

Security-Enhanced WireGuard Protocol Design using Quantum Key Distribution

Lutong Chen*, Kaiping Xue*[†], Jian Li*[†], Zhonghui Li*, Nenghai Yu*

* School of Cyber Science and Technology, University of Science and Technology of China, Hefei, 230027, China

[†] Corresponding Author: K. Xue (kpxue@ustc.edu.cn) and J. Li (lijian9@ustc.edu.cn)

Abstract—WireGuard is a pioneering and lightweight Virtual Private Network (VPN) protocol that has been merged into the Linux kernel. It leverages the Noise secure framework to provide advanced security functionalities, such as identity hiding and perfect forward security. Although WireGuard has an optional pre-shared key mode to ensure key security, the advanced security features are guaranteed by asymmetric cryptography algorithms, which cannot be held in the face of superior quantum computers. To achieve quantum-resistant security, WireGuard should avoid using vulnerable asymmetric cryptography algorithms that are currently deeply integrated into the WireGuard protocol. In this paper, we present a solution to enhance the security of WireGuard by integrating Quantum Key Distribution (QKD). We first change the security mode to tunnel-orient Pre-Shared Keys (PSK) as the authentication anchor. We also design QKD-assisted ephemeral keys and corresponding Key Encapsulation Mechanism (KEM) to achieve WireGuard’s advanced security properties without using asymmetric cryptography. We also integrate QKD keys during the key derivation to provide further security. Finally, we implement the entire protocol named WireGuard-QKD in Golang and evaluate its performance and security.

Index Terms—Network Security, Security Protocol, Quantum Key Distribution, Virtual Private Network, Quantum Computing

I. INTRODUCTION

Network security protocols serve as the fundamental components of modern network security, facilitating confidential and authorized communication between remote parties. Recently, WireGuard [1] stands out as an innovative Virtual Private Network (VPN) protocol that has been integrated into the Linux kernel since its 5.6 version in 2022. It is based on the Noise protocol framework, which is widely recognized as a novel framework for security protocols. It offers the function of Authenticated Key Exchange (AKE). Additionally, it incorporates advanced security features such as Identity Hiding (IH), and Perfect Forward Security (PFS) through up to four rounds of Elliptic-Curve Diffie–Hellman (ECDH) operations [2]. Accordingly, the lightweight nature, exceptional efficiency, and inherent security features lead to its widespread application in various commercial products, such as CloudFlare WARP [3], and Mozilla VPN [4], enabling robust privacy protection for users.

However, we notice that WireGuard may suffer from security downgrading when confronted with the challenges posed by the emergence of quantum computing [5]. Quantum computing represents a novel computing model that is expected to reach maturity in the next decade, and it introduces

significant challenges to the design of security protocols. For instance, the Shor algorithm has the potential to compromise the ECDH. While WireGuard incorporates an optional Pre-Shared Key (PSK) to partially mitigate this threat, we observe that the advanced security properties are still compromised as they heavily rely on ECDH.

There are two primary approaches to enhance WireGuard to mitigate the threat posed by quantum computing. The first approach involves the utilization of post-quantum cryptography algorithms [6]. However, most post-quantum cryptography algorithms often require more computational resources or larger key sizes [7]. In this paper, we focus on proposing a solution as the second category, using Quantum Key Distribution (QKD) [8] to provide a higher level of security guarantee. QKD can produce security keys between two parties with information-theoretic security based on the fundamental principles of quantum mechanics and has already been adopted to enhance several classical security protocols [9]. To the best of our knowledge, there currently exists no specific scheme for integrating QKD with WireGuard as it is not only a relatively new protocol but also in-principle relies on asymmetric cryptography.

We propose a variant implementation of WireGuard, WireGuard-QKD, aiming to replace the asymmetric cryptography algorithms and fully leverage the security provided by QKD networks. Firstly, we suggest a tunnel-based PSK authorization scheme as an alternative to the original one since ECDH is vulnerable to quantum computing. Secondly, we introduce a QKD-based Key Encapsulation Mechanism (KEM) to replace all four ECDH operations during the handshake procedure. Furthermore, we modify the Key Derivation Function (KDF) to fuse QKD keys for further security. Meanwhile, we prove that WireGuard-QKD still holds the original security properties by utilizing the flexibility and security of QKD keys. To validate our proposed protocol design, we implement WireGuard-QKD based on the official cross-platform WireGuard implementation, Wireguard-go [10]. We conduct experiments to show its high performance and efficiency. The contributions of our paper can be summarized as follows:

- We propose a QKD-based WireGuard protocol to mitigate quantum computing threats. We design a new tunnel-level pre-shared key authorization scheme to avoid using ECHD operations.
- We construct a QKD-based KEM method that leverages

only QKD technology and symmetric cryptographies. We prove its security is inherited from the high-level security of QKD keys. It also achieves all advanced security properties.

