EQRN: A Quantum-Secure Privacy Protocol Using SPHINCS-256 and ZK-STARKs

Whitepaper

February 8, 2023

Abstract

Legacy privacy architectures currently rely heavily on Elliptic Curve Cryptography (ECC) for stealth addresses, ring signatures, and confidential transactions. While secure against classical adversaries, ECC is fundamentally broken by Cryptographically Relevant Quantum Computers (CRQCs) running Shor's algorithm. We present EQRN, a full-stack post-quantum redesign. We replace ECC with stateless hash-based signatures (SPHINCS-256), Post-Quantum Key Encapsulation Mechanisms (PQ-KEMs), and Zero-Knowledge Scalable Transparent Arguments of Knowledge (ZK-STARKs). We detail a novel STARK-based Hash-Based Ring Signature (HBRS) construction, a HORST-derived key image mechanism, and a hash-based range proof protocol replacing Bulletproofs.

Contents

1 Introduction: The next era of Global Finance **3**
- 1.1 The Trust Paradox in a Trustless System . . . 3
- 1.2 The Computation Explosion: A New Threat Landscape . . . 3
- 1.3 The Privacy Dilemma: Transparency vs. Confidentiality . . . 4
- 1.4 A New Foundation: Security and Privacy by Design . . . 4
- 1.5 Towards the Next Financial Era . . . 4

2 Introduction and Threat Model **6**

3 Cryptographic Primitives & SPHINCS Mechanics **6**
- 3.1 The SPHINCS Signature Structure . . . 6

4 Hash-Based Ring Signatures (HBRS) via ZK-STARKs **6**
- 4.1 The STARK Statement . . . 6

5 STARK Arithmetization of SPHINCS **7**
- 5.1 Execution Trace and Hash Selection . . . 7
- 5.2 Transition Constraints . . . 7

6 Key Image Construction and Double-Spend Prevention **7**

7 Post-Quantum Confidential Transactions and Range Proofs **7**
- 7.1 Hash-Based Commitments . . . 8
- 7.2 STARK-Based Range Proofs . . . 8
- 7.3 Block Capacity and TPS . . . 8

8 Implementation Architecture **8**
8.1 Deprecating the Ed25519 Module 8
8.2 Redesigning the Ring Signature Layer (HBRS Generation) 8
8.3 Modifying Key Image Computation 9
8.4 Memory and State Management 9

9 Conclusion **9**

1 Introduction: The next era of Global Finance

When Bitcoin was introduced in 2008, it marked the beginning of a paradigm shift in how the world perceived trust, money, and financial systems. For the first time in modern history, individuals could transact peer-to-peer without relying on centralized intermediaries such as banks or governments. The concept of a trustless network, secured by cryptography and distributed consensus, ignited what is widely regarded as one of the most transformative technological revolutions of our time.

This innovation gave rise to Ethereum, smart contracts, and an entire ecosystem of decentralized applications. Decentralized finance (DeFi) emerged as a powerful alternative to traditional financial infrastructure, unlocking new models of lending, trading, and value exchange. Today, the digital asset market has surpassed trillions of dollars in value, signaling strong belief in the long-term potential of blockchain technology.

And yet, despite this momentum, a fundamental paradox remains.

Global adoption of blockchain-based financial systems still lingers below 8%. For a technology often described as a "once-in-a-century transformation," this limited penetration raises an important question:

What is preventing blockchain from becoming the default infrastructure for global finance?

1.1 The Trust Paradox in a Trustless System

At its core, blockchain replaces institutional trust with cryptographic certainty. Ownership is no longer determined by legal frameworks or custodians but by control over private keys. This introduces a radically different security model—one where the individual becomes the sole custodian of their assets.

However, this shift comes with significant implications.

In traditional finance, failures can be mitigated through regulatory oversight, legal recourse, and institutional safeguards. In contrast, within decentralized systems, the loss or compromise of private keys results in irreversible loss of assets, with no fallback mechanism. As a result, security is no longer a feature—it is the foundation.

