}$) adds comparatively modest overhead, as SHA-256 is a native precompile in all major zkVMs [10, 32].

The $\mathsf{PQ.Keygen}$ computation, however, introduces scheme-dependent overhead that may be substantial. Hash-based schemes (SLH-DSA, XMSS, SHRINCS [28, 15]) derive their key material through iterated hash function evaluations, which map efficiently to zkVM hash precompiles. Lattice-based schemes (ML-DSA [27], FN-DSA) generate keys through matrix sampling and polynomial arithmetic over moderate-size rings, which is more expensive inside a proof circuit but consists of operations (modular arithmetic, NTT transforms) that are well-suited to algebraic proof systems. SQIsign [8] represents the most expensive case: its key generation involves computing endomorphism rings of supersingular elliptic curves and quaternion algebra operations, which are computationally intensive even outside a proof circuit and would incur significant overhead when arithmetized inside a zkVM. The cost of proving $\mathsf{PQ.Keygen}(\mathsf{sk})$ inside the STARK may therefore vary by one or more orders of magnitude depending on the PQ scheme selected.

This variation is mitigated by the one-time nature of the migration: even if proving $\mathsf{PQ.Keygen}$ for a given scheme takes minutes rather than seconds, the user performs this computation exactly once per record (or once per seed, if seed-level migration is implemented per Section 8). The cost is borne by the individual user, not by the network, and does not affect verification time (which depends on the proof system parameters, not on the proven computation's complexity). Nonetheless, the choice of PQ scheme has a direct impact on the practical accessibility of the migration: a scheme whose keygen is prohibitively expensive to prove inside a zkVM would effectively exclude users without access to high-performance proving hardware or a trusted proving

service. This is an additional dimension of the PQ algorithm selection debate that has received limited attention in the existing literature [4, 13].

We defer precise end-to-end proving time estimates for the turnstile relation $\mathcal{R}$ (Definition 1) to future experimental work, as extrapolating from component benchmarks (which measure ECDSA verification, not the full turnstile circuit including PQ key derivation and hash computation) risks misleading precision. The benchmarks cited above ([46, 47, 48, 12]) characterize the dominant cost (elliptic curve scalar multiplication) and establish that the operation is within reach of consumer hardware. The hash (H) and $\mathsf{PQ.Keygen}$ components add overhead that will vary by PQ scheme and hash function but consist of operations (hash evaluation, pseudorandom expansion) that are native to zkVM proving pipelines. Proving costs across all platforms continue to decline rapidly: StarkWare's Stwo prover [11] achieves over 500,000 Poseidon2 hashes per second on a quad-core Intel i7, and RISC Zero's roadmap targets further order-of-magnitude improvements through hardware-optimized proving pipelines [10].

6.2 Proof Size and On-Chain Overhead

STARK proofs are larger than traditional SNARK proofs (which can be as small as 128 to 288 bytes for Groth16 [58]) but require no trusted setup and are post-quantum secure [38]. Recent benchmarks published on Delving Bitcoin [56] provide the first concrete measurements of STARK proof sizes for hash-committed ECDSA ownership proofs (ECDSA verification and SHA-256 hashing inside a ZK circuit). On an Apple M2 Max, proof sizes ranged from **2 MB** (RISC Zero) to **10 MB** (SP1), with STWO-Cairo at 5.6 MB and Ligero at 4.2 MB. These figures are substantially larger than general-purpose STARK benchmarks (which typically report 45 to 200 KB for simpler computations) because the ECDSA verification circuit (for secp256k1) is computationally expensive, requiring field arithmetic over a 256-bit prime that does not align natively with the small fields used by modern STARK provers [39]. The turnstile circuit (Definition 1) includes everything in the Delving Bitcoin benchmark plus domain-separated hashing and PQ key generation, so proof sizes for the full turnstile relation should be expected to be in the same range or moderately larger, depending on the PQ scheme's keygen complexity.

These proof sizes are large relative to typical transactions in space-constrained systems. For example, in Bitcoin a standard transaction is approximately 110 vbytes, and a 2 MB proof would consume a substantial fraction of a single block (see Section C for specific block weight analysis).

This overhead is significant and represents one of the primary practical challenges for the turnstile mechanism. It is bounded by several considerations:

One-time cost. Each record migrates exactly once. The migration proof is verified once, included in one block, and never revisited. Post-migration, spending transactions use standard PQ signatures (e.g., 2.4 KB for ML-DSA-44, 666 bytes for FN-DSA-512, 148 bytes for SQIsign [8, 43, 59, 42]), which are orders of magnitude smaller than the migration proof. The large proof size is a one-time migration cost, not a recurring burden on the network.

Witness discount. Systems may apply a discount factor to migration proof data, reducing its effective weight relative to persistent state data. The rationale is that data verified once and not stored long-term should be weighted less heavily than data that persists. Given current proof sizes, a substantial discount or an entirely separate data carrier (as discussed below) may be necessary for on-chain feasibility.

Comparison to alternatives. Commit-delay-reveal schemes [53, 55, 54] require multiple on-chain transactions (a commitment transaction, a delay period, and a reveal transaction), each carrying its own overhead. The turnstile achieves migration in a single transaction but at the

cost of a much larger proof payload. Whether the total on-chain footprint favors the turnstile or commit-reveal depends on the proof size achievable after aggregation and compression.

Proof compression and aggregation. STARK proof sizes have been declining but remain large for ECDSA-scale circuits. The Delving Bitcoin benchmarks [56] represent pre-optimization results from mid-2025; proving systems continue to improve rapidly, with StarkWare's Stwo reporting order-of-magnitude improvements over prior generations [11]. Recursive proof composition (proving a proof of a proof) can further compress the final on-chain artifact at the cost of additional proving time. Heilman's witness aggregation proposal [44] envisions a per-block "proofcarrier" sidecar that aggregates all PQ-related witness data into a single STARK, amortizing the per-record overhead across all migrations in a block. The Delving Bitcoin thread respondent ("light") explicitly suggests this approach for migration proofs [56]. BTQ Technologies' PQScale system [60] demonstrates that batches of 1,722 Falcon signatures can be aggregated to approximately 9% of their unaggregated size. Applied to migration proofs, per-block aggregation could reduce the effective per-record cost from megabytes to kilobytes, though this capability has not yet been demonstrated for ECDSA-circuit-scale proofs.

Off-chain registries as an interim path. The Delving Bitcoin benchmarks note that current proof sizes are "fine for off-chain registries but far too large for anything on-chain" [56]. An intermediate deployment path could use the turnstile proofs in an off-chain registry that maps legacy commitments to PQ public keys, with the registry serving as an attestation layer until on-chain proof verification or aggregation becomes feasible. This weakens the trust model (the registry becomes a trusted intermediary) but provides immediate practical value while the on-chain path matures.

6.3 Verification Cost

STARK verification is succinct: the verifier's running time is polylogarithmic in the size of the proven computation [38]. This property is essential for consensus systems where every full node must verify every transaction.

For ECDSA-scale computations (approximately 2^{20} to 2^{22} trace steps), STARK verification completes in single-digit milliseconds on standard hardware [12, 11]. For comparison, a single secp256k1 ECDSA signature verification takes approximately 0.02 ms on modern hardware [42], a single ML-DSA-44 signature verification takes approximately 0.028 ms [42], and SQIsign v2 verification takes approximately 1.5 ms [43]. A STARK verification at 1 to 5 ms is therefore comparable to a single SQIsign verification and roughly 50 to 250 times more expensive than a classical ECDSA signature check, but still well within the per-transaction verification budget of a full node (which currently allocates on the order of tens of milliseconds per transaction for complex scripts).

The key observation is that verification cost scales with the *proof system parameters* (security level, number of FRI query rounds, hash function), not with the size of the proven computation. A proof for a migration involving 10 records from the same key is only marginally more expensive to verify than a proof for a single record, because the verifier checks the same proof structure regardless of the number of relation instances aggregated inside.

7 Related Work

This section situates the turnstile mechanism within the broader landscape of post-quantum migration approaches. Fukuda et al. [61] survey post-quantum migration challenges for blockchain systems, analyzing the interplay between technical migration strategies and adoption barriers. We organize the discussion into five categories. For each, we identify the relationship to our

construction (whether complementary, alternative, or overlapping) and highlight the specific contributions of the turnstile that are not present in prior work.

7.1 ZK-Based Migration and Signature Lifting

The idea of exploiting hash-based key derivation as a post-quantum anchor for zero-knowledge migration has been independently discovered multiple times. We survey these proposals chronologically and position our construction against each.

Sattath and Wyborski [18] (2023; v2 July 2024). The earliest formal treatment. Sattath and Wyborski introduce "signature lifting": lifting a deployed pre-quantum signature scheme to a post-quantum one by exploiting the fact that a post-quantum one-way function (the hash) sits in the key derivation chain between secret and public key. They instantiate this with Picnic signatures, a ZK-proof-based signature scheme, and analyze the collision resistance of BIP-32 derivation paths. Their construction builds a new *signature scheme* (the lifted scheme produces signatures that are valid under both the classical and post-quantum verification algorithms), whereas our turnstile is a *migration protocol* that atomically transitions a record from one signing regime to another. They do not address HKDF-based deterministic PQ key binding, consensus-level migration transaction formats, eligible versus excluded key classes, or concrete proof size and verification benchmarks. Nonetheless, their formalization of the underlying observation predates all subsequent work by over a year, and their analysis of BIP-32 derivation security is directly relevant to our discussion of seed-level migration (Section 8). Their paper remains an ePrint preprint (presented as a contributed talk at the 2nd PQC Standardization & Migration Workshop) and has not appeared in conference proceedings.

Buterin [19] (2024). Buterin proposes an emergency hard-fork procedure for Ethereum in which classical spends are frozen and a new transaction type allows users to provide a STARK proof of knowledge of a private preimage whose public key hashes to their account address. Users transition to ERC-4337 smart contract wallets with post-quantum validation code. This is structurally similar to the turnstile but differs in several respects: it targets Ethereum's account model rather than UTXO-based systems; it is framed as an emergency response rather than proactive migration; it allows the user to choose arbitrary new validation code (no deterministic key binding); and it is an informal forum post without formal definitions, security theorems, or feasibility analysis.

