Toward Protocol-Level Quantum Safety in Bitcoin A Formal, Adversarial, and Invariant-Driven Treatment

Mayckon Giovani ·
Stigning

CITE AS
openxiv:cs.CE.2026.00001
ISSN
3120-9556 (online)
LICENSE
CC-BY-4.0
POSTED
2026-05-27
VERSION
v1
SUBJECT
cs.CE

AI DISCLOSURE
ASSISTANT

COVER EVIDENCE

TRANSPARENCY · partial | IDENTITY · strong | PROVENANCE · strong | CITATIONS · strong | MATH · partial | INTEGRITY · partial

Toward Protocol-Level Quantum Safety in Bitcoin

A Formal, Adversarial, and Invariant-Driven Treatment

Mayckon Giovani

April 2026 (Draft for review)

Abstract

"Quantum-safe Bitcoin" is not a property of a subset of well-behaved transactions. It is a global safety property of the consensus state machine: for every reachable consensus state and for every quantum-capable adversary, no consensus-valid execution may contain an unauthorized state transition. This paper specifies an execution model for Bitcoin at the level required for hostile review: explicit UTXO state, total validation predicates, deterministic transition functions for transactions and blocks, and a network/scheduler model sufficient to reason about mempool races and reorganizations. We define security goals in game-based form against quantum polynomial-time (QPT) adversaries, separate safety from liveness, and state concrete invariants (authorization integrity, state consistency, and determinism), proving that all invariants are preserved across valid transitions. We give a consensus-level construction for post-quantum authorization (via commit-and-reveal witness programs) and prove, via a complete game-hopping reduction, that unauthorized spends imply a break of the underlying post-quantum signature or hash binding assumptions — with a tight, non-rewinding reduction. We further define and exclude cross-input and cross-transaction witness replay attacks via sighash commitment axioms, formalize network execution as traces and prove that adversarial network control does not bypass PQ authorization, and address the quantum random oracle model. Finally, we formalize the migration dilemma — protocol-level quantum safety cannot be achieved without either freezing unmigrated legacy outputs or accepting open theft under Shor — and provide a formal migration trace model with monotonicity guarantees. All results are supported by formal artifacts: a TLA+ model exhaustively checked by TLC (zero invariant violations across 492 states; concrete counterexample for the migration dilemma) and Coq-mechanized proofs of spend predicate totality, determinism, and parse canonicality.

Contents

1 Introduction

Bitcoin's authorization model is cryptographic: the consensus rules grant spending power to whoever can produce a witness that satisfies the locking condition of an output. In the current protocol, that locking condition ultimately depends on secp256k1 signatures (ECDSA historically; Schnorr

via Taproot) [12, 20–22]. Under Shor's algorithm [18], this assumption collapses: discrete logarithms in elliptic curve groups become efficiently solvable by a quantum polynomial-time (QPT) adversary.

The usual failure mode in "quantum Bitcoin" discussions is local reasoning: "some transactions can be made post-quantum, therefore Bitcoin can be made post-quantum". This is structurally wrong. Consensus security is quantified over the entire reachable state space and over adversarially chosen executions. If any reachable state admits any consensus-valid unauthorized spend under a QPT adversary, the protocol is not quantum-safe at the protocol level.

1.1 Protocol-level quantum safety (informal)

At a minimum, protocol-level quantum safety requires:

1. a deterministic, total validation predicate and transition function (consensus is a state machine, not an API);
2. a game-based authorization definition against QPT adversaries;
3. invariants stated as quantifiers over *all* reachable states and all consensus-valid traces;
4. an explicit network model (mempool observation and transaction replacement are not accidents; they are adversarial levers);
5. a migration model that acknowledges lost keys and therefore the liveness-vs-safety dilemma.

1.2 Contributions and boundary of claims

What is proven. This paper proves the following results, all conditional on explicitly stated axioms: (i) consensus invariants (no double spend, state consistency, determinism) are preserved across all valid transitions (Theorems 1, 2), confirmed by exhaustive TLC model checking (Section 10.1); (ii) unauthorized spends of PQ-locked outputs imply a break of the underlying PQ signature scheme (EUF-CMA) or hash binding, via a tight, non-rewinding game-hopping reduction valid in the QROM (Theorem 4); (iii) cross-input and cross-transaction witness replay are excluded under the sighash commitment axiom (Theorem 5); (iv) adversarial network control (mempool observation, conflict injection, reorgs) does not bypass PQ authorization (Theorem 6); (v) no cryptographic-only protocol transition can simultaneously preserve safety and liveness for all outputs when lost keys exist (Theorems 8, 9), with a concrete counterexample produced by TLC; (vi) the PQ spend predicate is total, deterministic, and its witness parsing is canonical — machine-checked in Coq (Section 10.2, proof obligations PO-1, PO-2, PO-3, with PO-3 strengthened to full witness identity via varint-based encoding); (vii) the UTXO transition function is deterministic and preserves the no-double-spend invariant — machine-checked in Coq (PO-5); (viii) the cost function satisfies $\mathsf{Cost}(tx) = \mathsf{weight}(tx)$ with exact equality — machine-checked in Coq (PO-7).

What is assumed. The following are explicit axioms or implementation obligations, *not* proven in this paper: (a) the deployed sighash implementation satisfies the commitment property (Definition 5), including consensus-semantic injectivity — the Coq model of Sighash v2 discharges PO-4 under a SHA-256 collision-resistance axiom, and the deterministic Rust preimage transcript construction is checked against a Coq-extracted transcript function, while SHA-256 implementation correctness and compiler correctness remain outside the artifact boundary; (b) the transaction

id function h is collision-resistant against QPT adversaries (Definition 9); (c) the chosen PQ signature scheme is EUF-CMA secure against QPT adversaries, including in the QROM; (d) the mechanized spend predicate corresponds to the implementation used by consensus nodes — supported by bounded Coq-to-OCaml serializer extraction, golden-vector comparison, property-based correspondence tests, Kani source-level bounded parser-refinement harnesses, and release-binary translation validation against the Coq-extracted summaries, but not by a proof of compiler correctness (PO-8). All security theorems are conditional on these assumptions. Where an assumption is violated, the specific theorem that depends on it is identified.

2 System Model

We model the consensus-critical part of Bitcoin as a labeled transition system with explicit state and deterministic transitions. We intentionally separate (i) the *ledger state machine* (what blocks do) from (ii) the *network execution* (how transactions/blocks are observed, delayed, replaced).

2.1 Basic objects: outputs, outpoints, UTXO

Definition 1 (Outpoints and outputs)**.** *An* outpoint *is an identifier of the form*

$$op \in \mathsf{OutPoint} := \{0,1\}^{256} \times \mathbb{N},$$

interpreted as $(\mathit{txid}, \mathit{index})$*. An* output *is a tuple*

$$o \in \mathsf{Output} := \mathsf{Value} \times \mathsf{Script},$$

where $\mathsf{Value} \subset \mathbb{N}$ *is a satoshi-denominated value domain and* Script *is a consensus-interpreted locking program (e.g., a ScriptPubKey or witness program).*

Definition 2 (UTXO set)**.** *The UTXO set is a finite partial function*

$$U \in \mathsf{UTXO} := \mathsf{OutPoint} \rightharpoonup \mathsf{Output}.$$

We write $U[op]$ *for lookup when defined, and* $\mathrm{dom}(U) \subseteq \mathsf{OutPoint}$ *for its domain.*

2.2 Transactions and witness evaluation

We abstract a transaction as a structured object containing inputs and outputs. An input references a prior outpoint and carries a witness.

Definition 3 (Transactions)**.** *A transaction is a tuple*

$$tx \in \mathsf{Tx} := (I, O, \mathit{meta}),$$

where $I = (in_0, \ldots, in_{n-1})$ *is a list of inputs,* $O = (o_0, \ldots, o_{m-1})$ *is a list of outputs* $o_j \in \mathsf{Output}$*, and* meta *contains consensus-relevant fields (locktime, sequences, etc.). Each input is of the form* $in_i = (op_i, w_i)$ *with* $op_i \in \mathsf{OutPoint}$ *and* $w_i \in \mathsf{Witness}$*.*

Witness evaluation is treated as a deterministic total predicate:

$$\mathsf{Eval} : \mathsf{Script} \times \mathsf{Witness} \times \{0,1\}^{256} \to \{0,1\},$$

where the third argument is the message being authorized (a canonical *sighash*). This hides Script-level complexity while making explicit the consensus requirement: there exists a unique message m_i per input that is being authorized.

Definition 4 (Sighash)**.** *Let* $\mathsf{Sighash}(tx, i, \mathsf{ctx}) \in \{0,1\}^{256}$ *be the deterministic sighash function for input* i *of* tx *under a consensus context* ctx *(Taproot defines such a function for SegWit v1 spending) [21, 22]. We will write* $m_i := \mathsf{Sighash}(tx, i, \mathsf{ctx})$.

We require the following commitment property, which prevents cross-transaction witness replay and partial-commitment attacks.

Definition 5 (Sighash commitment)**.** *The sighash function satisfies* commitment *if:*

1. ***Injectivity (modulo input index):*** *For fixed consensus context* ctx *and fixed input index* i*,*

$$\mathsf{Sighash}(tx, i, \mathsf{ctx}) = \mathsf{Sighash}(tx', i, \mathsf{ctx}) \implies tx \equiv_{\mathrm{cs}} tx',$$

   where $tx \equiv_{\mathrm{cs}} tx'$ *denotes* consensus-semantic equivalence*: tx and tx' are identical in all consensus-critical fields (version, inputs with outpoints, outputs with values and scripts, locktime, and all fields committed to by the sighash). Transactions that differ only in consensus-irrelevant encoding details (e.g., witness serialization padding, future annex content not committed to by the sighash) are considered equivalent under* $\equiv_{\mathrm{cs}}$*. In particular, if* $tx \equiv_{\mathrm{cs}} tx'$ *then* $\mathsf{ValidTx}(U, tx) = \mathsf{ValidTx}(U, tx')$ *and* $\delta_{\mathsf{Tx}}(U, tx) = \delta_{\mathsf{Tx}}(U, tx')$ *for all* U*.*

2. ***Cross-input separation:*** *For* $i \neq j$ *in the same transaction,*

$$\mathsf{Sighash}(tx, i, \mathsf{ctx}) \neq \mathsf{Sighash}(tx, j, \mathsf{ctx}).$$

3. ***Field coverage:*** $\mathsf{Sighash}(tx, i, \mathsf{ctx})$ *commits to all consensus-critical fields of* tx *(version, inputs with their outpoints, outputs with their values and scripts, locktime) and to the spent output's script and value.*

Remark 1. *BIP 341 Taproot sighash satisfies these properties by construction: it hashes all input outpoints, all output scripts/values, the spent output's scriptPubKey and amount, and the input index [21]. Legacy sighash modes (e.g.,* `SIGHASH_NONE`*,* `SIGHASH_SINGLE`*) intentionally violate field coverage for specific use cases; outputs locked under such modes require separate analysis. The commitment property is an* axiom *of our model; any concrete instantiation must be verified against it.*

Definition 6 (Consensus-semantic equivalence)**.** *Two transactions* tx, tx' *are* consensus-semantically equivalent*, written* $tx \equiv_{\mathrm{cs}} tx'$*, if they agree on all fields that affect consensus validation and state transition:*

1. *version, locktime, and sequence numbers;*
2. *the list of input outpoints* $(op_0, \ldots, op_{n-1})$*;*
3. *the list of outputs* $(o_0, \ldots, o_{m-1})$ *including values and scripts;*
4. *all fields committed to by the sighash function.*

The equivalence class $[tx]_{\mathrm{cs}}$ *is the quotient of the transaction space by* $\equiv_{\mathrm{cs}}$*. Transactions in the same class may differ in consensus-irrelevant encoding details (e.g., witness serialization padding, non-committed annex content, or byte-level encoding choices that do not affect* $\mathsf{ValidTx}$ *or* δ_{Tx}*). By construction:* $tx \equiv_{\mathrm{cs}} tx'$ *implies* $\mathsf{ValidTx}(U, tx) = \mathsf{ValidTx}(U, tx')$*,* $\delta_{\mathsf{Tx}}(U, tx) = \delta_{\mathsf{Tx}}(U, tx')$*, and* $h(tx) = h(tx')$ *for all* U*.*

We formalize the quotient via a canonical normalization function*:*

$$\mathsf{Norm} : \mathsf{Tx} \to \mathsf{Tx}, \qquad \textit{such that } \mathsf{Norm}(tx) = \mathsf{Norm}(tx') \iff tx \equiv_{\mathrm{cs}} tx'.$$

Norm *strips all consensus-irrelevant fields and produces a unique representative of each equivalence class. Concretely,* $\mathsf{Norm}(tx)$ *retains exactly the fields enumerated in items 1–4 above and sets all other fields to a fixed canonical value (e.g., empty witness, zero-length annex). The sighash commitment property (Definition 5, item 1) can then be restated as:*

$$\mathsf{Sighash}(tx, i, \mathit{ctx}) = \mathsf{Sighash}(tx', i, \mathit{ctx}) \implies \mathsf{Norm}(tx) = \mathsf{Norm}(tx').$$

This eliminates any ambiguity about what "equivalent" means: two transactions are equivalent if and only if they have the same canonical representative.

