Open-Source Post-Quantum Encryptor: Design, Implementation and Deployment

Petr Tuma[1][a], Jan Hajny[1][b], Petr Muzikant[2][c], Jan Havlin[1][d], Lukas Malina[1][e], Patrik Dobias[1][f] and Jan Willemson[2][g]

[1]*Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic*

[2]*Cybernetica, Tartu, Estonia*

Keywords: Cryptography, Key Establishment, Post-Quantum Cryptography, Security, Quantum Key Distribution (QKD), Network Encryption.

Abstract: This article describes an open-source quantum-resistant network traffic encryptor for the Linux platform. Our encryptor uses a combination of quantum and post-quantum key establishment methods to achieve quantum resistance combined with a fast encryption speed of AES to make quantum-resistant encryption readily available to the public. The packet-by-packet encryption architecture ensures that every bit of information is properly authenticated and encrypted. The combination of multiple key sources further increases the encryptor's security – be it elliptic curve-based (Elliptic Curve Diffie Hellman, ECDH), quantum (Quantum Key Distribution, QKD) or post-quantum (CRYSTALS-Kyber). Without knowing all the keys obtained from different types of key sources, the final hybrid encryption key can only be obtained by brute-force means. Our contribution is very practical as the encryptor has reasonable performance, despite not being part of the Linux kernel.

1 INTRODUCTION

There are two ways to achieve quantum-safe key establishment to build a secure communication channel. The first approach is using Quantum Key Distribution (QKD) that requires usually expensive end devices and provides only key establishment between two end points. The second approach is deploying more flexible post-quantum cryptography (PQC) that enables a quantum-resistant key establishment, offers algorithms for digital signatures, and is simple to deploy on general computing platforms.

The aim of this work is to support the expansion of the usage of quantum-resistant cryptography by a wider public and small and medium-sized companies.

2 STATE OF THE ART

Commonly used security protocols for traffic encryption nowadays use either integer factorization or discrete logarithm problems to establish encryption keys for fast symmetric ciphers. For example, a secure connection to a web page uses the SSL/TLS protocol, which usually utilizes RSA for authentication, Diffie-Helmann key exchange, and AES for the actual payload encryption. The mentioned algorithms have several variants, and the resulting cipher suites are agreed upon between the communicating end nodes in the handshake stage of the protocol.

Similar methods are also used for other secure channel establishment protocols, including Virtual Private Networks (VPN), such as IPSec (Frankel and Krishnan, 2011), OpenVPN[1] or WireGuard[2].

2.1 Quantum-Resistant Encryption Solutions

There have already been a few quantum-resistant encryptors released on the market, including varia-

[a] https://orcid.org/0009-0009-8458-9361
[b] https://orcid.org/0000-0003-2831-1073
[c] https://orcid.org/0009-0008-7439-9508
[d] https://orcid.org/0009-0009-8025-9701
[e] https://orcid.org/0000-0002-7208-2514
[f] https://orcid.org/0000-0002-7321-7003
[g] https://orcid.org/0000-0002-6290-2099

[1] https://openvpn.net/
[2] https://www.wireguard.com/

Tuma, P., Hajny, J., Muzikant, P., Havlin, J., Malina, L., Dobias, P. and Willemson, J.
Open-Source Post-Quantum Encryptor: Design, Implementation and Deployment.
DOI: 10.5220/0012839200003767

In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 826-831
ISBN: 978-989-758-709-2; ISSN: 2184-7711

tions of OpenVPN and WireGuard. The OpenVPN variant uses Frodo or SIKE algorithms for key exchange. Both of these algorithms were submitted to the NIST's Post-Quantum Cryptography Standardization project, but none of them were chosen for standardization, and SIKE was even found to be insecure. PQ-WireGuard uses a combination of two Key Encapsulation Mechanisms (KEM), namely Classic McEliece and a passively secure variant of Saber.

Generally, post-quantum VPN software and hardware solutions are also offered by security vendors, e.g., QuantumNova's QS-P Network[3], or ExpressVPN's Lightway[4]. The system called FSP3000 ConnectGuard [5] claims to be the first in the world (as of July 2021) to provide the Layer 1 encryption for high-speed optical networks (up to 400 Gbit/s) with post-quantum cryptography support. Further, the Ribbon's Apollo solution[6] also provides the layer 1 optical encryption with AES-256-GCM and the parallel operation of a post-quantum cryptography key exchange mechanism similar to the Diffie-Hellman protocol. Nevertheless, closer specifications of those encryptors are usually not public.

