QryptCoin: A Peer-to-Peer Electronic Cash System with Consensus-Enforced Post-Quantum Authorization

Mikhael A.
info@qryptcoin.org

Abstract

QryptCoin is a peer-to-peer electronic cash system combining proof-of-work ordering with post-quantum transaction authorization using ML-DSA-65. The protocol introduces *consensus-enforced one-time public keys*: each ML-DSA-65 public key may be revealed and used at most once on the active chain, with any reuse rendering the containing block invalid. Outputs commit to public keys by hash and reveal them only at spend time via a canonical encoding scheme. A non-interactive stealth payment mechanism enables reusable payment identifiers without requiring recipient interaction, using ML-KEM-768 key encapsulation for post-quantum key agreement. This design minimizes long-term cryptographic exposure, eliminates key-reuse failure modes, and provides quantum resistance while maintaining compatibility with existing SHA-256 mining infrastructure.

1 Introduction

Decentralized electronic cash systems must enable direct value transfer over open networks without trusted intermediaries. Such systems face two fundamental challenges: preventing double spending through global transaction ordering, and securing authorization mechanisms against both current and future cryptanalytic advances.

Existing cryptocurrency protocols face two persistent sources of failure: (i) ambiguous consensus-critical encodings that cause implementation divergence, and (ii) long-lived authorization keys vulnerable to operational reuse and future quantum attacks. While current public-key cryptography remains secure against classical computers, the advent of cryptographically-relevant quantum computers poses an existential threat to systems relying solely on elliptic curve signatures.

QryptCoin addresses these challenges through two deliberate design choices:

1. **Post-quantum authorization with commitment-then-reveal:** Outputs commit to ML-DSA-65 public keys by hash; keys are revealed only at spend time, minimizing exposure windows.

2. **Consensus-enforced one-time keys:** Public-key reuse is treated as a consensus failure, not merely a wallet policy recommendation. Each revealed key may appear at most once on the active chain.

This paper describes the protocol mechanisms, security properties, and design rationale. Implementation constants and byte layouts are deferred to supplementary specifications.

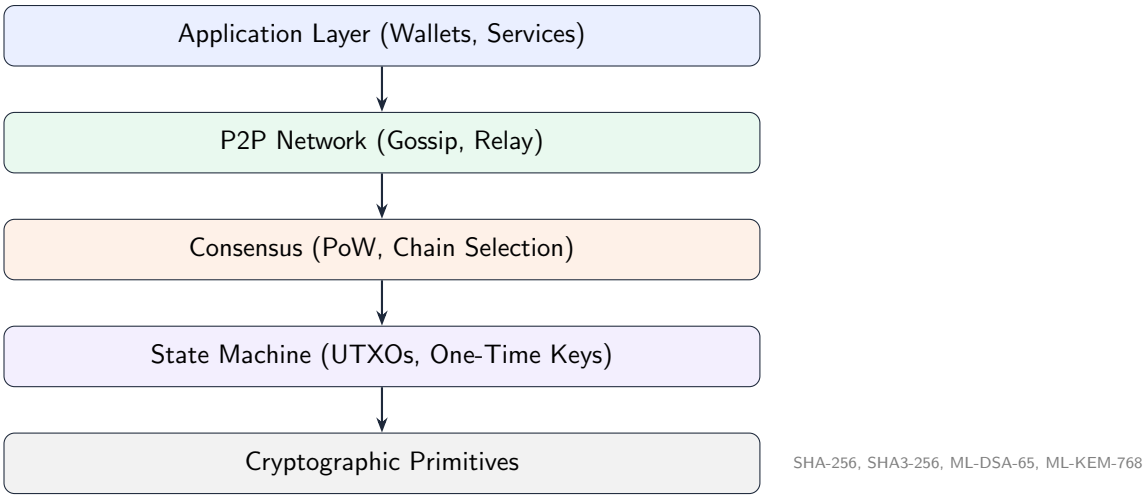


Figure 1: QryptCoin protocol stack. The state machine maintains both UTXOs and the global one-time key set as consensus-critical state.