Yet, the cryptographic backbone securing most blockchain networks today is based on the Elliptic Curve Digital Signature Algorithm (ECDSA)—a standard developed in 1985.

While ECDSA remains robust under classical computing assumptions, the context in which it operates has changed dramatically.

1.2 The Computation Explosion: A New Threat Landscape

Over the past four decades, computational power has grown exponentially. Processing capabilities have increased by billions of times since the 1980s, driven by advancements in semiconductor technology, parallel computing, and specialized hardware.

The rise of GPUs—originally designed for graphics rendering but now central to artificial intelligence and high-performance computing—has significantly accelerated the ability to perform complex mathematical operations at scale. Modern GPU clusters can execute parallel computations at speeds unimaginable at the time ECDSA was conceived.

Simultaneously, the rapid progress in quantum computing introduces an even more profound disruption. Unlike classical bits, quantum systems operate on qubits, enabling them to solve

certain classes of mathematical problems exponentially faster. Algorithms such as Shor's algorithm theoretically threaten the security assumptions underlying elliptic curve cryptography.

In practical terms, this means that the very cryptographic primitives securing billions of dollars in digital assets today may become vulnerable in the foreseeable future.

For institutions, governments, and global financial systems, this is not a theoretical concern—it is a systemic risk.

1.3 The Privacy Dilemma: Transparency vs. Confidentiality

While security forms one axis of concern, privacy represents another equally critical challenge.

Public blockchains, by design, offer full transparency. Every transaction is recorded on a distributed ledger that is accessible to anyone. While this enhances auditability and trust, it simultaneously creates an unintended consequence: financial surveillance.

Wallet balances, transaction histories, and behavioral patterns can be analyzed and traced, often linking pseudonymous addresses to real-world identities through data correlation techniques.

For individuals, this erodes financial privacy.
For institutions, it becomes a deal-breaker.

No government, enterprise, or large-scale financial entity can operate effectively in an environment where sensitive financial flows are publicly visible and traceable. The lack of built-in confidentiality mechanisms has therefore become a major barrier to institutional adoption.

1.4 A New Foundation: Security and Privacy by Design

The convergence of these challenges—aging cryptographic standards and the absence of native privacy—reveals a critical gap in the current blockchain paradigm.

To unlock true global adoption, decentralized systems must evolve beyond their first-generation design principles. They must offer:

- Quantum-resilient security capable of withstanding future computational breakthroughs
- Privacy-preserving architectures that protect transactional confidentiality
- Institution-grade trust frameworks without compromising decentralization

This is where EQRN emerges.

Built on NIST-approved post-quantum cryptographic standards, EQRN introduces a new class of blockchain infrastructure—one that is both quantum-secure and privacy-enabled by default. By re-architecting the foundational layers of security and data protection, EQRN aims to bridge the gap between decentralized innovation and institutional requirements.

1.5 Towards the Next Financial Era

The first wave of blockchain proved that trust could be decentralized.
The next wave must prove that it can be secured and privatized at scale.

As computational capabilities accelerate and global demand for secure digital infrastructure intensifies, the need for a new standard becomes inevitable.

EQRN represents not just an incremental upgrade, but a fundamental shift in how decentralized systems are designed, secured, and adopted.

The future of finance will not be defined solely by decentralization—
but by who can deliver it with uncompromising security and true privacy.

2 Introduction and Threat Model

Legacy architectures rely on the Ed25519 curve for signature generation and Curve25519 for Diffie-Hellman key exchanges (used in stealth addresses and Pedersen commitments). A CRQC capable of maintaining a few thousand logical qubits could solve the Elliptic Curve Discrete Logarithm Problem (ECDLP) in polynomial time, allowing an adversary to derive private spend keys from public keys and completely deanonymize the blockchain.

This paper proposes **EQRN**, a quantum-secure privacy protocol forked from the Monero protocol. We migrate the primary authentication mechanism to **SPHINCS-256**, a pioneering high-security stateless hash-based signature scheme. We choose SPHINCS over lattice-based schemes because its security relies purely on the preimage resistance of standard cryptographic hash functions, avoiding the complex mathematical assumptions of structured lattices.

