Postera: A Post-Quantum Privacy-Preserving Cryptocurrency

Technical Whitepaper v1.0

Postera Development Team
https://postera.network

January 2026

Abstract

We present Postera, a cryptocurrency designed to provide strong privacy guarantees while remaining secure against both classical and quantum adversaries. Postera combines NIST-standardized post-quantum digital signatures (ML-DSA-65, FIPS 204) with zero-knowledge proofs (Groth16 on BN254) to achieve transaction privacy through a note-based UTXO model similar to Zcash's Sapling protocol. Unlike existing privacy coins, Postera's signature scheme provides security against attacks by quantum computers, addressing a critical vulnerability in current blockchain systems. This paper describes the cryptographic foundations, transaction model, consensus mechanism, and security properties of the Postera protocol.

Contents

1 Introduction

The advent of practical quantum computers poses an existential threat to the security of existing blockchain systems. Most cryptocurrencies, including Bitcoin and Ethereum, rely on elliptic curve cryptography (ECDSA, EdDSA) for digital signatures. Shor's algorithm, when run on a sufficiently powerful quantum computer, can break these schemes in polynomial time, potentially allowing an attacker to forge signatures and steal funds.

Simultaneously, financial privacy has become increasingly important as blockchain analytics companies develop sophisticated tools to track and deanonymize transactions. While privacy-focused cryptocurrencies like Zcash and Monero address this concern, they remain vulnerable to quantum attacks.

Postera addresses both challenges by combining:

1. **Post-Quantum Signatures**: ML-DSA-65 (CRYSTALS-Dilithium), a NIST-standardized lattice-based signature scheme providing 128-bit post-quantum security.
2. **Zero-Knowledge Proofs**: Groth16 proofs on the BN254 curve enabling private transactions where amounts and recipients are hidden.
3. **Note-Based Privacy Model**: A UTXO-like system using encrypted notes, nullifiers, and commitment trees to prevent double-spending while maintaining privacy.

2 Threat Model and Security Goals

2.1 Adversary Capabilities

We consider adversaries with the following capabilities:

- **Classical Adversary**: Polynomial-time bounded classical computer with access to all public blockchain data.
- **Quantum Adversary**: Access to a cryptographically-relevant quantum computer capable of running Shor's and Grover's algorithms.
- **Network Adversary**: Ability to observe, delay, or reorder network messages (but not break cryptographic primitives).

2.2 Security Goals

Postera aims to achieve the following security properties:

Definition 1 (Transaction Integrity)**.** *No adversary can create a valid transaction that spends funds they do not own, even with quantum computing capabilities.*

Definition 2 (Balance Privacy)**.** *An observer of the blockchain cannot determine the balance of any address without knowledge of the corresponding viewing key.*

Definition 3 (Transaction Privacy)**.** *An observer cannot link the sender and receiver of a transaction, nor determine the transaction amount.*

Definition 4 (Double-Spend Prevention)**.** *No adversary can spend the same note twice, even across different transactions.*

3 Cryptographic Primitives

3.1 Post-Quantum Digital Signatures: ML-DSA-65

Postera uses ML-DSA-65 (Module-Lattice Digital Signature Algorithm), standardized as FIPS 204 by NIST. This scheme is based on the hardness of the Module Learning With Errors (MLWE) problem.

3.1.1 Parameters

Parameter	Value	Description
Security Level	3 (128-bit)	NIST security category
Public Key Size	1,952 bytes	Verification key
Secret Key Size	4,032 bytes	Signing key
Signature Size	3,309 bytes	Digital signature

Table 1: ML-DSA-65 parameters used in Postera

3.1.2 Security Basis

The security of ML-DSA relies on the hardness of two problems:

1. **Module-LWE**: Given $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ where $\mathbf{s}, \mathbf{e}$ are small, find $\mathbf{s}$.
2. **Module-SIS**: Given $\mathbf{A}$, find a short vector $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{0}$.

These problems are believed to be hard for both classical and quantum computers, with the best known quantum algorithms offering only a quadratic speedup via Grover's algorithm.

3.2 Zero-Knowledge Proofs: Groth16

For transaction privacy, Postera employs Groth16 zk-SNARKs on the BN254 (alt_bn128) curve. While the BN254 pairing is not post-quantum secure, it is used only for privacy (hiding amounts and participants), not for security of funds.

