Post-Quantum Authentication for Quantum Key Distribution Control Channels

Sylvain Cormier
Paraxiom Technologies Inc.
sylvain@paraxiom.org

September 2025 (original), February 2026 (revised)

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—including those compliant with the ETSI GS QKD 014 key delivery API—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 prove (Theorem 2) that **any production QKD deployment using RSA/ECDSA authentication will be retroactively compromised once quantum computers become available**, even if the quantum channel itself is unconditionally secure. A harvest-now-decrypt-later adversary can capture classical TLS handshakes today, later forge certificates via Shor's algorithm, impersonate QKD endpoints, and extract all quantum-derived keys.

As a constructive solution, we present the Post-Quantum Transport Gateway (PQTG), a transparent security layer that replaces classical TLS authentication with NIST-standardized post-quantum primitives (ML-KEM-768, Falcon-512, SPHINCS+-256f) while maintaining backward compatibility with existing QKD infrastructure. PQTG requires no vendor firmware modifications—it deploys as a localhost gateway on existing QKD hardware.

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 Rust implementation with performance benchmarks showing post-quantum operations outperform RSA-2048

by up to 125×. Critical components are formally verified with 549 machine-checked theorems in Lean 4.

Keywords: Quantum Key Distribution, Post-Quantum Cryptography, ETSI QKD 014, authenticated channels, Shor's algorithm, ML-KEM, Falcon, SPHINCS+, harvest-now-decrypt-later

Executive Summary (for non-specialist readers)

The Problem

Every QKD system deployed today uses classical TLS certificates (RSA or ECDSA) to authenticate its control channel—the channel that transports quantum-derived encryption keys between endpoints. A future quantum computer can forge these certificates, impersonate QKD endpoints, and steal every key. **The quantum channel is secure. The classical channel next to it is not.**

Who is affected

- Every ETSI QKD 014-compliant deployment worldwide
- Vendors: Nokia (1830 PSS), ID Quantique (Cerberis XG), Toshiba QKD, QuantumCTek
- Operators: national quantum networks, banking infrastructure, defence communications
- Standard: ETSI GS QKD 014 V1.1.1 specifies REST/HTTPS for key delivery—inherently TLS-dependent

The Solution: PQTG

The Post-Quantum Transport Gateway (PQTG) is a software gateway that installs directly on QKD hardware. It replaces the classical TLS endpoint with post-quantum cryptography (NIST-standardized ML-KEM, Falcon, SPHINCS+). The vendor's classical API is never exposed to the network. No vendor firmware changes required. Open-source. Patent-free. MIT licensed.

Why this matters now

Adversaries can record classical TLS traffic from QKD control channels **today**. When quantum computers arrive, they can retroactively decrypt these recordings, forge certificates, and extract every quantum key ever transported over those channels. This is the harvest-now-decrypt-later (HNDL) threat. The window for mitigation is **before** data is captured, not after.

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 ETSI QKD 014 Control Channel

The ETSI GS QKD 014 standard [6] defines the application interface for QKD key delivery. It specifies a REST/HTTPS API through which key management systems retrieve quantum-derived keys from QKD endpoints. In production deployments, this API is authenticated using mutual TLS (mTLS) with X.509 client certificates signed by RSA or ECDSA.

This means the channel that *transports quantum keys* is itself protected only by classical cryptography. The quantum channel generates the keys; the classical channel delivers them. If the classical channel is compromised, the keys are compromised—regardless of the quantum channel's security.

1.3 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.

1.4 Affected Infrastructure

This vulnerability is not theoretical or limited to a single vendor. It affects every QKD deployment that uses the ETSI QKD 014 API with classical TLS, including:

- **Nokia 1830 PSS** — uses HTTPS/REST with X.509 mTLS for key rotation between the security management server and QKD endpoints
- **ID Quantique Cerberis XG** — ETSI 014-compliant REST API with TLS client certificates
- **Toshiba QKD System** — management API over HTTPS
- **QuantumCTek** — similar HTTPS-based key management
- **National quantum testbeds** — Canada (Kirq), UK (UKQN), Europe (EuroQCI) — all built on ETSI 014-compliant infrastructure

1.5 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.

6. **Formal verification**: Critical security properties are verified with 549 machine-checked theorems in Lean 4 [19], with zero unresolved assumptions.

1.6 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. No prior work has formalized this specific vulnerability in the context of ETSI QKD 014 deployments, nor provided a constructive mitigation with formal security proofs.

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:*

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

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. The ETSI GS QKD 014 standard specifies HTTPS as the transport for key delivery, making TLS authentication a *protocol requirement*, not merely an implementation choice.

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 per ETSI QKD 014), and Certificate Authority.

In a typical deployment, a key management system (e.g., Nokia 1830 PSS) communicates with QKD hardware via the ETSI 014 REST API over HTTPS. The API call `GET /api/v1/keys/{slave_SAE_ID}/enc_keys` retrieves quantum-derived keys. This call is authenticated by mutual TLS with X.509 client certificates.

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 handshakes on the ETSI 014 REST channel, including X.509 certificates and encrypted key-delivery payloads
2. Time t_1: $\mathcal{A}_{\text{future}}$ obtains quantum computer
3. Time t_1: Breaks RSA/ECDSA, recovers CA private key, forges certificates, impersonates QKD endpoints, extracts all quantum keys from $[t_0, t_1]$

3.3 Threat Scenarios

Harvest-Now-Decrypt-Later: Adversary captures certificates and encrypted sessions from the ETSI 014 channel, later uses Shor's algorithm to extract keys. Every quantum key transported over this channel during $[t_0, t_1]$ is compromised.