2.2 Related Work

Currently, several research works deal with PQC deployment in VPN and security protocols. For example, in 2019 van Heesch *et al.* (van Heesch et al., 2019) described how OpenVPN and OpenSSL can be configured to establish quantum-safe connections, and provided the performance analysis of PQC KEMs and signatures on OpenVPN in TLS 1.2 and TLS 1.3. In 2022, Bae *et al.* (Bae et al., 2022) delivered state-of-the-art performance evaluation results of PQC algorithms in the IPsec protocol, namely, in the strongswan library implementation. They showed the trade-offs between the security level and performance of PQC algorithms in the IPsec protocol. In 2022, Marrok *et al.* (Marrok et al., 2022) presented a hybrid cryptography approach to integrate post-quantum security into VPN protocols. Their approach utilizes the Kyber KEM in combination with the WireGuard VPN protocol.

Recently, Cano Aguilera *et al.* (Aguilera et al., 2023) introduced a quantum resilient secure end-to-end communication solution based on PQC algorithms between two data-processing units (DPU) employing on-board ARM processors in 2023. They used CRYSTALS-Kyber (Avanzi et al., 2019) for KEM, CRYSTALS-Dilithium (Ducas et al., 2018) for signing, and AES-256 for encryption. While they used DPUs on servers for the acceleration of PQC schemes, they did not investigate a hybrid approach.

Another relevant work published by Schatz *et al.* (Schatz et al., 2023) in 2023 dealt with combining orthogonal methods for quantum-resistant key exchanges, i.e., PQC, quantum key distribution (QKD) and multipath key reinforcement (MKR) in IKE protocols. They presented mainly the design of the IKE proxy concept for virtual private networks in the quantum era and concrete steps for key combining. However practical implementation and performance results were not provided.

2.3 Contribution

In this article, we describe our open-source implementation of a site-to-site quantum-resistant network traffic encryption system for Linux platforms. Our system uses both quantum and post-quantum cryptography algorithms, which are currently chosen for standardization by NIST [7], as a quantum-resistant key establishment. Established keys are combined into a hybrid key, which provides another layer of security: Without the knowledge of each component, the attacker is unable to calculate the hybrid key. The payload itself is encrypted with the AES symmetric algorithm for faster encryption speed. The main contribution of the paper is the fast open-source implementation of the encryptor including the PQC and QKD methods, and practical evaluation of the resulting software in real networks. Our implementation is very simple to deploy and freely accessible at (Tuma and Havlin, 2024). By sharing our implementation and this paper, we follow the main objective, which is the support of PQC deployment and integration in practice.

3 PROPOSED ARCHITECTURE

In this section, we present the cryptographic architecture of our site-to-site encryption system, that is depicted in Fig. 1 as Encryption Gateways. We also describe our design goals and selection of cryptographic primitives.

3.1 Design Goals

During the creation of the cryptographic architecture, we considered the following main design goals:

[3]https://quantumnova.xyz/itsolutions/offer/pqvpn

[4]https://www.expressvpn.com/lightway

[5]https://www.adva.com/

[6]https://ribboncommunications.com/

[7]https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization

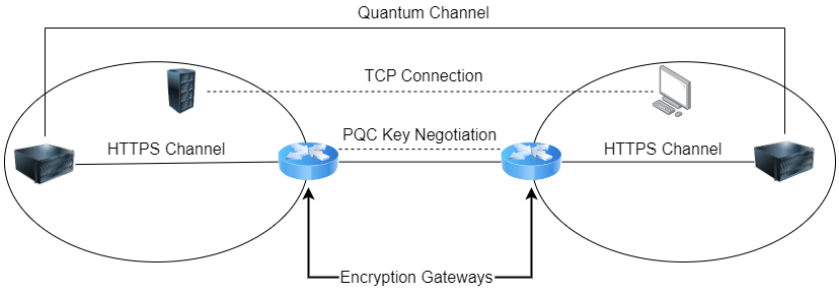

Figure 1: General Architecture.

- **Post-Quantum Security:** Our approach combines quantum-resistant schemes mitigating the risk of quantum attacks. Moreover, we deployed only cryptographic primitives that have no known vulnerabilities and are considered secure for the medium-term future. Our encryptor achieves IND-CCA security against quantum adversaries.
- **Compliance:** Only primitives that comply with the recommendations of renowned authorities are used in our implementation.
- **Cryptographic Flexibility:** Cryptographic flexibility means easy and fast replacement of cryptographic components in case any weaknesses arise. Should an algorithm become vulnerable, it must be easy to replace without affecting the remaining components in the resulting system.