- We implement a cross-platform demonstration named WireGuard-QKD. By adopting a dynamic rekey period, it can fully leverage the QKD keys to secure the communication.

This paper is organized as follows. First, we brief the background of QKD and analyze the quantum threats in the WireGuard protocol in Section II. Then, we propose our WireGuard-QKD protocol design in Section III and a secure analysis in Section IV. In Section V, we present our WireGuard-QKD implementation and conduct a performance evaluation. Finally, we conclude our work in Section VI.

II. BACKGROUND AND PROBLEM STATEMENT

In this section, we introduce the QKD technology, and brief the WireGuard protocol, especially analyze its security under quantum computing.

A. Quantum Key Distribution (QKD)


QKD is a novel quantum information technology that facilitates the secure distribution of cryptographic keys among remote parties [8], [11]. Since the inception of the pioneering BB84 protocol proposed in 1984, subsequent protocols [12], [13] have made significant advancements in terms of both security and efficiency, making it possible to construct QKD networks as a robust security infrastructure. As a result, several countries and regions [14], [15] are devoted to QKD research and constructing their QKD networks.

The major function of QKD is to negotiate secure keys, offering the following advantages. First, QKD generates a greater number of keys compared to classical schemes. In most classical schemes, all encryption keys are derived from a fixed-length master key. For example, in WireGuard, it is a 32-byte string. Oppositely, QKD devices [16] can produce secret keys at speeds of up to 100 Mbps. This capability enables the encryption of traffic in a One-Time Password (OTP) manner and achieves information-theoretical security. Second, QKD keys offer quantum intrinsic randomness and are independent of each other, which simplifies the provision of Perfect Forward Secrecy (PFS) [17]. Third, the QKD mechanism guarantees that the keys are confidently shared between two parties. Thus, it is possible to build a quantum AKE protocol using QKD keys [18] where two parties authorize each other during the handshake procedure.


B. Security Analysis on WireGuard in post-quantum model

The WireGuard Protocol is a new VPN protocol designed to surpass the performance and security capabilities of traditional alternatives. It operates as a connection-less protocol and is built into the Linux kernel to achieve heightened efficiency and reduced latency. From a security perspective, WireGuard adopts the Noise framework [2] to authenticate parties and establish session keys for subsequent traffic encryption.


Static Public Key	Static Private Key	Listen Endpoint
spk_{Alice}	ssk_{Alice}	0.0.0.0:10000
Peer Public Key	Allowed Source IPs	Peer Endpoint
spk_{Bob}	10.1.0.0/24	2.2.2.2:10000
spk_{David}	10.2.0.0/24	3.3.3.3:10000



static public keys
as the identity



routing items



WireGuard
VPN endpoint

Fig. 1. A cryptokey routing table for node Alice.

In particular, it utilizes Curve25519 for ECDH [19] key exchange, Chacha20_Poly1305 for symmetric authenticated encryption, and BLAKE2s for hashing [20]. The handshake procedure involves two parties, the initiator and the responder, and is completed within two packets. The first packet is transmitted from the initiator to the responder, while the second packet is sent in the opposite direction.

However, the emergence of quantum computing introduces new threats to WireGuard, particularly concerning the four rounds of Curve25519 ECDH operations in the handshake. Therefore, one primary goal of this paper is to substitute all ECDH operations using QKD keys while preserving the protocol's original security properties. Additionally, we aim to enhance the security of the protocol by harnessing QKD technologies. Through a comprehensive analysis of the original WireGuard protocol, we conclude three key aspects that require a new QKD-enhanced design.

The first and primary aspect is the authentication scheme. In the original protocol, parties utilize asymmetric cryptography based on Curve25519 to authenticate each other. Specifically, each node maintains a cryptokey routing table. For an example of the initiator, it possesses a static private key ssk_i (corresponding public key is spk_i) to authenticate its identity and also stores the public keys of all its peers, such as spk_r for the responder (corresponding secret key is ssk_r). In the cryptokey routing table, the node also employs the public keys to identify its peers. Fig. 1 illustrates the routing table on node Alice with two peers, Bob and David. However, since Curve25519 is not secure under quantum computing threat, we attempt to avoid using Curve25519 here.