Lemma 1 (Norm-preserving validation and transition)**.** *For all UTXO sets* U *and transactions* tx*:*

1. $\mathsf{ValidTx}(U, tx) = \mathsf{ValidTx}(U, \mathsf{Norm}(tx))$*;*
2. $\delta_{\mathsf{Tx}}(U, tx) = \delta_{\mathsf{Tx}}(U, \mathsf{Norm}(tx))$*;*
3. $h(tx) = h(\mathsf{Norm}(tx))$*.*

Proof. By definition, Norm preserves all fields enumerated in Definition 6 (items 1–4) and only modifies consensus-irrelevant fields. ValidTx depends only on syntax (which Norm preserves by retaining all structural fields), input outpoints, output values/scripts, and witness evaluation. Witness evaluation depends on the spend predicate and sighash, both of which are functions of the consensus-critical fields only. δ_{Tx} depends only on the input outpoints (for removal) and the transaction id and output list (for creation), all of which are consensus-critical. $h(tx)$ is computed from the consensus-critical serialization (SegWit txid excludes witness data), so $h(tx) = h(\mathsf{Norm}(tx))$. □

Remark 2 (Scope of the sighash axiom)**.** *All security theorems in this paper (Theorems 4, 5, 6) are conditional on the deployed sighash satisfying the commitment property (Definition 5). The Coq development now proves the modeled Sighash v2 commitment property under a SHA-256 collision-resistance axiom and exposes an extractable transcript constructor for the final preimage assembly (PO-4 in Section 10.3). The extraction pipeline compares that transcript constructor against the deployed Rust byte construction, including outpoint serialization, output serialization, spent-output serialization, and final preimage assembly with supplied sub-hashes. What remains outside that proof is SHA-256 implementation/refinement, the SHA-256 collision-resistance assumption itself, and compiler/toolchain correctness. Our results should be read as: "*if *the deployed sighash transcript corresponds to the verified model and SHA-256 satisfies the stated collision-resistance axiom,* then *the stated security guarantees hold." A full mechanization of BIP 341 itself would require formalizing the BIP 341 specification and proving refinement to the deployed serialization — a valuable but separate effort.*

2.3 Validation predicates and transition functions

The critique that "you cannot prove anything without δ" is correct. We therefore define total predicates and deterministic transition functions.

Definition 7 (Transaction validation)**.** *Transaction validation is a predicate*

$$\mathsf{ValidTx} : \mathsf{UTXO} \times \mathsf{Tx} \to \{0,1\}$$

defined as

$$\mathsf{ValidTx}(U,tx) = 1 \iff \Big(\mathit{Syntax}(tx) \wedge \mathit{NoDupInputs}(tx) \wedge \mathit{InputsExist}(U,tx) \wedge \mathit{ValueConservation}(U,tx) \wedge \mathit{WitnessOK}(U$$

where $\mathit{WitnessOK}(U,tx)$ *expands to:*

$$\forall i \in \{0,\ldots,|I|-1\} : \ \ \mathsf{Eval}(\pi_i, w_i, m_i) = 1,$$

with $(\cdot,\pi_i) = U[op_i]$ *and* $m_i = \mathsf{Sighash}(tx,i,\mathit{ctx})$*.*

Definition 8 (Transaction transition)**.** *The transaction transition function*

$$\delta_{\mathsf{Tx}} : \mathsf{UTXO} \times \mathsf{Tx} \to \mathsf{UTXO}$$

is defined (for all inputs) by removing spent outpoints and adding newly created outpoints:

$$\delta_{\mathsf{Tx}}(U,tx) := (U \setminus \{op_i\}_i) \ \cup \ \{(h(tx),j) \mapsto o_j\}_j,$$

where $h(tx) \in \{0,1\}^{256}$ *is the canonical transaction id and* j *ranges over the output indices.*

Remark 3. *In actual Bitcoin, the txid and outpoint formation depends on SegWit/non-SegWit serialization (wtxid, etc.). For protocol-level authorization analysis, the essential property is uniqueness and determinism: outpoint identifiers are stable and collision-resistant.*

We formalize this as an explicit axiom, since the invariant preservation proof (Section 4.5) depends on it.

Definition 9 (Transaction id collision resistance)**.** *The transaction id function* $h : \mathsf{Tx} \to \{0,1\}^{256}$ *satisfies* collision resistance *if for all QPT adversaries* $\mathcal{A}_q$*:*

$$\mathbb{P}[\mathcal{A}_q \text{ outputs } (tx,tx') \text{ with } tx \neq tx' \text{ and } h(tx) = h(tx')] \leq \mathrm{negl}(\lambda).$$

The system requirement is not generic collision search in a vacuum but collision against the active identifier set*: for any reachable state* U *and any new transaction* tx*, the newly created outpoints* $(h(tx),j)$ *must not collide with any* $op \in \mathrm{dom}(U)$*. This is a weaker requirement than arbitrary collision resistance: the adversary must find a collision against a specific, evolving set of existing identifiers rather than against an arbitrary second preimage.*

Remark 4 (Concrete collision budget)**.** *Bitcoin uses double-SHA-256 for txid computation (*$n = 256$ *output bits). The best known quantum collision attack (BHT algorithm) achieves* $O(2^{n/3}) = O(2^{85})$ *query complexity for generic collisions. However, the system's actual requirement is collision avoidance against the active outpoint set of size* $|\mathrm{dom}(U)|$*. Let* N_{total} *denote the total number of distinct outpoints ever introduced into the reachable state space across the system's lifetime. The probability that any pair among* N_{total} *identifiers collides is bounded by the birthday paradox:*

$$p_{\mathrm{coll}} \ \leq \ \binom{N_{\mathrm{total}}}{2} \cdot 2^{-256} \ \approx \ \frac{N_{\mathrm{total}}^2}{2^{257}}.$$

Under quantum speedup (BHT), an adversary actively searching for collisions achieves advantage $O(N_{\text{total}}^{1/3} \cdot 2^{-256/3})$ per query, but the total collision probability over N_{total} identifiers remains dominated by $N_{\text{total}}^2/2^{257}$ for passive collision (birthday) and $O(N_{\text{total}}/2^{85})$ for active quantum search against the existing set.

With $N_{\text{total}} \approx 10^9$ (cumulative outpoints over Bitcoin's lifetime), the passive birthday bound gives $p_{\text{coll}} \approx 2^{-197}$; the active quantum search bound gives $\approx 2^{-55}$. For $N_{\text{total}} \approx 10^{12}$ (aggressive long-horizon estimate), these become $\approx 2^{-177}$ and $\approx 2^{-45}$ respectively. Both remain negligible for practical security parameters. If future quantum capabilities or UTXO growth erode this margin, the txid function can be upgraded (e.g., to a 384-bit or 512-bit hash) as part of a witness version bump. The invariant preservation proof (Theorem 1) assumes collision resistance of h; if this assumption fails, StateConsistency may be violated by outpoint overwrites.

Definition 10 (Block validation and transition)**.** *A block is $B \in \mathsf{Blk} := (\mathit{hdr}, (tx_0, \ldots, tx_{k-1}))$. Block validation is a predicate $\mathsf{ValidBlk} : \mathsf{UTXO} \times \mathsf{Blk} \to \{0,1\}$ requiring PoW validity and sequential transaction validity (coinbase rules elided here for brevity). The block transition function $\delta_{\mathsf{Blk}} : \mathsf{UTXO} \times \mathsf{Blk} \to \mathsf{UTXO}$ applies δ_{Tx} sequentially:*

$$\delta_{\mathsf{Blk}}(U, B) := \delta_{\mathsf{Tx}}(\cdots \delta_{\mathsf{Tx}}(\delta_{\mathsf{Tx}}(U, tx_0), tx_1) \cdots, tx_{k-1}).$$

2.4 Determinism as an invariant

Consensus is well-defined only if ValidTx and δ_{Tx} are deterministic and total. We capture this explicitly as a safety invariant (Section 4) rather than treating it as "implementation detail".

3 Adversary and Network Model

We treat the adversary as QPT with explicit network control. This is not ideological; it is necessary to reason about mempool races and reorg windows.

3.1 Quantum capabilities

We assume $\mathcal{A}_q$ is a quantum polynomial-time adversary with access to:

1. **Shor**: polynomial-time discrete logarithms and factoring [18].
2. **Grover**: quadratic speedup for unstructured search and preimage search [9, 23].

3.2 Network/scheduler capabilities

We model the network as a set of nodes $\mathcal{N}$ that exchange messages (transactions, blocks). The delivery schedule is controlled by a scheduler with adversarial influence:

$$\mathsf{Sched} : (\mathsf{msgs}, \mathsf{time}) \to \mathsf{deliveries}.$$

The adversary's network control NetCtl includes:

- observing transactions in mempool prior to confirmation;
- injecting conflicts and fee-bumping replacements;
- delaying propagation (eclipse/partition-like behavior);

- exploiting reorganizations in the chain selection rule.

For chain consistency and reorg reasoning, we lean on the Bitcoin Backbone model [7], which formalizes safety/liveness of Nakamoto consensus under bounded delay and adversarial mining power.

4 Security Goals and Invariants

The paper is invariant-driven. This requires stating invariants in a form that can be checked against traces and used in proofs. We also separate *safety* (nothing bad happens) from *liveness* (something good eventually happens).

4.1 Safety vs liveness

Definition 11 (Safety)**.** *A safety property is a predicate P such that for every execution trace τ of the consensus state machine, if τ violates P then there exists a finite prefix of τ that already violates P. Operationally: an explicit "bad event" occurs.*

Definition 12 (Liveness)**.** *A liveness property is a predicate Q such that for every finite execution prefix, there exists an extension where Q holds. Operationally: assuming adequate network and mining conditions, honest transactions eventually confirm.*

Protocol-level quantum safety is primarily a *safety* requirement: prevent unauthorized transitions. It does not automatically imply liveness under censorship or under fee-market adversaries.

4.2 UTXO consistency invariants

Let U_t be the UTXO set after applying a prefix of blocks. We require:

$$\begin{aligned} \mathsf{NoDoubleSpend}: &\quad \forall op, \text{ op is removed from } U_t \text{ at most once across the trace,} \\ \mathsf{StateConsistency}: &\quad \forall op,\ op \notin \mathrm{dom}(U_t) \ \Rightarrow \ \text{either op never existed or has been spent.} \end{aligned}$$

4.3 Transition determinism

$$\mathsf{Determinism}: \quad \forall U, tx,\ \mathsf{ValidTx}(U, tx) = 1 \Rightarrow \exists!\ U' \text{ such that } U' = \delta_{\mathsf{Tx}}(U, tx).$$

4.4 Authorization integrity: game-based definition

We need a definition that a hostile reviewer recognizes: a concrete game whose advantage must be negligible. We use EUF-CMA style authorization games, extended to QPT adversaries [4, 8].