2 System Architecture

QryptCoin employs a layered architecture (Figure 1) with clearly separated concerns:

Cryptographic Layer. Two hash families provide domain separation: $H_2(x)$ denotes SHA-256 (used for proof-of-work), while $H_3(x)$ denotes SHA3-256 (used for commitments and signatures). Transaction authorization uses ML-DSA-65 (FIPS 204) exclusively.

State Machine. Ledger state comprises the UTXO set and a global revealed-key set $\mathcal{S}$. Both are updated atomically during block connection and rolled back during reorganizations.

Consensus Layer. Proof-of-work provides probabilistic finality. Nodes select the valid chain with greatest cumulative work.

Network Layer. Peer-to-peer gossip disseminates transactions and blocks. Optional post-quantum transport encryption (ML-KEM-768 + ChaCha20-Poly1305) provides confidentiality against network adversaries.

2.1 Units and Primitives

The native unit is the QRY, with 1 QRY $= 10^8$ Miks. All consensus amounts are denominated in Miks.

Domain separation for signature digests uses literal ASCII tags in the hashed preimage:

$$\text{sighash}_i = H_3(\texttt{"QRY-SIGHASH-V1"} \parallel M_i)$$

3 Transaction Model

3.1 UTXO Structure

QryptCoin uses an unspent transaction output (UTXO) model. Each output contains a value in Miks and a *locking descriptor* specifying authorization conditions. An input references a prior

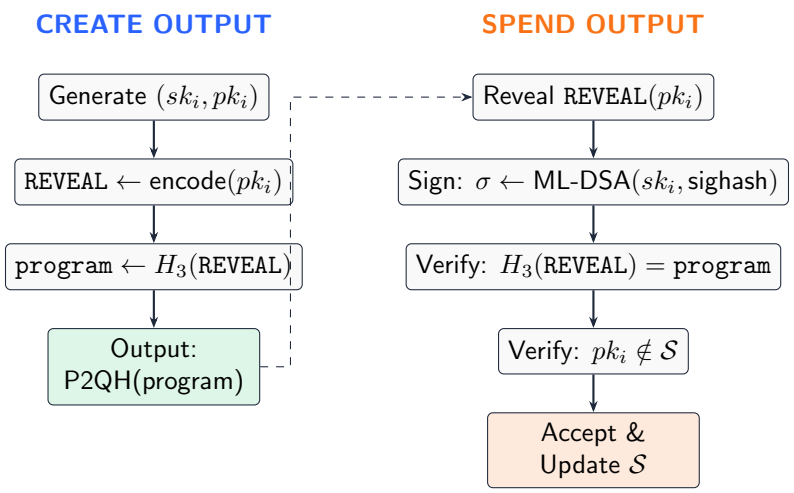


Figure 2: P2QH commitment-then-reveal protocol. The public key is revealed only at spend time and must not already exist in the global set $\mathcal{S}$.

output by outpoint (`txid`, `vout`) and provides witness data satisfying the locking condition.

3.2 Pay-to-Quantum-Hash (P2QH)

The primary output type is *pay-to-quantum-hash*, implementing a commitment-then-reveal scheme. Outputs commit to a 32-byte program:

$$\texttt{program} = H_3(\texttt{REVEAL\_V1})$$

where `REVEAL_V1` is a canonical encoding of the ML-DSA public key. The public key remains hidden until spend time, minimizing exposure to quantum attacks.

Witness Structure. For a P2QH spend, the witness contains exactly two items: (1) `REVEAL_V1`—the canonical public-key reveal, and (2) the ML-DSA signature over a domain-separated digest.

Canonical Encoding. The `REVEAL_V1` format is fixed-width with no optional fields or trailing bytes:

Field	Size
version	1 byte (must be `0x01`)
algo_id	1 byte (must be `0x01`)
params_id	1 byte (must be `0x01`)
reserved	2 bytes (must be `0x0000`)
pk_len	2 bytes (must be 1952)
pk_bytes	1952 bytes

Non-canonical encodings are rejected at consensus level. Nodes do not normalize malformed encodings; any deviation from the canonical format renders the transaction invalid.

