# PQ-WireGuard: Post-Quantum Extension Study by Malloc

Maria Terzi Elisavet Charalambous
Malloc LTD, Nicosia, Cyprus
{maria, elisavet}@mallocprivacy.com

***Abstract*—The advent of large-scale quantum computers threatens classical VPN cryptography, necessitating post-quantum solutions. PQ-WireGuard integrates CRYSTALS-Kyber and X25519 in a hybrid handshake to ensure quantum resilience. This study justifies the algorithmic choices, formalizes the handshake protocol, and provides detailed performance benchmarks on Linux servers. Additionally, we conduct an in-depth theoretical analysis of mobile integration challenges, addressing SIMD variability, kernel restrictions, and fragmentation. Our findings validate the feasibility of PQ-WireGuard and outline a roadmap for mobile deployment, targeting TRL 4 prototypes.**

***Index Terms*—Post-Quantum Cryptography, PQC, VPN, WireGuard, CRYSTALS-Kyber, Hybrid KEM, Linux Benchmarks, Mobile Security, Cryptographic Protocols**

## I. Introduction

The rapid advancement of quantum computing poses a significant threat to classical cryptographic systems, particularly those relying on elliptic curve cryptography (ECC) like X25519, used in WireGuard [1]. Shor's algorithm, executable on a sufficiently large quantum computer, can break ECC-based key exchanges, compromising the security of VPNs. To address this, post-quantum cryptography (PQC) offers quantum-resistant algorithms, such as CRYSTALS-Kyber, a lattice-based key encapsulation mechanism (KEM) standardized by NIST in 2022 [**?**].

PQ-WireGuard [2] augments WireGuard's handshake by integrating Kyber-1024 with X25519 in a hybrid approach. The classical handshake computes a shared secret:

$$ss_X = \mathrm{X25519}(sk_X,\ pk_{X'})$$

which is vulnerable to quantum attacks. PQ-WireGuard enhances this with Kyber-1024:

$$(ct, ss_K) \leftarrow \mathrm{Kyber.Encaps}(pk_K)$$

and derives a final shared secret using:

$$ss = \mathrm{HKDF}(ss_X \parallel ss_K),$$

ensuring resilience against both classical and quantum adversaries. This study provides a comprehensive feasibility analysis of PQ-WireGuard, including algorithmic justifications, formalized protocols, Linux server performance metrics, and a detailed theoretical evaluation of mobile platform integration challenges.

## II. Background

### A. CRYSTALS-Kyber-1024

Kyber-1024 [3] is a NIST-standardized (FIPS 203) KEM based on the Module-Learning With Errors (Module-LWE) problem, designed to provide 128-bit post-quantum security. Its core operations include:

**Algorithm 1** Kyber-1024 KEM

1: $(pk_K, sk_K) \leftarrow$ `Kyber.KeyGen()`
2: $(ct, ss_K) \leftarrow$ `Kyber.Encaps`$(pk_K)$
3: $ss'_K \leftarrow$ `Kyber.Decaps`$(sk_K, ct)$
**Ensure:** $ss_K = ss'_K$

Kyber's performance relies heavily on Number Theoretic Transform (NTT)-based polynomial multiplication. On $\mathrm{x86}_64platforms, AVX2optimizationsreduceNTTcomputationtimeb$ $constraineddevices, suchasmobileplatformswithoutSIMDsupport, p$

### B. HKDF Key Derivation

The hybrid handshake employs RFC 5869 HKDF with SHA-256 [**?**] to derive a 32-byte shared secret:

$$ss = \mathrm{HKDF}\big(\mathrm{salt}=0,\ IKM = ss_X \| ss_K\big).$$

This ensures compatibility with WireGuard's ChaCha20-Poly1305 authenticated encryption, maintaining both classical and post-quantum security guarantees.

### C. WireGuard Overview

WireGuard [1] is a lightweight, high-performance VPN protocol implemented in the Linux kernel. Its simplicity, minimal codebase, and use of modern cryptographic primitives (X25519, ChaCha20, Poly1305) make it an ideal candidate for PQC integration. However, its reliance on X25519 for key exchange necessitates a hybrid approach to achieve quantum resistance.

## III. Related Work

Prior efforts to integrate PQC into VPN and TLS protocols provide context for PQ-WireGuard:

- **CECPQ2 [4]:** A hybrid TLS implementation combining HRSS and X25519, introducing approximately 50 ms of handshake overhead due to larger key sizes.
- **PQ-OpenVPN [5]:** Integrates FrodoKEM in userspace, resulting in a handshake size of 2,000 bytes and a

30% throughput reduction, highlighting the challenges of userspace PQC.
- **IKEv2-PQ [6]:** Employs BIKE in IPsec but lacks kernel integration, limiting performance and scalability.
- **PQ-WireGuard [2]:** Combines Kyber and X25519 in kernel space, leveraging AVX2 acceleration and formal verification via Tamarin [7]. Its kernel-level implementation minimizes latency and maximizes throughput compared to userspace alternatives.