Certificate Authority Compromise: With sk_{CA} from Shor's algorithm, adversary generates valid certificates for any endpoint. All QKD endpoints trusting this CA are now impersonatable.

Long-Term Key Exposure: X.509 certificates valid for years become retroactively compromised. A certificate issued today with a 10-year validity period is a 10-year window of vulnerability.

Active Impersonation: After t_1, adversary can actively impersonate a QKD endpoint in real time, requesting keys from the ETSI 014 API with forged client certificates. The key management system (e.g., Nokia 1830) will deliver quantum keys to the adversary, believing it is communicating with a legitimate endpoint.

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 on the ETSI 014 channel.

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—including all current ETSI QKD 014-compliant systems—will be retroactively compromised once quantum computers become available.*

Remark 1. *This result applies regardless of the QKD protocol used (BB84, E91, CV-QKD, etc.), regardless of the QKD hardware vendor, and regardless of whether the quantum channel itself achieves perfect information-theoretic security. The vulnerability is in the classical authentication layer, not the quantum layer.*

5 Post-Quantum Transport Gateway Protocol

5.1 Design Overview

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

Architecture: External interface uses PQ-TLS; internal interface connects to vendor API via localhost; vendor TLS is never exposed externally. The gateway is transparent to both the QKD hardware (which sees

only a localhost client) and the key management system (which sees a PQ-authenticated endpoint).

Key design principle: PQTG requires zero modifications to vendor firmware or hardware. It installs as a software service on the same machine as the QKD endpoint, intercepting and re-securing the ETSI 014 API.

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. All ETSI 014 API calls (`/api/v1/keys/*`) are proxied transparently.

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/Paraxiom/pq-transport-gateway
License: MIT — patent-free, no licensing dependencies.

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. PQTG introduces **no performance penalty** compared to classical TLS—it is strictly faster for key operations.

7.3 Deployment

PQTG supports major QKD vendors: ID Quantique (Cerberis XG), Toshiba QKD System, QuantumCTek, and Nokia 1830 PSS. Deployment involves: install on QKD hardware, configure firewall, generate PQ certificates, add authorized keys. Total deployment time: under one hour per endpoint.

7.4 Formal Verification

Critical security properties of the Paraxiom cryptographic stack—including Falcon-512 parameter correctness, SPHINCS+ security bounds, ML-KEM field arithmetic, and transport frame integrity—are formally verified with 549 machine-checked theorems in Lean 4 [19] using Mathlib v4.27.0, with **zero unresolved assumptions** (`sorry`-free).

8 Discussion

8.1 Why Hybrid TLS Is Insufficient

A common response to the post-quantum threat is to deploy hybrid TLS, which adds post-quantum key exchange to TLS 1.3 while retaining classical authentication. This is insufficient for QKD control channels because:

1. Hybrid TLS typically retains RSA/ECDSA *signatures* for certificate authentication
2. Shor's algorithm breaks the signature scheme, enabling certificate forgery
3. With a forged certificate, an adversary can impersonate a QKD endpoint even if the key exchange is quantum-safe
4. The ETSI 014 API uses mutual TLS—both client *and* server certificates must be post-quantum

PQTG eliminates *all* classical cryptographic primitives from the authentication path.

8.2 No Alternative Solutions Exist

To our knowledge, PQTG is the only deployed mitigation for this vulnerability:

- **Vendor firmware updates**: Would require each QKD vendor (Nokia, IDQ, Toshiba, etc.) to independently implement and certify PQ-TLS. No vendor has announced plans to do so.
- **Pre-shared keys from QKD**: Creates a circular dependency—QKD keys cannot authenticate the channel that delivers QKD keys.
- **VPN tunneling**: Wrapping ETSI 014 in a PQ VPN is functionally equivalent to PQTG but less integrated, less auditable, and not purpose-built for QKD infrastructure.

8.3 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.

QRNG seeding: All PQ key generation can optionally be seeded from quantum random number generators (QRNG), providing hardware-grade entropy independent of algorithmic PRNGs.

8.4 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 proved that **every ETSI QKD 014-compliant deployment using RSA/ECDSA authentication is retroactively compromisable** (Corollary 3).

We presented PQTG as a constructive solution providing post-quantum authentication with formal security proofs, practical implementation, and

formal verification in Lean 4. PQTG is the only known deployed mitigation for this vulnerability. It requires no vendor modifications, introduces no performance penalty, and is available as open-source software under the MIT license.

As quantum computing advances and nations invest in quantum communication infrastructure, ensuring that the classical layers of these systems are quantum-safe is not optional—it is urgent. The window for harvest-now-decrypt-later mitigation is closing.

9.1 Future Work

- FIPS 140-3 certification of PQTG
- Extended formal verification of the PQTG handshake protocol in Lean 4
- Integration with ETSI QKD 004 (device interface) in addition to 014
- Standardization proposal for post-quantum ETSI QKD API authentication
- High-throughput optimization for backbone QKD deployments

Acknowledgments

We thank the NIST Post-Quantum Cryptography project and Open Quantum Safe initiative for foundational work in post-quantum cryptography. We acknowledge the operators of Canada's Kirq quantum testbed for providing a real-world environment for QKD infrastructure research.

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.

[19] S. Cormier, "549 Lean 4 Theorems: Formal Verification Across Seven Post-Quantum Systems," *Zenodo*, DOI: 10.5281/zenodo.14805498, 2026.

[20] Empire Club of Canada, "Quantum Power and National Security: Canada's New Strategic Imperative," Panel discussion, February 25, 2026.

A Deployment Guide

Installation:

```
git clone https://github.com/Paraxiom/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.