Secure Quantum Zero-Knowledge Proofs: Implementation, Analysis, and Optimization

Author: Nicolas Cloutier
ORCID: 0009-0008-5289-5324
GitHub: https://github.com/Grillcheese-AI/qzkp-academic/

Affiliation: GrillCheese AI

Date: May 24th, 2025 *(Revised: Feb 21st, 2026)*

Abstract

We present an end-to-end **reference implementation (prototype)** and structured security analysis of a quantum zero-knowledge proof (QZKP)-style system, with emphasis on preventing **transcript leakage** in practical encodings of quantum witnesses. The protocol supports configurable **soundness** levels (32–256 bits) via repeated challenges and combines standard cryptographic components for integrity and aggregation, including hash-based commitments (SHA-256 / BLAKE3), Merkle-tree aggregation, and post-quantum authentication using **ML-DSA** (derived from CRYSTALS-Dilithium).

Our core contribution is a construction we call *probabilistic entanglement*, intended to enable verification of quantum-dependent relations while limiting what a verifier can infer about the underlying witness from protocol transcripts. We state explicit threat models and assumptions, provide a simulator-based **computational** zero-knowledge claim (unless strengthened), and supply empirical leakage regression tests showing that naive transcript designs can reveal witness structure whereas the secure variant avoids direct inclusion of witness amplitude components.

We report experimental executions on IBM Quantum hardware and characterize proof sizes and runtimes across security settings. Finally, we release the source code, test suite, and evidence artifacts to support reproducibility, regression testing, and external cryptographic review.

Keywords: quantum cryptography, zero-knowledge proofs, post-quantum cryptography, transcript leakage, soundness analysis

1. Introduction

Quantum zero-knowledge proofs (QZKP) represent a fundamental advancement in cryptographic protocols, enabling verification of quantum witness knowledge without revealing the witness itself. While theory provides formal quantum ZK notions, practical implementations can fail due to transcript design choices, serialization, and inadvertent disclosure of witness-related information. This work focuses on implementation-driven failure modes and mitigation strategies.

1.1 Problem Statement

Designing a practical QZKP system presents several critical challenges:

1. **Information Leakage Prevention**: avoiding direct or indirect witness disclosure via transcripts

2. **Commitment Schemes**: commitment mechanisms with clear binding/hiding assumptions and digest-length guidance

3. **Randomness Requirements**: cryptographically secure randomization and repeatability controls

4. **Post-Quantum Security**: authentication primitives compatible with standardized PQC

1.2 Contributions

- **Security Analysis**: practical vulnerability assessment of naive QZKP transcript designs

- **Secure Implementation Goals**: a secure-by-design prototype targeting transcript non-inclusion of witness amplitudes

- **Performance Characterization**: proof sizes and runtimes across security levels

- **Post-Quantum Authentication**: PQ signatures (ML-DSA / Dilithium-derived) for integrity/authentication

- **Open Source**: complete implementation, tests, and reproducibility artifacts

2. Theoretical Foundations

2.1 Quantum Zero-Knowledge Proofs

Quantum zero-knowledge extends classical ZK into the quantum setting. We use standard properties:

- **Completeness**: honest prover convinces honest verifier for valid instances
- **Soundness**: cheating prover succeeds on invalid instances with small probability
- **Zero-Knowledge**: verifier learns nothing beyond validity of the statement, formalized via simulation

This paper emphasizes **implementation-level** zero-knowledge risks and mitigations.

2.2 Post-Quantum Cryptography (Terminology)

We use post-quantum authentication compatible with NIST-standardized primitives. In modern terminology:

- **ML-DSA** *(FIPS 204; derived from CRYSTALS-Dilithium)* for signatures
- (Optionally) **ML-KEM** *(FIPS 203; derived from CRYSTALS-Kyber)* for key establishment where needed

Hash-based commitments use SHA-256 and/or BLAKE3; security depends on digest length and standard assumptions.