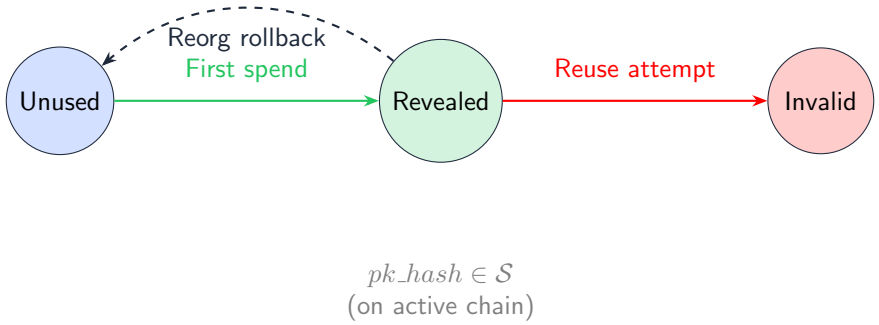


Figure 3: Public key state transitions. Each key transitions from unused to revealed exactly once. Reuse attempts cause transaction (and block) rejection.

3.3 Consensus-Enforced One-Time Keys

Key Innovation. QryptCoin enforces one-time public keys at the consensus level—not as wallet policy, but as a validity rule.

Let `pk_bytes` be the ML-DSA public key extracted from `REVEAL_V1`. Define:

$$\texttt{pk\_hash} = H_3(\texttt{pk\_bytes})$$

Nodes maintain a consensus-critical set $\mathcal{S}$ of all `pk_hash` values revealed on the active chain. Transaction validation requires:

1. The input reveals a canonical `REVEAL_V1`
2. The revealed `pk_hash` $\notin \mathcal{S}$
3. The ML-DSA signature is valid

Upon block acceptance, new `pk_hash` values are inserted into $\mathcal{S}$. Reorganizations deterministically apply insertions and removals. The revealed-key set $\mathcal{S}$ grows monotonically with transaction activity on the active chain and is rolled back only during chain reorganizations.

Security Rationale. This design provides three benefits:

- *Reduced exposure window:* Public keys exist on-chain only after their associated funds are spent
- *Quantum resistance:* Even if ML-DSA is partially weakened, attackers cannot exploit revealed keys for future spends
- *Eliminated operational failures:* Address reuse—a common source of privacy and security failures—becomes impossible

4 Block Structure and Consensus

4.1 Block Format

Each block contains an 80-byte header and an ordered list of transactions beginning with a coinbase transaction.

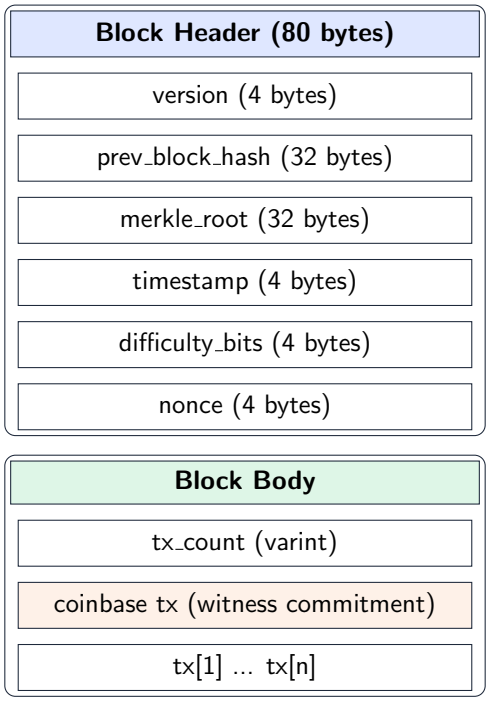


Figure 4: Block structure. The header commits to the transaction Merkle root; the coinbase transaction contains the witness commitment.

Proof-of-Work. Blocks are valid only if:

$$H_2^2(\text{header}_{80}) \leq T$$

where T is the target encoded in `difficulty_bits`. Using double SHA-256 maintains compatibility with existing mining hardware.

Witness Commitment. The coinbase transaction embeds a fixed-format witness commitment binding witness data to block validity:

$$\texttt{QRYW} \parallel \texttt{0x01} \parallel H_3(\text{witness\_merkle\_root})$$

This prevents witness malleability without changing the header hash.