4.5 Invariant preservation

Stating invariants is necessary but not sufficient. We must prove that valid transitions preserve them.

Theorem 1 (Invariant preservation under δ_{Tx})**.** *Let U be a UTXO set satisfying $\mathsf{NoDoubleSpend}(U)$, $\mathsf{StateConsistency}(U)$, and $\mathsf{Determinism}(U)$. Let tx be a transaction with $\mathsf{ValidTx}(U, tx) = 1$. Assume collision resistance of h (Definition 9). Then $U' := \delta_{\mathsf{Tx}}(U, tx)$ satisfies all three invariants.*

Proof. We verify each invariant in turn.

Determinism. δ_{Tx} is defined by a single expression (set difference plus union) with no branching on non-deterministic state. Given U and tx, the result U' is uniquely determined.

NoDoubleSpend. $\mathsf{ValidTx}$ requires $\mathsf{InputsExist}(U, tx)$: every op_i referenced by tx is in $\mathrm{dom}(U)$. $\mathsf{NoDupInputs}(tx)$ ensures no op_i appears twice within tx. δ_{Tx} removes each op_i from U exactly once. Since $op_i \in \mathrm{dom}(U)$ and U satisfies $\mathsf{NoDoubleSpend}$, each op_i has been created exactly once and not previously spent. After removal, $op_i \notin \mathrm{dom}(U')$, so it cannot be spent again.

StateConsistency. For any $op \notin \mathrm{dom}(U')$, either: (a) $op \notin \mathrm{dom}(U)$ and op is not a newly created outpoint of tx, so by the inductive hypothesis on U, op either never existed or was previously spent; (b) $op \in \mathrm{dom}(U)$ and $op = op_i$ for some input of tx, so op has now been spent; or (c) op is a newly created outpoint $(h(tx), j)$ — but these are added to U' by construction, so $op \in \mathrm{dom}(U')$, contradicting the assumption.

Additionally, collision resistance of h (Definition 9) ensures that newly created outpoints $(h(tx), j)$ do not collide with existing outpoints in $\mathrm{dom}(U) \setminus \{op_i\}_i$, preventing silent overwrites that would violate state consistency. □

Theorem 2 (Invariant preservation under δ_{Blk})**.** *If U satisfies all three invariants and $\mathsf{ValidBlk}(U, B) = 1$, then $\delta_{\mathsf{Blk}}(U, B)$ satisfies all three invariants.*

Proof. δ_{Blk} applies δ_{Tx} sequentially. $\mathsf{ValidBlk}$ requires each tx_k to be valid against the intermediate state produced by applying $tx_0, \ldots, tx_{k-1}$. By induction on k and Theorem 1, each intermediate state satisfies the invariants, and therefore so does the final state. □

Corollary 1 (Global invariant preservation)**.** *For any execution trace $\tau = (U_0, B_1, U_1, B_2, U_2, \ldots)$ where U_0 satisfies the invariants and each $\mathsf{ValidBlk}(U_{t-1}, B_t) = 1$, every reachable state U_t satisfies NoDoubleSpend, StateConsistency, and Determinism.*

4.6 Cryptographic primitives and authorization games

We now define the authorization predicate and the game-based security notion.

Cryptographic primitives. Let $\mathsf{PQC\text{-}Sig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ be a post-quantum signature scheme intended for consensus use (e.g., ML-DSA [13] or SLH-DSA [14]). Let $H : \{0,1\}^\star \to \{0,1\}^{256}$ be a fixed hash function (e.g., SHA-256).

Definition 13 (PQ spend predicate)**.** *For a commitment $P \in \{0,1\}^{256}$ and an input message $m \in \{0,1\}^{256}$, define*

$$\mathsf{SpendPred}_{\mathrm{PQ}}(P, m, w) = 1$$

iff the witness parses as $w \mapsto (\mathsf{pk}, \sigma)$, satisfies $H(\mathsf{pk}) = P$, and $\mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1$.

Definition 14 (Authorization predicate)**.** *A transaction tx with $\mathsf{ValidTx}(U, tx) = 1$ satisfies the* authorization predicate $\mathsf{Auth}(U, tx)$ *if, for every input i of tx spending output op_i with commitment P_i, the witness w_i could only have been produced by an entity with access to the signing oracle $\mathsf{Sign}(\mathsf{sk}_i, \cdot)$ where $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Kg}$ and $H(\mathsf{pk}_i) = P_i$. Formally, $\mathsf{Auth}(U, tx)$ holds iff for every input i:*

$$\exists\, q \in \mathcal{Q}_i : \; m_i = q,$$

where $\mathcal{Q}_i$ is the set of messages queried to $\mathsf{Sign}(\mathsf{sk}_i, \cdot)$ and $m_i = \mathsf{Sighash}(tx, i, \mathit{ctx})$. In other words, a transaction is authorized iff every input's sighash was submitted to the legitimate signing oracle. This definition eliminates ambiguity about delegation: authorization is defined purely by oracle access, not by intent or identity.

Remark 5 (Delegation and multi-party signing)**.** *The oracle-based definition of* Auth *naturally accommodates delegation and multi-party signing (MPC/threshold) flows: if a party delegates signing authority, the delegate obtains access to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *(or a functional equivalent that produces valid signatures). In the game-based model, this is captured by the delegate being allowed to query the signing oracle. A delegated signature on* m_i *results in* $m_i \in \mathcal{Q}_i$*, so* Auth *holds. The definition does not require that the original key holder personally invokes the oracle — only that the oracle was invoked on the relevant message by some entity with legitimate access. Unauthorized spend means no entity with oracle access authorized the sighash, regardless of organizational structure.*

Definition 15 (Unauthorized spend)**.** *A consensus-valid transaction is* unauthorized *if:*

$$\mathsf{Unauthorized}(U, tx) \iff \mathsf{ValidTx}(U, tx) = 1 \ \wedge \ \neg\, \mathsf{Auth}(U, tx).$$

A trace τ *violates authorization integrity if* $\exists\, t,\ tx \in B_t :\ \mathsf{Unauthorized}(U_{t-1}, tx)$*.*

Definition 16 (Authorization break game)**.** *The game* $\mathsf{AuthBreak}^{\mathcal{A}_q}(\lambda)$ *is:*

1. *Challenger samples* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^\lambda)$ *and sets* $P := H(\mathsf{pk})$*.*
2. $\mathcal{A}_q$ *receives* P *and oracle access to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$*.*
3. $\mathcal{A}_q$ *outputs* $(m^\star, w^\star)$*.*
4. $\mathcal{A}_q$ *wins iff* $\mathsf{SpendPred}_{\mathrm{PQ}}(P, m^\star, w^\star) = 1$ *and* $m^\star$ *was not previously queried to the signing oracle.*

Define $\mathrm{Adv}^{\mathcal{A}_q}_{\mathrm{Auth}}(\lambda) := \mathbb{P}[\mathsf{AuthBreak}^{\mathcal{A}_q}(\lambda) = 1]$*.*

Definition 17 (Protocol-level authorization integrity)**.** *We say that PQ-locked outputs satisfy authorization integrity against QPT adversaries if for all* $\mathcal{A}_q \in \mathsf{QPT}$*,*

$$\mathrm{Adv}^{\mathcal{A}_q}_{\mathrm{Auth}}(\lambda) \leq \mathrm{negl}(\lambda).$$

5 Why secp256k1 Authorization Fails Under Shor

Bitcoin's legacy spend predicates are instantiations of $\mathsf{SpendPred}_{\mathrm{EC}}$ that verify secp256k1 signatures. Under Shor, secp256k1 key recovery becomes feasible.

Lemma 2 (Key recovery under Shor)**.** *Let* $Q = dG$ *be a secp256k1 public key. In the presence of Shor's algorithm, there exists a QPT algorithm that computes* d *from* Q*.*

Theorem 3 (Impossibility with active ECDLP surface)**.** *If the reachable state space contains any spendable output whose consensus spend predicate depends on secp256k1 signatures (ECDSA or Schnorr), then Bitcoin cannot satisfy protocol-level quantum safety against* $\mathcal{A}_q$*.*

Proof. Let U be a reachable consensus state containing a spendable output $o = (v, \pi)$ at outpoint $op \in \mathrm{dom}(U)$, where π is a locking script that accepts a witness containing a valid secp256k1 signature under public key Q. We construct a QPT adversary $\mathcal{A}_q$ that produces an unauthorized spend.

Step 1 (Key recovery). Q is either directly on-chain (P2PK, P2TR key path) or revealed in the mempool upon an honest spend attempt (P2PKH, P2WPKH). In either case, $\mathcal{A}_q$ obtains Q. By Shor's algorithm applied to the secp256k1 curve [16, 18], $\mathcal{A}_q$ computes $d \in \mathbb{Z}_n$ such that $Q = dG$ in polynomial time.

Output type	Verifier exposed?	Attack window under Shor
P2PK (legacy)	yes (on-chain)	immediate
P2TR (key path)	yes (on-chain)	immediate
P2PKH / P2WPKH	no (hash until spend)	mempool race at first spend
P2SH / P2WSH	no (script until spend)	mempool race upon reveal

Table 1: Exposure timing of secp256k1 verifiers in Bitcoin.

Step 2 (Witness construction). $\mathcal{A}_q$ constructs a transaction tx' that spends op to an adversary-controlled output. Using d, $\mathcal{A}_q$ computes a valid ECDSA (or Schnorr) signature σ' on $m' := \mathsf{Sighash}(tx', 0, \mathsf{ctx})$. The witness $w' := (\sigma', Q)$ satisfies $\mathsf{Eval}(\pi, w', m') = 1$.

Step 3 (Consensus acceptance). Since $op \in \mathrm{dom}(U)$, $\mathsf{InputsExist}$ holds. tx' is constructed to satisfy Syntax, $\mathsf{NoDupInputs}$, and $\mathsf{ValueConservation}$. $\mathsf{WitnessOK}$ holds because $\mathsf{Eval}(\pi, w', m') = 1$. Therefore $\mathsf{ValidTx}(U, tx') = 1$, and the spend is consensus-valid.

Step 4 (Authorization violation). The spend is unauthorized: $\mathcal{A}_q$ was never given d by the legitimate owner, and the witness was not produced on behalf of the owner. This violates authorization integrity (Definition 16 adapted to $\mathsf{SpendPred}_{\mathrm{EC}}$).

Since U was an arbitrary reachable state containing such an output, protocol-level quantum safety fails. □

5.1 Public key exposure and mempool races

Key exposure is not homogeneous across output types. Some outputs reveal secp256k1 public keys on-chain (immediate exposure), while others reveal them only when spent (mempool window).

Lemma 3 (Mempool race theft)**.** *Consider an output whose verifier is revealed only in the witness of an honest spend (e.g., P2WPKH). If (i) Shor key recovery is fast relative to time-to-confirmation and (ii) $\mathcal{A}_q$ can inject and preferentially propagate conflicts, then $\mathcal{A}_q$ has a non-negligible probability of replacing the honest spend with a conflicting spend to an adversarial output.*

Remark 6. *Quantitative analyses of this race appear in quantum attack treatments of Bitcoin [1]. The point for protocol design is qualitative: hiding a secp256k1 key until spend does not yield quantum safety; it only shifts the break into a race condition.*

6 A Consensus-Level PQ Authorization Construction

The minimal consensus construction is to introduce PQ-locked outputs whose spend predicate is *only* $\mathsf{SpendPred}_{\mathrm{PQ}}$, and to couple this with a migration/enforcement policy (Section 9).

6.1 Witness-program commitment

SegWit witness programs provide a clean versioning mechanism [11]. Abstractly, an output is identified by a pair (v, P) where v is a version byte and P is a bounded-size program. For PQ, the program must remain compact; therefore we commit to the PQ verifying key using a 32-byte hash:

$$P := H(\mathsf{pk}) \in \{0,1\}^{256}.$$

The witness reveals (pk, σ).