Delving Bitcoin ECDSA-in-STARK benchmarks [56] (2026). olkurbatov published benchmarks for STARK-based proofs of hash-committed ECDSA ownership without public key revelation. Their circuit proves knowledge of a public key P and a valid ECDSA signature σ such that $\mathsf{SHA256}(P)$ matches a declared hash. Their construction differs from the turnstile in two respects. First, their witness is (P, σ) rather than sk: the user signs a predetermined message on their signing device and hands the signature to a separate proving device, preserving classical key isolation. The turnstile uses sk directly as the witness, which requires sk on the proving device but enables deterministic PQ key derivation inside the proof. These are different trade-offs for different user populations. Second, their proof establishes address ownership only; it does not bind the owner to a PQ public key inside the proof. The turnstile extends the proof statement to include domain-separated PQ key derivation (Definition 1), achieving atomic re-registration in a single step. Their approach would require a separate mechanism to bind the ownership proof to a PQ key and register the migration on chain. Their benchmarks provide the first concrete proof size and proving time measurements for ECDSA-in-STARK circuits and are an essential reference for evaluating practical feasibility (Section 6.2, Section 6.3).

Kiraz and Kardas [21] (2026). Concurrent with this work, Kiraz and Kardas propose STARK circuits for migrating Bitcoin and Ethereum addresses to post-quantum schemes, with SP1 benchmarks and proposed Bitcoin opcodes (`OP_CHECKQUANTUMSIG`, `OP_CHECKSTARKPROOF`). Their circuit design addresses both HASH160-committed Bitcoin addresses and Keccak-committed Ethereum addresses. The two papers are complementary: Kiraz and Kardas focus on circuit construction, address-format coverage, and implementation benchmarks; we focus on the protocol-level migration mechanism, providing a formal NP relation definition (Definition 1), correctness and soundness theorems with a full security reduction (Theorem 1, Theorem 2), and deterministic PQ key binding via domain-separated HKDF, none of which appear in their work. Conversely, their SP1 benchmarks and multi-chain circuit designs provide implementation-level detail that we defer to future work.

Baldimtsi, Chalkias, and Roy [22] (2025). Baldimtsi et al. analyze the post-quantum readiness of EdDSA key derivation chains (presented as a poster at CCS 2025 and a talk at RWC 2025). They show that BIP-32-style ECDSA wallets are *not* PQ-ready without breaking changes, because non-hardened derivation exposes scalar relationships that a quantum adversary can exploit. In contrast, SLIP-0010 hash-chain-based EdDSA private key derivation *is* PQ-ready and supports ZK proofs of possession. Crucially, they explicitly state that their seed-based approach is "currently infeasible for ECDSA-based chains" due to the absence of deterministic key derivation in the ECDSA key generation model. Our HKDF-based key binding (Assumption 4) fills precisely this gap for ECDSA-based systems, providing the deterministic derivation mechanism that Baldimtsi et al. identify as missing. Their analysis also strengthens the case for our sk-level (not seed-level) witness approach: seed-level migration via BIP-32 non-hardened derivation would require proving the hash-chain computation inside the STARK to avoid exposing scalar relationships, whereas sk-level migration sidesteps this issue entirely.

Tan and Zhou [23] (2023). Tan and Zhou study the impact of migrating blockchains away from ECDSA for post-quantum security, focusing on BIP-39 seed compatibility and the effect on users and applications. Their work provides context for the practical migration challenges that motivate our construction.

7.2 Commit-Delay-Reveal Schemes

The commit-delay-reveal approach to quantum-safe record migration was first proposed by Ruffing [53] in 2018, building on Adam Back's 2013 committed transactions idea. The protocol proceeds in three steps: (1) the user broadcasts a commitment (a hash of the intended transaction) without revealing the public key; (2) a delay period elapses to ensure the commitment is buried; (3) the user reveals the full transaction, including the public key and ECDSA signature. The delay ensures that a quantum adversary who observes the public key in step 3 cannot produce a competing transaction before the committed transaction is confirmed.

Stewart et al. [54] formalized this approach in *Royal Society Open Science*, proposing a substantial delay phase (approximately six months) deployable as a soft fork. Dryja [55] proposed a variant based on Bonneau and Miller's Fawkescoin [62], using a three-hash commit/reveal scheme that avoids pre-selecting a PQ signature algorithm. Wandersleb [63] proposed a preemptive commit/reveal mechanism using OP_RETURN Merkle roots deployable without consensus changes.

The turnstile differs from commit-delay-reveal in several respects. It is *non-interactive*: a single migration request replaces the multi-step commit-delay-reveal sequence, eliminating the delay period during which funds are in a liminal state. The public key P is *never revealed* at any point, providing a strictly stronger privacy and security property than commit-reveal, where P is exposed during the reveal step. The turnstile's security depends on hash preimage

resistance rather than timing assumptions about quantum computation speed. The trade-off is that commit-reveal requires less complex consensus changes (no STARK verification) and can potentially be deployed as a soft fork [54, 55]. The two approaches are not mutually exclusive; commit-reveal could serve as a near-term stopgap while STARK verification capability is developed.

7.3 PQ Output Types and Witness Aggregation

BIP-360 [13, 64, 65] proposes a SegWit v3 output type (P2MR) for post-quantum signatures in Bitcoin. BIP-360 and the turnstile are complementary: BIP-360 provides the *destination* (the PQ-secured output type), while the turnstile provides the *migration path* from frozen legacy records to PQ-secured outputs. See Section C.7 for detailed analysis.

Heilman [44] proposed Non-Interactive Witness Aggregation (NIWA), in which a block producer aggregates all PQ signatures in a block into a single STARK proof, compressing PQ transactions to approximately 76 bytes each. Witness aggregation and the turnstile operate at different points in the migration lifecycle: the turnstile handles one-time migration, while witness aggregation handles ongoing compression of PQ signatures in routine post-migration transactions. They share a common dependency on consensus-level STARK verification (Assumption 3), and their simultaneous adoption would amortize the consensus change cost. A further synergy exists in proof aggregation during mass migration: a block producer could aggregate multiple turnstile proofs into a single per-block STARK using the same infrastructure. BTQ Technologies' PQScale system [60] demonstrates feasibility, achieving approximately 9% of unaggregated proof size for batches of 1,722 Falcon signatures.

7.4 Script-Level Approaches

Rubin [66] and Heilman [67] demonstrated that OP_CAT and OP_SIZE can enable quantum-safe Lamport signatures in Bitcoin transactions without new signature scheme consensus changes. These constructions occupy a different point in the design space: they provide PQ signing capability using existing or minimally modified script, at the cost of extreme space overhead (kilobytes to tens of kilobytes per signature). They do not address the migration problem for frozen legacy records. See Section C.8 for details.

7.5 Migration Frameworks

The QBIP proposal [24, 68] defines a three-phase migration framework for Bitcoin: Phase A restricts sending to quantum-vulnerable addresses; Phase B freezes remaining legacy UTXOs; Phase C, described as "optional" and "pending further research," proposes recovery of frozen UTXOs through a zero-knowledge proof of seed possession.

The turnstile can be understood as a concrete instantiation of Phase C, differing in several substantive respects: it uses the ECDSA private key sk as the witness (strictly more general than QBIP's BIP-39 seed phrase requirement), it provides deterministic PQ key binding via HKDF inside the STARK, and it includes a formal security analysis, none of which are present in the QBIP Phase C placeholder, whose specification text states "This section is a placeholder for the technical pseudocode." We view the turnstile as additive to QBIP: Phases A and B provide the incentive structure and freeze mechanism that our construction assumes (Assumption 2), and the turnstile fills the gap left by Phase C's placeholder. See Section C.6 for the detailed comparison.

8 Open Questions and Future Work

- **Seed-level migration:** The construction as presented uses the individual ECDSA private key sk as the witness. In practice, most users derive their private keys from a single seed via hierarchical deterministic (HD) key derivation. A user with n records across n distinct derivation paths must currently produce n separate STARK proofs (or a single proof covering n instances of $\mathcal{R}$, each with a different sk_i). A more efficient approach would prove knowledge of the *seed itself* and derive all sk_i values inside the STARK circuit via the HD chain code computation (iterated HMAC-SHA512 and scalar addition). This is feasible inside a zkVM, as the derivation consists entirely of hash functions and scalar arithmetic, but the circuit size scales linearly with the number of derivation paths, and the verifier must know which paths to check against which on-chain commitments. A seed-level turnstile would require either a canonical set of supported derivation paths or a mechanism for the prover to declare paths within the proof. We consider this a promising direction for reducing per-user migration overhead but defer a full treatment to future work (see Section C for Bitcoin-specific derivation path considerations).

- **Multisig migration:** Extending the turnstile to k-of-n multisig requires proving knowledge of k private keys and binding to a PQ multisig or threshold construction.

- **Script-path records:** Records locked to complex scripts (timelocks, hashlocks) require proving satisfaction of the script conditions inside the STARK.

- **Proof aggregation:** Can multiple record migrations from the same seed be batched into a single proof?

- **Proof aggregation and recursive compression:** Current STARK proof sizes for ECDSA-scale circuits range from 2 to 10 MB [56], which is prohibitive for direct on-chain inclusion under current consensus rules. Proof aggregation (batching multiple migration proofs into a single per-block proof) and recursive proof composition (proving a proof of a proof to compress the final artifact) are the most promising paths to reducing the on-chain footprint to feasible levels. These techniques trade proving time for proof size: recursive composition may require minutes or hours of additional prover computation to achieve kilobyte-scale final proofs. Whether this trade-off is acceptable for a one-time migration event (where proving latency is less critical than proof size) is an open engineering question. Heilman's Bitzip proposal [44] and the "proofcarrier" sidecar concept discussed in the Delving Bitcoin thread [56] suggest architectural paths, but neither has been implemented or benchmarked for ECDSA-circuit-scale proofs. Demonstrating end-to-end proof aggregation for turnstile migrations, from individual record proofs through recursive compression to a single per-block artifact of acceptable size, is a critical prerequisite for on-chain deployment and a high-priority direction for future work.