3 Cryptographic Primitives & SPHINCS Mechanics

SPHINCS is a stateless scheme built upon a hyper-tree of Merkle trees.

3.1 The SPHINCS Signature Structure

A SPHINCS signature σ consists of three main components:

1. **HORST (Hash to Obtain Random Subset Tree):** A few-time signature scheme used to sign the actual message hash.
2. **WOTS+ (Winternitz One-Time Signature):** Used to sign the public keys (roots) of the HORST trees.
3. **Hypertree Authentication Paths:** The Merkle paths connecting the WOTS+ signatures up to the single global public key root.

The signature is mathematically represented as:

$$\sigma = (\sigma_{\text{HORST}}, \sigma_{\text{WOTS}}, \text{AuthPaths})$$

For the 256-bit security level, the resulting signature size is exactly 41 KB. Storing 41 KB per input leads to insurmountable chain bloat. Thus, EQRN introduces a STARK-based signature compression layer.

4 Hash-Based Ring Signatures (HBRS) via ZK-STARKs

SPHINCS public keys are Merkle roots, possessing no algebraic structure to natively combine into a ring signature. To achieve Untraceability, EQRN wraps the SPHINCS verification algorithm inside a ZK-STARK.

4.1 The STARK Statement

Instead of posting the 41 KB SPHINCS signature σ, the spender generates a STARK proof π. The ring is defined as a set of n public keys: $R = \{P_1, P_2, \ldots, P_n\}$.

The STARK proves the following statement: *I know a private key SK, an index $i \in [1, n]$, and a SPHINCS signature σ such that:*

$$P_i \in R \tag{1}$$

$$\text{SPHINCS.Verify}(P_i, \sigma, \text{TxHash}) = \text{True} \tag{2}$$

$$SK \text{ mathematically corresponds to } P_i \tag{3}$$

Because STARKs are purely hash-based, this construction maintains the strict post-quantum assumptions of EQRN.

5 STARK Arithmetization of SPHINCS

To generate the STARK proof efficiently, the SPHINCS verification algorithm must be encoded into an Algebraic Intermediate Representation (AIR).

5.1 Execution Trace and Hash Selection

Standard SPHINCS utilizes hash functions like ChaCha12 or BLAKE-256, which require extensive bitwise operations that are highly inefficient in arithmetic circuits. EQRN replaces the internal hash function of the SPHINCS instance with an arithmetization-friendly hash function, such as **Poseidon2**, defined over a prime field $\mathbb{F}_p$ where p is a 64-bit Goldilocks prime ($p = 2^{64} - 2^{32} + 1$).

The execution trace of the STARK is a matrix T of dimensions $N \times W$, where W is the number of state registers and N is the number of steps required to verify the Hypertree and WOTS+ paths.

5.2 Transition Constraints

For the Poseidon2 permutation $P(S)$, where S is the state vector, the STARK enforces transition constraints at each row j:

$$T[j + 1, \dots] = P(T[j, \dots])$$

These polynomial constraints ensure that every Merkle branch in the SPHINCS authentication path is correctly computed. By utilizing the FRI protocol, the prover commits to this trace, allowing the verifier to check the 41 KB signature verification in logarithmic time.

6 Key Image Construction and Double-Spend Prevention

In legacy systems, the key image is $I = xH_p(P)$. We tie the post-quantum key image to the deterministic randomness inherent in the HORST structure of SPHINCS. We define the Key Image KI as a cryptographic commitment to the bottom-most secret HORST values used in the signature, mathematically constrained inside the STARK proof:

$$KI = H(\text{HORST}_{\text{secret\_nodes}} \parallel \text{TxHash})$$

If a KI is seen twice, the network rejects the transaction as a double spend.

7 Post-Quantum Confidential Transactions and Range Proofs

Legacy protocols use Pedersen commitments ($C = xG + aH$) and Bulletproofs to hide transaction amounts and prove $a \in [0, 2^{64} - 1]$. Because Pedersen commitments rely on the ECDLP, they must be replaced.