6.2 Spend predicate

Given an input message $m = \mathsf{Sighash}(tx, i, \mathsf{ctx})$, the consensus spend predicate is:

$$\mathsf{SpendPred}_{\mathrm{PQ}}(P, m, w) = 1 \iff \exists(\mathsf{pk}, \sigma) : \ \mathsf{Parse}(w) = (\mathsf{pk}, \sigma) \ \wedge \ H(\mathsf{pk}) = P \ \wedge \ \mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1.$$

6.3 Determinism, parsing, and DoS bounds

Consensus cannot tolerate ambiguous encoding. Witness parsing must be fully specified and canonical. Additionally, verification cost must be bounded and accounted for in the block resource model. We model this as a cost function $\mathsf{Cost}(tx)$ and require:

$$\forall tx : \quad \mathsf{ValidTx}(U, tx) = 1 \Rightarrow \mathsf{Cost}(tx) \leq \alpha \cdot \mathsf{weight}(tx),$$

for a fixed engineering constant α.

At the block level, we additionally require a global cost cap:

Definition 18 (Block cost invariant)**.** *Block validation enforces:*

$$\forall B : \quad \mathsf{ValidBlk}(U, B) = 1 \Rightarrow \sum_{tx \in B} \mathsf{Cost}(tx) \leq C_{\max},$$

where $C_{\max}$ is a consensus-enforced constant (analogous to the 4 MWU weight limit in current Bitcoin).

This invariant ensures that even if individual PQ witnesses are expensive to verify, the total per-block verification cost is bounded. Without it, an adversary could construct blocks that are cryptographically valid but operationally infeasible to verify, degrading network liveness without breaking authorization. The constant $C_{\max}$ must be parameterized jointly with the choice of PQC-Sig and the target verification throughput (see Section 11).

7 Security Theorems: From Unauthorized Spends to Cryptographic Breaks

The core requirement is a reduction: a consensus-valid unauthorized spend of a PQ-locked output should imply a break of the underlying assumptions. We follow the game-hopping methodology [3, 19].

7.1 Hash binding game

Define the hash binding game $\mathsf{BindH}^{\mathcal{B}}$: the challenger samples $\mathsf{pk} \leftarrow \{0,1\}^\star$ from the key distribution of Kg, sets $P := H(\mathsf{pk})$, and $\mathcal{B}$ wins if it outputs $\mathsf{pk}' \neq \mathsf{pk}$ such that $H(\mathsf{pk}') = P$. Let $\mathrm{Adv}^{\mathcal{B}}_{\mathrm{BindH}}$ be the winning probability.

7.2 Full game-hopping reduction

We now give the complete reduction via a sequence of games $G_0 \to G_1 \to G_2$.

Definition 19 (Game G_0: real authorization game)**.** *G_0 is identical to $\mathsf{AuthBreak}^{\mathcal{A}_q}(\lambda)$ (Definition 16). The challenger samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^\lambda)$, sets $P := H(\mathsf{pk})$, gives P and signing oracle access to $\mathcal{A}_q$. $\mathcal{A}_q$ outputs $(m^\star, w^\star)$ and wins if $\mathsf{SpendPred}_{\mathrm{PQ}}(P, m^\star, w^\star) = 1$ with $m^\star$ unqueried.*

$$\mathbb{P}[G_0 = 1] = \mathrm{Adv}^{\mathcal{A}_q}_{\mathrm{Auth}}(\lambda) =: \epsilon(\lambda).$$

Definition 20 (Game G_1: binding enforcement)**.** *G_1 is identical to G_0 except: after $\mathcal{A}_q$ outputs $(m^\star, w^\star)$ with $\mathsf{Parse}(w^\star) = (\mathsf{pk}^\star, \sigma^\star)$, if $H(\mathsf{pk}^\star) = P$ but $\mathsf{pk}^\star \neq \mathsf{pk}$, the game sets* $\mathsf{bad} := \mathsf{true}$ *and the adversary* still wins. *Otherwise the game proceeds identically.*

Lemma 4 (Transition $G_0 \to G_1$)**.** *G_0 and G_1 are identical games (no behavioral change). The flag* bad *is purely bookkeeping:*

$$\mathbb{P}[G_0 = 1] = \mathbb{P}[G_1 = 1].$$

Proof. The only difference is the introduction of the flag bad, which does not alter any output or oracle response. The winning condition is unchanged. □

Definition 21 (Game G_2: abort on binding break)**.** *G_2 is identical to G_1 except: if* $\mathsf{bad} = \mathsf{true}$ *(i.e.,* $\mathsf{pk}^\star \neq \mathsf{pk}$ *but $H(\mathsf{pk}^\star) = P$), the adversary* loses *(the game outputs* 0*).*

Lemma 5 (Transition $G_1 \to G_2$)**.**

$$|\,\mathbb{P}[G_1 = 1] - \mathbb{P}[G_2 = 1]| \leq \mathrm{Adv}^{\mathcal{B}_H}_{\mathrm{BindH}}(\lambda).$$

Proof. G_1 and G_2 differ only when $\mathsf{bad} = \mathsf{true}$. By the fundamental lemma of game-playing [3]:

$$|\,\mathbb{P}[G_1 = 1] - \mathbb{P}[G_2 = 1]| \leq \mathbb{P}[\mathsf{bad} = \mathsf{true} \text{ in } G_1].$$

We construct $\mathcal{B}_H$ that breaks hash binding whenever bad is set. $\mathcal{B}_H$ receives a challenge P (where $P = H(\mathsf{pk})$ for unknown pk). $\mathcal{B}_H$ cannot sign directly, but it simulates the signing oracle using the real sk (which it generates itself; the binding game is about finding a second preimage of P, not about the signing key).

Concretely: $\mathcal{B}_H$ samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^\lambda)$, computes $P := H(\mathsf{pk})$, and runs $\mathcal{A}_q$ with this P and honest signing oracle. If $\mathcal{A}_q$ outputs $(\mathsf{pk}^\star, \sigma^\star)$ with $H(\mathsf{pk}^\star) = P$ and $\mathsf{pk}^\star \neq \mathsf{pk}$, then $\mathcal{B}_H$ outputs $\mathsf{pk}^\star$ as a second preimage. Therefore $\mathbb{P}[\mathsf{bad}] \leq \mathrm{Adv}^{\mathcal{B}_H}_{\mathrm{BindH}}(\lambda)$.

Reduction model note. $\mathcal{B}_H$ generates its own $(\mathsf{sk}, \mathsf{pk})$ rather than receiving P from an external challenger. This is a standard-model reduction (not black-box challenger-driven): $\mathcal{B}_H$ knows pk and can therefore simulate the signing oracle perfectly. The reduction is valid because the binding game only requires $\mathcal{B}_H$ to output a second preimage $\mathsf{pk}^\star \neq \mathsf{pk}$ with $H(\mathsf{pk}^\star) = H(\mathsf{pk})$; it does not require $\mathcal{B}_H$ to be ignorant of pk. An alternative formulation where $\mathcal{B}_H$ receives P from an external challenger and simulates signing without sk would require additional assumptions (e.g., a signing oracle from the challenger); we choose the self-generated formulation for simplicity and tightness. □

Remark 7 (Binding game model)**.** *The hash binding game* BindH *as defined in this paper is a* self-generated challenge *game: the reductor $\mathcal{B}_H$ chooses its own* pk *and therefore knows the preimage. This is strictly weaker than a game where the challenger provides P and the reductor must find any preimage — our game only requires finding a* second *preimage. The self-generated formulation is the correct model here because: (i) the reductor must simulate the signing oracle, which requires* sk*; (ii) the binding property we need is second-preimage resistance (given* pk*, find* $\mathsf{pk}' \neq \mathsf{pk}$ *with $H(\mathsf{pk}') = H(\mathsf{pk})$), not collision resistance in general; (iii) second-preimage resistance of SHA-256 with 256-bit output provides 2^{128} quantum security under Grover, matching the system's target. The bound $\mathrm{Adv}^{\mathcal{B}_H}_{\mathrm{BindH}}(\lambda)$ should therefore be interpreted as the second-preimage advantage, not the generic collision advantage.*

Lemma 6 (Reduction from G_2 to EUF-CMA)**.**

$$\mathbb{P}[G_2 = 1] \leq \mathrm{Adv}^{\mathcal{B}_{\mathrm{sig}}}_{\text{EUF-CMA}}(\lambda).$$

Proof. In G_2, a win requires $\mathsf{pk}^\star = \mathsf{pk}$ (since $\mathsf{pk}^\star \neq \mathsf{pk}$ is aborted) and $\mathsf{Vfy}(\mathsf{pk}, m^\star, \sigma^\star) = 1$ with $m^\star$ unqueried. We construct $\mathcal{B}_{\text{sig}}$ against the EUF-CMA game of PQC-Sig.

$\mathcal{B}_{\text{sig}}$ receives pk from the EUF-CMA challenger and sets $P := H(\mathsf{pk})$. It gives P to $\mathcal{A}_q$ and forwards signing queries to the EUF-CMA signing oracle. When $\mathcal{A}_q$ outputs $(m^\star, w^\star)$, $\mathcal{B}_{\text{sig}}$ parses $w^\star \mapsto (\mathsf{pk}^\star, \sigma^\star)$. If $\mathsf{pk}^\star \neq \mathsf{pk}$, $\mathcal{B}_{\text{sig}}$ aborts (this corresponds to the G_2 abort). Otherwise, $\mathcal{B}_{\text{sig}}$ outputs $(m^\star, \sigma^\star)$ as an EUF-CMA forgery.

The simulation is perfect: $\mathcal{A}_q$'s view is identical to G_2. The signing oracle is faithfully forwarded. $\mathcal{B}_{\text{sig}}$ makes exactly the same number of queries as $\mathcal{A}_q$. Therefore $\mathbb{P}[G_2 = 1] \leq \text{Adv}^{\mathcal{B}_{\text{sig}}}_{\text{EUF-CMA}}(\lambda)$. □

Theorem 4 (Authorization reduction — full)**.** *For any QPT adversary $\mathcal{A}_q$ that wins $\mathsf{AuthBreak}^{\mathcal{A}_q}(\lambda)$ with probability $\epsilon(\lambda)$, there exist QPT adversaries $\mathcal{B}_{\text{sig}}$ and $\mathcal{B}_H$ such that:*

$$\epsilon(\lambda) \;\leq\; \text{Adv}^{\mathcal{B}_{\text{sig}}}_{\text{EUF-CMA}}(\lambda) \;+\; \text{Adv}^{\mathcal{B}_H}_{\text{BindH}}(\lambda).$$

Moreover, the reduction is tight*: $\mathcal{B}_{\text{sig}}$ makes exactly q signing queries if $\mathcal{A}_q$ does, and $\mathcal{B}_H$ runs in time $\mathsf{Time}(\mathcal{A}_q) + O(\lambda)$.*

Proof. Combining Lemmas 4, 5, and 6:

$$\epsilon(\lambda) = \mathbb{P}[G_0 = 1] = \mathbb{P}[G_1 = 1] \leq \mathbb{P}[G_2 = 1] + \text{Adv}^{\mathcal{B}_H}_{\text{BindH}}(\lambda) \leq \text{Adv}^{\mathcal{B}_{\text{sig}}}_{\text{EUF-CMA}}(\lambda) + \text{Adv}^{\mathcal{B}_H}_{\text{BindH}}(\lambda).$$

Tightness follows from the constructions: $\mathcal{B}_{\text{sig}}$ forwards queries one-to-one with no rewinding; $\mathcal{B}_H$ runs $\mathcal{A}_q$ once with $O(\lambda)$ overhead for hashing. □

Corollary 2 (PQ outputs are safe if assumptions hold)**.** *If PQC-Sig is EUF-CMA secure against QPT adversaries and H is binding against QPT adversaries (with 256-bit output for post-Grover margin), then $\text{Adv}^{\mathcal{A}_q}_{\text{Auth}}(\lambda)$ is negligible for all $\mathcal{A}_q \in \mathsf{QPT}$.*

7.3 Tightness and concrete security

The reduction above is tight in the sense that no security is lost in the game transitions: $\mathcal{B}_{\text{sig}}$ does not guess or rewind, and $\mathcal{B}_H$ runs $\mathcal{A}_q$ exactly once. This means the concrete security of the PQ authorization scheme is:

$$\epsilon_{\text{system}}(\lambda) \leq \epsilon_{\text{sig}}(\lambda) + \epsilon_{\text{hash}}(\lambda),$$

where ϵ_{sig} and ϵ_{hash} are the concrete advantages against the signature scheme and hash function, respectively.