- **Privacy:** The migration request links $H(P)$ to $\mathsf{pk}_{\mathsf{pq}}$ on chain, creating a public mapping between old and new addresses. Can this be avoided?

- **PQ scheme agility:** If the initially chosen PQ scheme is later broken, can the turnstile be re-invoked to migrate to a replacement scheme?

- **Nonce-hardened migration and fresh key generation:** The base construction derives the PQ keypair deterministically from sk via domain-separated hashing (Assumption 4). This is a deliberate design choice that preserves the property that the user's existing backup is the sole secret required for migration. However, it means that classical compromise of sk (e.g., stolen seed phrase, malware during proof generation) would allow an attacker to derive the same PQ keypair and race the legitimate owner.

Two variants could address this at different points in the design space:

Nonce-hardened derivation: Include an additional random nonce r in the seed derivation: $\mathsf{seed_{pq}} = \mathrm{HKDF}(\mathsf{sk} \| r)$, with r as an additional private witness in the STARK. This ensures that compromise of sk alone is insufficient to derive $\mathsf{seed_{pq}}$. The trade-off is that r becomes new secret material that must be independently generated, stored, and backed up.

Fresh key generation: Decouple the PQ keypair entirely from sk. The user generates a fresh PQ keypair from independent randomness, and the STARK proves only ownership of the legacy record (conditions 1–3 of $\mathcal{R}$) while the PQ public key is supplied as an unconstrained public input. This provides complete forward security: compromise of sk after migration reveals nothing about the PQ key. However, it fully breaks the single-backup property, requires the user to securely generate and store entirely new key material, and removes the deterministic key binding (Section 4.4) that prevents key substitution attacks. A malicious prover who obtains sk could register any PQ key of their choosing.

The choice among these variants reflects a fundamental trade-off between backup simplicity and forward security. The base protocol optimizes for the broadest user base (single backup, deterministic derivation, no new secrets). The nonce variant is appropriate for institutional custodians with HSM infrastructure that can manage additional secrets. The fresh key variant provides the strongest post-migration security but introduces operational complexity and a new attack surface (key substitution). We consider the formal analysis of these variants, including their interaction with seed-level migration and custodial key management workflows, an important direction for future work.

Additional Bitcoin-specific open questions (activation mechanism, incentive compatibility) are discussed in Section C.11.

9 Conclusion

This paper has provided the first formal construction and security analysis of a migration protocol for hash-committed public key systems, building on an observation independently identified by Sattath and Wyborski [18], Buterin [19], Ray [20], and others: that the hash commitment $H(P)$ is quantum-safe by virtue of hash preimage resistance, and that the private key sk can serve simultaneously as the ECDSA secret and as a pseudorandom seed for PQ key generation. Our contribution is the formalization of this observation into a complete migration primitive, the turnstile, with a formal NP relation definition, correctness and soundness theorems, deterministic PQ key binding via domain-separated HKDF (absent from all prior proposals), and concrete feasibility analysis. A STARK proof binds legacy ownership to a deterministically derived PQ keypair, enabling the system to verify ownership and register a new PQ signing key in a single non-interactive step, without revealing the private key or public key at any point during the transition.

We believe this construction is sound. The security argument reduces to well-studied assumptions: the preimage resistance of the hash function against quantum adversaries (supported by NIST's position that hash functions do not require post-quantum migration [5]), and the computational soundness of the STARK proof system (supported by Chiesa et al. [38] for PCP-based arguments, with the IOP-based extension to FRI remaining formally open as discussed in Section A). The gap in the formal proof literature for FRI specifically is a known limitation that we have documented honestly; it reflects the state of the art rather than evidence of a vulnerability, and we have confidence in the construction's security given that no quantum attack on any hash-based proof system is known.

Several practical realities temper this confidence with appropriate caution.

The mechanism covers a specific, bounded class of records: those whose public keys have never been revealed. Records with exposed public keys fall outside the scope of this proposal. No

zero-knowledge proof mechanism can protect such records post-quantum, because an adversary who derives sk from an exposed P can produce the same proof as the legitimate owner. The turnstile does not claim to solve the full migration problem; it addresses the subset protected by hash commitments.

Bitcoin as the primary application. Bitcoin is the largest deployed system employing hash-committed public keys, and the primary motivation for this work. The Bitcoin instantiation (Section C) covers an estimated two-thirds of circulating supply; the remainder resides in outputs with exposed public keys, which fall outside the turnstile's scope and require proactive migration before a CRQC exists [4, 24].

On-chain costs are non-trivial. STARK proofs in the range of 2 to 10 MB per migration request (based on current benchmarks for ECDSA-scale circuits [56]) represent a substantial overhead, and a mass migration event would place significant pressure on system resources. Proof aggregation techniques [44, 60] offer a path toward amortizing this cost, but they introduce additional complexity and are not yet demonstrated for ECDSA-circuit-scale proofs.

The system-level changes required are substantial. The mechanism presupposes a post-quantum signature scheme (Assumption 1), disabled classical signing (Assumption 2), and STARK verification capability (Assumption 3). Each of these is a major change. We note that the technical mechanism is ready to be deployed if and when the necessary infrastructure is in place.

The choice of proving backend requires care. As documented in Remark 9, not all zkVM platforms maintain post-quantum security end-to-end; some introduce discrete-log or pairing-based assumptions that defeat the purpose of a PQ migration. The choice of hash function internal to the proving system (e.g., Poseidon versus SHA-256) involves a trade-off between proving efficiency and cryptographic maturity. These are implementation-level decisions that do not affect the construction's formal security argument but matter substantially for deployment.

Looking forward, we identify three directions for future work. First, extending the turnstile to cover multisig records, complex scripts, and cross-key aggregation would broaden the eligible record set. Second, formal verification of the STARK circuit implementing $\mathcal{R}$ (Definition 1), including the elliptic curve scalar multiplication, hash computation, and PQ key derivation, would provide stronger assurance than the proof-level arguments presented here. Third, we intend to pursue a direct implementation of the turnstile mechanism on a forthcoming internal research chain, providing empirical validation of the proof generation costs, on-chain overhead, and integration challenges discussed in Section 6.

Post-quantum migration for systems with hash-committed public keys is not a single event but a years-long process involving multiple complementary mechanisms. The turnstile is one piece of that process, addressing the specific problem of transitioning signing authority for hash-protected records after classical signing paths are disabled. It does not replace proactive migration, commit-reveal schemes, or PQ signature adoption; it complements them by providing a fallback path for credentials that would otherwise be permanently frozen. The convergence of post-quantum signature standards, practical zero-knowledge proof systems, and community-driven migration frameworks has opened a design space that did not exist even two years ago. This paper is an attempt to fill one well-defined corner of that space with a construction that is formally grounded, practically feasible, and honest about its limitations.

A On the Post-Quantum Security of FRI-Based STARKs

The construction proposed in this paper relies on STARKs as the proof system for the migration turnstile. A natural question is whether STARKs themselves are secure against quantum adversaries. The practical consensus in the cryptographic community is that they are, on the grounds that STARKs rely only on collision-resistant hash functions and error-correcting codes,

neither of which admits a known efficient quantum attack. We briefly survey the formal state of this question.

The strongest result to date is due to Chiesa, Ma, Spooner, and Zhandry [38], who proved that Kilian's four-message succinct argument system is post-quantum secure in the standard model when instantiated with any probabilistically checkable proof (PCP) and any *collapsing* hash function. Collapsing hash functions [37] are the quantum analogue of collision-resistant hash functions; SHA-256 is widely believed to be collapsing. This result establishes post-quantum security for PCP-based succinct arguments.

However, modern STARK constructions (including those used in production zkVM systems such as RISC Zero, SP1, and Stwo) are not PCP-based. They are built on *interactive oracle proofs* (IOPs), with the FRI (Fast Reed-Solomon Interactive Oracle Proof of Proximity) protocol serving as the core polynomial commitment scheme. The IOP model differs from the PCP model in the interaction pattern and oracle access structure, and the Chiesa et al. result does not directly cover it.

Chiesa, Dall'Agnol, Di, Guan, and Spooner [52] addressed this gap by proving post-quantum security for IOP-based succinct arguments via a new quantum rewinding strategy that works across multiple rounds. The original preprint (arXiv:2411.05360, November 2024) was subsequently revised and published at TCC 2025 (Lecture Notes in Computer Science, vol. 16270). The result establishes post-quantum security for interactive BCS-transformed arguments in the standard model when the vector commitment scheme is collapsing; the extension to the non-interactive (Fiat-Shamir compiled) setting used in production STARK deployments remains formally open.

This gap was noted by one of the present authors (Ray) in an exchange on the Bitcoin development mailing list in April 2025 [69], to which Ben-Sasson responded by citing both the Chiesa et al. and Di et al. results.

We emphasize that this is a gap in the formal proof literature, not evidence of an actual vulnerability. No quantum attack on FRI or any hash-based IOP is known, and the underlying cryptographic assumptions (collision resistance and collapsing properties of SHA-256) are identical to those in the PCP setting where post-quantum security has been formally established. This type of gap between conjectured security and formal proof is not unique to STARKs; it is the norm across post-quantum cryptography. The NIST-standardized PQ signature schemes (ML-DSA, SLH-DSA, FN-DSA) are "believed" secure against quantum adversaries based on the conjectured hardness of their underlying problems (Module-LWE, hash function security, NTRU lattices), but none has a proof of security against quantum adversaries in the standard model without idealized assumptions. Even ECDSA's classical security relies on the assumed hardness of the discrete logarithm problem, which has never been proven hard in a complexity-theoretic sense. The practical security of all deployed cryptographic systems rests on conjectured hardness plus extensive cryptanalytic scrutiny over time. STARKs are younger than ECDSA but older than the NIST PQ standards, and their hash-based assumption surface is among the most conservative in all of post-quantum cryptography. The security of the migration mechanism proposed in this paper rests on these hash function assumptions rather than on any specific property of the IOP model. Nonetheless, a formal proof of post-quantum security for IOP-based succinct arguments in the standard model remains an open problem and would strengthen the theoretical foundation of any STARK-based Bitcoin consensus mechanism.