3.2 Selected Primitives

We list all the necessary components of the hybrid key establishment system. Our system is composed of:

- **Quantum Key Distribution (QKD) Key Source:** Our system is compatible with any QKD system that can provide keys in JSON format through its API using the ETSI GS QKD 014 standard (ETSI, 2019).
- **Classical Cryptography Key Source:** Key source can be used along with other key sources or used instead of a QKD key source to remove the dependency on expensive hardware – this makes the encryptor a more viable solution from a market point of view. Elliptic-Curve Diffie Hellman was selected as it represents the current standard for key establishment. Our encryptor uses a 512-bit implementation of the sect571k1 curve.
- **PQC Key Source:** CRYSTALS-Kyber (Avanzi et al., 2019) was selected as the post-quantum source of keys. CRYSTALS-Kyber is currently the only key establishment mechanism approved for standardization by the NIST PQC competition. The system uses the Kyber-768 implementation, which aims to be as secure as AES-192. Key sizes are 1184 bytes for the public key and 2400 bytes for the secret key. Our system establishes two shared secrets via CRYSTALS-Kyber, which are then used in encryption and decryption key derivation. Encryptor currently lacks a quantum resistant authentication mechanism as public keys are not backed by certification authorities or chain of trust. One way to mitigate the possibility of man-in-the-middle attacks is to use long-term pre-shared public keys.
- **Symmetric Block Cipher:** We use the Advanced Encryption Standard (AES) algorithm in the Galois-Counter Mode (GCM) which provides both confidentiality and integrity of transferred data. The 256-bit variant is considered quantum safe (Bonnetain et al., 2019).
- **Key Management System:** the Key Management System (KMS) provides the logic for the derivation and updating of encryption and decryption keys. Data encrypted with AES-GCM-256 should be encrypted using a unique combination of the {nonce, key} pair. This property is achieved with periodic key updates.

3.3 Practical Implementation

The implementation of our system is divided into server and client sides of communication. The implementation can be divided into four components.

- **Key Establishment:** The system establishes three keys – one with QKD, one with ECDH, and one with PQC. The QKD key is obtained through REST APIs from QKD servers. Every key from the QKD source comes with an ID to ensure that both sides receive the same key. The client sends GET request to QKD-Alice to obtain the key and key ID, which is transferred to the server. The server sends POST request with the key ID to QKD-Bob, and receives the same key as the client. The classical asymmetric key

is obtained by ECDH-512 key exchange on the sect571k1 curve. The PQC key is obtained with CRYSTALS Kyber-768 key encapsulation mechanism. The client generates public and private keys. The public key is sent to the server. The server generates a 256-bit secret, encrypts it with the client's public key, and sends the encrypted secret to the client. The client decrypts the message with the private key, resulting in both parties obtaining the same key. The library used for CRYSTALS Kyber can be found at (Roy, 2023).

- **Key Combining:** The established keys are combined into one hybrid key. The key combination ensures that even with the compromised input key, an attacker is still unable to decrypt the communication between the endpoints. The combiner uses the 512-bit version of SHA3 as the hash function (SHA3) and as the basic component in the HMAC function. The combiner takes the established keys from the partial key establishment methods (K_1 from Kyber, K_2 from ECDH, K_3 from QKD) and publicly known values transferred during these methods as inputs. The chosen 3-key combiner method is described in Ricci *et al.* (Ricci et al., 2024) that compared the benefits and drawbacks of eleven combiner methods from simple XOR and PRF approaches to more robust nested PRF approaches. The brief comparison of various key combiners and their security robustness can be also found in (Rossi, 2023). The combiner is based on the extension of the dual-PRF combiner to work with three keys as inputs. This is the simplest known construction to work with three keys as inputs that is secure in the quantum standard model and is indistinguishable under Chosen-Ciphertext Attack (IND-CCA). The full description, comparison, and security analysis of the 3-key combiner method can be found in (Ricci et al., 2024).
- **Packet Encryption:** Our system utilizes virtual interfaces to intercept packets for secure transfer. Packets are encrypted with the AES-GCM-256 algorithm utilizing the hybrid key described above. AES-GCM-256 uses 16-byte long nonce value and 16-byte long message authentication. Every packet is encrypted individually and transferred through a public network in a tunneling fashion. Encrypted packets are transferred using the UDP protocol between the gateways. The encrypted packet structure is depicted in Figure 2. AES-GCM was implemented with the Crypto++ library found at (Wei, 2023).
- **KMS – Rekey:** Due to the properties of AES-GCM, the same combination of key and nonce cannot be used for encryption indefinitely. Because of this, our system implements a rekeying mechanism. Rekeying obtains a new key from the QKD system and combines it with the PQC key, which results in a new hybrid key to be used for encryption. AES-GCM can encrypt a maximum of 2^{32} messages before rekeying. Considering the packet size of 1500 bytes and encryption speed of 1 Gbps, the encryptor needs to change the key at least once approximately every 14 hours[8]. The rekeying mechanism is by default triggered every hour.