The second challenge is that the ECDH scheme needs to be changed to address the threat posed by quantum computing. In WireGuard, to achieve advanced security properties, both the initiator and responder possess static and long-term private keys, and also generate ephemeral keys during each handshake. Let esk_i and esk_r represent the ephemeral private keys for the initiator and responder, respectively, with epk_i and epk_r denoting their corresponding public keys. Consequently, one handshake procedure involves four rounds of ECDH operations: two occur in the first packet, and two in the second packet. As depicted in Fig. 2, each of the four ECDH

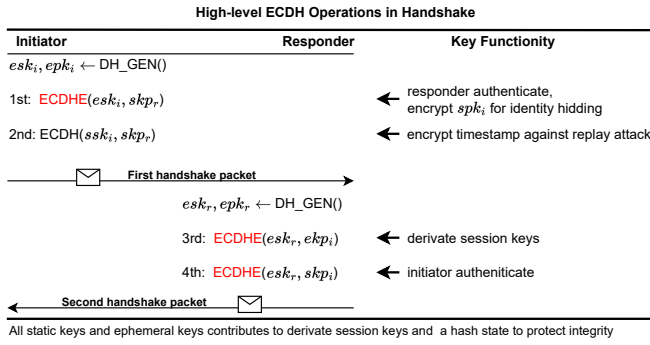


Fig. 2. High-level handshake procedures in WireGuard protocol.

operations serves a distinct purpose, but they all contribute to derive session keys (to encryption the following traffic) and a hashed handshake state (to ensure integrity). Again, since ECDH should not be used anymore, we intend to use QKD keys to provide a similar function as the original ECDH does.

The last challenge is to upgrade the KDF procedure to fully exploit QKD keys to enhance security. Currently, the original protocol utilizes a fixed-length key to derive a session key for all subsequent traffic. However, considering the enhanced PFS offered by QKD keys, we propose incorporating QKD keys into this KDF procedure. Moreover, we can fully leverage QKD keys and minimize key reuse by shortening the rekey period. In WireGuard, the rekey scheme periodically performs a new handshake to negotiate new session keys. By leveraging a large amount of available QKD keys, it becomes feasible to reduce key reuse further, thereby approaching the OTP encryption method.

III. WIREGUARD-QKD DESIGN

A. Overview

Based on the aforementioned analysis, we now introduce our proposed design for the WireGuard-QKD protocol. The protocol is specifically tailored to fully utilize QKD technology to address quantum computing threats and enhance overall security. First, we adopt a tunnel-level authentication scheme instead of relying on asymmetric cryptography algorithms for mutual authentication. A tunnel-level PSK is employed as the shared secret for authenticating peers and contributing to the distribution of session keys, which is presented in Section III-B. Next, we introduce a novel cryptographic primitive that utilizes QKD keys to construct a KEM in Section III-C. The goal is to replace all ECDH operations in the handshake process. Finally, by integrating QKD keys into the key derivation procedures, we aim to achieve a higher level of PFS in Section III-D.

In general, our WireGuard-QKD protocol will use four 32-byte QKD keys during each handshake, including three keys used in key exchange in replacing ECDH operations, and one used in the derivation procedure. As a result, at most 128 bytes of the QKD key will be consumed in one handshake. The original WireGuard protocol introduces a

constant rekey scheme that performs handshakes at about 90 seconds. However, it will lead to an inefficient use of QKD keys, considering the QKD keys are produced faster than the current consumption rate. As a result, we also dynamically adjust the rekey period so that it can fully use the QKD keys to approach an OTP encryption manner.

B. A Tunnel-level PSK-based Authentication

In the original WireGuard protocol, a node-based asymmetric cryptography authentication is employed, wherein nodes utilize static asymmetric keys as a foundation for mutual authentication. Although the security of Curve25519 is deemed vulnerable in quantum computing models, the presence of a pre-configured secure anchor remains essential. Therefore, we introduce a tunnel-level PSK-based authentication scheme in WireGuard-QKD.

