Post-Quantum Authentication for Quantum Key Distribution Control Channels

Sylvain Cormier
Paraxiom Technologies Inc.
sylvain@paraxiom.org

November 2025

Abstract

Quantum Key Distribution (QKD) protocols achieve information-theoretic security by leveraging quantum mechanics, but their security proofs explicitly assume authenticated classical channels between endpoints. Contemporary QKD implementations realize this authentication using TLS 1.2/1.3 with RSA or ECDSA certificates—primitives vulnerable to polynomial-time quantum attacks via Shor's algorithm. We formalize this architectural inconsistency as a theorem demonstrating that any QKD deployment whose classical authentication relies on factoring or discrete logarithm assumptions is insecure against quantum adversaries, even if the quantum channel itself is unconditionally secure. We prove that a harvest-now-decrypt-later adversary can retroactively compromise authentication, enabling endpoint impersonation and key extraction. As a constructive solution, we present the Post-Quantum Transport Gateway (PQTG), a protocol that replaces classical TLS authentication with NIST-standardized post-quantum primitives (ML-KEM-768, Falcon-512, SPHINCS+) while maintaining backward compatibility with existing QKD infrastructure. We provide formal security analysis under the quantum random oracle model, prove IND-CCA2 security for key exchange and EUF-CMA security for authentication, and demonstrate practical deployment through an open-source implementation with performance benchmarks showing post-quantum operations outperform RSA-2048 by up to 125×.

Keywords: Quantum Key Distribution, Post-Quantum Cryptography, authenticated channels, Shor's algorithm, ML-KEM, Falcon, SPHINCS+

1 Introduction

1.1 Quantum Key Distribution and Classical Authentication

Quantum Key Distribution [1, 2] enables two parties to establish a shared secret key with security guaranteed by the laws of quantum mechanics rather than computational assumptions. The security of QKD protocols has been rigorously proven [3,4], establishing information-theoretic confidentiality against eavesdropping on the quantum channel.

However, all QKD security proofs share a critical assumption: *authenticated classical communication channels* [5]. Without authentication, an adversary can mount man-in-the-middle attacks during basis reconciliation, error correction, privacy amplification, endpoint verification, and key management operations.

The standard approach in deployed QKD systems is to implement this authenticated classical channel using TLS 1.2 or 1.3 [6], relying on RSA (2048–4096 bit) or ECDSA (P-256, P-384) for digital signatures and X.509 certificate-based authentication.

1.2 The Post-Quantum Authentication Gap

Shor's algorithm [7] enables quantum computers to factor integers and solve discrete logarithms in polynomial time, breaking RSA and elliptic curve cryptography. While the timeline for cryptographically relevant quantum computers remains uncertain (estimates range from 2030–2040 [8]), the *harvest-now-decrypt-later* (HNDL) threat is immediate: adversaries can capture encrypted communications today and decrypt them retroactively once quantum computers become available [9].

This creates a fundamental contradiction in current QKD deployments:

> *QKD systems designed to provide quantum-safe key distribution rely on classical authentication that is explicitly vulnerable to quantum attacks.*

1.3 Contributions

This paper makes the following contributions:

1. **Formal vulnerability analysis**: We prove that QKD deployments using classical public-key authentication are insecure against polynomial-time quantum adversaries (Theorem 2).

2. **Threat model formalization**: We define a precise adversary model capturing harvest-now-decrypt-later attacks, certificate authority compromise, and long-term key exposure specific to QKD infrastructure.

3. **Protocol design**: We present the Post-Quantum Transport Gateway (PQTG), a transparent security layer that replaces classical TLS authentication with NIST-standardized post-quantum primitives.

4. **Security proofs**: We prove PQTG achieves IND-CCA2 security for key encapsulation and EUF-CMA security for digital signatures under the quantum random oracle model (QROM).

5. **Implementation and evaluation**: We provide an open-source implementation and demonstrate that post-quantum operations achieve lower latency than RSA-based handshakes.

1.4 Related Work

Post-quantum cryptography. NIST has standardized ML-KEM [10] (lattice-based key encapsulation), ML-DSA and SLH-DSA (signature schemes). These provide security against quantum adversaries under well-studied hardness assumptions.

Hybrid TLS. The IETF is standardizing hybrid key exchange [11]. The Open Quantum Safe project [12] provides liboqs for post-quantum TLS. Google has deployed hybrid Kyber in Chrome [13].

QKD security. Prior work has identified implementation vulnerabilities [14], side-channel attacks [15], and authentication requirements [5]. However, post-quantum insecurity in QKD control channels has received limited formal treatment.