3. Security Analysis and Framework

3.1 Security Model

We consider a malicious verifier V^* that can adaptively choose challenges and record full transcripts. Unless otherwise stated, the adversary is quantum polynomial-time (QPT).

3.1.1 Zero-Knowledge Property Theorem 1 (Computational Zero-Knowledge). For any quantum polynomial-time verifier V^*, there exists a quantum polynomial-time simulator S such that for all valid instances x:

$$\text{View}_{V^*}(P, V^*)(x) \approx_c S(x).$$

where $\approx_c$ denotes computational indistinguishability.

Note. This paper does not claim information-theoretic zero-knowledge unless explicitly proven under a stronger framework.

3.1.2 Soundness Theorem 2 (Soundness with Repetition). For k independent binary challenges:

$$\Pr[\langle P^*, V\rangle(x) = 1 \mid x \notin L] \leq 2^{-k}.$$

This provides a direct mapping between "security level" and challenge count.

3.2 Practical Security Objectives

Our framework targets practical security challenges:

1. **State Representation**: prevent transcript leakage via naive state serialization
2. **Commitment Scheme**: binding/hiding commitments with explicit assumptions and adequate digest length
3. **Randomness**: cryptographically secure randomization using OS entropy (`crypto/rand`)
4. **Adversary Model**: resistance to transcript-based reconstruction strategies tested in the suite

We developed a testing framework to quantify transcript leakage as an engineering goal (see §5 and Appendix B).

4. Probabilistic Entanglement Framework

4.1 Theoretical Foundations

Our work introduces a framework called *probabilistic entanglement* that aims to address implementation-level leakage by keeping witness-bearing information inside quantum operations while limiting transcript exposure. The intended flow is:

1. **Probabilistic Encoding**: map classical data into amplitudes and phases
2. **Quantum State Formation**: prepare a state encoding witness-dependent relations
3. **Entanglement/Control Structure**: couple verification operations to the witness state
4. **Measurement + Commit/Challenge**: commit to verifier-checkable outcomes without serializing witness amplitudes

4.2 Mathematical Formulation (High Level)

Step 1: Probabilistic Encoding

Given a classical bitstring $d \in \{0,1\}^n$, define:

$$|\psi_d\rangle = \frac{1}{\sqrt{Z}} \sum_{x \in \{0,1\}^n} f(x,d)\,|x\rangle,$$

where Z is a normalization factor and $f(x,d)$ defines the (magnitude, phase) structure.

Step 2: Proof State Formation

$$|\psi_{\text{proof}}\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle|\psi_d\rangle + |1\rangle\, U|\psi_d\rangle\right),$$

where U is a unitary transformation implementing verification-related structure.

Step 3: Measurement Compatibility (Correct Terminology)

If validity checks and secret-bearing measurements are intended to be jointly measurable, the relevant condition is **commutation/compatibility**, not "orthogonality":

$$[\mathcal{O}_s, \mathcal{O}_v] = 0.$$

(When observables act on disjoint subsystems, the lifted operators commute trivially.)

Step 4: Quantum Verification

A verification projector $M_v = |\phi_v\rangle\langle\phi_v|$ yields:

$$P_{\text{verify}} = |\langle\phi_v|\psi_{\text{proof}}\rangle|^2.$$

4.3 Implementation Details (Prototype)

```
def create_qzkp_circuit(data_bytes, security_level=256):
    """
    Prototype circuit builder: prepares an encoded state + verification structure.
    Note: quantum hardware returns measurement results; statevectors exist only in simulator m
    """
    quantum_state = bytes_to_quantum_amplitudes(data_bytes)
    qc = QuantumCircuit(security_level // 8)  # e.g., 32 qubits for 256-bit
    qc = apply_probabilistic_entanglement(qc, quantum_state)
    return qc
```

4.4 Proof Size and Soundness Mapping

For k independent binary challenges, soundness error is bounded by 2^{-k}. The implementation exposes security levels via $k \in 32, 64, 80, 96, 128, 256$.