B On Applicability to Other Chains

The turnstile mechanism is presented in the context of Bitcoin, but its underlying logic is not Bitcoin-specific. It relies on three structural properties of the target chain: (1) the existence of on-chain commitments that hide the public key behind a hash, (2) a private key that is a scalar or byte string suitable as a pseudorandom seed for PQ key generation, and (3) the ability to

verify a zero-knowledge proof at the consensus or application layer. Any chain satisfying these properties is a candidate for the same construction. Rather than enumerate compatible chains, we find it more useful to characterize where the mechanism does *not* apply, as the failure modes are more instructive than the success cases.

Account-model chains with address-as-public-key. The most significant class of incompatible chains comprises account-model systems where the user's address is derived directly from (or is identical to) the public key, and that public key is visible on chain from the moment the account is created or first receives funds. Solana is the canonical example: an account address is the base58-encoded ed25519 public key itself [4]. There is no hash commitment; the public key is the address. Every funded Solana account is therefore in the equivalent of Bitcoin's "exposed P2PK" category. A quantum adversary who can run Shor's algorithm on ed25519 (which requires approximately 1,500 to 2,000 logical qubits [1]) can derive the private key for any account directly from its address. No zero-knowledge migration mechanism can help, because the adversary has the same information as the legitimate owner. The same limitation applies to any chain where the public key is structurally exposed at the account or address level.

Ethereum: a partial case. Ethereum occupies a middle ground. Ethereum addresses are derived via $\mathsf{Keccak256}(\mathsf{pubkey})[12:]$, a truncated hash of the secp256k1 public key. This is structurally similar to Bitcoin's P2PKH: the public key is hidden behind a hash until the account signs a transaction. However, Ethereum's account model means that once an account has sent any transaction (not just received), the public key is exposed in the transaction's signature recovery data (the v, r, s values allow recovery of the full public key). Given Ethereum's high transaction frequency, the vast majority of funded accounts have sent at least one transaction, making them quantum-vulnerable. The subset of Ethereum accounts that have only received (never sent) is substantially smaller than Bitcoin's equivalent set [4]. The turnstile mechanism is theoretically applicable to this subset, but the eligible set may be too small to justify the consensus engineering cost. The Ethereum Foundation is pursuing a different approach based on account abstraction (ERC-4337) and STARK-based proofs that does not require a UTXO-style turnstile [70].

Chains with non-scalar private keys. Some signature schemes use private keys that are not simple scalars or byte strings. For example, certain threshold signature constructions distribute key shares across multiple parties in a way that no single party holds a complete private key. The turnstile requires a single witness sk that can be fed to both ECDSA (or ed25519) key derivation and PQ key generation. If the private key material cannot be reconstituted into a single seed, the mechanism does not apply in its current form. This is analogous to the multisig limitation discussed in Section 2.3 and Section 8.

Chains without hash-committed outputs. Any chain that stores raw public keys in its state representation (whether UTXO-based or account-based) lacks the hash commitment that serves as the quantum-safe anchor for the turnstile. Without $H(P)$ on chain, there is no quantum-resistant commitment to bind the STARK proof to. These chains must rely on proactive migration (moving funds to PQ-secured addresses before a CRQC exists) rather than post-freeze recovery mechanisms.

Where it does apply. Beyond Bitcoin, the mechanism applies most naturally to UTXO-based chains that inherited Bitcoin's address structure and use ECDSA or Schnorr signatures with hash-committed outputs. This includes Bitcoin Cash, Litecoin, and other Bitcoin-derived chains, as well as any chain that adopted a similar hash-then-encode address derivation pipeline. Chains using ed25519 with hashed addresses (rather than raw public key addresses) are also

candidates, provided the ed25519 private key (a 32-byte scalar) can serve as a PQ keygen seed, which is the case for all NIST-approved PQ schemes [27, 28]. The construction generalizes cleanly: replace secp256k1 scalar multiplication with ed25519 key derivation inside the STARK circuit, adjust the hash function to match the target chain's address derivation, and the rest of the mechanism is identical.

The general lesson is that the turnstile's applicability is determined by a single architectural question: does the chain store a hash of the public key, or the public key itself? Chains in the former category have an accidental quantum-safe anchor that the turnstile can exploit. Chains in the latter category do not, and must solve the migration problem through other means.

C Application to Bitcoin

This appendix collects the Bitcoin-specific details of the turnstile mechanism. The generic construction is presented in the main body; here we instantiate it for the Bitcoin UTXO model, specifying output types, eligibility criteria, transaction formats, consensus rules, and relationships to Bitcoin-specific proposals.

C.1 Bitcoin Output Types and Hash Commitments

Bitcoin transactions consume *unspent transaction outputs* (UTXOs) and create new ones. Each UTXO is locked by a *ScriptPubKey* that specifies the conditions under which it can be spent. The spending transaction provides a *scriptSig* or *witness* satisfying those conditions. For the output types relevant to this paper, the ScriptPubKey contains a hash commitment to the public key rather than the public key itself [4].

P2PKH (Pay-to-Public-Key-Hash). The ScriptPubKey is:

$$\mathsf{OP\_DUP}\ \ \mathsf{OP\_HASH160}\ \ \langle H_{\text{chain}} \rangle\ \ \mathsf{OP\_EQUALVERIFY}\ \ \mathsf{OP\_CHECKSIG}$$

where $H_{\text{chain}} = \mathsf{HASH160}(\mathsf{encode}(P))$ is a 20-byte value. To spend, the user provides (P, σ) in the scriptSig, where σ is an ECDSA signature. The script engine verifies that $\mathsf{HASH160}(\mathsf{encode}(P)) = H_{\text{chain}}$ and that σ is a valid signature under P. Crucially, P is revealed only at spend time. An unspent P2PKH output exposes only H_{chain}.

P2WPKH (Pay-to-Witness-Public-Key-Hash). Native SegWit v0 outputs use a 20-byte witness program:

$$\mathsf{OP\_0}\ \ \langle H_{\text{chain}} \rangle$$

The spending witness contains (P, σ). The verification logic is equivalent to P2PKH but the data is placed in the segregated witness, which receives a 75% weight discount. As with P2PKH, P is revealed only upon spending.

P2SH-P2WPKH (Nested SegWit). A backwards-compatible wrapper around P2WPKH. The outer ScriptPubKey is:

$$\mathsf{OP\_HASH160}\ \ \langle \mathsf{HASH160}(\text{redeemScript}) \rangle\ \ \mathsf{OP\_EQUAL}$$

where the redeem script is $\mathsf{OP\_0}\ \ \langle H_{\text{chain}} \rangle$. Both P and the redeem script are revealed only upon spending.

The hash as a quantum-safe commitment. The function $\mathsf{HASH160} = \mathsf{RIPEMD160} \circ \mathsf{SHA256}$ produces a 160-bit output from the 33-byte compressed public key. Recovering P from H_{chain} requires inverting this hash, which is a preimage problem. Grover's algorithm provides at most a quadratic speedup for unstructured preimage search [33], but the relevant search space is the 256-bit scalar field $\mathbb{F}_q$ (the domain of sk), not the 160-bit hash output (see Section 5.2 for the detailed argument). NIST does not consider hash functions to require migration as part of post-quantum standardization [5]. The hash commitment H_{chain} is therefore quantum-safe for all output types where P has not been revealed through other means.

Output types where P is exposed. Two output types expose the public key on chain at the time of receipt rather than at spend time. P2PK (Pay-to-Public-Key) outputs embed P directly in the ScriptPubKey ($\langle P \rangle$ $\mathsf{OP\_CHECKSIG}$). P2TR (Taproot, SegWit v1) outputs embed a 32-byte x-only public key in the witness program [71, 72]. These output types are excluded from the turnstile mechanism (Section C.3) because a quantum adversary can derive sk directly from the visible P.

C.2 Eligible Record Classes

The eligible UTXO classes, conditioned on Assumption 5, are:

- **P2PKH outputs (never spent from):** The ScriptPubKey contains $\mathsf{OP\_DUP}$ $\mathsf{OP\_HASH160}$ $\langle \mathsf{HASH160}(P) \rangle$ $\mathsf{OP\_EQUALVERIFY}$ $\mathsf{OP\_CHECKSIG}$. The public key P is revealed only when the output is spent, as the spending transaction must include P in the scriptSig for $\mathsf{OP\_CHECKSIG}$ to verify. If the address has never been spent from, P remains hidden behind the $\mathsf{HASH160} = \mathsf{RIPEMD160}(\mathsf{SHA256}(\cdot))$ commitment.
- **P2SH-wrapped SegWit (P2SH-P2WPKH), never spent from:** The outer ScriptPubKey commits to $\mathsf{HASH160}$(redeemScript), where the redeem script itself contains the witness program $\mathsf{OP\_0}$ $\langle \mathsf{HASH160}(P) \rangle$. Both P and the redeem script are revealed only upon spending.
- **P2WPKH (native SegWit v0), never spent from:** The witness program is $\mathsf{OP\_0}$ $\langle \mathsf{HASH160}(P) \rangle$ (20 bytes). As with P2PKH, P appears in the witness data only when the output is spent.

Size of the eligible set. Quantifying the eligible set precisely requires a full UTXO set scan distinguishing between outputs whose public keys have and have not been revealed. Forensic analysis by Dr. Anthony Milton, presented at the Quantum Bitcoin Summit (July 2025) and detailed in the Chaincode Labs report [4, 73], estimates approximately **6.51 million BTC** (roughly one-third of circulating supply) with exposed public keys, vulnerable to a quantum adversary. The vulnerable set breaks down as follows: approximately 1.72 million BTC in P2PK outputs (public key embedded directly in the ScriptPubKey), approximately 4.49 million BTC exposed through address reuse (public key revealed by prior spends), and approximately 184,000 BTC in P2TR key-path outputs (x-only public key visible from receipt) [4, 71]. Institutional custodians are disproportionately represented in the address reuse category, as exchange deposit addresses are frequently reused for accounting convenience [73].