4.2 Chain Selection

Nodes select the valid chain with greatest cumulative proof-of-work. Settlement is probabilistic: for an adversary controlling fraction $q < 1/2$ of hashpower, the probability of reorganizing k

confirmations is bounded by:

$$P_{\text{reorg}}(k) \leq \left(\frac{q}{1-q}\right)^k$$

k	$q = 0.10$	$q = 0.20$	$q = 0.30$	$q = 0.40$
1	0.111	0.250	0.429	0.667
6	1.9×10^{-6}	2.4×10^{-4}	6.2×10^{-3}	0.088
10	2.9×10^{-10}	9.5×10^{-7}	9.6×10^{-5}	0.017

Table 1: Reorganization probability upper bounds for various adversary hashpower fractions and confirmation depths.

4.3 Difficulty Adjustment

Difficulty adjusts every 2016 blocks targeting 600-second mean block spacing. The adjustment uses observed timestamps with bounded clamping to prevent abrupt swings. Specifically, the observed timespan is clamped to the range $[T/4, 4T]$ before computing the new target, where T is the target timespan.

4.4 Fork Resolution

When forks occur, nodes resolve by selecting the chain with greater cumulative work. State transitions (UTXO set and revealed-key set $\mathcal{S}$) are applied and rolled back deterministically during block connect/disconnect operations.

5 Network Protocol

5.1 Message Propagation

Transactions and blocks propagate via inventory announcements. Nodes independently validate all received data before relay, ensuring that invalid transactions and blocks do not propagate.

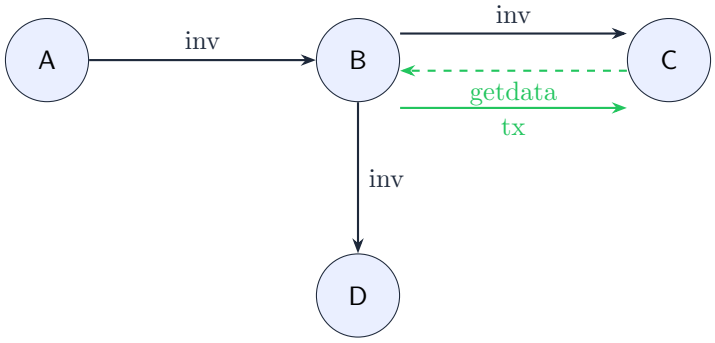


Figure 5: Inventory-based gossip protocol. Nodes announce data availability; peers request items they lack.

5.2 Authenticated Transport

Peers may establish encrypted channels using ML-KEM key establishment and ML-DSA transcript authentication, with subsequent messages protected by ChaCha20-Poly1305. This improves confidentiality against network adversaries but does not affect consensus—nodes must still validate all data locally.

5.3 Mempool and Relay Policy

Commit-delay-reveal. To mitigate front-running of revealed public keys, nodes may implement commit-delay policies requiring transaction commitment to mempool before reveal propagation. This is a policy-layer protection, not consensus.

Fee floor decay. The mempool enforces a dynamic minimum fee that decays over time, preventing spam while allowing legitimate low-fee transactions during quiet periods.

6 Economics and Incentives

6.1 Monetary Policy

Block producers receive a subsidy plus transaction fees. The subsidy begins at 50 QRY and halves every 210,000 blocks, yielding a maximum supply of 21,000,000 QRY.

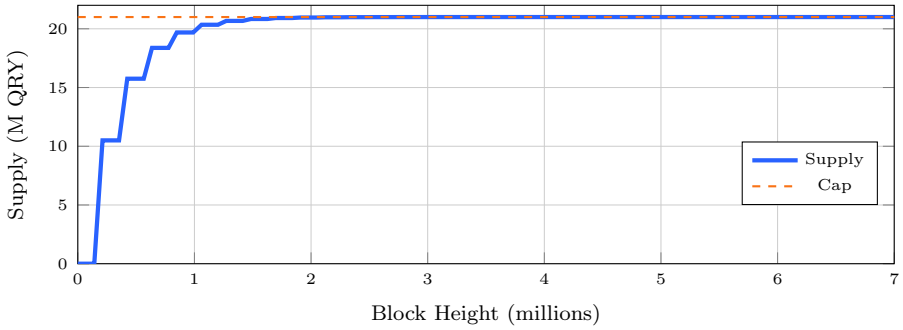


Figure 6: Asymptotic supply curve approaching 21M QRY maximum.