Specifically, WireGuard-QKD requires that both the initiator and responder peers maintain a tunnel-level PSK. Each node generates a random static secure key, referred to as ssk . Additionally, the two peers involved in a VPN tunnel share a pre-configured tunnel-level PSK, denoted as $tpsk$, which serves as the authenticator secret during the handshake process. Furthermore, the $tpsk$ can also be utilized as the peer's identity within the cryptokey routing table. The pre-configured $tpsk$ can be manually set or derived from the nodes' ssk according to the following procedure:

$$tpsk = \text{HKDF}(\mathbf{H}, k, \mathbf{H}(ssk_i) || \mathbf{H}(ssk_r)), \quad (1)$$

where HKDF is a hash-based KDF standardized in [21], \mathbf{H} is the BLAKE2s hashing function [20], and k is a fixed string. The ssk_i and ssk_r are two parties' static secure keys.

The $tpsk$, replacing the static public keys spk , are used in mainly two aspects. First, it is used to authenticate peers and derive the session keys (for the following packets' encryption). Second, it is also used as the integrity key to protect the handshake's integrity as follows:

$$\text{mac1} = \text{MAC}(\mathbf{H}(\text{LABEL_MAC1} || tpsk), msg), \quad (2)$$

where mac1 is a field to protect the handshake packet's integrity, MAC is a keyed Blake2s used in WireGuard, LABEL_MAC1 is a constant string, and msg is the packet content (except the mac1 and mac2 fields [1]).

In most scenarios, the use of PSK instead of asymmetric cryptography may introduce some limitations in key management. Users would need to manually maintain the PSKs and keep track of all peers' keys. However, these limitations will not occur here. Firstly, WireGuard adopts a Trust-on-First-Use (TOFU) approach [22], where nodes are still required to manually maintain the public keys of other nodes due to the absence of a Certificate Authority (CA) infrastructure. Secondly, the utilization of a QKD network to provide security keys typically necessitates a static network topology, where the VPN tunnels are not likely to be dynamic.

Nevertheless, we acknowledge that using tunnel-based PSKs does introduce changes to the handshake's decryption procedure when a node has multiple peers. In the WireGuard

protocol, a node can directly utilize its static secret key ssk to process all incoming packets no matter which peer sends. However, in WireGuard-QKD, nodes now maintain different $tpsk$ for each peer and now need to iteratively try all $tpsk$ s to decrypt incoming packets. Fortunately, WireGuard employs an authenticated encryption scheme, enabling the node to detect a failed decryption attempt using a particular $tpsk$ and subsequently try the next available $tpsk$. As a result, the decryption overhead increases from $O(1)$ to $O(n)$, but it is generally acceptable since nodes in WireGuard scenarios typically communicate with a limited number of peers.

C. QKD-based Key Encapsulation Mechanism

The KEM [23] is a modern cryptographic primitive to replace traditional Key Exchange or Public Key Encryption (PKE) methods. It facilitates the secure transmission of keys between multiple parties. In this paper, we propose a novel QKD-based KEM to replace the existing ECDH operations using QKD keys and a hashing function. To ensure compatibility with the original WireGuard protocol, we modify the common KEM definitions [7] for the following reasons: 1) The classic KEM typically employs one pair of asymmetric keys, whereas the WireGuard utilizes ECDH operations that are associated with the key pairs of two nodes. Consequently, we need to ensure seamless integration of WireGuard-QKD with the original protocol. 2) We need to include QKD keys in the KEM procedure to guarantee security.

As a result, we present our QKD-based **QKEM** = (**KeyGen**, **Qkd**, **Encaps**, **Decaps**) as a tuple of probability algorithms over a key space \mathcal{K} . It runs between two parties, namely a node and its peer:

- **KeyGen** returns a pair of ephemeral keys (sk_e, pk_e) . The ephemeral secret sk_e is randomly chosen from $\{0, 1\}^{|\mathcal{K}|}$, where $|\mathcal{K}|$ is the key's length, and the ephemeral public key $pk_e = \mathbf{H}(sk_e)$.
- **Qkd** is the function to produce a QKD key $k_q = \mathbf{Qkd}()$, with two properties: 1) It is statistically indistinguishable from $\{0, 1\}^{|\mathcal{K}|}$. 2) The QKD keys are used in an OTP manner, where each pair of **Encaps** and **Decaps** operations invokes the **Qkd** function to obtain the same k_q . However, the QKD keys are independent across multiple rounds of **Encaps** and **Decaps** operations.
- **Encaps** is a key encapsulation algorithm that uses a QKD key k_q , node's ephemeral secret key sk_e , and an additional nodes' public key pk_p (for compatibility consideration) to provide a key $k \in \mathcal{K}$, and a ciphertext ct . Here, $ct = \mathbf{H}(sk_e)$, and

$$k = \mathbf{HMAC}(\mathbf{H}, k_q, \mathbf{H}(sk_e) \otimes pk_p), \quad (3)$$

where **HMAC** is the Keyed-Hashing for Message Authentication algorithm defined in [24].