By difference, approximately **13.5 million BTC** (roughly 67% of the circulating supply of 19.99 million) reside in UTXOs with unexposed public keys and are therefore candidates for the turnstile mechanism. At current prices (approximately US$68,000 per BTC as of February 2026), the eligible set represents roughly **US$918 billion** in value that could be secured through the proposed mechanism. CoinShares [74] provides a more conservative estimate of the vulnerable set at approximately 1.6 million BTC restricted to P2PK outputs, which would place the eligible

set even higher. Under any of these estimates, the eligible set represents a substantial majority of Bitcoin's total value.

C.3 Excluded Record Classes

The following UTXO classes are *not* eligible for the proposed mechanism. For each, we state the reason and, where applicable, note alternative migration paths discussed in the literature.

- **P2PK (Pay-to-Public-Key) outputs.** The public key P is directly embedded in the ScriptPubKey ($\langle P \rangle$ OP_CHECKSIG). A quantum adversary with access to P can derive sk via Shor's algorithm [1, 50] and produce a valid STARK proof indistinguishable from the legitimate owner's. No zero-knowledge migration mechanism can protect these outputs post-quantum. Approximately 1.72 million BTC reside in P2PK UTXOs [4], including an estimated 968,000 to 1.1 million BTC attributed to Satoshi Nakamoto. Lopp [75] argues these coins should be considered permanently frozen (or burned) rather than recoverable.

- **P2TR (Taproot, SegWit v1) key-path outputs.** Taproot outputs embed a 32-byte x-only public key directly in the witness program [71]. This key is visible from the moment of receipt, making P2TR key-path outputs vulnerable to the same quantum attack as P2PK. Ruffing [72] proves that Taproot's *script-path* spending remains post-quantum secure when the key-path is disabled, which is the approach taken by BIP-360 [13]. However, existing P2TR UTXOs with active key-paths are excluded from our mechanism. Approximately 184,000 BTC are held in P2TR outputs [4], though many of these contain negligible sat amounts from ordinals and inscriptions.

- **Any address previously spent from (address reuse).** When a UTXO is spent, the spending transaction reveals P in the scriptSig (for P2PKH) or witness (for P2WPKH). If the same address subsequently receives new funds, those new UTXOs are locked to the same $H(P)$, but P is now public from the prior spend. An estimated 4.49 million BTC are exposed through address reuse [4, 76], with exchange deposit addresses (particularly Binance) accounting for a disproportionate share.

- **Addresses exposed via fork chains.** Bitcoin Cash (August 2017), Bitcoin Gold (October 2017), and other forks sharing Bitcoin's key hierarchy allow users to spend forked coins using the same private keys. If a user spent coins on a fork chain, the corresponding public key P was revealed on that fork's blockchain. An adversary monitoring fork chains can collect these public keys and use them to attack the corresponding Bitcoin UTXOs. The magnitude of this exposure has not been systematically quantified in the literature.

- **Multisig outputs (bare, P2SH, P2WSH).** Multisig constructions require k-of-n public keys, any subset of which may have been revealed through partial spending or other on-chain activity. Even if only one constituent key is exposed, a quantum adversary can derive that key's private counterpart. Extending the turnstile to multisig requires proving knowledge of k private keys simultaneously and binding to a PQ threshold or multisig construction. This is a substantially more complex problem that interacts with ongoing research on PQ threshold signatures [77] and is deferred to future work.

- **UTXOs whose owners have lost the private key or seed.** The turnstile requires the user to provide a valid STARK proof, which in turn requires knowledge of sk. If the owner has lost access to sk (or the seed from which it is derived), migration is impossible and the funds remain permanently frozen. This outcome is identical to the current behavior when a user loses their private key, and is arguably preferable to the alternative in which a quantum adversary eventually recovers the key and steals the funds [75].

C.4 Migration Transaction Format

The migration transaction is a special-purpose transaction type recognized by consensus after the legacy spend freeze (Assumption 2). It differs from a standard Bitcoin transaction in two respects: the input is satisfied by a STARK proof rather than a signature, and the output is locked to a PQ public key under the PQ spending rules (Assumption 1).

We describe the transaction structure at a conceptual level, deferring the precise serialization format to a future BIP-level specification.

Input. The transaction references one or more frozen UTXOs by their outpoint (transaction ID and output index), exactly as in a standard Bitcoin transaction. The scriptSig or witness field does *not* contain an ECDSA signature or public key. Instead, the witness contains:

- The STARK proof π.
- The PQ public key $\mathsf{pk}_{\mathsf{pq}}$ (which is a public input to π and must be provided in the clear for the verifier).

The on-chain commitment H_{chain} is not included in the witness, as it can be looked up from the referenced UTXO's ScriptPubKey by the verifying node.

Output. The transaction creates one or more new UTXOs locked to $\mathsf{pk}_{\mathsf{pq}}$ under the PQ output type. If BIP-360 is the adopted PQ standard, this would be a SegWit v3 output (`bc1r` address) [13]. The output value is the input value minus the transaction fee, as in a standard transaction.

Value preservation. The migration does not alter the quantity of bitcoin held; it changes only the spending authority. The frozen UTXO's value is transferred in full (minus fees) to the PQ-secured output.

Multiple UTXOs from the same key. If a user holds multiple frozen UTXOs locked to the same $H(P)$ (i.e., address reuse where P was never revealed because none of the UTXOs were spent from), all can be migrated in a single transaction by referencing multiple inputs. The STARK proof covers all inputs, since the same sk satisfies $\mathcal{R}$ for all of them. This is a natural optimization that reduces on-chain overhead.

Multiple UTXOs from different keys. A user who holds UTXOs derived from different private keys (e.g., multiple addresses derived from the same BIP-32 seed via different derivation paths) must produce a separate proof for each distinct sk. These could be included in a single transaction with multiple inputs and multiple proofs, or in separate transactions. Proof aggregation across distinct keys is a potential optimization discussed in Section 8.

C.5 Consensus Validation Rules

A validating node that receives a migration transaction must enforce a set of consensus rules beyond STARK proof verification. These rules ensure that the turnstile is applied only to eligible UTXOs and that on-chain invariants are maintained. We specify them here because the security of the mechanism depends not only on the cryptographic properties of the STARK but also on the network's ability to reject migration attempts for UTXOs that fall outside the eligible set (Section 2.2).

Output type check. The node must verify that the referenced UTXO is of an eligible output type: P2PKH, P2WPKH, or P2SH-P2WPKH. Migration transactions referencing P2PK, P2TR (key-path), bare multisig, or any other output type where the public key is embedded directly in the ScriptPubKey must be rejected. This check is deterministic and requires only inspection of the referenced output's script structure, which is available in the UTXO set.

Address reuse check. The node must verify that the public key P corresponding to H_{chain} has never been revealed on the Bitcoin blockchain. In practice, this means checking that no prior transaction has spent from the same address, since spending reveals P in the scriptSig or witness. This can be implemented by checking that no input in any confirmed transaction references an output locked to the same H_{chain}, or equivalently, that the address associated with H_{chain} has a transaction count of zero on the spending side. This check is enforceable at consensus because the Bitcoin blockchain is a complete record of all spends.

Limits of on-chain enforcement. The output type and address reuse checks exhaust what the network can enforce through consensus. There exist scenarios in which P has been revealed through channels that the Bitcoin blockchain cannot observe:

- **Fork chain exposure.** If the user spent coins on Bitcoin Cash, Bitcoin Gold, or another fork sharing the same key hierarchy, P was revealed on the fork chain. The Bitcoin network has no visibility into fork chain transactions and cannot detect this exposure.
- **Off-chain disclosure.** The user may have disclosed P through a signed message, a proof-of-reserves protocol, a key-sharing arrangement, or any other off-chain mechanism. The network cannot detect such disclosures.
- **Side-channel leakage.** Implementation vulnerabilities in wallet software, hardware devices, or network protocols may have leaked P without the user's knowledge.

In all of these cases, a quantum adversary who obtains P can derive sk and produce a valid migration transaction that passes all consensus checks. The turnstile mechanism cannot protect against off-chain public key exposure; this is a fundamental limitation shared by all proposals that rely on hash commitment as the quantum-safe anchor, including commit-delay-reveal schemes [53, 54, 55] and QBIP Phase C [24]. Users who suspect their public key may have been exposed through any channel should migrate proactively via standard ECDSA-to-PQ transactions during the QBIP Phase A window, before the legacy spend freeze takes effect, rather than relying on the turnstile as a fallback.

C.6 Relationship to QBIP and Phase C

The QBIP proposal by Lopp and Papathanasiou [24, 68] is the most directly relevant prior work. QBIP defines a three-phase migration framework: Phase A restricts sending to quantum-vulnerable addresses, incentivizing proactive migration via standard transactions; Phase B (approximately five years after Phase A) freezes all remaining legacy UTXOs by invalidating ECDSA and Schnorr spends; and Phase C, described as "optional" and "pending further research," proposes a hard fork to allow recovery of frozen legacy UTXOs through a zero-knowledge proof of possession of a BIP-39 seed phrase [24].

The turnstile mechanism proposed in this paper can be understood as a concrete instantiation of Phase C. However, it differs from the QBIP description in several substantive respects:

Witness generality. QBIP Phase C specifies "ZK proof of knowledge of a BIP-39 seed phrase" [24]. Our construction uses the ECDSA private key sk as the witness, not the mnemonic. This is strictly more general: any BIP-39 holder can derive sk via standard BIP-32 hierarchical deterministic derivation [4], but the mechanism also covers users who hold raw private keys, paper wallets, brainwallets, or keys generated through non-BIP-39 processes. This distinction matters because early Bitcoin wallets (pre-2013) predate the BIP-39 standard, and a mechanism restricted to BIP-39 seed phrases would exclude a meaningful fraction of legacy UTXO holders.

Deterministic key binding. QBIP Phase C describes "recovery" of frozen funds but does not specify how the recovered funds are secured post-migration. Our construction binds sk deterministically to a PQ public key via $\mathsf{PQ.Keygen}(\mathsf{sk})$ inside the STARK proof (Definition 1). This eliminates a degree of freedom that would otherwise allow an adversary who obtains sk to register an alternative PQ key, reducing the attack surface to a race condition on transaction inclusion rather than a key substitution attack (Section 4.4). The QBIP text does not address this binding property.