Security Level	Challenges (k)	Proof Size	Soundness Error
32-bit	32	13.5 KB	$2^{-32} \approx 2.33 \times 10^{-10}$
64-bit	64	17.6 KB	$2^{-64} \approx 5.42 \times 10^{-20}$
80-bit	80	19.6 KB	$2^{-80} \approx 8.27 \times 10^{-25}$
96-bit	96	21.6 KB	$2^{-96} \approx 1.26 \times 10^{-29}$
128-bit	128	25.7 KB	$2^{-128} \approx 2.94 \times 10^{-39}$
256-bit	256	41.9 KB	$2^{-256} \approx 8.64 \times 10^{-78}$

5. Security Analysis of Existing Implementations (Empirical)

Methodology (Empirical Leakage Evaluation):

1. Generate distinctive witness/test vectors
2. Generate transcripts using the target implementation
3. Scan transcripts for direct witness inclusion or trivially reconstructible patterns
4. Attempt heuristic reconstruction where applicable
5. Report leakage as an engineering metric (not a formal proof)

Results (Summary):

- **Insecure implementation**: observed high leakage under naive transcript designs
- **Secure-by-design prototype**: passes "no-direct-inclusion" regression tests under the implemented suite

  Important: These tests provide evidence and regression protection; they do not replace a full cryptographic proof.

5.1 Attack Scenarios

5.1.1 State Reconstruction Attack (Transcript-Level) **Objective**: reconstruct witness structure from proof data **Method**: extract serialized witness components or match amplitude patterns **Observed**: succeeds against naive implementations that serialize amplitudes or probabilities

5.1.2 Commitment Inversion / Weak Randomness **Objective**: exploit deterministic or weakly randomized commitments **Method**: pattern analysis and brute-force in reduced spaces **Observed**: succeeds against weak commitment schemes; mitigated by strong randomness + proper digest sizing

6. Secure Implementation Design

6.1 SecureQuantumZKP Protocol

We designed `SecureQuantumZKP` to address implementation leakage risks with these **engineering goals**:

- **Cryptographic Commitments**: SHA-256/BLAKE3 with secure randomization
- **Post-Quantum Authentication**: ML-DSA (Dilithium-derived) for authentication/integrity
- **Merkle Tree Aggregation**: efficient proof composition
- **Transcript Non-Inclusion Goal**: avoid direct embedding of witness amplitudes/components in transcripts

Protocol Structure (illustrative):

```
SecureProof {
  ProofID: UUID,
  Commitments: []CryptographicCommitment,
  Challenges: []Challenge,
  Responses: []Response,
  MerkleRoot: MerkleTree(responses),
  Signature: PQSignature,
  Metadata: SecureMetadata
}
```

6.2 Cryptographic Components

Hash Functions

- SHA-256: standard hash for commitments
- BLAKE3: high-performance hash alternative
- Digest length: MUST be chosen to meet security goals; truncation reduces security margin

Digital Signatures

- ML-DSA (Dilithium-derived): post-quantum authentication primitive

Randomness

- OS CSPRNG (`crypto/rand`) for commitment randomness and protocol nonces

6.3 Security Properties (Scope)

- **Completeness (engineering)**: valid proofs should verify with high probability under expected conditions
- **Soundness**: bounded by 2^{-k} under independent challenge repetition
- **Zero-Knowledge (claim)**: simulator-based computational ZK under explicit assumptions; empirical leakage tests support non-inclusion goals

7. Performance Analysis

7.1 Proof Size Analysis

Proof sizes scale approximately linearly with the number of challenges k (Table in §4.4).

7.2 Performance Benchmarking (Prototype)

Generation Performance (example reported):

- 80-bit security: 0.57 ms average generation time
- 128-bit security: 0.72 ms average generation time
- 256-bit security: 1.72 ms average generation time

Verification Performance (example reported):

- < 0.2 ms verification time per challenge (implementation-dependent)
- Overall verification scales approximately $O(k)$ (not $O(1)$)

  Note: Microbenchmarks measure CPU code paths and should be reported with platform/toolchain details for reproducibility.