Definition 5 (zk-SNARK)**.** *A zero-knowledge Succinct Non-interactive ARgument of Knowledge is a proof system where:*

- ***Completeness****: Honest provers can convince verifiers of true statements.*
- ***Soundness****: No adversary can prove false statements.*
- ***Zero-Knowledge****: Proofs reveal nothing beyond the statement's validity.*
- ***Succinctness****: Proofs are small and fast to verify.*

3.2.1 Circuit Design

Postera uses two circuits implemented in Circom:

Spend Circuit proves:

1. Knowledge of a note (v, pk_h, r) with value v, recipient hash pk_h, and randomness r
2. The note commitment exists in the commitment tree (Merkle proof)

3. The nullifier was correctly derived from the note and nullifier key

Output Circuit proves:

1. A new note commitment was correctly computed from (v, pk_h, r)

3.3 Hash Functions

3.3.1 Poseidon Hash

For in-circuit hashing (commitments, nullifiers, Merkle tree), Postera uses the Poseidon hash function, optimized for arithmetic circuits over prime fields.

$$H_{\text{Poseidon}} : \mathbb{F}_p^n \to \mathbb{F}_p \tag{1}$$

Poseidon uses a sponge construction with the following domain separators:

Domain	Separator
Note Commitment	1
Value Commitment Hash	2
Nullifier	3
Merkle Empty Leaf	4
Merkle Node	5

Table 2: Poseidon domain separators

3.3.2 SHA-256 and Blake2

For proof-of-work and key derivation, Postera uses SHA-256 and Blake2s256 respectively, both of which provide adequate post-quantum security (128-bit against Grover's algorithm with 256-bit output).

3.4 Authenticated Encryption: ChaCha20-Poly1305

Note encryption uses ChaCha20-Poly1305, an AEAD scheme providing:

- 256-bit key security
- Authentication and confidentiality
- 128-bit post-quantum security against Grover's algorithm

4 Transaction Model

4.1 Notes

The fundamental unit of value in Postera is a **note**, analogous to a UTXO but with privacy properties.

Definition 6 (Note)**.** *A note $\mathcal{N}$ is a tuple (v, pk_h, r) where:*

- $v \in \{0, 1, \ldots, 2^{64} - 1\}$ *is the value*
- $pk_h \in \mathbb{F}_p$ *is the recipient's public key hash*
- $r \in \mathbb{F}_p$ *is random blinding factor*

4.2 Commitments

Notes are stored on-chain as commitments, hiding their contents.

Definition 7 (Note Commitment)**.** *The commitment to a note* $\mathcal{N} = (v, pk_h, r)$ *is:*

$$cm = H_{Poseidon}(1, v, pk_h, r) \tag{2}$$

where 1 *is the domain separator for note commitments.*

4.3 Nullifiers

To prevent double-spending without revealing which note is spent, Postera uses nullifiers.

Definition 8 (Nullifier)**.** *The nullifier for a note at position* ρ *with nullifier key* nk *is:*

$$nf = H_{Poseidon}(3, nk, \rho) \tag{3}$$

When a note is spent, its nullifier is published. The blockchain maintains a nullifier set; any transaction attempting to publish an existing nullifier is rejected.

4.4 Commitment Tree

All note commitments are stored in a Merkle tree of depth 32, supporting up to 2^{32} notes.

Definition 9 (Commitment Tree)**.** *A binary Merkle tree where:*

- *Leaves are note commitments (or empty leaf hash for unused positions)*
- *Internal nodes:* $H_{Poseidon}(5, left, right)$
- *Empty leaf:* $H_{Poseidon}(4, 0)$

4.5 Transaction Structure

A shielded transaction $\mathcal{T}$ consists of:

$$\begin{aligned} \mathcal{T} = \{&\text{spends} : [(\text{nf}_i, \text{anchor}_i, \pi_{\text{spend},i})], && (4)\\ &\text{outputs} : [(\text{cm}_j, \text{enc}_j, \pi_{\text{output},j})], && (5)\\ &\text{binding\_sig}, && (6)\\ &\text{fee}\} && (7) \end{aligned}$$

Where:

- nf_i is the nullifier for spent note i
- anchor_i is the Merkle root at time of proof generation
- $\pi_{\text{spend},i}$ is the Groth16 proof for spend i
- cm_j is the commitment for output note j
- enc_j is the encrypted note data
- $\pi_{\text{output},j}$ is the Groth16 proof for output j
- binding_sig proves value balance (inputs = outputs + fee)

4.6 Value Balance and Binding Signatures

To ensure that transactions preserve value (no inflation), Postera uses Pedersen commitments and binding signatures.