Formal security argument. QBIP Phase C contains no technical pseudocode or security analysis; the specification text states "This section is a placeholder for the technical pseudocode. Detailed implementation logic will be added here" [68]. Our construction provides a formal definition of the NP relation (Definition 1), correctness and soundness theorems (Theorem 1, Theorem 2), and analysis of zero-knowledge, non-malleability, and front-running resistance (Section 5).

Turnstile vs. recovery framing. QBIP describes Phase C as "recovery," suggesting a one-time claim process. Our framing as a *turnstile* emphasizes that the mechanism is a seamless, atomic transition of spending authority: the same secret controls spending before and after migration, expressed through different cryptographic schemes. This is not a recovery from a lost state but a protocol-mediated change of authentication mechanism. The distinction matters for user mental models and wallet implementation: the user's backup (seed phrase, private key) remains the sole secret throughout, and no new secret material is introduced.

Relationship. We view the turnstile as additive to the QBIP framework, not competitive with it. Phases A and B of QBIP provide the incentive structure and freeze mechanism that our construction assumes (Assumption 2). The turnstile fills the gap left by Phase C's placeholder specification. A natural deployment path would be to adopt QBIP Phases A and B as proposed, with the turnstile mechanism serving as the concrete implementation of Phase C.

The QBIP mailing list discussion [68] includes several responses relevant to our construction. Pieter Wuille noted that commit/reveal protocols "only work for addresses whose pubkeys haven't been exposed yet," which aligns precisely with our scoping in Assumption 5. Sjors Provoost expressed skepticism that Phase C is viable, arguing that "almost all dormant bitcoins that would be locked by Phase B are already permanently lost." Our mechanism does not resolve this empirical question, but it ensures that any owner who *does* retain their key material can migrate, regardless of how long the funds have been dormant.

Prior articulation. The "turnstile" framing for Bitcoin-specific quantum migration was proposed by one of the present authors (Ray) in a February 2025 response to Cruz's QRAMP proposal on the Bitcoin-dev mailing list [20], drawing an analogy to the Zcash shielded-pool turnstile. The underlying cryptographic observation, that hash-based key derivation provides a post-quantum anchor exploitable via zero-knowledge proofs, was previously formalized by Sattath and Wyborski [18] in 2023 ("signature lifting") and independently proposed by Buterin [19] for Ethereum in 2024. The present paper provides the formal construction, security analysis,

and practical feasibility assessment that were absent from all of these prior proposals, as well as the deterministic PQ key binding via HKDF that distinguishes the turnstile from earlier work.

C.7 Relationship to BIP-360

BIP-360 [13, 64, 65], authored by Hunter Beast, Ethan Heilman, and Isabel Foxen Duke, proposes a new SegWit v3 output type (`bc1r` addresses) called Pay-to-Merkle-Root (P2MR). P2MR removes Taproot's quantum-vulnerable key-path spend while preserving the script-path spending capability, which Ruffing [72] has formally proven to be post-quantum secure. BIP-360 supports multiple PQ signature algorithms (ML-DSA-44, SLH-DSA-Shake-128s, FN-DSA) and proposes a quantum witness discount to offset the larger PQ signature sizes. A reference implementation is available as `libbitcoinpqc` [78].

BIP-360 and the turnstile mechanism are complementary. BIP-360 provides the *destination*: the PQ-secured output type to which funds are migrated. The turnstile provides the *migration path*: the mechanism by which frozen legacy UTXOs transition to BIP-360 outputs without revealing the ECDSA public key. In the QBIP framework [24], BIP-360 is explicitly listed as a prerequisite for Phase A. Our construction assumes that a PQ output type exists (Assumption 1) and is agnostic to whether it is realized via BIP-360 or a successor proposal.

BIP-360's quantum witness discount is also relevant to the turnstile's practical feasibility. If migration proofs are classified as quantum witness data and receive a similar or greater discount, the effective on-chain cost of migration transactions decreases substantially (Section 6.2). We note that BIP-360's current algorithm selection (ML-DSA, SLH-DSA, FN-DSA) reflects a deliberate conservatism: SQIsign was originally included but deprecated on the basis of verification times that have since improved by over 30x [64, 43]. The turnstile mechanism is agnostic to this algorithm selection debate; it requires only that the chosen PQ scheme accepts a pseudorandom seed for key generation (Assumption 4), a property satisfied by all current candidates including SQIsign.

C.8 OP_CAT and Lamport Signatures

Rubin [66] demonstrated in 2021 that OP_CAT (proposed for reactivation in BIP-347 [79]) can enable quantum-safe Lamport signatures in Bitcoin transactions without introducing new signature scheme consensus changes. Heilman [67] extended this in 2024, showing that Lamport signatures can be constructed using OP_SIZE to sign the size of an ECDSA signature, requiring no consensus changes at all. This approach was covered in Bitcoin Optech Newsletter #301 [80].

These constructions occupy a different point in the design space. They provide PQ signing capability using only existing or minimally modified Bitcoin script, avoiding the need for new signature schemes at consensus. The trade-off is extreme space overhead: Lamport signatures are on the order of kilobytes to tens of kilobytes per signature, with no efficient aggregation path. They are best understood as emergency fallback mechanisms that could be deployed rapidly if a CRQC emerged before more comprehensive solutions (BIP-360, Bitzip, the turnstile) were ready.

The OP_CAT approach does not address the migration problem for frozen UTXOs. It provides a new way to *spend* going forward but does not provide a mechanism for transitioning spending authority from a legacy ECDSA key to a PQ key without revealing the ECDSA public key. The turnstile and OP_CAT/Lamport approaches are therefore orthogonal: one solves migration, the other solves ongoing PQ signing.

C.9 Wallet Implementation

The turnstile mechanism introduces new requirements for wallet software. We outline the necessary capabilities and discuss compatibility with existing wallet architectures.

Required capabilities. A wallet supporting turnstile migration must be able to:

1. **Derive sk from the user's secret material.** For BIP-39 wallets, this involves the standard mnemonic-to-seed derivation (PBKDF2 with 2048 rounds of HMAC-SHA512) followed by BIP-32 hierarchical deterministic derivation along the appropriate path (e.g., `m/44'/0'/0'/0/0` for the first receiving address) [4]. For raw private key holders, sk is used directly.

2. **Compute the PQ keypair.** $(\mathsf{pk}_{\mathsf{pq}}, \mathsf{sk}_{\mathsf{pq}}) \leftarrow \mathsf{PQ.Keygen}(\mathsf{sk})$, using the PQ library specified by the consensus rules. If BIP-360 is adopted, the reference implementation `libbitcoinpqc` [78] provides the necessary PQ key generation functions.

3. **Generate the STARK proof.** This is the most computationally intensive step and may be performed locally (on the user's device) or delegated to a proving service. Local proving requires a zkVM runtime (e.g., RISC Zero [10], SP1 [32]) and takes 5 to 30 seconds with GPU acceleration (Section 6.1). Delegated proving introduces a trust assumption: the proving service learns sk (since it is the witness), so delegation is only appropriate to a trusted party or via a secure enclave. An alternative is client-side proving in a browser or mobile environment [48], which avoids delegation entirely.

4. **Construct and broadcast the migration transaction.** The wallet assembles the transaction (Section 4.3), includes π and $\mathsf{pk}_{\mathsf{pq}}$ in the witness, and broadcasts to the Bitcoin network via the standard peer-to-peer protocol.

Proving environment. The most natural environment for STARK proof generation is a desktop wallet application, following the pattern established by Bitcoin Core and other full-node wallet software that has been in continuous use since Bitcoin's earliest days. Desktop wallets routinely handle private key material for transaction signing, and STARK proof generation is operationally similar: the wallet loads sk, executes the zkVM prover, and produces π. Users seeking additional isolation can run the proving software on a security-focused operating system such as Qubes OS or Tails, or on a dedicated air-gapped machine, transferring only the resulting proof (which contains no secret material) to a network-connected device for broadcast. These practices mirror existing operational security patterns for high-value Bitcoin transactions and do not require any novel security infrastructure.

Hardware wallet considerations. Users who store their keys exclusively on hardware wallets (e.g., Ledger, Trezor) face a narrower constraint: current hardware wallet processors lack the computational capacity for STARK proof generation. These users must either (a) derive sk from the BIP-39 recovery phrase they recorded during hardware wallet setup and generate the proof on a desktop machine, which is standard practice for any operation requiring direct key access [81], or (b) wait for future hardware wallet firmware that supports proof generation or secure export to an external prover. Option (a) is practical today and represents the same security posture as any user who manages keys via desktop wallet software.

For institutional clients, custodians operating under qualified custodian frameworks can perform the proving step within their existing secure infrastructure (HSMs, secure enclaves) without exposing client key material to general-purpose hardware. This is operationally identical to how institutional custodians already perform transaction signing on behalf of clients.

Timeline for wallet support. The wallet implementation is not conceptually complex: it combines existing BIP-32/39 derivation logic with a PQ key generation library and a zkVM prover. The primary engineering effort is integrating the zkVM runtime into wallet software, which is a one-time development cost. Given the multi-year migration timeline proposed by

QBIP [24] and the Chaincode roadmap [4], wallet developers would have substantial lead time to implement and test this functionality before the freeze activates.

C.10 Network Load During Mass Migration

If the legacy spend freeze (Assumption 2) is enacted with a fixed deadline, a large number of UTXO owners may attempt to migrate in a concentrated time window, creating a surge in block space demand. We consider the scale of this demand and potential mitigations.

Scale. Pont et al. [82] estimate a lower bound of 76.16 days of cumulative block space for full UTXO migration (across all 186.7 million UTXOs as of late 2024), assuming each migration consumes a standard transaction's worth of block space. With STARK proofs at 100 KB per migration, the block space requirement increases substantially. However, not all UTXOs require the turnstile mechanism: UTXOs with exposed keys are ineligible (Section 2.3), and many users will migrate proactively before the freeze via standard ECDSA-to-PQ transactions during the QBIP Phase A window [24]. The turnstile is a fallback for UTXOs that remain frozen after the deadline.