7.3 Comparison with Other ZK Systems (Cautious)

Any cross-system comparison must specify:

- statement size and witness model
- setup assumptions
- hardware and implementation details
- security model and threat assumptions

We include comparisons as contextual motivation; definitive claims require standardized benchmarking protocols.

8. Conclusion

This work presents a reproducible QZKP prototype and an engineering-focused security analysis aimed at preventing transcript leakage in practical implementations. We provide explicit threat modeling, a simulator-based computational ZK claim under stated assumptions, empirical leakage regression tests, and IBM Quantum hardware execution evidence. The release is intended to support external review, reproduction, and future strengthening toward more formal guarantees.

References

[1] John Watrous. (2009). *Zero-Knowledge against Quantum Attacks*. **SIAM Journal on Computing**, 39(1), 25–58.

[2] Anne Broadbent, Christian Schaffner. (2016). *Quantum Cryptography Beyond Quantum Key Distribution*. **Designs, Codes and Cryptography**, 78(1), 351–382.

[3] Hirotada Kobayashi. (2003). *Non-interactive Quantum Perfect and Statistical Zero-Knowledge*. In **ISAAC 2003**, LNCS 2906, pp. 178–188. Springer.

[4] Jack O'Connor, Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn. (2020). *BLAKE3: One Function, Fast Everywhere*. IACR ePrint 2020/1143.

[5] Ralph C. Merkle. (1988). *A Digital Signature Based on a Conventional Encryption Function*. In **CRYPTO '87**, LNCS 293, pp. 369–378. Springer.

Corresponding Author: Nicolas Cloutier (ORCID: 0009-0008-5289-5324)

Appendix A: Implementation Details

A.1 Core Data Structures

QuantumState Representation:

```
type QuantumState struct {
    Amplitudes []complex128
    Dimension  int
    Normalized bool
}
```

Secure Proof Structure:

```
type SecureProof struct {
    ProofID     string
    Commitments []CryptographicCommitment
    Challenges  []Challenge
    Responses   []Response
    MerkleRoot  []byte
    Signature   []byte
    Metadata    SecureMetadata
}
```

Cryptographic Commitment:

```
type CryptographicCommitment struct {
    Hash       []byte
    Randomness []byte
    Timestamp  time.Time
}
```

A.2 Security Configuration

```
const (
    SecurityLevel32Bit  = 32
    SecurityLevel64Bit  = 64
    SecurityLevel80Bit  = 80
    SecurityLevel96Bit  = 96
    SecurityLevel128Bit = 128
    SecurityLevel256Bit = 256
)
```

- Hash: SHA-256 / BLAKE3
- RNG: `crypto/rand`

- PQ signatures: ML-DSA (Dilithium-derived)

A.3 Performance Optimizations (Implementation Notes)

- Allocation discipline for frequently created structures
- Parallelizable steps where safe (challenge generation / verification batching)
- Caching where appropriate (Merkle paths, signature verification if applicable)

Appendix B: Security Analysis Details (Engineering + Proof Outline)

B.1 Leakage Testing Framework (Regression)

Test Vector Generation (illustrative):

```
func GenerateDistinctiveVectors() []QuantumState {
    return []QuantumState{
        {Amplitudes: []complex128{0.6+0.2i, 0.3+0.1i, 0.5+0.4i, 0.2+0.3i}},
        {Amplitudes: []complex128{0.8+0.1i, 0.2+0.3i, 0.4+0.2i, 0.1+0.5i}},
        {Amplitudes: []complex128{0.7+0.3i, 0.1+0.2i, 0.3+0.6i, 0.4+0.1i}},
        {Amplitudes: []complex128{0.5+0.5i, 0.4+0.1i, 0.2+0.3i, 0.6+0.2i}},
    }
}
```