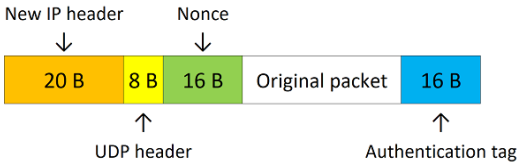

Figure 2: Encrypted packet structure.

3.4 System Architecture

Our system architecture deploys software implementations of a client and a server (labeled as trust zones). Encryptor implementation uses two network interfaces. The first interface is used for communication between encryptor parts and encrypted data transfer. The second interface provides a connection to the internal network and source of quantum-generated keys. The connection between the QKD points uses its own optical connection. Key-generation blocks (Kyber and QKD) generate independent 256-bit keys, which are then forwarded to the key combiner. The key combiner combines the keys and outputs the 256-bit hybrid encryption and decryption keys. Keys are used by AES for data encryption.

4 PERFORMANCE

This section presents the measured performance results such as throughput, delay, and the influence of rekeying.

- **Throughput** is one of the main properties of every encryption system. Our system was tested in a virtualized environment to eliminate transfer delays between the communicating parties as much as possible. The whole architecture was virtualized using VirtualBox[9] and Intel Core i7 1065G7 Ice Lake processor. The testing scenario was site-

[8] $1500 \cdot 8 \cdot 2^{32} / 10^9 / 3600 \approx 14.31$h

[9] https://www.virtualbox.org/

to-site with 2 gateways and 2 clients. The gateways were given 2 threads each, and the clients were given 1 thread each. Throughput was measured with iperf3[10] over the course of 30 minutes, and compared with other popular encryption solutions. The results can be seen in Figure 3. The encryptor performed better than OpenVPN, which is limited to running on one thread only, but still was no match for Wireguard and its low-level component implementation within the Linux kernel.

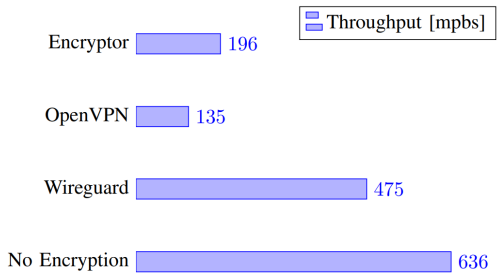

Figure 3: Throughput measurement.

- **Round-Trip Time** (RTT) is the time from signal transmission until receiving the response. Figure 4 displays a comparison between the solutions; the presented values are averages of 15 consecutive pings.

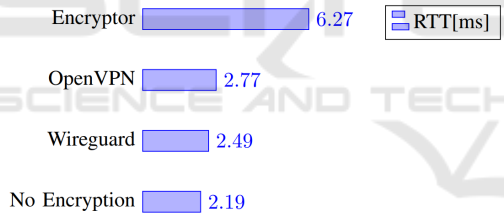

Figure 4: Round-trip time measurement.

Our encryptor has a significantly greater RTT compared to other solutions. This is caused by the encryptor scalability logic – the main program thread creates encryption threads upon data receipt, which is a source of increased RTT.

- **Rekeying:** The encryptor changes encryption keys regularly, which causes packets sent during this process to fail the integrity check due to a key mismatch.

5 CZECH-ESTONIAN PILOT

To evaluate the software in a real deployment, we selected geographically distant areas (Czech Republic and Estonia) and established a quantum-safe communication channel between them. The challenges of selecting distant locations are a higher delay and a much higher probability of packet reordering, which both may cause significant problems in data transmission, in particular of voice and live video. We selected Nextcloud Talk, an open-source live collaboration solution, as the application to test a high-volume transmission.

5.1 Architecture and Setup

We began with configuring virtual machines (4 virtual CPU, 4 GB of RAM and 40 GB of disk space) on both sides with connectivity to target pilot participants. Afterwards, the Post-Quantum Encryptor repository was cloned on both virtual machines and the software was installed using available scripts setting the other network's address spaces as an argument. Then, we have established an additional networking route rule at the Nextcloud Server as well as the participants' computers. Without it, the devices would try to send packets (targeted to the other network) to the default gateways which would not know what to do with them. Next, we have lowered the Maximum Transmission Unit (MTU) value on the Nextcloud Server, participant's computers, and the TUNTAP interfaces on both virtual machines to ensure the transfer of the encrypted packets (with additional header) over the internet.