Graduated migration. The QBIP framework [24] proposes a multi-year migration window. Phase A (approximately 3 years after activation) restricts sending to quantum-vulnerable addresses, incentivizing proactive migration via standard transactions. Phase B (approximately 5 years after Phase A) freezes remaining legacy UTXOs. Phase C (the turnstile) then operates on the residual set. This phased approach spreads migration load over years rather than concentrating it at a single deadline. The Chaincode Labs report [4] recommends a dual-track strategy with a 2-year contingency plan and a 7-year comprehensive migration path.

Block space allocation. A dedicated portion of block weight could be reserved for migration transactions, ensuring that ordinary payment traffic is not displaced. This is analogous to proposals for segregating different transaction types within the block weight limit [44]. Alternatively, the quantum witness discount proposed in BIP-360 [13] could be calibrated to ensure migration transactions are economically competitive with standard transactions without crowding them out.

Rate limiting and UTXO consolidation. An additional mitigation strategy is to deploy the turnstile mechanism *before* Q-day, during a proactive migration period (analogous to QBIP Phase A [24]). In this pre-emergency setting, consensus rules could impose rate limits on turnstile migrations, such as restricting the number of migration transactions per block or setting minimum UTXO value thresholds for turnstile eligibility. This would encourage users to consolidate small UTXOs via standard ECDSA transactions (which are still safe pre-Q-day) before migrating the consolidated UTXO via the turnstile, reducing the total number of migration proofs required. Users holding many small UTXOs across multiple addresses derived from the same seed would benefit from consolidating into a single address before the freeze, then migrating once. This approach trades pre-freeze coordination for post-freeze efficiency.

Proof aggregation. The most promising approach to reducing per-UTXO overhead is batch aggregation. A miner (or a third-party aggregation service) could collect multiple migration proofs from the mempool and aggregate them into a single STARK proof covering all migrations in the block. This is the approach proposed by Heilman [44] for ongoing PQ signature compression, and it applies equally to one-time migration proofs. BTQ Technologies' PQScale system [60] demonstrates that 1,722 Falcon signatures can be aggregated to approximately 9% of their unaggregated size using ZK proof aggregation. Applied to migration proofs, aggregation

could reduce the per-UTXO on-chain cost from approximately 100 KB to a few hundred bytes, bringing it in line with standard PQ transaction sizes.

C.11 Bitcoin-Specific Open Questions

The following open questions are specific to the Bitcoin instantiation of the turnstile mechanism:

- **Activation mechanism:** How is the freeze and migration capability activated? Soft fork vs. hard fork trade-offs.
- **Incentive compatibility:** Fee economics for migration transactions. Should migration proofs receive a witness discount?

References

[1] M. Roetteler, M. Naehrig, K.M. Svore, K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," *ASIACRYPT 2017*, 2017. https://eprint.iacr.org/2017/598.pdf

[2] T. Häner, S. Jaques, M. Naehrig, M. Roetteler, M. Soeken, "Improved Quantum Circuits for Elliptic Curve Discrete Logarithms," *PQCrypto 2020*, 2020. https://eprint.iacr.org/2020/077

[3] M. Mosca, M. Piani, "Quantum Threat Timeline Report 2024," Global Risk Institute, December 2024. https://globalriskinstitute.org/publication/2024-quantum-threat-timeline-report/

[4] A. Milton, C. Shikhelman, "Bitcoin and Quantum Computing: Current Status and Future Directions," Chaincode Labs Technical Report, May 2025. https://chaincode.com/bitcoin-post-quantum.pdf

[5] NIST, "Transition to Post-Quantum Cryptography Standards," NIST IR 8547 (Initial Public Draft), November 2024. https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf

[6] J. Mascelli, M. Rodden, "Harvest Now Decrypt Later: Examining Post-Quantum Cryptography and the Data Privacy Risks for Distributed Ledger Networks," Federal Reserve Board, FEDS 2025-093, September 2025. https://www.federalreserve.gov/econres/feds/harvest-now-decrypt-later-examining-post-quantum-cryptography-and-the-data-privacy-risks.htm

[7] NIST, "NIST Releases First 3 Finalized Post-Quantum Encryption Standards," News release, August 2024. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

[8] M.A. Aardal, et al., "SQIsign Algorithm Specifications and Supporting Documentation, Version 2.0," NIST PQC Additional Signatures Round 2, February 2025. https://csrc.nist.gov/Projects/pqc-dig-sig

[9] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046

[10] RISC Zero, "Precompiles," Developer documentation, 2025. https://dev.risczero.com/api/zkvm/precompiles

[11] StarkWare, "StarkWare sets new proving record with Stwo," Blog post, July 2024. https://starkware.co/blog/starkware-new-proving-record/

[12] Fenbushi Capital, "Benchmarking zkVMs: Current State and Prospects," August 2025. https://fenbushi.vc/2025/08/29/benchmarking-zkvms-current-state-and-prospects/

[13] Hunter Beast, E. Heilman, I. Foxen Duke, "BIP-360: Pay-to-Merkle-Root (P2MR)," Bitcoin Improvement Proposal, December 2024. https://bip360.org/bip360.html

[14] Blockstream, "Blockstream Research Demonstrates Quantum-Resistant Transaction Signing on Liquid Using Simplicity Smart Contracts," Blockstream Blog, March 3, 2026. https://blog.blockstream.com/blockstream-research-demonstrates-quantum-resistant-transaction-signing-on-liquid-using-

[15] M. Kudinov, J. Nick, "Hash-based Signature Schemes for Bitcoin," IACR ePrint 2025/2203, December 2025. https://eprint.iacr.org/2025/2203

[16] Ethereum Foundation, "Post-Quantum Security for Ethereum," March 2026. https://pq.ethereum.org/

[17] J. Drake, "Strawmap," EF Protocol, February 25, 2026. https://strawmap.org/

[18] O. Sattath, S. Wyborski, "Protecting Quantum Procrastinators with Signature Lifting: A Case Study in Cryptocurrencies," IACR ePrint 2023/362, 2023 (v2 July 2024). https://eprint.iacr.org/2023/362

[19] V. Buterin, "How to hard-fork to save most users' funds in a quantum emergency," Ethereum Research forum, March 2024. https://ethresear.ch/t/how-to-hard-fork-to-save-most-users-funds-in-a-quantum-emergency/18901

[20] D. Ray, Response to QRAMP proposal, Bitcoin-dev mailing list, February 12, 2025. https://groups.google.com/g/bitcoindev/c/8PM6iZCeDMc

[21] M.S. Kiraz, S. Kardas, "Migrating Bitcoin and Ethereum Addresses to the Quantum Blockchain Era," IACR ePrint 2026/352, February 2026. https://eprint.iacr.org/2026/352

[22] F. Baldimtsi, K. Chalkias, A. Roy, "Post-Quantum Readiness in EdDSA Chains," IACR ePrint 2025/1368, 2025. https://eprint.iacr.org/2025/1368

[23] T.G. Tan, J. Zhou, "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 (DPM/CBT 2022)*, Lecture Notes in Computer Science, vol. 13619, pp. 308–316, Springer, 2023. https://link.springer.com/chapter/10.1007/978-3-031-25734-6_19

[24] J. Lopp, C. Papathanasiou, et al., "Post Quantum Migration and Legacy Signature Sunset," July 2025. https://qbip.org/

[25] H. Krawczyk, P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, Internet Engineering Task Force, May 2010. https://www.rfc-editor.org/rfc/rfc5869

[26] A. Cruz, "Proposal for Quantum-Resistant Address Migration Protocol (QRAMP) BIP," Bitcoin-dev mailing list, February 2025. https://groups.google.com/g/bitcoindev/c/8PM6iZCeDMc

[27] NIST, "FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA)," August 2024. https://csrc.nist.gov/pubs/fips/204/final

[28] NIST, "FIPS 205: Stateless Hash-Based Digital Signature Standard (SLH-DSA)," August 2024. https://csrc.nist.gov/pubs/fips/205/final

[29] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," *Advances in Cryptology – CRYPTO 2010*, Lecture Notes in Computer Science, vol. 6223, pp. 631–648, Springer, 2010. https://doi.org/10.1007/978-3-642-14623-7_34

[30] E. Barker, "Recommendation for Key Management: Part 1 – General," NIST Special Publication 800-57 Part 1, Revision 5, May 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5

[31] T. Pornin, "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)," RFC 6979, Internet Engineering Task Force, August 2013. https://www.rfc-editor.org/rfc/rfc6979

[32] Succinct Labs, "SP1 Turbo: The World's Fastest zkVM Just Got Faster," Blog post, 2025. https://blog.succinct.xyz/sp1-turbo/

[33] National Academies of Sciences, Engineering, and Medicine, "Quantum Computing: Progress and Prospects," The National Academies Press, 2019. https://www.nationalacademies.org/read/25196/chapter/6

[34] M. Boyer, G. Brassard, P. Høyer, A. Tapp, "Tight Bounds on Quantum Searching," *Fortschritte der Physik*, vol. 46, no. 4–5, pp. 493–505, 1998. https://doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P

[35] J. Ha, J. Lee, J. Heo, "Resource Analysis and Modifications of Quantum Computing with Noisy Qubits for Elliptic Curve Discrete Logarithms," *Scientific Reports*, 14:3927, 2024. https://www.nature.com/articles/s41598-024-54434-w

[36] C. Gidney, "How to Factor 2048 Bit RSA Integers with Less Than a Million Noisy Qubits," arXiv:2505.15917, May 2025. https://arxiv.org/abs/2505.15917

[37] D. Unruh, "Computationally Binding Quantum Commitments," *EUROCRYPT 2016*, Lecture Notes in Computer Science, vol. 9666, pp. 497–527, 2016. https://doi.org/10.1007/978-3-662-49896-5_18

[38] A. Chiesa, F. Ma, N. Spooner, M. Zhandry, "Post-Quantum Succinct Arguments: Breaking the Quantum Rewinding Barrier," *Proceedings of the 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 49–58, 2021. https://doi.org/10.1109/FOCS52979.2021.00014