- **Decaps** is a decapsulation algorithm that runs on the peer and decapsulates the key k from ct , the QKD key k_q , and the peer's public key pk_p as follows:

$$k = \mathbf{HMAC}(\mathbf{H}, k_q, ct \otimes pk_p). \quad (4)$$

It is now possible to construct a similar adversary game $G_{\mathbf{QKEM}}^A$ like other KEM schemes, and we present the advantage of the adversary A as $\mathbf{Adv}_{\mathbf{QKEM}}^{\text{CCA}}(A) =$

$$\Pr \left[\begin{array}{l} sk_e, pk_e \leftarrow \mathbf{KeyGen}(); \\ b \leftarrow \{0, 1\}; \\ (ct^*, k_0^*) \leftarrow \mathbf{Encaps}(\mathbf{Q}(), sk_e, pk_p); \\ k_1^* \leftarrow \mathcal{K}; \\ b' = A^{\mathbf{Decaps}(\mathbf{Q}(), \cdot)}(k_b^*, ct^*, pk_p) \end{array} \right], \quad (5)$$

where $\mathbf{Decaps}(\mathbf{Q}(), \cdot)$ is the decapsulation oracle that the adversary cannot access to the QKD keys, as it is used only once in each operation. The security of our **QKEM** relies on the security of the QKD keys and that **HMAC** is a secure Pseudo-Random Function (PRF) [25], in our construction. However, it does not depend on the secrecy of pk_p and pk_e , as they might be public information. Here, we present a brief explanation. First, we employ the QKD keys in an OTP manner, where each QKD key is confidential to the adversary and used only once for a specific pair of **Encaps** and **Decaps** operations. Consequently, the adversary's queries to the **Decaps** oracles will not reveal the actual value of k_q used in encapsulating k_0^* . It indicates that the CCA adversary A falls back to a single-round Known Ciphertext Attack (KCA) one. Further, since k_q is statistically indistinguishable from $\{0, 1\}^{|\mathcal{K}|}$ and **HMAC** is a secure PRF, the derived key k_0^* achieves computationally indistinguishable from a random distribution. This property ensures the security of the encapsulated key k_0^* in our QKEM scheme.

We now explain how we use our **QKEM** in the WireGuard-QKD protocol, as illustrated in Fig. 3. We treat the static ECDH operation (the 2nd one) and the ephemeral ECDH operations (specifically the 1st, 3rd, and 4th ones) differently. For the ephemeral ECDH operations, we replace them with the **QKEM** scheme. We produce the ephemeral esk and epk using the **QKEM.KeyGen**. As for the static keys ssk and spk , we use the tunnel-based PSK $tpsk$ instead. Finally, we use **QKEM.Encaps** to replace the ECDH operations. As for the 2nd static ECDH, it originally uses the two nodes' static keys to calculate, which all refer to the $tpsk$ in WireGuard-QKD. Thus, we use the $tpsk$ as the output directly.

D. QKD Key Derivation and Dynamic Rekey Period

We present how to further integrate the QKD keys into the key derivation procedure. Fig. 4 shows how session keys are generated. In Fig. 4(a), the session keys are derived from a chaining key using **HMAC** [24] three times to produce the encryption keys for two directions for the following traffic. Furthermore, the chaining key is generated in rotation in the aforementioned handshake procedure with each ECDH or **QKEM** in Fig. 4(b). Here, we use a trivial but effective way to integrate the QKD keys to provide a further PFS. After finishing the handshake, the initiator and the responder will request a 32-byte QKD key, namely k_{q4} , and use it to replace the empty string $\{0\}^{256}$ used in the key derivation procedure.