7.1 Hash-Based Commitments

We replace algebraic commitments with perfectly hiding, computationally binding hash commitments:

$$C = H(a \parallel r)$$

Where a is the transaction amount and r is a 256-bit blinding factor.

7.2 STARK-Based Range Proofs

To prove that a is not a negative number, we utilize the same STARK framework used for the ring signature. The prover adds columns to the execution trace representing the binary decomposition of a: $(b_0, b_1, \ldots, b_{63})$. The transition constraints enforce:

$$b_k \times (1 - b_k) = 0 \quad \text{(Boolean constraint)} \tag{4}$$

$$a = \sum_{k=0}^{63} b_k \cdot 2^k \tag{5}$$

$$C = \text{Poseidon2}(a \parallel r) \tag{6}$$

Because this uses the same proving system as the HBRS, the range proof is batched with the signature verification proof, yielding immense data savings.

7.3 Block Capacity and TPS

Assuming a dynamic block size targeting roughly 2 MB, and an average transaction size of 6 KB (including KEM ciphertexts):

$$N_{\text{tx}} = \frac{2 \text{ MB}}{6 \text{ KB}} \approx 333 \text{ tx/block}$$

Assuming a 120-second block time, the throughput is ≈ 2.77 TPS. Block verification is dominated by STARK verification ($\sim$ 10 ms per proof), taking $\sim$ 3.3 seconds sequentially or < 1 second parallelized.

8 Implementation Architecture

Transitioning from the legacy codebase (which relies on `crypto-ops` for Ed25519 arithmetic) to EQRN requires a modular overhaul of the transaction pipelines.

8.1 Deprecating the Ed25519 Module

The legacy `crypto::generate_ring_signature` and `crypto::check_ring_signature` functions rely heavily on scalar multiplication and elliptic curve point additions. In EQRN, the `crypto` namespace is abstracted. The Ed25519 scalar fields are replaced with elements from the Goldilocks prime field $\mathbb{F}_p$ to natively support the STARK arithmetization.

8.2 Redesigning the Ring Signature Layer (HBRS Generation)

The generation of a transaction input signature is delegated to a localized STARK prover:

1. **Decoy Selection:** The wallet software queries the daemon for $n - 1$ random public keys (SPHINCS Merkle roots). Let the true spend key be at secret index s.

2. **Native Signing:** The wallet computes the raw 41 KB SPHINCS signature σ_{raw} over the transaction hash Tx_{hash} using the private spend key SK_s.

3. **Trace Generation:** The STARK prover constructs the execution trace matrix. The public inputs are the ring members $R = \{P_0, \ldots, P_{n-1}\}$ and Tx_{hash}. The private witness includes s and σ_{raw}.

4. **FRI Commitment:** The prover applies the FRI protocol to commit to the polynomial representation of the trace.

5. **Proof Output:** The prover outputs the succinct STARK proof π, discarding σ_{raw}. The payload appends π in place of the legacy signature.

8.3 Modifying Key Image Computation

Legacy implementations compute the key image within the same scalar multiplication loop as the ring signature. EQRN decouples the signature from algebraic groups; the key image extraction is implemented as a parallel hash commitment constraint within the STARK AIR.

$$KI = \text{Poseidon2}(\text{HORST}_{\text{nodes}} \parallel Tx_{\text{hash}})$$

The daemon's memory pool and consensus layers are updated to index KI as a 256-bit hash rather than an elliptic curve point.

8.4 Memory and State Management

Generating a proof for a ring size of $n = 16$ with a full SPHINCS verification trace requires approximately 200 MB of RAM for the prover. However, the daemon's verifier memory footprint remains negligible (< 1 MB per transaction), preserving the ability to run full nodes on low-resource hardware like Raspberry Pis.

9 Conclusion

We demonstrate that a fully quantum-secure privacy cryptocurrency is practically viable using SPHINCS-256, ML-KEM, and ZK-STARKs. By bridging stateless hash signatures with arithmetization-friendly STARK proofs, EQRN preserves untraceability and unlinkability while entirely eliminating elliptic curve discrete logarithm assumptions from the protocol.