[39] J. Thaler, "17 Misconceptions about SNARKs (and Why They Hold Us Back)," a16z crypto, July 2023. https://a16zcrypto.com/posts/article/17-misconceptions-about-snarks/

[40] J. Nick, "SHRINCS: 324-byte stateful post-quantum signatures with static backups," Delving Bitcoin, December 11, 2025. https://delvingbitcoin.org/t/shrincs-324-byte-stateful-post-quantum-signatures-with-static-backups/2158

[41] Conduition, "Hash-Based Signature Schemes for Post-Quantum Bitcoin," 2024. https://conduition.io/cryptography/quantum-hbs/

[42] R. Campbell, "Hybrid Post-Quantum Signatures for Bitcoin and Ethereum: A Protocol-Level Integration Strategy," *Journal of the British Blockchain Association (JBBA)*, 2025. https://jbba.scholasticahq.com/api/v1/articles/154321-hybrid-post-quantum-signatures-for-bitcoin-and-ethereum-a-protocol-level-integrat pdf

[43] Wouter Castryck and Thomas Decru, "An efficient key recovery attack on SIDH," Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975

[44] E. Heilman, "Post Quantum Signatures and Scaling Bitcoin," Bitcoin-dev mailing list, April 2025. https://groups.google.com/g/bitcoindev/c/wKizvPUfO7w

[45] Succinct Labs, "SP1 Hypercube Is Now Live on Mainnet," Blog post, 2025. https://blog.succinct.xyz/sp1-hypercube-is-now-live-on-mainnet/

[46] 0xPARC, "circom-ecdsa," GitHub, 2022. https://github.com/0xPARC/circom-ecdsa

[47] D. Tehrani, "Efficient ECDSA Signature Verification using Circom," Ethereum Research, September 2022. https://ethresear.ch/t/efficient-ecdsa-signature-verification-using-circom/13629

[48] Personae Labs, "Spartan-ECDSA," GitHub, 2023. https://github.com/personaelabs/spartan-ecdsa

[49] L.K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 212–219, 1996. https://doi.org/10.1145/237814.237866

[50] S. Jaques, M. Naehrig, M. Roetteler, F. Virdia, "Implementing Grover Oracles for Quantum Key Search on AES and LowMC," *EUROCRYPT 2020*, 2020. https://eprint.iacr.org/2019/1146

[51] C. Zalka, "Grover's Quantum Searching Algorithm is Optimal," *Physical Review A*, vol. 60, no. 4, pp. 2746–2751, 1999. https://doi.org/10.1103/PhysRevA.60.2746

[52] A. Chiesa, M. Dall'Agnol, Z. Di, Z. Guan, N. Spooner, "Quantum Rewinding for IOP-Based Succinct Arguments," *TCC 2025*, Lecture Notes in Computer Science, vol. 16270, Springer, 2025. (Original preprint arXiv:2411.05360, November 2024; revised as IACR ePrint 2025/947.) https://eprint.iacr.org/2025/947

[53] T. Ruffing, "Transition to post-quantum," Bitcoin-dev mailing list, February 2018. https://gnusha.org/pi/bitcoindev/1518710367.3550.111.camel@mmci.uni-saarland.de/

[54] I. Stewart, D. Ilie, A. Zamyatin, S. Werner, M.F. Torshizi, W.J. Knottenbelt, "Committing to Quantum Resistance: A Slow Defence for Bitcoin Against a Fast Quantum Computing Attack," *Royal Society Open Science*, 5(6):180410, 2018. https://royalsocietypublishing.org/doi/10.1098/rsos.180410

[55] T. Dryja, "Post-Quantum commit/reveal Fawkescoin variant as a soft fork," Bitcoin-dev mailing list, May 2025. https://groups.google.com/g/bitcoindev/c/LpWOcXMcvk8

[56] olkurbatov, "PQ provers for P2PKH outputs," Delving Bitcoin forum, February 2026. https://delvingbitcoin.org/t/pq-provers-for-p2pkh-outputs/2287

[57] Ethereum Foundation, "Poseidon Cryptanalysis Initiative 2024–2026," 2024. https://www.poseidon-initiative.info/

[58] S. Chaliasos, et al., "Analyzing and Benchmarking ZK-Rollups," *AFT 2024*, IACR ePrint 2024/889, 2024. https://eprint.iacr.org/2024/889.pdf

[59] B. Westerbaan, "Sizing Up Post-Quantum Signatures," Cloudflare Blog, November 2021. https://blog.cloudflare.com/sizing-up-post-quantum-signatures/

[60] BTQ Technologies, "Introducing PQScale: A Scaling Solution for Post-Quantum Signatures," Blog post, 2025. https://www.btq.com/blog/introducing-pqscale-a-scaling-solution-for-post-quantum-signatures

[61] K. Fukuda, S. Matsuo, Y. Suga, T. Ito, "The Grand Challenge of PQC Migration: Analysis of Modern Blockchain and Intertwined Human Egoisms," IACR ePrint 2025/1626, 2025. https://eprint.iacr.org/2025/1626

[62] J. Bonneau, A. Miller, "Fawkescoin," *Security Protocols XXII*, Lecture Notes in Computer Science, vol. 8809, pp. 350–358, Springer, 2014. https://doi.org/10.1007/978-3-319-12400-1_35

[63] L. Wandersleb, "Pre-emptive commit/reveal for quantum-safe migration (poison-pill)," Bitcoin-dev mailing list, June 2025. https://groups.google.com/g/bitcoindev/c/oa4nDmlLzN4

[64] Hunter Beast, "P2QRH / BIP-360 Update," Bitcoin-dev mailing list, 2025. https://groups.google.com/g/bitcoindev/c/oQKezDOc4us

[65] Hunter Beast, "Major BIP 360 Update," Delving Bitcoin, 2025. https://delvingbitcoin.org/t/major-bip-360-update/2170

[66] J. Rubin, "OP_CAT Makes Bitcoin Quantum Secure," Bitcoin-dev mailing list, July 2021. https://gnusha.org/pi/bitcoindev/CAD5xwhgzR8e5r1e4H-5EH2mSsE1V39dd06+TgYniFnXFSBqLxw@mail.gmail.com/

[67] E. Heilman, "Signing a Bitcoin Transaction with Lamport Signatures (no changes needed)," Bitcoin-dev mailing list, April 2024. https://mailing-list.bitcoindevs.xyz/bitcoindev/CAEM=y+XyW8wNOekw13C5jDMzQ-dOJpQrBC+qR8-uDot25tM=XA@mail.gmail.com/

[68] J. Lopp, "A Post Quantum Migration Proposal," Bitcoin-dev mailing list discussion thread, July 2025. https://groups.google.com/g/bitcoindev/c/uEaf4bj07rE

[69] D. Ray, E. Ben-Sasson, Exchange on post-quantum security of FRI/STARKs (within the "Post Quantum Signatures and Scaling Bitcoin" thread), Bitcoin-dev mailing list, April 4–6, 2025. https://groups.google.com/g/bitcoindev/c/wKizvPUfO7w

[70] T. Mallick, M. Zeldin, M. Cenk, C. Nita-Rotaru, "Quantum Disruption: An SoK of How Post-Quantum Attackers Reshape Blockchain Security and Performance," arXiv:2512.13333, December 2025. https://arxiv.org/abs/2512.13333

[71] D. Nugent, "Quantum Vulnerability of Bitcoin Addresses," Project Eleven Blog, July 2025. https://blog.projecteleven.com/posts/quantum-vulnerability-of-bitcoin-addresses

[72] T. Ruffing, "The Post-Quantum Security of Bitcoin's Taproot as a Commitment Scheme," IACR ePrint 2025/1307, July 2025. https://eprint.iacr.org/2025/1307

[73] Presidio Bitcoin, "Insights from the Quantum Bitcoin Summit," July 2025. https://presidiobitcoin.substack.com/p/insights-from-the-quantum-bitcoin

[74] C. Bendiksen, "Quantum Vulnerability in Bitcoin: A Manageable Risk," CoinShares Research, February 2026. https://coinshares.com/us/insights/research-data/quantum-vulnerability-in-bitcoin-a-manageable-risk/

[75] J. Lopp, "Against Allowing Quantum Recovery of Bitcoin," Bitcoin-dev mailing list, March 2025. https://mailing-list.bitcoindevs.xyz/bitcoindev/CADL_X_cF=UKVa7CitXReMq8nA_4RadCF==kU4YG+0GYN97P6hQ@mail.gmail.com/

[76] Ainvest, "32.7% of Bitcoin Supply at Quantum Risk as Address Reuse Exposes 6.36 Million BTC to Potential Attacks," July 2025. https://www.ainvest.com/news/bitcoin-news-today-32-7-bitcoin-supply-quantum-risk-address-reuse-exposes-6-36-million-b

[77] M. Buser, R. Dowsley, M.F. Esgin, et al., "A Survey on Exotic Signatures for Post-quantum Blockchain: Challenges and Research Directions," *ACM Computing Surveys*, 2023. https://dl.acm.org/doi/10.1145/3572771

[78] Hunter Beast, "libbitcoinpqc: Post-Quantum Cryptography for use with Bitcoin according to BIP-360," GitHub, 2025. https://github.com/bitcoin/libbitcoinpqc

[79] E. Heilman, A. Sabouri, "BIP-347: OP_CAT in Tapscript," Bitcoin Improvement Proposal, December 2023. https://github.com/bitcoin/bips/blob/master/bip-0347.mediawiki

[80] Bitcoin Optech, "Newsletter #301," May 8, 2024. https://bitcoinops.org/en/newsletters/2024/05/08/

[81] Ledger, "Should Crypto Fear Quantum Computing?" Blog post, 2024. https://www.ledger.com/blog/should-crypto-fear-quantum-computing

[82] J.J. Pont, J.J. Kearney, J. Moyler, C.A. Perez-Delgado, "Downtime Required for Bitcoin Quantum-Safety," arXiv:2410.16965, October 2024. https://arxiv.org/abs/2410.16965