Distinction from prior work. While hybrid TLS provides post-quantum key exchange, it typically retains classical signatures for authentication. Our work eliminates all quantum-vulnerable primitives from QKD authentication, which is necessary because certificate forgery enables endpoint impersonation even with quantum-safe key exchange.

2 Preliminaries

2.1 Quantum Key Distribution Model

Definition 1 (QKD Security)**.** *A QKD protocol achieves ϵ-security if, for any adversary controlling the quantum channel, the protocol outputs either a*

key K satisfying:

$$\Pr[abort] \leq \epsilon_{rob}, \quad \Pr[K_A \neq K_B | no\ abort] \leq \epsilon_{cor}$$

and the key is ϵ_{sec}-close to uniform and independent of the adversary's quantum state.

Critical assumption: All QKD security proofs assume authenticated classical channels [4].

2.2 Classical Authentication via TLS

Contemporary QKD implementations [6, 18] use TLS 1.2/1.3 with X.509 certificates signed using RSA or ECDSA.

2.3 Quantum Cryptanalysis

Definition 2 (Quantum Adversary)**.** *A quantum adversary $\mathcal{A}$ has polynomial-time access to a quantum computer with arbitrary quantum operations on polynomially many qubits.*

Theorem 1 (Shor's Algorithm [7])**.** *Given an RSA modulus N or elliptic curve discrete logarithm instance, a quantum adversary can recover the secret key in time $O((\log N)^3)$.*

Implications: RSA-2048 requires ≈4098 logical qubits [17]; ECDSA P-256 requires ≈2330 qubits.

2.4 Post-Quantum Primitives

Definition 3 (Key Encapsulation Mechanism)**.** *A KEM consists of* $(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$. *A KEM is IND-CCA2 secure if no polynomial-time adversary can distinguish encapsulated keys from random.*

Definition 4 (Digital Signature Scheme)**.** *A signature scheme consists of* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. *It is EUF-CMA secure if no polynomial-time adversary can forge signatures on new messages.*

We use ML-KEM-768 [10] (NIST Level 3), Falcon-512 [16] (Level 1), and SPHINCS+-256f [10] (Level 5).

3 System Model and Threat Analysis

3.1 System Architecture

A QKD network consists of: QKD endpoints (Alice, Bob), quantum channel, classical channel, management API (REST/HTTPS), and Certificate Authority.

3.2 Adversary Model

We define $\mathcal{A} = (\mathcal{A}_{\text{now}}, \mathcal{A}_{\text{future}})$:

- $\mathcal{A}_{\textbf{now}}$: Captures traffic passively, cannot break RSA/ECDSA classically
- $\mathcal{A}_{\textbf{future}}$: Has quantum computer, can run Shor's algorithm, forge certificates

Attack timeline:

1. Time t_0: $\mathcal{A}_{\text{now}}$ captures TLS traffic
2. Time t_1: $\mathcal{A}_{\text{future}}$ obtains quantum computer
3. Time t_1: Breaks RSA/ECDSA, forges certificates, impersonates endpoints

3.3 Threat Scenarios

Harvest-Now-Decrypt-Later: Adversary captures certificates and encrypted sessions, later uses Shor's algorithm to extract keys.

Certificate Authority Compromise: With sk_{CA} from Shor's algorithm, adversary generates valid certificates for any endpoint.

Long-Term Key Exposure: Certificates valid for years become retroactively compromised.

4 Main Security Result

Theorem 2 (Authentication Collapse under Quantum Adversary)**.** *Let Π_{QKD} be a QKD protocol proven ϵ-secure under authenticated classical channels. Let Π_{Auth} be classical authentication using RSA/ECDSA. The composite protocol $\Pi = (\Pi_{QKD}, \Pi_{Auth})$ is insecure against the adversary $\mathcal{A} = (\mathcal{A}_{now}, \mathcal{A}_{future})$.*

Specifically, $\mathcal{A}$ can extract all quantum keys from $[t_0, t_1]$ and impersonate endpoints after t_1 with probability $1 - negl(\lambda)$.

Proof. **Phase 1 (Harvest):** $\mathcal{A}_{\text{now}}$ captures TLS handshakes with RSA/ECDSA public keys and encrypted API traffic.

Phase 2 (Decrypt): $\mathcal{A}_{\text{future}}$ applies Shor's algorithm to factor N or solve discrete log in time $O(\text{poly}(\log N))$ (Theorem 1), recovering sk_{CA}.

Phase 3 (Impersonate): Using sk_{CA}, adversary generates valid certificate $\text{cert}_M = \mathsf{Sign}(sk_{\text{CA}}, M)$ for malicious endpoint M, presents it to client C when connecting to legitimate endpoint A. Client verifies $\mathsf{Verify}(pk_{\text{CA}}, \text{cert}_M) = 1$ and accepts M as A.