6.2 Transaction Fees

Fees are denominated in Miks per virtual byte, where virtual bytes derive from transaction weight:

$$\text{weight} = 4 \times \text{base\_bytes} + \text{witness\_bytes}$$

Post-quantum witnesses are larger than classical signatures, increasing fees under fee-market conditions. This creates natural economic pressure for efficient witness construction.

6.3 Parameter Rationale

- *21M supply cap:* A deliberately finite ceiling chosen for predictable scarcity; fixed at genesis and enforced by consensus.
- *10-minute blocks:* Balances orphan rate against confirmation latency given post-quantum transaction sizes.
- *100-block maturity:* Prevents spending rewards from orphaned chains.
- *8M weight limit:* Accommodates approximately 1,500 single-input post-quantum transactions per block.

7 Wallet Architecture

7.1 Key Derivation

The reference wallet uses a 24-word BIP-39 compatible mnemonic. The derivation pipeline produces deterministic ML-DSA keypairs:

1. Mnemonic $\rightarrow$ PBKDF2-HMAC-SHA512 $\rightarrow$ 64-byte seed
2. 64-byte seed $\rightarrow H_3 \rightarrow$ 32-byte master seed
3. Per-index: $H_3(\texttt{"QRY-MLDSA-KEYGEN-V1"} \parallel \text{master} \parallel i)$ seeds SHAKE256-XOF
4. XOF output feeds ML-DSA.KeyGen deterministically

7.2 Payment Codes (Non-Interactive Stealth Payments)

Because consensus enforces one-time keys, addresses cannot be safely reused. *Payment Codes* provide reusable identifiers that allow senders to derive fresh one-time addresses without any interaction with the recipient. This preserves usability for donations, recurring payments, and offline recipients without relaxing consensus rules.

7.2.1 Payment Code Structure

A Payment Code is a compact, reusable identifier encoded as a Bech32m string with prefix `qrypay1`. The payload encodes:

Field	Description
`scan_pubkey`	ML-KEM-768 encapsulation key ek (1184 bytes)
`spend_pubkey`	ML-DSA-65 public key (1952 bytes)
`net_id`	4-byte network identifier

7.2.2 Protocol

When a sender wishes to pay a Payment Code recipient:

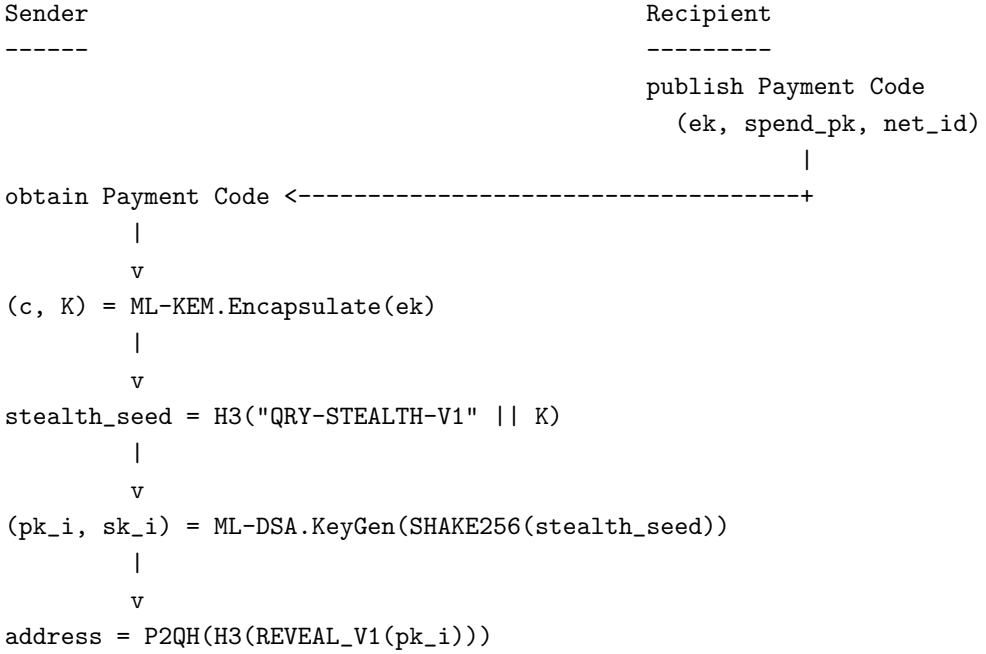
1. **Encapsulation.** The sender performs ML-KEM-768 encapsulation against the recipient's scan public key (ek), producing a shared secret K and ciphertext c: $(c, K) \leftarrow \text{ML-KEM.Encaps}(ek)$.
2. **Key Derivation.** A domain-separated stealth seed is derived:

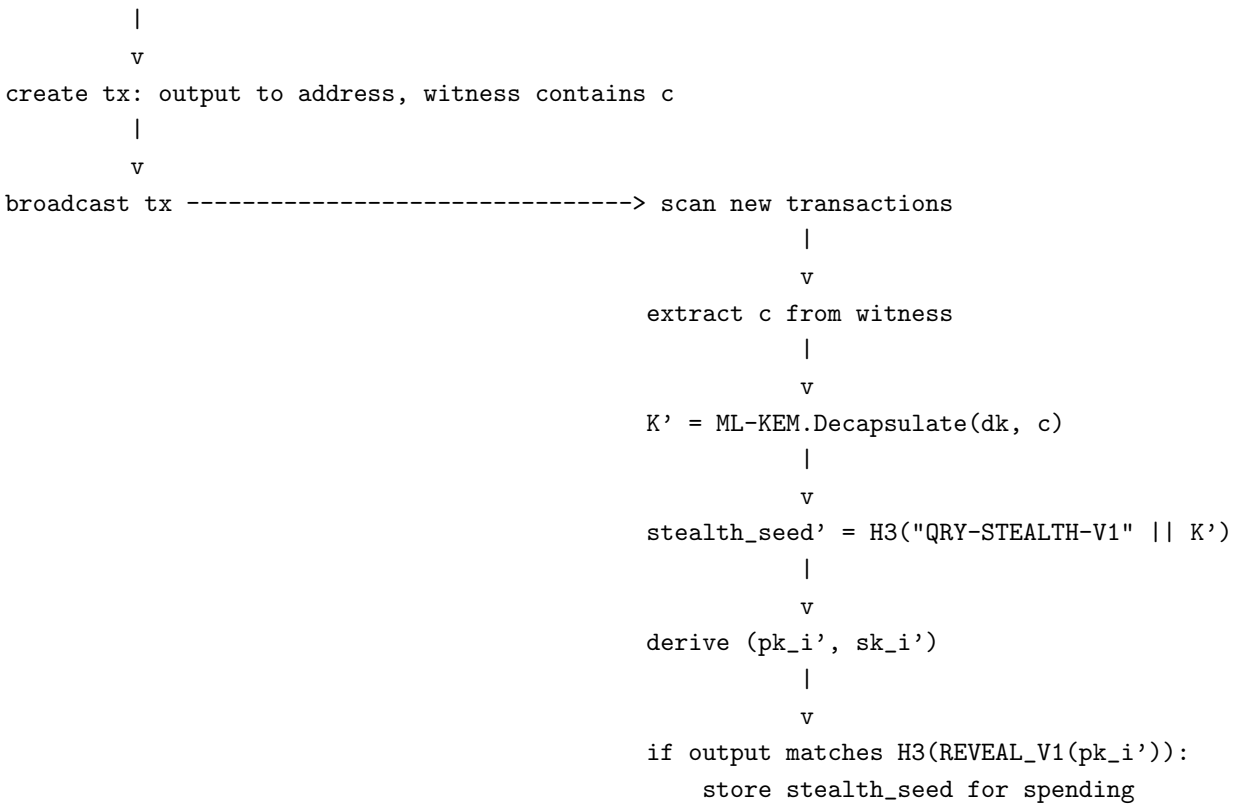
$$\text{stealth\_seed} = H_3(\texttt{"QRY-STEALTH-V1"} \parallel K)$$

3. **One-Time Key Generation.** The stealth seed is expanded via SHAKE256-XOF to produce a fresh ML-DSA-65 keypair unique to this transaction.
4. **Address Construction.** The derived public key is encoded as a standard P2QH address using the commitment-then-reveal scheme.
5. **Witness Payload.** The ciphertext c is embedded in the transaction witness, enabling the recipient to recover the shared secret and derive the corresponding private key.
6. **Recipient Recovery.** The recipient scans transactions, decapsulates $K \leftarrow \text{ML-KEM.Decaps}(dk, c)$, rederives the stealth seed, and recovers spend authority.