5.2 Demonstration and Performance

In order to demonstrate the functionality of the encryptor, we have conducted both a voice call and a video call between two participants using the NextCloud Talk feature. *Both test instances were stable and without any voice or video stutters.*

The key establishment described in section 3.3 (excluding QKD keys) took on average 213.1 ms with a standard deviation of 5.4 ms on one machine, and 163.2 ms with a standard deviation of 2.1 ms on the other machine. Differences between machines are most probably caused by having access to different CPU models.

Round Trip Time between two machines (using an ICMP echo and reply) averaged at 48.3 ms with a standard deviation of 0.6 ms.

6 FUTURE EXTENSIONS

In this section, we list potential future extensions:

- Kernel Module: The system could be implemented as a kernel module to eliminate switching

[10]https://iperf.fr/

between the user and kernel spaces, thus increasing the throughput of the file transfer.

- Authentication Improvement: Authentication of the encryptor sides is not currently quantum resistant if pre-shared PQC keys are not used. This extension would add a certification authority and certificates with PQC elements for authentication to mitigate the potential man-in-the-middle attacks by a quantum adversary.
- Hardware Acceleration: The software may be adapted to be compatible with hardware-accelerated solution described in (Ricci et al., 2024).

7 CONCLUSION

In this article, we introduced an open-source quantum-resistant encryption system for real-time data transfer that can be used in site-to-site settings. The system utilizes a combination of several different quantum-resistant key-establishment methods to add extra layers of security. The software can be used as a low-cost demonstrator and experimental tool for testing PQC. This could also be used for system demonstration to companies seeking high throughput before purchasing high-speed solutions like FPGA-accelerated solutions. Therefore, it serves as a tool to raise awareness and support of post-quantum cryptography deployment and integration to existing systems. The implementation of the proposed system is freely accessible at (Tuma and Havlin, 2024).

ACKNOWLEDGEMENTS

Funded by the European Union under Grant Agreement No. 101087529. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

This work is supported by the Ministry of the Interior of the Czech Republic under Grant VJ01010008.

REFERENCES

Aguilera, A. C., Clemente, X. A. i., Lawo, D., Monroy, I. T., and Olmos, J. V. (2023). First end-to-end pqc protected dpu-to-dpu communications. *Electronics Letters*, 59(17):e12901.

Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. (2019). Crystals-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2(4):1–43.

Bae, S., Chang, Y., Park, H., Kim, M., and Shin, Y. (2022). A performance evaluation of ipsec with post-quantum cryptography. In *International Conference on Information Security and Cryptology*, pages 249–266. Springer.

Bonnetain, X., Naya-Plasencia, M., and Schrottenloher, A. (2019). Quantum security analysis of aes. *IACR Transactions on Symmetric Cryptology*, 2019(2):55–93.

Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. (2018). Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268.

ETSI (2019). Etsi gs qkd 014: Quantum key distribution (qkd) protocol and data format of rest-based key delivery api. https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf.

Frankel, S. and Krishnan, S. (2011). Ip security (ipsec) and internet key exchange (ike) document roadmap. Technical report.

Marrok, A., Boukhelef, S., and Chikouche, N. (2022). Pqh-wireguard: Post-quantum hybrid cryptography-based wireguard vpn protocol. In *International Conference on Information Technology and Applications*, pages 283–292. Springer.

Ricci, S., Dobias, P., Malina, L., Hajny, J., and Jedlicka, P. (2024). Hybrid keys in practice: Combining classical, quantum and post-quantum cryptography. *IEEE Access*.

Rossi, M. (2023). Pqc transition in france anssi views. In *Real World Post-Quantum Crypto*.

Roy, A. (2023). kyber. *https://github.com/itzmeanjan/kyber*.

Schatz, D., Altheide, F., Koerfgen, H., Rossberg, M., and Schaefer, G. (2023). Virtual private networks in the quantum era: A security in depth approach. pages 486–494.

Tuma, P. and Havlin, J. (2024). Linux network traffic encryptor. *https://github.com/gabsssq/Linux-network-traffic-encryptor*.

van Heesch, M., van Adrichem, N., Attema, T., and Veugen, T. (2019). Towards quantum-safe vpns and internet. *Cryptology ePrint Archive*.

Wei, D. (2023). Crypto++ library. *http://www.cryptopp.com/*.