Impact of Grover's algorithm. Grover's algorithm [9] provides a quadratic speedup for unstructured search. For a hash function with n-bit output, the best generic quantum preimage attack requires $O(2^{n/2})$ queries [23]. With $n = 256$ (SHA-256), this yields $\sim 2^{128}$ quantum security — matching the post-quantum security target. For the signature scheme, ML-DSA (FIPS 204) at security level 2 targets ≥ 128 bits of quantum security [13]; SLH-DSA (FIPS 205) at equivalent parameters provides comparable margins [14].

Concrete parameter guidance. A deployment must ensure:

- $\epsilon_{\text{sig}}(\lambda) \leq 2^{-128}$ under QPT adversaries (NIST Level 1 or above);
- $\epsilon_{\text{hash}}(\lambda) \leq 2^{-128}$ for second-preimage resistance under Grover (satisfied by SHA-256 with 256-bit output);
- combined: $\epsilon_{\text{system}} \leq 2^{-127}$, which is negligible for any practical λ.

7.4 Quantum random oracle model considerations

The reduction in Section 7 treats H as a classical oracle: the reductions $\mathcal{B}_{\text{sig}}$ and $\mathcal{B}_H$ evaluate H on classical inputs. In the quantum random oracle model (QROM) [5], the adversary $\mathcal{A}_q$ may query H in superposition, which affects the feasibility of certain proof techniques.

Impact on the binding transition ($G_1 \to G_2$). The transition relies on extracting a second preimage when bad is set. In the classical ROM, this is straightforward: the reductor observes $\mathsf{pk}^\star$ in the clear. In the QROM, if $\mathcal{A}_q$ computes $H(\mathsf{pk}^\star)$ via a quantum query, the reductor cannot directly observe the query input without disturbing the state. However, our construction does *not* require online extraction from hash queries. The reductor only needs to inspect $\mathcal{A}_q$'s *final classical output* $(\mathsf{pk}^\star, \sigma^\star)$, which is a classical string. The check $H(\mathsf{pk}^\star) = P$ with $\mathsf{pk}^\star \neq \mathsf{pk}$ is performed on this classical output. Therefore, the $G_1 \to G_2$ transition does not require quantum query recording or the compressed oracle technique; it remains valid in the QROM.

Impact on the EUF-CMA reduction (G_2). The reduction $\mathcal{B}_{\text{sig}}$ forwards signing queries and inspects the final output. If $\mathsf{PQC\text{-}Sig}$ is proven EUF-CMA secure in the QROM (as ML-DSA and SLH-DSA are designed to be [13, 14]), then $\mathcal{B}_{\text{sig}}$'s advantage bound carries over directly.

Summary. The reduction is valid in the QROM without modification, because: (i) extraction occurs on classical outputs, not quantum query transcripts; (ii) the hash oracle is only programmed implicitly (no reprogramming is needed); (iii) the signature scheme's QROM security is assumed. Crucially, our proof design avoids both oracle reprogramming and rewinding — two techniques that are known to be non-trivial or impossible in the QROM [5, 24]. This is a deliberate architectural choice: by structuring the spend predicate as hash-then-verify (commit to $H(\mathsf{pk})$, reveal pk in the witness), we ensure that the reduction only needs to inspect classical outputs, sidestepping the fundamental difficulties of quantum query extraction. This construction is deliberately QROM-friendly: it avoids oracle programmability entirely, making the proof structure robust against the known barriers to QROM security (measurement disturbance, lack of extractability from superposition queries). If future analysis reveals a gap in the QROM security of a specific $\mathsf{PQC\text{-}Sig}$ candidate, the reduction's validity depends on that scheme's QROM proof, not on our game-hopping structure. For a comprehensive treatment of random oracles under quantum access, see Boneh et al. [5] and Zhandry [24].

8 Network Model Implications: Ordering, Reorgs, and Races

8.1 Replay and cross-input attacks

The authorization game (Definition 16) covers single-output forgery. A complete treatment must also exclude witness replay across inputs and across transactions. We define two additional games and show they reduce to the sighash commitment property (Definition 5).

Definition 22 (Cross-input replay game)**.** *The game* $\mathsf{CrossInput}^{\mathcal{A}_q}(\lambda)$ *is:*

1. *Challenger samples* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^\lambda)$*, sets* $P := H(\mathsf{pk})$*.*
2. *Challenger creates a transaction* tx *with two inputs* $i \neq j$*, both spending outputs locked to* P*, and provides* $\mathcal{A}_q$ *with a valid witness* w_i *for input* i*.*

3. *$\mathcal{A}_q$ wins if it produces a valid witness w_j for input j without querying the signing oracle on $m_j := \mathsf{Sighash}(tx, j, \mathsf{ctx})$.*

Lemma 7 (Cross-input replay is infeasible)**.** *Under the sighash commitment property (Definition 5) and EUF-CMA security of* PQC-*Sig:*

$$\mathbb{P}[\mathsf{CrossInput}^{\mathcal{A}_q}(\lambda) = 1] \leq \mathrm{Adv}^{\mathcal{B}_{\mathrm{sig}}}_{\text{EUF-CMA}}(\lambda).$$

Proof. By the cross-input separation property of $\mathsf{Sighash}$ (Definition 5, item 2), $m_i \neq m_j$. Therefore a valid signature σ_i on m_i is not a valid signature on m_j (except with negligible probability under EUF-CMA). Any witness w_j that passes verification must contain a fresh signature on m_j, which constitutes an EUF-CMA forgery if m_j was not queried. □

Definition 23 (Cross-transaction replay game)**.** *The game $\mathsf{CrossTx}^{\mathcal{A}_q}(\lambda)$ is:*

1. *Challenger samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^\lambda)$, sets $P := H(\mathsf{pk})$.*
2. *Challenger creates transaction tx spending an output locked to P at input i, and provides $\mathcal{A}_q$ with the valid witness w.*
3. *$\mathcal{A}_q$ constructs a different transaction $tx' \neq tx$ with an input i' spending an output locked to P.*
4. *$\mathcal{A}_q$ wins if w is a valid witness for input i' of tx' without querying the signing oracle on $m' := \mathsf{Sighash}(tx', i', \mathsf{ctx})$.*

Lemma 8 (Cross-transaction replay is infeasible)**.** *Under the sighash injectivity property (Definition 5, item 1) and EUF-CMA security of* PQC-*Sig:*

$$\mathbb{P}[\mathsf{CrossTx}^{\mathcal{A}_q}(\lambda) = 1] \leq \mathrm{Adv}^{\mathcal{B}_{\mathrm{sig}}}_{\text{EUF-CMA}}(\lambda).$$

Proof. By sighash injectivity, $tx \not\equiv_{\mathrm{cs}} tx'$ implies $m := \mathsf{Sighash}(tx, i, \mathsf{ctx}) \neq \mathsf{Sighash}(tx', i', \mathsf{ctx}) =: m'$ (for any i, i'). The witness w contains a signature valid under message m. For w to also be valid under m', the signature must verify on a distinct message, which is an EUF-CMA forgery. □

Theorem 5 (Complete authorization security)**.** *Under the sighash commitment property, EUF-CMA security of* PQC-*Sig against QPT adversaries, and hash binding of H against QPT adversaries, the PQ spend predicate resists:*

1. *direct forgery (Theorem 4);*
2. *cross-input witness replay (Lemma 7);*
3. *cross-transaction witness replay (Lemma 8).*

The combined advantage is bounded by:

$$\mathrm{Adv}^{\mathcal{A}_q}_{\mathrm{total}}(\lambda) \leq 3 \cdot \mathrm{Adv}^{\mathcal{B}_{\mathrm{sig}}}_{\text{EUF-CMA}}(\lambda) + \mathrm{Adv}^{\mathcal{B}_H}_{\mathrm{BindH}}(\lambda).$$

8.2 Network-level authorization resilience

The authorization reduction above is consensus-local: it says that if a PQ output is spent, the witness cannot be forged except by breaking primitives. However, Bitcoin users do not interact with δ_{Blk} directly; they broadcast transactions into a hostile network.

We now formalize the connection between network execution and authorization safety.

Definition 24 (Execution trace)**.** *An execution trace is a sequence*

$$\tau = (U_0 \xrightarrow{B_1} U_1 \xrightarrow{B_2} U_2 \xrightarrow{B_3} \cdots),$$

where U_0 is the genesis UTXO set and each transition satisfies $\mathsf{ValidBlk}(U_{t-1}, B_t) = 1$ *and* $U_t = \delta_{\mathsf{Blk}}(U_{t-1}, B_t)$*. We write $\tau[0..t]$ for the prefix up to state U_t.*

Definition 25 (Network-valid trace)**.** *A trace τ is* network-valid *under scheduler* Sched *and adversary $\mathcal{A}_q$ with network control* NetCtl *if:*

1. *each block B_t is delivered according to* Sched *(possibly with adversarial delay);*
2. *$\mathcal{A}_q$ may inject transactions into any block B_t (subject to* ValidBlk*);*
3. *$\mathcal{A}_q$ may observe mempool contents and inject conflicting transactions;*
4. *chain selection follows the longest-chain rule (or heaviest-chain variant).*

Definition 26 (Unauthorized spend in network trace)**.** *A transaction $tx \in B_t$ in a network-valid trace contains an unauthorized spend if* $\mathsf{Unauthorized}(U_{t-1}, tx)$ *(Definition 15). In the context of the authorization break game, this means the witness for at least one input constitutes a winning output in* AuthBreak *(Definition 16): the entity producing the witness was not given* sk *and did not obtain a signature on the relevant sighash from* $\mathsf{Sign}(\mathsf{sk}, \cdot)$*.*

Theorem 6 (Network-resilient authorization safety)**.** *Let τ be any network-valid trace under QPT adversary $\mathcal{A}_q$. If every spendable output in every reachable state U_t is PQ-locked (i.e., its spend predicate is* $\mathsf{SpendPred}_{\mathrm{PQ}}$*), then for all prefixes $\tau[0..t]$:*

$$\mathbb{P}[\textit{unauthorized spend in } \tau[0..t]] \leq t \cdot k_{\max} \cdot n_{\max} \cdot (\mathrm{Adv}^{\mathcal{B}_{\mathrm{sig}}}_{\text{EUF-CMA}}(\lambda) + \mathrm{Adv}^{\mathcal{B}_H}_{\text{BindH}}(\lambda)),$$

where $k_{\max}$ is the maximum number of transactions per block and $n_{\max}$ is the maximum number of inputs per transaction.

Proof. Each input of each transaction in each block is independently validated by ValidTx via WitnessOK. By Theorem 4, each individual input's witness can only be forged with advantage at most $\mathrm{Adv}_{\text{EUF-CMA}} + \mathrm{Adv}_{\text{BindH}}$. A union bound over all inputs (at most $n_{\max}$ per transaction), all transactions (at most $k_{\max}$ per block), and all blocks (at most t) yields the stated bound. Since both advantages are negligible and t, $k_{\max}$, $n_{\max}$ are polynomial, the total remains negligible.