Definition 10 (Value Commitment)**.** *For a value v with randomness r:*

$$cv = v \cdot G + r \cdot H \tag{8}$$

where G, H are generator points on BN254.

The binding signature proves that:

$$\sum_i \text{cv}_{\text{spend},i} = \sum_j \text{cv}_{\text{output},j} + \text{fee} \cdot G \tag{9}$$

This is verified using a Schnorr signature over the sum of randomness values.

5 Consensus Mechanism

5.1 Proof of Work

Postera uses SHA-256 based proof-of-work, similar to Bitcoin. A valid block must satisfy:

$$H_{\text{SHA256}}(\text{block\_header}) < \text{target} \tag{10}$$

The target is expressed as a difficulty d, requiring d leading zero bits.

5.2 Difficulty Adjustment

Difficulty adjusts every 10 blocks to maintain approximately 10-second block times:

Algorithm 1 Difficulty Adjustment

1: window $\leftarrow$ 10 blocks
2: target_time $\leftarrow$ 100 seconds (10 blocks Œ 10 seconds)
3: actual_time $\leftarrow t_{\text{current}} - t_{\text{window\_start}}$
4: ratio $\leftarrow$ target_time/actual_time
5: ratio $\leftarrow \text{clamp}(\text{ratio}, 0.5, 2.0)$ ▷ Limit adjustment
6: $d_{\text{new}} \leftarrow d_{\text{current}} \times \text{ratio}$
7: $d_{\text{new}} \leftarrow \text{clamp}(d_{\text{new}}, d_{\min}, d_{\max})$

5.3 Block Structure

```
Block {
    header: {
        version: u32,
        prev_hash: [u8; 32],
        merkle_root: [u8; 32],
        commitment_root: [u8; 32],
        nullifier_root: [u8; 32],
        timestamp: u64,
        difficulty: u64,
        nonce: u64,
    },
    transactions: Vec<ShieldedTransaction>,
}
```

5.4 Mining Rewards

Each block produces a coinbase transaction creating 50 POSTERA to the miner's address. The reward is implemented as an output note in the coinbase transaction.

6 Privacy Model

6.1 Viewing Keys

Users possess three types of keys:

1. **Spending Key** (sk): ML-DSA-65 secret key, required to spend notes
2. **Viewing Key** (vk): Derived from sk, allows decrypting incoming notes
3. **Public Key Hash** (pk_h): Address for receiving funds

6.2 Note Encryption

Output notes are encrypted to the recipient's viewing key using ChaCha20-Poly1305:

$$\text{enc} = \text{ChaCha20-Poly1305}_k(\text{nonce}, (v, r)) \tag{11}$$

where k is derived from a Diffie-Hellman-like key exchange (using the recipient's public key).

6.3 Privacy Guarantees

Theorem 1 (Sender Privacy)**.** *Given a transaction $\mathcal{T}$, an observer without the spending key cannot determine which notes were spent (beyond the published nullifiers revealing that* some *notes were spent).*

Theorem 2 (Receiver Privacy)**.** *Given a transaction $\mathcal{T}$, an observer without the recipient's viewing key cannot determine the recipient of any output.*

Theorem 3 (Amount Privacy)**.** *Given a transaction $\mathcal{T}$, an observer cannot determine the value of any input or output.*

6.4 Anonymity Set

The anonymity set for a spend is the entire set of unspent notes in the commitment tree. With a tree depth of 32, this supports up to $2^{32} \approx 4$ billion notes.

7 Security Analysis

7.1 Post-Quantum Security of Signatures

Theorem 4 (Signature Unforgeability)**.** *Under the hardness of Module-LWE and Module-SIS, ML-DSA-65 signatures are existentially unforgeable under chosen message attacks, even against quantum adversaries.*

This ensures that an attacker, even with a quantum computer, cannot forge transactions or steal funds.