The full sender and recipient flows are illustrated below.

Figure 7: Non-interactive stealth payment flow. The sender derives a one-time keypair using ML-KEM-768 key encapsulation without any interaction with the recipient.





7.2.3 Security Properties

Payment Codes provide the following security guarantees:

- *Forward secrecy:* Each transaction uses a fresh encapsulation. Compromise of past ciphertexts does not affect future payments.
- *Unlinkability:* Observers cannot link payments to the same Payment Code without the scan private key.
- *Post-quantum security:* Both the key agreement (ML-KEM-768) and signature (ML-DSA-65) schemes are NIST-standardized post-quantum algorithms.
- *One-time key compliance:* Derived keys satisfy the consensus-enforced one-time public key rule; each stealth output uses a unique keypair.

8 Security Analysis

8.1 Threat Model

QryptCoin assumes adversaries may: create conflicting transactions, delay or selectively drop messages, exploit encoding ambiguities, and attempt key theft. The protocol does not prevent majority hashpower attacks—an inherent limitation of proof-of-work.

8.2 Double-Spend Resistance

Double spending requires building an alternative chain with greater cumulative work. For $q < 1/2$, success probability decays exponentially with confirmation depth (Table 1).

8.3 Post-Quantum Security

Pre-spend protection. Outputs commit only to $H_3(\texttt{REVEAL})$; public keys remain hidden until spend time. A quantum adversary observing outputs cannot extract keys to forge spends.

Post-spend protection. Revealed keys cannot be reused by consensus rule. Even if ML-DSA is partially weakened, attackers cannot exploit revealed keys for new transactions.

Proof-of-work compatibility. SHA-256 mining remains secure: Grover's algorithm provides only quadratic speedup, addressable by difficulty adjustment.

8.4 Encoding Determinism

Strict canonical encodings eliminate cross-implementation divergence. Non-canonical `REVEAL_V1` payloads are rejected, preventing consensus splits from parsing ambiguities.

8.5 Key Compromise Limitations

One-time keys limit exposure duration but do not prevent theft under endpoint compromise. If a signing key is stolen before its output is spent, an attacker can produce a valid spend. This is mitigated by operational security, not protocol design.

9 Privacy Considerations

QryptCoin does not provide network-layer anonymity. The one-time key rule prevents revealed-key reuse but does not eliminate transaction graph analysis. Users requiring strong privacy should employ additional privacy-enhancing technologies.

10 Reclaiming Disk Space

A fully validating node does not need to retain all historical spent outputs to validate new blocks. Instead, nodes maintain a compact chainstate consisting of:

- The UTXO set (current spendable outputs)
- The revealed-public-key set $\mathcal{S}$ (keys already used on the active chain)
- A block index and headers needed for chain selection and reorganization

The reference implementation persists both the UTXO snapshot and the revealed-key snapshot atomically, binding them to the active tip. These snapshots accelerate restart and reindex workflows. Because the one-time key set is part of consensus state, it cannot be discarded without losing validation capability.

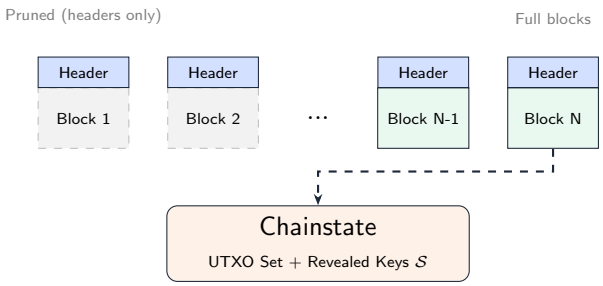


Figure 8: Nodes can prune old block bodies while retaining headers and chainstate. The chainstate includes both UTXOs and the one-time key set $\mathcal{S}$.

10.1 Scalability of the Revealed-Key Set

The set $\mathcal{S}$ grows monotonically with chain history. For long-term scalability:

Current implementation. The reference node uses an append-only LevelDB index keyed by pk_hash. At 0.25 TPS average, $\mathcal{S}$ grows by approximately 8M entries per year.