Crucially, the adversary's network capabilities (NetCtl) — mempool observation, conflict injection, delay, reorgs — do not help forge witnesses for PQ-locked outputs. The witness must contain a valid PQ signature on the sighash, and the sighash is determined by the transaction the adversary constructs. Network control allows the adversary to *choose which transactions to include* but not to *bypass the authorization predicate.* □

Remark 8 (Conservatism of the union bound)**.** *The bound $t \cdot k_{\max} \cdot n_{\max} \cdot (\cdot)$ is a worst-case union bound that treats each input's forgery event as independent. In practice, a QPT adversary may correlate attacks across inputs (e.g., reusing intermediate quantum computations) or adapt queries based on observed witnesses. However, since each input's sighash is distinct (by the sighash commitment property) and each forgery requires a fresh EUF-CMA break on a distinct message, the union bound remains valid: correlation does not help the adversary produce a valid signature on a message it has not queried. Even under correlated or amortized quantum attacks (e.g., reusing intermediate quantum states across inputs), each successful forgery requires a valid signature on a fresh, distinct sighash message; the EUF-CMA game is defined per-message, so amortization does not reduce the per-input advantage. Tighter bounds under structured adversaries (e.g., adversaries that control block construction) are a natural refinement but do not affect the negligibility conclusion.*

Remark 9 (Block-constructing vs. observing adversaries)**.** *The bound in Theorem 6 is worst-case: it assumes the adversary controls block construction and can place adversarial transactions in every slot. For a weaker adversary that can only observe the mempool and inject conflicting transactions (but does not mine), the effective $k_{\max}$ and $n_{\max}$ are limited to the adversary's own injected transactions, not the entire block. Under the Backbone model [7], an adversary with mining fraction $\beta < 1/2$ controls at most a $\beta/(1-\beta)$ fraction of blocks in expectation, tightening the bound to $\approx t \cdot \frac{\beta}{1-\beta} \cdot k_{\max} \cdot n_{\max} \cdot (\cdot)$. We state the worst-case bound for generality; the tighter bound follows by substituting the adversary's effective block rate.*

Remark 10. *This theorem does* not *guarantee liveness (honest transactions may be censored or delayed) or protection against reorg-based double-spending of the adversary's own funds. It guarantees that the authorization predicate is not the failure point: no network-level attack converts into an authorization break for PQ-locked outputs. For liveness and chain quality under bounded adversarial mining power, we rely on Backbone-style results [7, 15].*

Two network-level facts remain relevant for legacy (non-PQ) outputs:

1. **Exposure-to-confirmation windows** create theft races for schemes where the verifier/secret becomes computable after observation (secp256k1 under Shor).
2. **Reorgs and censorship** affect liveness and the reliability of migration strategies.

Backbone-style results [7] provide a framework to state conditions under which honest transactions are eventually included (liveness) and honest views have common prefix (safety). Protocol-level quantum safety requires that the *authorization predicate* is not the point of failure under those executions.

9 Migration Model and the Liveness–Safety Dilemma

The hard part is not defining PQ outputs. The hard part is the existing UTXO set: legacy outputs do not commit to PQ verifiers. There is no cryptographic "reinterpretation" that turns a secp256k1-locked output into a PQ-locked output without the owner's participation.

9.1 Migration transactions and traces

Definition 27 (Migration transaction)**.** *A transaction tx is a* migration transaction *if:*

1. *at least one input spends a legacy (non-PQ) output;*

2. all outputs are PQ-locked (their spend predicates are $\mathsf{SpendPred}_{\mathrm{PQ}}$*).*

Definition 28 (PQ fraction)**.** *For a UTXO set* U*, define the* PQ fraction *as:*

$$\rho(U) := \frac{|\{op \in \mathrm{dom}(U) : \textit{output at op is PQ-locked}\}|}{|\,\mathrm{dom}(U)|}.$$

Definition 29 (Migration trace)**.** *A migration trace is an execution trace* $\tau = (U_0, B_1, U_1, \dots)$ *in which honest participants broadcast migration transactions. We say* τ *is* honest-migration *if no honest participant creates new legacy outputs after the migration announcement height* H_a*.*

Proposition 7 (Migration monotonicity)**.** *In an execution trace where consensus rules reject the creation of legacy outputs after the migration announcement height* H_a *(i.e.,* $\mathsf{ValidBlk}$ *requires all new outputs in blocks* $> H_a$ *to be PQ-locked), the PQ fraction is monotonically non-decreasing:*

$$\forall t' > t \geq H_a : \quad \rho(U_{t'}) \geq \rho(U_t).$$

Proof. After H_a, $\mathsf{ValidBlk}$ rejects any transaction that creates a legacy output. Therefore every new output in $\mathrm{dom}(U_{t+1}) \setminus \mathrm{dom}(U_t)$ is PQ-locked. Legacy outputs can only leave $\mathrm{dom}(U)$ by being spent (migrated or otherwise); they are never created. Let $L_t := |\{op \in \mathrm{dom}(U_t) : \text{legacy}\}|$ and $Q_t := |\{op \in \mathrm{dom}(U_t) : \text{PQ}\}|$. In each block after H_a: $L_{t+1} \leq L_t$ (legacy count can only decrease or stay) and new PQ outputs are added. Therefore $\rho(U_{t+1}) = Q_{t+1}/(L_{t+1} + Q_{t+1}) \geq Q_t/(L_t + Q_t) = \rho(U_t)$. □

Remark 11. *The monotonicity guarantee is enforced by consensus, not by honest behavior. Even an adversarial miner cannot create legacy outputs after* H_a*, because* $\mathsf{ValidBlk}$ *would reject the block. This is strictly stronger than the behavioral assumption in earlier drafts. Under partial deployment (e.g., a soft fork not yet activated by all miners), non-upgraded miners may produce blocks containing legacy outputs that upgraded nodes reject. In this case, monotonicity holds on the chain viewed by upgraded nodes; non-upgraded nodes see a different chain. The structural result (monotonicity under full consensus adoption) is unaffected; partial deployment weakens the operational guarantee but not the formal property.*

9.2 States, transformation, and cutover

Let U_0 be the current UTXO set. Let U_{PQ} denote a UTXO set where every spendable output is PQ-locked. Migration is not a single function applied once; it is a sequence of on-chain transitions that induces a partial transformation:

$$\Phi : \; U_0 \rightsquigarrow U_{\mathrm{PQ}}.$$

To eliminate ECDLP from the consensus attack surface, the protocol must eventually enforce a cutover height H_c such that, for blocks beyond H_c, legacy spend predicates are rejected:

$$\forall \text{inputs in blocks } > H_c : \quad \mathsf{SpendPred} \equiv \mathsf{SpendPred}_{\mathrm{PQ}}.$$

This can be expressed as a consensus restriction (soft-fork style), but it has a consequence: any unmigrated legacy output becomes unspendable after H_c.

Definition 30 (Migration completeness)**.** *Migration is* complete *at height* H_c *if* $\rho(U_{H_c}) = 1$*, i.e., every output in the UTXO set is PQ-locked. Migration is* incomplete *if* $\rho(U_{H_c}) < 1$*, meaning some legacy outputs remain.*

9.3 Unavoidable trade-off

Theorem 8 (No-Free-Migration)**.** *Under cryptographic-only authorization (no external trust assumptions such as social recovery, multisig escrow with trusted parties, or covenant-based recovery requiring protocol extensions beyond* $\mathsf{SpendPred}_{\mathrm{PQ}}$ *and* $\mathsf{SpendPred}_{\mathrm{EC}}$*), there exists no protocol transition* Φ *that simultaneously:*

1. *preserves authorization integrity against QPT adversaries for all outputs (safety);*
2. *preserves spendability for all outputs including those with irrecoverably lost keys (liveness).*

Formally: $\neg\exists\,\Phi$ *such that* $\forall\, op \in \mathrm{dom}(U_0)$*, both* $\mathsf{AuthIntegrity}(\Phi, op)$ *and* $\mathsf{Spendable}(\Phi, op)$ *hold under Shor, where authorization is determined solely by cryptographic witness verification.*

Proof. Immediate from Theorem 9 below: for any lost-key output, safety and liveness are mutually exclusive under cryptographic-only authorization. Since at least one lost-key output exists, no Φ can satisfy both properties universally. □

Remark 12. *The restriction to cryptographic-only authorization is essential. Mechanisms that introduce external trust assumptions (e.g., social recovery via a quorum of trusted parties, or time-locked covenants that transfer control to a recovery entity) can in principle break the dilemma — but they do so by weakening the authorization model, not by solving the cryptographic problem. Such mechanisms are outside the scope of this paper's consensus-level analysis.*

Theorem 9 (Lost keys force a dilemma)**.** *Assume there exist legacy outputs whose secrets are irrecoverably lost. Then under Shor, any protocol evolution that keeps those outputs indefinitely spendable necessarily violates authorization integrity (they become stealable by* $\mathcal{A}_q$*); conversely, any evolution that enforces protocol-level quantum safety must forfeit liveness for those outputs (they must be frozen/burned).*

Proof. We prove both directions by contradiction.

Direction 1: keeping legacy outputs spendable violates safety. Suppose the protocol keeps a legacy output *op* spendable indefinitely, with spend predicate $\mathsf{SpendPred}_{\mathrm{EC}}$ depending on secp256k1 public key Q. By Theorem 3, $\mathcal{A}_q$ can compute d from Q via Shor's algorithm and construct a consensus-valid unauthorized spend. This directly violates authorization integrity (Definition 26).

Direction 2: enforcing PQ-only predicates forfeits liveness for lost-key outputs. Suppose the protocol enforces $\mathsf{SpendPred} \equiv \mathsf{SpendPred}_{\mathrm{PQ}}$ after cutover height H_c. A legacy output *op* can only be migrated if its owner produces a valid legacy witness (requiring knowledge of d) to spend it into a PQ-locked output. If d is irrecoverably lost, no valid legacy witness can be produced before H_c. After H_c, the legacy spend predicate is rejected by consensus. Therefore *op* is permanently unspendable: it cannot be spent via legacy rules (rejected) or via PQ rules (no PQ commitment exists). This is a liveness failure for *op*.

Since at least one such lost-key output exists (an empirical fact: early Bitcoin outputs with demonstrably lost keys), the protocol must choose one direction. No third option exists: any spend predicate either depends on secp256k1 (vulnerable under Shor) or does not (requiring owner participation to migrate). □

Remark 13. *This is not a policy statement; it is a structural constraint imposed by cryptography and by operational reality (lost keys exist). Any proposal that avoids stating which side it chooses is incomplete.*

10 Formal Verification Artifacts

The goal is to produce artifacts that reduce consensus risk, not to "prove Bitcoin". We couple an executable model (for counterexample finding) with mechanized proofs (for critical components). Both artifacts are available in the companion repository.

10.1 TLA+ model checking: UTXO transitions

We implemented two finite-state TLA+ specifications of the UTXO transition system.

Model 1: single-input transactions (`BitcoinPQ.tla`). The base model has explicit state, two output types (legacy and PQ-locked), single-input single-output transactions, migration announcement and cutover heights, and the consensus rule that rejects legacy output creation after announcement. The specification was model-checked with TLC under the following instantiation: 2 genesis outputs (1 legacy, 1 PQ-locked), outpoint universe of size 6, migration announcement at height 1, cutover at height 2.

TLC exhaustively explored 492 states (260 distinct) to depth 8 with **zero violations** of the structural invariant, which conjoins:

- NoDoubleSpend: spent outpoints are disjoint from the UTXO domain;
- ValueBound: total value never exceeds the genesis total;
- StateConsistency: all outpoint identifiers are fresh and properly tracked.

Model 2: multi-input transactions with value conservation (`BitcoinPQMulti.tla`). The extended model strengthens PO-6 by supporting multi-input (1–2 inputs) and multi-output (1–2 outputs) transactions with explicit value conservation ($\sum \mathsf{inputs} \geq \sum \mathsf{outputs}$, with the difference as fee), freeze enforcement after cutover, and a MigrationMonotonicity invariant that tracks the PQ fraction $\rho(U_t)$ as a state variable and verifies it is non-decreasing after the announcement height. The model was instantiated with 3 genesis outputs (total value 4), outpoint universe of size 8, migration announcement at height 1, and cutover at height 3. TLC exhaustively explored **58,237 states (6,365 distinct)** to depth 10 with **zero violations** of all four invariants (NoDoubleSpend, ValueBound, StateConsistency, MigrationMonotonicity).