Success probability is $1 - \text{negl}(\lambda)$ since Shor succeeds with overwhelming probability. The authenticated channel assumption of Π_{QKD} is violated, voiding all security guarantees. □ □

Corollary 3. *Any production QKD deployment using RSA/ECDSA authentication will be retroactively compromised once quantum computers become available.*

5 Post-Quantum Transport Gateway Protocol

5.1 Design Overview

PQTG replaces classical TLS authentication with post-quantum primitives via transparent gateway co-located with QKD endpoints.

Architecture: External interface uses PQ-TLS; internal interface connects to vendor API via localhost; vendor TLS never exposed externally.

5.2 Cryptographic Protocols

PQTG uses ML-KEM-768 for key establishment and dual signatures (Falcon-512 for online, SPHINCS+-256f for long-term).

Algorithm 1 PQTG Handshake

1: $(pk_G, sk_G) \leftarrow \mathsf{ML\text{-}KEM.KeyGen}()$
2: $C \rightarrow G$: ClientHello
3: $G \rightarrow C$: pk_G, cert_G, σ_G (Falcon signature)
4: C verifies: $\mathsf{Falcon.Verify}(pk_{\text{CA}}, \text{cert}_G, \sigma_G) = 1$
5: $(ct, K) \leftarrow \mathsf{ML\text{-}KEM.Encaps}(pk_G)$
6: $C \rightarrow G$: ct
7: G: $K \leftarrow \mathsf{ML\text{-}KEM.Decaps}(sk_G, ct)$
8: Derive session keys: $K_{\text{enc}}, K_{\text{mac}} \leftarrow \mathsf{HKDF}(K)$

5.3 Protocol Specification

PQTG listens on `0.0.0.0:8443` (external) and connects to `127.0.0.1:443` (vendor API). Firewall blocks external port 443 access.

6 Security Analysis

Theorem 4 (PQTG IND-CCA2 Security)**.** *Under ML-KEM-768 IND-CCA2 security in QROM, PQTG key establishment achieves IND-CCA2 security against quantum adversaries.*

Proof. Security reduces to ML-KEM-768. If adversary $\mathcal{A}$ breaks PQTG with advantage ϵ, we construct $\mathcal{B}$ breaking ML-KEM: $\mathcal{B}$ receives challenge (pk^*, ct^*, K_b), simulates PQTG using pk^*, forwards $\mathcal{A}$'s guess. If $\mathcal{A}$ succeeds with ϵ, then $\mathcal{B}$ breaks ML-KEM with $\epsilon - \text{negl}(\lambda)$. By ML-KEM security, $\epsilon \leq \text{negl}(\lambda)$. □ □

Theorem 5 (PQTG EUF-CMA Security)**.** *Under Falcon-512 and SPHINCS+-256f EUF-CMA security in QROM, PQTG authentication achieves EUF-CMA security.*

Proof. Dual-signature construction: certificates contain Falcon-512 keys signed by SPHINCS+ CA. To forge, adversary must break either Falcon (for session forgery) or SPHINCS+ (for certificate forgery). By EUF-CMA security of both schemes, forgery probability is $\text{negl}(\lambda)$. □ □

Proposition 6 (Forward Secrecy)**.** *PQTG provides forward secrecy: compromise of long-term signing keys does not compromise past session keys (established via ephemeral ML-KEM).*

Proposition 7 (Post-Compromise Security)**.** *PQTG supports key rotation, enabling recovery from compromise.*

7 Implementation and Performance

7.1 Implementation

PQTG is implemented in Rust using: `pqcrypto` (NIST reference implementations), `rustls` (TLS 1.3), `tokio` (async runtime).

Open source: https://github.com/QuantumVerseProtocols/pq-transport-gateway

7.2 Performance Evaluation

Benchmarks on Intel Xeon 2.4 GHz:

Operation	**RSA-2048**	**ML-KEM-768**	**Speedup**
Key generation	50ms	0.02ms	2500×
Encapsulation	0.05ms	0.03ms	1.7×
Decapsulation	5ms	0.04ms	125×
Handshake	∼100ms	∼80ms	1.25×

Table 1: Performance: Classical vs. Post-Quantum

Post-quantum algorithms outperform RSA in most operations.

7.3 Deployment

PQTG supports major QKD vendors: ID Quantique (Cerberis XG), Toshiba QKD System, QuantumCTek. Deployment involves: install on QKD hardware, configure firewall, generate PQ certificates, add authorized keys.

8 Discussion

8.1 Practical Considerations

Hybrid mode: PQTG supports hybrid classical+PQ for transitional deployments, though pure PQ is recommended.