Sparse Merkle Tree roadmap. A future consensus upgrade may replace the flat set with a Sparse Merkle Tree (SMT), enabling:

- Compact membership proofs for light clients
- State snapshots with cryptographic commitments
- Potential pruning of deep history with SMT root anchors

11 Simplified Verification

Lightweight verification can be performed by maintaining only block headers and requesting Merkle proofs for specific transactions. Such clients can verify:

- Proof-of-work on headers
- Inclusion of a transaction in a block via Merkle proof over `txid`s

However, lightweight clients cannot independently enforce all consensus rules—notably full UTXO validation and the one-time public-key rule—without additional authenticated state. Simplified verification therefore requires trust in one or more full nodes for complete validation. Users with high-value transactions or stringent security requirements should operate full nodes.

12 Transaction Size and Throughput Analysis

Post-quantum signatures impose significant size overhead. A typical single-input P2QH spend requires:

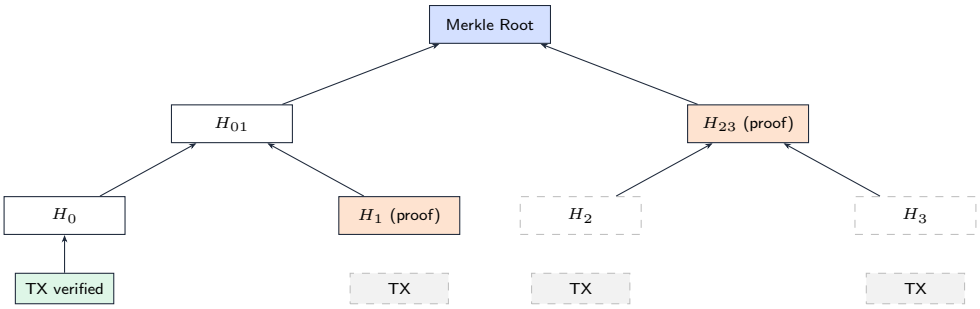


Figure 9: Simplified Payment Verification (SPV). A lightweight client verifies transaction inclusion using a Merkle proof (orange nodes) without downloading full blocks.

Component	Size
ML-DSA-65 public key	1,952 bytes
ML-DSA-65 signature	3,309 bytes
`REVEAL_V1` header	7 bytes
Total witness per input	$\approx$ 5,268 bytes

With a 1 MB block size cap and 10-minute target spacing, theoretical throughput for single-input transactions is approximately 0.25 TPS. QryptCoin is designed as a settlement layer where individual transactions represent high-value or batched settlements, not retail payment volume.

Mitigation strategies.

- *Output batching:* Aggregate multiple payments into single transactions.
- *Layer-2 protocols:* Payment channels and rollups for high-frequency transfers.
- *Future signature aggregation:* Potential consensus upgrades for witness compression.

13 Conclusion

QryptCoin demonstrates that post-quantum security and consensus-enforced key hygiene can be achieved without sacrificing the proven properties of proof-of-work ordering. By committing to public keys by hash, revealing them only at spend time, and enforcing one-time usage at the consensus level, QryptCoin eliminates a significant class of operational and cryptographic failures while providing quantum resistance. The non-interactive stealth payment mechanism further extends usability by enabling reusable payment identifiers without compromising the one-time key invariant.

The protocol prioritizes explicit rules, deterministic encodings, and minimal consensus surface area over maximal expressiveness. Future work may extend the authorization model to support threshold signatures and time-locked contracts while preserving the one-time key invariant.

References

1. NIST. *FIPS 204: Module-Lattice-Based Digital Signature Standard.* 2024.
2. NIST. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard.* 2024.
3. NIST. *FIPS 202: SHA-3 Standard.* 2015.
4. NIST. *FIPS 180-4: Secure Hash Standard.* 2015.
5. Merkle, R. *Protocols for Public Key Cryptosystems.* IEEE S&P, 1980.
6. Dwork, C. and Naor, M. *Pricing via Processing.* CRYPTO, 1992.
7. IETF. *RFC 8439: ChaCha20-Poly1305.* 2018.