Result 1: structural invariants. Both models confirm zero violations of all structural invariants across every reachable state.

Result 2: migration dilemma (counterexample). When AuthIntegrityPQ (all outputs PQ-locked after cutover) is checked as an invariant in the base model, TLC produces a **concrete counterexample** in 5 states: a legacy output created before the announcement height survives to the cutover height without being migrated. The extended model produces an analogous counterexample when FreezeEnforcement is checked. This is the migration dilemma (Theorem 8) demonstrated mechanically: protocol-level quantum safety requires either complete migration or freezing of unmigrated outputs. The counterexample is not a bug in the model; it is the expected behavior that the impossibility theorem predicts.

10.2 Coq mechanization: spend predicate properties

We mechanized the PQ spend predicate and witness parsing in Coq (Rocq 9.1) across three modules.

Module 1: spend predicate and witness encoding (`SpendPredPQ.v`). This module proves PO-1, PO-2, and PO-3 with a varint-based witness encoding model. The witness parse function uses an axiomatized varint encoder/decoder (6 axioms capturing round-trip, positive length, trailing-data tolerance, prefix determinism, canonical uniqueness, and consumed-length equality) whose concrete discharge is faithful to Bitcoin's compact-size encoding through the single-byte and `0xFD`/u16 cases used by the current Coq model. Properties proved:

- **PO-1 (Totality):** $\forall P, m, w : \ \mathsf{SpendPred}_{\mathrm{PQ}}(P, m, w) \in \{\mathsf{true}, \mathsf{false}\}$. Machine-checked as `spend_pred_pq_total`.
- **PO-2 (Determinism):** $\forall P, m, w :$ $\mathsf{SpendPred}_{\mathrm{PQ}}(P, m, w)$ yields the same result on every evaluation. Machine-checked as `spend_pred_pq_deterministic` (reflexive) and `spend_pred_pq_deterministic_` (extensional).
- **PO-3 (Parse canonicality — strengthened):** If two witnesses parse to the same (pk, σ) pair, the witnesses are *byte-identical*: $w_1 = w_2$. Machine-checked as `parse_varint_injective`, which is strictly stronger than the original PO-3 (which only showed the relevant regions match). The proof proceeds via `parse_varint_witness_determines_serialize`: any accepting witness w satisfies $w = \mathsf{Serialize}(\mathsf{pk}, \sigma)$, establishing full structural canonicality.
- **Anti-malleability:** If $\mathsf{SpendPred}_{\mathrm{PQ}}(P, m, w_1) = 1$ and $\mathsf{SpendPred}_{\mathrm{PQ}}(P, m, w_2) = 1$ and both witnesses parse to the same (pk, σ), then $w_1 = w_2$. Machine-checked as `spend_pred_pq_anti_malleability`.
- **Round-trip:** $\mathsf{Parse}(\mathsf{Serialize}(\mathsf{pk}, \sigma)) = \mathsf{Some}(\mathsf{pk}, \sigma)$ for all non-empty pk, σ. Machine-checked as `parse_serialize_round_trip`.

Additionally, we proved a complete characterization of the spend predicate (`spend_pred_pq_iff`): the predicate accepts if and only if parsing succeeds, the hash matches, and signature verification passes. The cryptographic primitives (H, Vfy) are axiomatized as parameters, matching the paper's approach: the proofs hold for any instantiation.

Module 2: UTXO transitions and cost model (`UTXOTransitions.v`). This module mechanizes PO-5 and PO-7. The UTXO set is modeled as an association list with lookup, remove, and add operations. The transition function δ_{Tx} is defined as removal of spent outpoints followed by addition of new outpoints with fresh identifiers. Properties proved:

- **PO-5 (Transition determinism):** δ_{Tx} is a pure function: identical inputs produce identical outputs. Machine-checked as `delta_tx_deterministic_ext`. Additionally, `delta_tx_preserves_no_double` proves that the $\mathsf{NoDoubleSpend}$ invariant is preserved across transitions.
- **PO-7 (Cost boundedness):** $\mathsf{Cost}(tx) \leq \alpha \cdot \mathsf{weight}(tx)$ with $\alpha = 1$. Machine-checked as `cost_bounded_by_weight`. The stronger result `cost_equals_weight` proves exact equality: $\mathsf{Cost}(tx) = \mathsf{weight}(tx)$ for the SegWit witness discount model, with cost constants matching the implementation (input overhead 144 WU, base overhead 40 WU, per-output 164 WU).

Module 3: varint axiom discharge with Bitcoin compact-size encoding (`VarintConcrete.v`). This module provides a concrete implementation of Bitcoin's compact-size varint encoding — the actual multi-byte format used in the Rust implementation — and proves that all 6 varint axioms from `SpendPredPQ.v` are satisfied. The encoding covers two cases: $n \leq 252$ maps to $[n]$ (1 byte);

$253 \leq n \leq 65535$ maps to [253; n mod 256; $n/256$] (3 bytes, little-endian u16). The decoding function enforces canonicality: it rejects non-minimal encodings (e.g., using the 3-byte prefix for values that fit in 1 byte). The key proof technique for canonicality (Axiom 5) uses Nat.Div0.mod_add and Nat.div_add to show that $(lo + hi \cdot 256) \bmod 256 = lo$ and $(lo + hi \cdot 256)/256 = hi$ when $lo < 256$, establishing that the decoded prefix equals the canonical encoding. The proofs are packaged as a `VarintAxioms` record. The module also proves concrete witness canonicality and parse injectivity for the bounded parser/serializer: every accepted concrete witness is exactly the canonical serialization of its parsed public key and signature, and two accepted witnesses with the same parsed components are byte-identical. It further proves that the protocol witness cap used by the Rust implementation, `MAX_WITNESS_SIZE`=16,000, remains within the modeled u16 varint domain, that capped parsed or serialized concrete witnesses have public-key and signature component lengths inside that domain, and that the consensus-domain parser equals the byte-level parser below the cap while rejecting witnesses above the cap. Additionally, the extraction pipeline produces **seven golden witness vectors** whose witness bytes are generated by a Coq-extracted serializer and verified byte-for-byte against the Rust implementation (including the ML-DSA-44 public key length 1,312 and signature length 2,420), exhaustively compares the Coq-extracted varint encoder/decoder against the deployed Rust functions over every modeled value $0, \ldots, 65535$ plus canonical rejection cases, and compares the Coq-extracted single-signature witness serializer/parser/consensus-domain parser/canonicality behavior and parser decision traces against the Rust functions over a deterministic boundary and malformed-input refinement matrix plus 111,111 symbolic bounded witnesses over modeled-domain critical bytes. On the Rust side, the parser is factored through an allocation-free witness layout function used by the public parser, consensus parser, trace hook, and canonicality predicates; five Kani harnesses verify bounded symbolic source-level alignment between those deployed Rust functions, including rejection before parsing for oversized consensus witnesses. The CI pipeline additionally builds the refinement examples as optimized release binaries, executes those compiled artifacts, compares their outputs against the Coq-extracted golden, varint-refinement, and witness-refinement summaries, and emits a certificate containing Rust toolchain metadata plus source, lockfile, binary, and generated-output hashes. This addresses PO-8 at the bounded encoding level: the Coq model uses the same single-byte and `0xFD`/u16 compact-size formats as the Rust code for the consensus-valid witness subset, while the Rust-only `0xFE` and `0xFF` ranges remain outside the current Coq proof boundary for general-purpose CompactSize and are rejected by the executable consensus-domain witness parser.

10.3 Proof obligations: status

Table 2 summarizes the status of each proof obligation.

10.4 Mechanizing Script/witness semantics

Script semantics and witness parsing demand mechanization to avoid consensus bugs. The Coq development covers the PQ spend predicate with varint-based witness encoding (PO-1, PO-2, PO-3), the UTXO transition function with no-double-spend preservation (PO-5), and the cost model with exact weight equality (PO-7). The varint axiom system is proved consistent by a concrete implementation (`VarintConcrete.v`). PO-4 (sighash commitment) is proved for the Coq Sighash v2 model under a SHA-256 collision-resistance axiom and supported at the implementation layer by transcript refinement and executable evidence: a Coq-extracted `sighash_preimage_from_hashes` function is compared against Rust's deployed outpoint serialization, output serialization, spent-output serialization, and final preimage assembly, including release-binary validation; a `verify_sighash_commitment_property`

ID	Property	Status	Artifact
PO-1	Spend predicate totality	**Verified**	Coq (`SpendPredPQ.v`)
PO-2	Spend predicate determinism	**Verified**	Coq (`SpendPredPQ.v`)
PO-3	Witness parsing canonicality	**Verified (strengthened)**	Coq (`SpendPredPQ.v`)
PO-4	Sighash commitment	**Verified model + transcript evidence**	Coq model + extracted tr
PO-5	Transition determinism	**Verified**	Coq (`UTXOTransitions.v`
PO-6	Invariant preservation	**Model-checked**	TLA+/TLC (2 models)
PO-7	Cost boundedness	**Verified**	Coq (`UTXOTransitions.v`
PO-8	Implementation correspondence	**Bounded + source + binary evidence**	Coq extraction + Kani +

Table 2: Status of proof obligations. "Verified" = machine-checked proof in Coq. "Verified (strengthened)" = machine-checked proof that is strictly stronger than the original obligation (full witness identity rather than region matching). "Model-checked" = exhaustively verified by TLC over finite state space. "Executable evidence" = property-based tests verifying the property across hundreds of randomly generated inputs. "Verified model + transcript evidence" = machine-checked proof for the Coq model under explicit cryptographic axioms, plus Coq-extracted vs Rust transcript/preimage refinement and executable tests of the concrete implementation. "Bounded + source + binary evidence" = the concrete Coq varint model covers the single-byte and `0xFD`/u16 compact-size ranges; concrete parser/serializer canonicality and parse injectivity are machine-checked for that bounded model; the protocol witness-size cap is proved/checked to remain within that domain; Coq-extracted varint encode/decode behavior is exhaustively compared against Rust over all modeled values; Coq-extracted witness serialize/parse/consensus-domain parse/canonicality behavior and parser decision traces are compared against Rust over a deterministic boundary/rejection matrix and a symbolic bounded state-space; witness bytes generated by the Coq-extracted serializer match the Rust implementation byte-for-byte on seven golden vectors; all six varint axioms are proved for that bounded model; 28 Rust tests exercise the corresponding implementation properties; five Kani harnesses verify source-level bounded parser/layout/canonicality alignment over symbolic Rust inputs; and optimized release refinement binaries are executed and required to reproduce the Coq-extracted summaries with source/binary hashes recorded. Compiler correctness remains outside the artifact boundary.

function programmatically checks injectivity, cross-input separation, and field coverage for consensus-encodable transactions; and 9 property-based tests exercise these properties across hundreds of randomly generated transactions with BIP 340-style tagged hashes for domain separation. The Coq statement is deliberately scoped to well-formed transactions, spent outputs, and u32 input indices with fixed-width consensus fields, and is phrased over consensus-semantic input outpoints rather than full witness-bearing input records, matching the fact that sighash excludes witness bytes. The remaining PO-4 boundary is SHA-256 primitive correctness/collision resistance and compiler/toolchain correctness, not deterministic transcript serialization. The current evidence for PO-8 consists of (i) all 6 varint axioms proved for a concrete bounded encoding in Coq covering the single-byte and `0xFD`/u16 CompactSize cases, (ii) direct Coq proofs that the bounded concrete parser is canonical and injective on the accepting domain, (iii) direct Coq proofs that the consensus-domain parser accepts exactly the byte-level parser below the cap, rejects above the cap, and only accepts canonical modeled-domain witnesses, (iv) exhaustive Coq-extracted vs Rust varint refinement over all modeled values $0, \ldots, 65535$ plus canonical rejection cases, (v) Coq and Rust guards showing that the consensus witness cap remains inside that domain, including an executable Rust consensus-domain parser used by the spend predicate, (vi) a Coq-extracted vs Rust

witness-level refinement matrix covering serialization, parsing, consensus-domain parsing, canonicality behavior, and parser decision traces over deterministic length-boundary and malformed-input representatives plus 111,111 symbolic bounded witnesses, (vii) seven golden witness vectors generated through the Coq-extracted serializer and verified byte-for-byte against the Rust implementation (including ML-DSA-44 key/signature lengths), (viii) five Kani harnesses that verify source-level bounded parser/layout/canonicality alignment over symbolic Rust byte arrays, (ix) release-binary translation validation against the Coq-extracted summaries with toolchain/source/binary hashes recorded, (x) 28 Rust tests that exercise the corresponding implementation properties, and (xi) 3 adversarial boundary tests that exercise the security properties from Theorems 4 and 5 (commitment second-preimage resistance, cross-input replay resistance, cross-transaction replay resistance). Full compiler-correctness closure remains a separate artifact class. For PO-4, the transcript/preimage serialization layer is extracted and compared against Rust, while SHA-256 itself remains an explicit cryptographic primitive boundary. For PO-8, the consensus-valid witness subset is bounded by u16, varint-level refinement is exhaustive, witness-level extracted-function plus consensus-domain operational-trace refinement is checked by CI, the Rust source parser is bounded-verified by Kani, and the release binaries are translation-validated over the modeled PO-8 domains. Extending the Coq model to `0xFE`/`0xFF` remains necessary only for a general-purpose CompactSize verification claim or if the witness cap is raised above 65,535 bytes.