Certificate lifecycle: SPHINCS+ signatures enable long-term certificate validity (10+ years) without quantum vulnerability.

Vendor integration: PQTG requires no vendor modifications, only deployment as gateway.

8.2 Limitations

Trust assumptions: PQTG assumes NIST PQC algorithms are secure. Cryptanalytic advances could require algorithm updates.

Implementation attacks: Side-channel resistance requires constant-time implementations (available in `pqcrypto`).

Deployment complexity: Operators must manage PQ certificates and key rotation.

9 Conclusion

QKD systems rely on classical authentication vulnerable to quantum attacks, undermining their quantum security guarantees. We formalized this vulnerability (Theorem 2) and presented PQTG as a constructive solution providing post-quantum authentication with formal security proofs and practical implementation.

As quantum computing advances, transitioning QKD infrastructure to post-quantum authentication is essential. PQTG enables immediate mitigation through transparent deployment without vendor modifications.

9.1 Future Work

- Formal verification using Tamarin prover
- FIPS 140-3 certification
- Extended protocol support (ETSI QKD API)
- High-throughput optimization

Acknowledgments

We thank the NIST Post-Quantum Cryptography project and Open Quantum Safe initiative for foundational work in post-quantum cryptography.

References

[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Proc. IEEE Int. Conf. Computers, Systems and Signal Processing*, pp. 175–179, 1984.

[2] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, vol. 67, no. 6, p. 661, 1991.

[3] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, no. 1, p. 145, 2002.

[4] P. W. Shor and J. Preskill, "Simple proof of security of the BB84 quantum key distribution protocol," *Phys. Rev. Lett.*, vol. 85, no. 2, p. 441, 2000.

[5] V. Scarani et al., "The security of practical quantum key distribution," *Rev. Mod. Phys.*, vol. 81, no. 3, p. 1301, 2009.

[6] ETSI, "Quantum Key Distribution (QKD); Application Interface," *ETSI GS QKD 014 V1.1.1*, 2023.

[7] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[8] M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[9] NSA, "Quantum Computing and Post-Quantum Cryptography," *Cybersecurity Information Sheet*, 2022.

[10] NIST, "Post-Quantum Cryptography: Selected Algorithms 2024," *FIPS 203, 204, 205*, 2024.

[11] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid key exchange in TLS 1.3," *IETF Internet-Draft*, draft-ietf-tls-hybrid-design-09, 2023.

[12] D. Stebila and M. Mosca, "Post-quantum key exchange for the Internet and the Open Quantum Safe project," *Int. Conf. Selected Areas in Cryptography*, pp. 1–24, 2016.

[13] A. Langley, "Protecting Chrome Traffic with Hybrid Kyber KEM," *Google Security Blog*, 2023.

[14] F. Xu, X. Ma, Q. Zhang, H. K. Lo, and J. W. Pan, "Secure quantum key distribution with realistic devices," *Rev. Mod. Phys.*, vol. 92, no. 2, p. 025002, 2020.

[15] V. Makarov et al., "Creation of backdoors in quantum communications via laser damage," *Phys. Rev. A*, vol. 94, no. 3, p. 030302, 2016.

[16] T. Pornin, "New Efficient, Constant-Time Implementations of Falcon," *Post-Quantum Cryptography: 11th Int. Conf.*, pp. 307–324, 2020.

[17] M. Roetteler et al., "Quantum resource estimates for computing elliptic curve discrete logarithms," *Advances in Cryptology–ASIACRYPT*, pp. 241–270, 2017.

[18] M. Mehic et al., "Quantum key distribution: a networking perspective," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–41, 2020.

A Deployment Guide

Installation:

```
git clone https://github.com/QuantumVerseProtocols/pq-transport-gateway
cd pq-transport-gateway
cargo build --release
sudo ./install.sh
```

Configuration:

```
[proxy]
listen = "0.0.0.0:8443"

[qkd]
vendor_api = "https://127.0.0.1:443"

[security]
pq_kex = "ml-kem-768"
pq_sig = "falcon512"
```

Firewall:

```
# Block external access to vendor API
iptables -A INPUT -p tcp --dport 443 ! -s 127.0.0.1 -j DROP
# Allow PQTG
iptables -A INPUT -p tcp --dport 8443 -j ACCEPT
```

B Security Parameters

NIST security levels:

- Level 1: $\geq$ AES-128 (64-bit quantum security)
- Level 3: $\geq$ AES-192 (96-bit quantum security)
- Level 5: $\geq$ AES-256 (128-bit quantum security)

PQTG uses Level 3 (ML-KEM-768) for 50% margin above AES-128.