Leakage Detection (illustrative):

```
func DetectInformationLeakage(proof []byte, originalState QuantumState) float64 {
    leakedComponents := 0
    totalComponents := len(originalState.Amplitudes)
    for _, amplitude := range originalState.Amplitudes {
        if ContainsAmplitude(proof, amplitude) {
            leakedComponents++
        }
    }
    return float64(leakedComponents) / float64(totalComponents)
}
```

B.2 Attack Simulation Results (Scope)

- Reconstruction attacks can succeed against naive transcript serialization
- Secure-by-design prototype aims to prevent direct transcript inclusion; regression tests check common failure modes
- These results are empirical evidence and engineering validation, not a formal ZK proof

B.3 Soundness Error Analysis

For k independent binary challenges:

$$P(\text{cheat success}) \leq \left(\frac{1}{2}\right)^k.$$

B.4 Proof Details (Outline)

- Assumptions and reduction targets
- Simulator construction
- Soundness amplification proof
- Commitment security discussion (digest length, collision/preimage assumptions)

Technical Appendix: Rigorous Analysis of Probabilistic Entanglement Framework

Response to technical inquiry

Author: Nicolas Cloutier

Date: May 27th, 2025

Context: Mathematical analysis addressing measurement compatibility, noise discussion, and security bounds under explicit assumptions

1. Measurement Compatibility (Commutation) Conditions

Definition 1.1 (Secret Observable).
On the data Hilbert space $\mathcal{H}_d$:

$$\mathcal{O}_s = \sum_{i=0}^{2^n-1} \alpha_i \, |\psi_i\rangle\langle\psi_i|, \quad |\psi_i\rangle \in \mathcal{H}_d, \ \alpha_i \in \mathbb{R}.$$

Definition 1.2 (Validity Observable).
On the verification Hilbert space $\mathcal{H}_v$:

$$\mathcal{O}_v = \sum_{j=0}^{2^m-1} \beta_j \, |\phi_j\rangle\langle\phi_j|, \quad |\phi_j\rangle \in \mathcal{H}_v, \ \beta_j \in \{0,1\}.$$

Theorem 1.1 (Subsystem Commutation).
On $\mathcal{H} = \mathcal{H}_d \otimes \mathcal{H}_v$:

$$[\mathcal{O}_s \otimes \mathbb{I}_v, \ \mathbb{I}_d \otimes \mathcal{O}_v] = 0.$$

Proof. Operators acting on disjoint tensor factors commute. □

2. Noise and Decoherence (Clarified Scope)

Noise can be modeled as a CPTP map:

$$\mathcal{N}(\rho) = \sum_k E_k \, \rho \, E_k^\dagger, \quad \sum_k E_k^\dagger E_k = \mathbb{I}.$$

Important clarification. Local CPTP maps preserve tensor-factor locality, but do not, in general, preserve commutators under arbitrary pictures without additional assumptions.

3. Zero-Knowledge Security Bounds (Assumptions Required)

Mutual information:

$$I(A:B)_\rho = S(\rho_A) + S(\rho_B) - S(\rho_{AB}).$$

Bounds of the form:

$$I(\text{Secret} : \text{Transcript})_\rho \leq \varepsilon \cdot \log_2 |\mathcal{S}|$$

require an explicit definition of the transcript state, an adversary model, and a proof (often via trace-distance and/or simulation arguments).

4. Practical Implementation Bounds

- Soundness via repetition: $\varepsilon_{\text{sound}} \leq 2^{-k}$
- Verification cost scales approximately $O(k)$ (not $O(1)$)
- Scalability summary matches Table 4.4

5. Conclusion (Technical Appendix)

This appendix clarifies measurement compatibility conditions (commutation), highlights what is automatic from tensor product structure, and scopes noise/security-bound statements to avoid overclaiming without full formal proofs.