7.2 Privacy vs. Post-Quantum Security Trade-off

The Groth16 proofs use BN254 pairings, which are vulnerable to quantum attacks. However:

1. **Privacy degradation, not fund theft**: A quantum attacker could potentially break privacy by forging proofs, but cannot steal funds (protected by ML-DSA signatures).
2. **Future upgradability**: The protocol can migrate to post-quantum proof systems (e.g., STARKs, lattice-based SNARKs) as they mature.
3. **Current security**: BN254 remains secure against classical adversaries, and cryptographically-relevant quantum computers are not yet available.

7.3 Double-Spend Prevention

Theorem 5 (Double-Spend Resistance)**.** *A note can only be spent once. Any attempt to spend a note twice will be rejected due to nullifier collision.*

Proof. Each note has a unique nullifier determined by $nf = H(nk, \rho)$. When spent, the nullifier is added to the nullifier set. Any subsequent transaction using the same nullifier is rejected by consensus rules. □

7.4 Inflation Resistance

Theorem 6 (No Inflation)**.** *No transaction can create value. The total supply is determined solely by mining rewards.*

Proof. The binding signature proves that input values equal output values plus fees. The Groth16 proofs ensure that the values in the commitments are correctly formed. Together, these prevent creation of value from nothing. □

8 Implementation

8.1 Architecture

Postera is implemented in Rust with the following components:

- **Core**: Block, transaction, and state management
- **Crypto**: ML-DSA signatures, Poseidon hashing, proof verification
- **Consensus**: Proof-of-work mining with difficulty adjustment
- **Network**: P2P gossip, block synchronization, REST API
- **Wallet**: Note scanning, balance tracking, transaction building

8.2 Client-Side Proving

Zero-knowledge proofs are generated in the browser using:

- **Circom**: Circuit compiler for R1CS constraints
- **snarkjs**: JavaScript library for Groth16 proving
- **WebAssembly**: Circuit witness computation

This ensures that sensitive data (spending keys, note values) never leaves the user's device.

8.3 Verification

Proof verification is performed server-side using:

- **arkworks**: Rust library for elliptic curve operations
- **ark-groth16**: Groth16 verifier implementation

Verification keys are committed to the repository with SHA-256 checksums to ensure all nodes use identical parameters.

9 Performance

Operation	Time	Notes
Block mining (diff 20)	100ms	2 threads, M1 Mac
Proof generation (spend)	3s	Browser, WebAssembly
Proof generation (output)	1s	Browser, WebAssembly
Proof verification	5ms	Server-side, arkworks
Transaction validation	15ms	Including proof verification

Table 3: Performance benchmarks

10 Future Work

1. **Post-Quantum ZK Proofs**: Migrate to lattice-based or hash-based proof systems (e.g., STARKs) for full post-quantum privacy.
2. **Light Clients**: Implement compact block filters and SPV-like verification for mobile wallets.
3. **Layer 2 Scaling**: Payment channels and rollups for higher throughput.
4. **Multi-Signature**: Threshold signatures using lattice-based schemes.
5. **Smart Contracts**: Private smart contract execution using zero-knowledge virtual machines.

11 Conclusion

Postera represents a significant step toward cryptocurrency systems that are secure in a post-quantum world while preserving financial privacy. By combining NIST-standardized post-quantum signatures with zero-knowledge proofs, Postera achieves:

- **Quantum-Resistant Fund Security**: ML-DSA-65 signatures protect against quantum attacks on fund ownership.
- **Transaction Privacy**: Groth16 proofs hide transaction amounts and participants.
- **Practical Performance**: Client-side proving in browsers enables decentralized, private transactions.

As quantum computing advances, the importance of post-quantum cryptography will only grow. Postera provides a foundation for private, quantum-secure digital currency.

Acknowledgments

We thank the developers of the cryptographic libraries that make Postera possible: FIPS 204 reference implementation, arkworks, snarkjs, Circom, and light-poseidon.

References

[1] National Institute of Standards and Technology. *FIPS 204: Module-Lattice-Based Digital Signature Standard*. 2024.

[2] Groth, J. *On the Size of Pairing-based Non-interactive Arguments*. EUROCRYPT 2016.

[3] Hopwood, D., Bowe, S., Hornby, T., Wilcox, N. *Zcash Protocol Specification*. 2023.

[4] Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M. *Poseidon: A New Hash Function for Zero-Knowledge Proof Systems*. USENIX Security 2021.

[5] Shor, P. *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*. FOCS 1994.

[6] Ducas, L., et al. *CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme*. TCHES 2018.