11 Engineering Reality and Parameterization

Post-quantum signatures are larger and often more expensive to verify. This impacts:

- block bandwidth (*weight*);
- signature verification throughput;
- mempool policy and fee market dynamics;
- hardware requirements for full nodes.

These are not reasons to avoid quantum safety; they are constraints that must be internalized into the cost model (Cost) and the activation plan (migration windows, enforcement height, and wallet operational invariants).

11.1 Witness size and verification cost comparison

Table 3 compares the witness footprint of current and candidate PQ schemes. All sizes are approximate and reflect single-signature spending; multisig and threshold constructions multiply accordingly.

Weight budget impact. A standard SegWit block has a weight limit of 4 MWU (4,000,000 weight units), with witness data discounted at 1 WU per byte. A single ML-DSA-44 witness ($\sim$3.7 KB) consumes roughly 37$\times$ the weight of a Schnorr witness ($\sim$98 B). This reduces the maximum number of single-input, single-output transactions per block from $\sim$10,000 (Schnorr) to $\sim$270 (ML-DSA-44) or $\sim$120 (SLH-DSA-128s). Batch verification techniques, witness aggregation, or increased block weight limits are engineering levers that affect throughput but do not change the security analysis.

Scheme	Public key (bytes)	Signature (bytes)	Witness total (bytes)	Security (quantum bits)
ECDSA (secp256k1)	33	72	∼107	0 (broken)
Schnorr (BIP 340)	32	64	∼98	0 (broken)
ML-DSA-44 (FIPS 204)	1,312	2,420	∼3,734	128
ML-DSA-65 (FIPS 204)	1,952	3,309	∼5,263	192
SLH-DSA-128s (FIPS 205)	32	7,856	∼7,890	128
SLH-DSA-128f (FIPS 205)	32	17,088	∼17,122	128

Table 3: Witness sizes for current and post-quantum signature schemes. "Witness total" includes public key, signature, and minimal encoding overhead. Quantum security of 0 means the scheme is broken under Shor. Sizes for ML-DSA and SLH-DSA are from NIST FIPS 204 and FIPS 205 [13, 14].

Verification throughput. ML-DSA verification is comparable to Ed25519 in speed (∼10,000 verifications/second on commodity hardware). SLH-DSA verification is slower (∼1,000–3,000/s for the "f" variants, faster for "s" variants). The cost function $\mathsf{Cost}(tx)$ must reflect these differences to prevent DoS via blocks filled with expensive-to-verify witnesses.

12 Related Work

Quantum attacks on Bitcoin. Aggarwal et al. [1] provide the first systematic analysis of quantum threats to Bitcoin, quantifying the advantage of a quantum miner (via Grover) and the feasibility of key recovery (via Shor) under various timing assumptions. Their treatment is primarily quantitative and attack-oriented; it does not formalize consensus as a state machine, define authorization integrity as a game, or address migration. Our work complements theirs by providing the formal framework in which their attack scenarios can be stated as theorem violations.

Bitcoin Backbone protocol. Garay, Kiayias, and Leonardos [7] formalize the safety and liveness properties of Nakamoto consensus (common prefix, chain quality, chain growth) under synchronous networks with bounded adversarial mining power. Pass, Seeman, and Shelat [15] extend this to asynchronous settings. These results provide the network-level foundation we rely on (Section 8), but they do not address the authorization layer: their model assumes honest transaction validity, whereas our contribution is precisely to formalize what "valid authorization" means under QPT adversaries and to prove that PQ predicates preserve it.

Post-quantum signature standardization. NIST has standardized ML-DSA (FIPS 204, lattice-based) [13] and SLH-DSA (FIPS 205, hash-based) [14] as post-quantum signature schemes. Chen et al. [6] provide the initial NIST report motivating the transition. Our construction is parametric in the choice of $\mathsf{PQC\text{-}Sig}$; the security theorems hold for any scheme satisfying EUF-CMA under QPT adversaries.

Quantum random oracle model. Boneh et al. [5] initiated the study of cryptographic security when adversaries can query random oracles in superposition. Zhandry [24] developed techniques for quantum-accessible random functions. Boneh and Zhandry [4] proved that certain signature schemes remain secure under quantum chosen-message attacks. We discuss the QROM implications for our reduction in Section 7.4.

Bitcoin-specific PQ proposals. Heilman et al. [10] explore using `OP_CAT` to enable Lamport-like signatures within Bitcoin Script, providing a script-level path to quantum resistance without new opcodes. BIP-360 [2] proposes a "Pay to Quantum Resistant Hash" output type. These proposals address the construction layer (how to encode PQ witnesses in Bitcoin) but do not provide the formal consensus model, game-based security definitions, invariant preservation proofs, or migration formalization that we contribute.

Shor's algorithm for elliptic curves. Proos and Zalka [16] analyze the concrete quantum circuit complexity of Shor's algorithm applied to elliptic curve groups. Roetteler et al. [17] provide refined resource estimates, suggesting that ∼2,500 logical qubits suffice for 256-bit curves. These estimates inform the timeline urgency of migration but do not affect our formal results, which assume Shor is available as a QPT oracle.

13 Conclusion

Protocol-level quantum safety is a global safety property of the Bitcoin consensus state machine. It cannot be achieved by adding a "post-quantum opcode" while leaving secp256k1 spend paths reachable. A credible design must (i) specify UTXO transitions and validation deterministically, (ii) define authorization integrity as a QPT game, (iii) provide a tight reduction from unauthorized spends to PQ signature security and hash binding via explicit game-hopping, (iv) prove that consensus invariants are preserved across all valid transitions, (v) exclude witness replay and cross-input attacks via sighash commitment properties, (vi) model network execution as formal traces and prove that network-level adversarial capabilities do not bypass PQ authorization, (vii) address the quantum random oracle model, (viii) formalize migration with monotonicity guarantees and explicitly choose how to handle unmigrated (including lost-key) outputs, and (ix) discharge proof obligations via mechanized verification.

This paper addresses all nine requirements. The formal artifacts — TLC model checking of UTXO transitions (two models: single-input with 492 states and multi-input with 58,237 states and value conservation; zero structural invariant violations; concrete migration dilemma counterexample in both models), Coq-mechanized proofs of spend predicate totality, determinism, and parse canonicality (strengthened to full witness identity via axiomatized varint model with concrete bounded compact-size discharge, consensus-size guard, golden test vectors, exhaustive varint refinement, and witness-level consensus-domain operational-trace refinement matrices), Kani source-level bounded verification of the deployed Rust witness parser layout/canonicality relation, release-binary translation validation against the Coq-extracted PO-8 summaries, Coq-mechanized proofs of the Sighash v2 commitment model under a SHA-256 collision-resistance axiom plus Coq-extracted vs Rust transcript/preimage refinement for PO-4, transition determinism (PO-5), and cost boundedness with exact weight equality (PO-7), and a reference implementation in Rust with FIPS 204 ML-DSA-44 signatures, 34 property-based tests including 9 sighash commitment properties, 28 PO-8 correspondence tests with bounded golden vectors, a Rust consensus-size guard, and 3 adversarial boundary tests — provide machine-checked and empirically validated evidence while leaving cryptographic primitive implementation correctness and compiler correctness as explicit implementation-correspondence boundaries.

References

[1] Divesh Aggarwal, Gavin K. Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them. https://arxiv.org/abs/1710.10377, 2017. Accessed 2026-04-09.

[2] Hunter Beast. Bip-360: Qubit – pay to quantum resistant hash. https://github.com/bitcoin/bips/blob/master/bip-0360.mediawiki, 2024. Accessed 2026-04-09.

[3] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology – EUROCRYPT*, pages 409–426. Springer, 2006. doi: 10.1007/11761679_25.

[4] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. https://eprint.iacr.org/2013/088, 2013. Accessed 2026-04-09.

[5] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology – ASIACRYPT*, pages 41–69. Springer, 2011. doi: 10.1007/978-3-642-25385-0_3.

[6] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. In *NIST Internal Report 8105*, 2016. Accessed 2026-04-09.

[7] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology – EUROCRYPT*, pages 281–310. Springer, 2015. doi: 10.1007/978-3-662-46803-6_10.

[8] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. doi: 10.1137/0217017.

[9] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 212–219. ACM, 1996. doi: 10.1145/237814.237866.

[10] Ethan Heilman, Tadge Dryja, Madars Virza, and Andrew Poelstra. Op_cat and quantum-resistant bitcoin. https://arxiv.org/abs/2401.15768, 2024. Accessed 2026-04-09.

[11] Eric Lombrozo, Johnson Lau, and Pieter Wuille. Bip141: Segregated witness (consensus layer). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki, 2016. Accessed 2026-04-09.

[12] National Institute of Standards and Technology. Fips 186-5: Digital signature standard (dss). https://csrc.nist.gov/publications/detail/fips/186/5/final, 2023. Accessed 2026-04-09.

[13] National Institute of Standards and Technology. Fips 204: Module-lattice-based digital signature standard. https://csrc.nist.gov/publications/detail/fips/204/final, 2024. Accessed 2026-04-09.

[14] National Institute of Standards and Technology. Fips 205: Stateless hash-based digital signature standard. https://csrc.nist.gov/publications/detail/fips/205/final, 2024. Accessed 2026-04-09.

[15] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Advances in Cryptology – EUROCRYPT*, pages 643–673. Springer, 2017. doi: 10.1007/978-3-319-56614-6_22.

[16] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4):317–344, 2003. arXiv:quant-ph/0301141.

[17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *Advances in Cryptology – ASIACRYPT*, pages 241–270. Springer, 2017. doi: 10.1007/978-3-319-70697-9_9.

[18] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 124–134. IEEE, 1994. doi: 10.1109/SFCS.1994.365700.

[19] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. In *Cryptology ePrint Archive, Report 2004/332*, 2004. Accessed 2026-04-09.

[20] Pieter Wuille, Jonas Nick, and Tim Ruffing. Bip340: Schnorr signatures for secp256k1. https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki, 2020. Accessed 2026-04-09.

[21] Pieter Wuille, Andrew Poelstra, Jonas Nick, and Tim Ruffing. Bip341: Taproot: Segwit version 1 spending rules. https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki, 2021. Accessed 2026-04-09.

[22] Pieter Wuille, Andrew Poelstra, Jonas Nick, and Tim Ruffing. Bip342: Validation of taproot scripts. https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki, 2021. Accessed 2026-04-09.

[23] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4): 2746–2751, 1999. doi: 10.1103/PhysRevA.60.2746.

[24] Mark Zhandry. How to construct quantum random functions. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 679–688. IEEE, 2012. doi: 10.1109/FOCS.2012.37.