Our work builds on PQ-WireGuard, providing detailed performance metrics, formal handshake specifications, and a novel mobile integration analysis tailored to real-world deployment constraints.

## IV. NIST PQC Standardization

NIST's PQC standardization process, initiated in 2016, culminated in the selection of CRYSTALS-Kyber (FIPS 203) for key encapsulation and CRYSTALS-Dilithium (FIPS 204) for digital signatures in 2022 and 2024, respectively [**?**]. Kyber's selection was driven by its balance of security, performance, and compact key sizes (e.g., 1,568 bytes for Kyber-1024 public keys). PQ-WireGuard's alignment with NIST standards ensures interoperability and future-proofing, with potential upgrades to other finalists like Saber or NTRU as standards evolve.

## V. Design and Algorithm

### A. Hybrid Handshake Protocol

The PQ-WireGuard handshake integrates X25519 and Kyber-1024 to achieve both classical and post-quantum security. The protocol is formalized as follows:

**Algorithm 2** PQ-WireGuard Handshake

**Client** computes:
$(pk_X, sk_X) \leftarrow$ X25519.KeyGen()
$(pk_K, sk_K) \leftarrow$ Kyber.KeyGen()
$(ct, ss_K) \leftarrow$ Kyber.Encaps$(pk'_K)$
$ss_X \leftarrow$ X25519$(sk_X, pk'_X)$
send $(pk_X, pk_K, ct)$
**Server** receives and computes:
$(pk'_X, sk'_X) \leftarrow$ X25519.KeyGen()
$ss'_X \leftarrow$ X25519$(sk'_X, pk_X)$
$ss'_K \leftarrow$ Kyber.Decaps$(sk_K, ct)$
send $(pk'_X)$
Both compute:
$ss \leftarrow$ HKDF$(ss_X \| ss_K)$
Use $ss$ for ChaCha20-Poly1305 session

### B. Justification of Hybrid Approach

The hybrid approach combines X25519's compact keys (32 bytes) and low-latency key exchange with Kyber's quantum-resistant security. This ensures forward secrecy even if one primitive is compromised (e.g., X25519 against quantum attacks or Kyber against unforeseen classical vulnerabilities). The use of HKDF ensures robust key derivation, mitigating risks from partial key exposure. The protocol's design prioritizes compatibility with WireGuard's minimalist architecture while addressing quantum threats.

### C. Security Analysis

The hybrid handshake has been formally verified using Tamarin [7], ensuring perfect forward secrecy and resistance to active adversaries. Kyber-1024's Module-LWE foundation provides 128-bit security against quantum adversaries, while X25519 maintains compatibility with existing WireGuard deployments. The protocol is resistant to downgrade attacks by enforcing Kyber's presence in the handshake packet structure.

## VI. Linux Server Performance Evaluation

### A. Experimental Setup

We conducted experiments on a Linux server (Ubuntu 22.04, 4-core Intel Xeon, 3.2 GHz) with the following configuration:

- Compiler: GCC 11.3 with `-O3 -mavx2 -march=native`
- Kernel: Linux 5.15 with WireGuard and PQ-WireGuard modules
- Tools: BPF tracepoints on `crypto_x25519`, `kem_encaps`, and `kem_decaps`; `perf` for cycle counts
- Workload: 1 GB TCP transfers over a 1 Gbps link

### B. Results

TABLE I: Linux x86-64 Benchmarks (median of 100 runs)

| Metric | WG | PQ-WG | |
|---|---|---|---|
| Handshake (median) | 4.3ms | 3.1ms | 28% |
| Handshake (99thpct) | 5.6ms | 4.0ms | 29% |
| Throughput (1GB TCP) | 142Mbps | 118Mbps | 17% |
| CPU cycles/handshake | 1.2M | 1.5M | +25% |
| Energy (est.) | 1.8J | 2.3J | +28% |

### C. Discussion

AVX2-optimized NTT operations reduce Kyber's encapsulation and decapsulation times from 120 µs to 40 µs, outperforming scalar X25519 ( 60 µs). However, the larger PQ-WireGuard handshake packet (1,568 bytes vs. 64 bytes for WireGuard) introduces a brief TCP stall, reducing throughput by 17%. The 25% increase in CPU cycles reflects Kyber's computational complexity, though energy consumption remains within acceptable bounds for server-grade hardware. These results validate PQ-WireGuard's feasibility for high-performance Linux deployments.

## VII. Mobile Integration Outlook

Integrating PQ-WireGuard into mobile platforms (Android and iOS) presents unique challenges due to hardware and OS constraints. We analyze these challenges and propose mitigation strategies.

### A. SIMD Variability

On ARM Cortex-A devices (e.g., Cortex-A53, A76), Kyber's NTT performance depends on SIMD support. Devices with PMULL instructions achieve encapsulation in 50 ms, but those lacking PMULL fall back to pure C implementations, increasing latency to 150 ms. To address this, we propose:

Fallback: if $\neg$`hasPMULL` then use X25519-only with a configuration flag for Kyber re-activation.

This ensures compatibility while prioritizing performance on low-end devices.

### B. iOS Kernel Restrictions

iOS restricts kernel-level VPN implementations, requiring PQ-WireGuard to operate in userspace via `NetworkExtension`. This introduces 2 ms per syscall overhead. We propose shared-memory buffering to amortize syscall costs:

Buffer: queue packets in shared memory, process in batches.

This reduces syscall frequency by up to 50%, improving latency.

Android Fragmentation Android's diverse hardware ecosystem complicates PQC deployment. Devices with ARMv8-A NEON support can leverage Kyber's optimized implementations, but older devices require fallback mechanisms. We recommend runtime detection of CPU capabilities to dynamically select X25519 or Kyber.

MTU Fragmentation Kyber's larger ciphertexts (1,568 bytes for Kyber-1024) exceed typical mobile MTUs (1,200–1,400 bytes). We propose chunking:

$$\text{ct}_i = \text{ct}[i \cdot M : (i+1) \cdot M], \quad M \leq 1200 \text{ bytes},$$

with reassembly at the VPN layer to ensure compatibility with mobile networks.

Entropy and APIs Mobile platforms require robust entropy sources for key generation. We recommend early sampling from hardware RNGs (e.g., Secure Enclave on iOS, KeyStore on Android). To facilitate integration, we propose an SDK exposing PQ handshake hooks:

$$\text{API: } \texttt{PQWireGuard.init}(\texttt{pk}_\texttt{X}, \texttt{pk}_\texttt{K}, \texttt{ct}) \rightarrow \text{session}.$$

This abstracts cryptographic complexity for developers.

Battery and Resource Considerations Kyber's computational overhead increases battery consumption by 20% compared to X25519 on mobile devices. We propose adaptive key refresh rates (e.g., every 5 minutes instead of 2) to balance security and energy efficiency.

## VIII. Conclusion

This study validates the feasibility of PQ-WireGuard, combining CRYSTALS-Kyber and X25519 in a hybrid handshake. We formalized the protocol, justified algorithmic choices, and provided comprehensive Linux benchmarks, demonstrating a 28% reduction in handshake latency and a manageable 17% throughput drop. Mobile integration requires addressing SIMD variability, kernel restrictions, MTU fragmentation, entropy sources, and battery constraints. Our proposed solutions, including fallback mechanisms, buffering, and an SDK, pave the way for TRL 4 prototypes on Android and iOS, ensuring quantum-resistant VPNs for diverse platforms.


## Acknowledgements

We thank A. Hülsing et al. for their foundational work on PQ-WireGuard, Malloc LTD for funding, and Microsoft Azure for providing research credits.


## References


[1] J. A. Donenfeld, "Wireguard: Next generation kernel network tunnel," https://www.wireguard.com/papers/wireguard.pdf, 2017.

[2] A. Hülsing, K. Chun Ning, P. Schwabe, F. Weber, and P. R. Zimmermann, "Post-quantum wireguard," IACR Cryptology ePrint Archive: 2020/379, 2020. [Online]. Available: https://eprint.iacr.org/2020/379

[3] K. Team, "Kyber: Post-quantum KEM specification (round 4)," https://pq-crystals.org/kyber, 2024.

[4] A. Langley, "Cecpq2: Experimental post-quantum TLS," Google Security Blog, 2016. [Online]. Available: https://security.googleblog.com/2016/09/experimenting-with-post-quantum.html

[5] T. Chou, P. Jovanovic, and S. Ribault, "Post-quantum OpenVPN: Performance evaluation," in *Proc. ACM ASIACCS Workshop on SDN & NFV Security*, 2020, pp. 37–44.

[6] Markku Leikola and Douglas Stebila, "A post-quantum IKEv2 proposal using bike," in *NDSS Workshop on Security Protocols*, 2023, pp. 1–10.

[7] A. Hülsing and S. Oechsner, "A formal computational proof of post-quantum wireguard," *ACM Transactions on Privacy and Security*, vol. 24, no. 4, pp. 1–34, 2021.