To further leverage the QKD keys, we implement our WireGuard-QKD platform with a dynamic rekey period. The

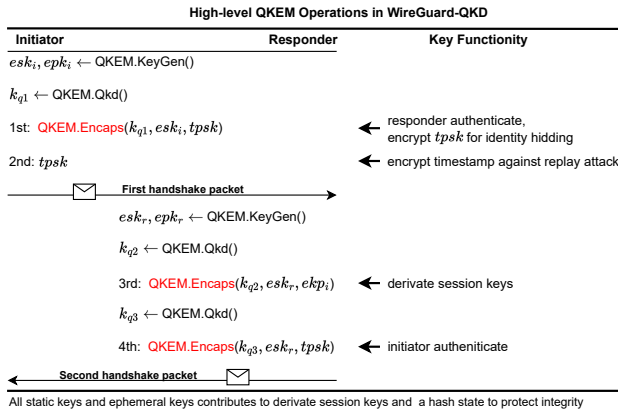


Fig. 3. High-level handshake procedures in WireGuard-QKD protocol.

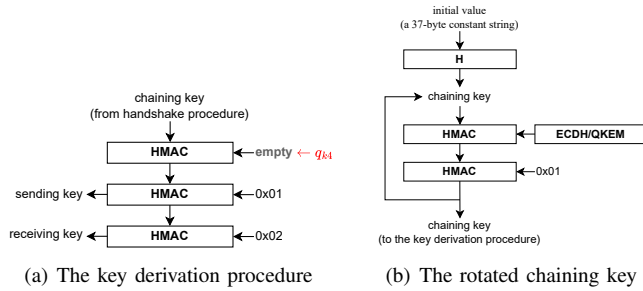


Fig. 4. The modification of integrating QKD keys in WireGuard-QKD

rekey period is a constant value in the original WireGuard protocol, at about 90 seconds. It indicates that the initiator should reinitiate a new handshake procedure to re-negotiate the session keys. In our WireGuard-QKD, we adopt a dynamic rekey period p_r based on the available QKD keys. Note that one handshake will use four 32-byte QKD keys, we set the rekey period to

$$p_r = \max\left\{\frac{128}{r_q}, mp_r\right\}, \quad (6)$$

where r_q is the rate for the QKD network to provide keys, and mp_r is the minimal value of p_r to avoid handshake overhead.

IV. SECURITY ANALYSIS

We employ the formal security protocol verification tool Tamarin [26] to evaluate the security of the WireGuard-QKD protocol. Tamarin is a widely used tool that has been extensively utilized for the verification of numerous commonly used protocols [27]. In this study, we treat all four QKD keys and the tunnel-level PSK as independent secret keys. We then generate Tamarin rules and lemmas to establish the security properties of the protocol. Here, we explain how we achieve these security properties.

Basic Security of the session keys. The security of the derivated session keys is protected by both the QKD keys and the $tpsk$. Since HMAC can be viewed as a security PRF, the derivated session keys will be secure if any of the QKD keys or the $tpsk$ are secure.

Perfect Forward Security. PFS indicates that the leak of long-term secrets ($tpsk$ here) will not influence the session

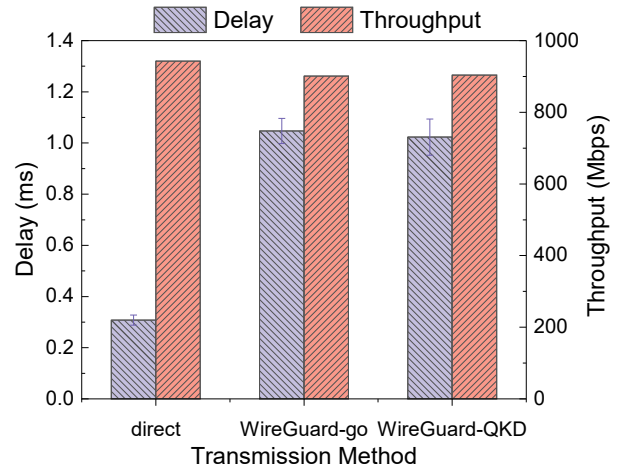


Fig. 5. The delay and throughput in peer-to-peer experiment.

keys. It is achieved by the PFS property of QKD keys. The QKD keys are confidentially shared between nodes, and they use one QKD key only once. It guarantees that the adversary cannot recover the session keys since they have no information about QKD keys used in that handshake even if the $tpsk$ are leaked. Besides, both two parties will generate a random esk , which can also provide PFS.

Identity Hidding. It requires the third party not to obtain the identity of the communication parties. In WireGuard-QKD, the tunnel's identity $tpsk$ is reviewed as a secret (not public information) and is also transmitted in encryption. Meanwhile, unless the peer acknowledges both $tpsk$ and the current QKD keys, it cannot decrypt the ciphertext either.

Authenticated Key Exchange. AKE is required to authenticate the two parties. Here, we also use $tpsk$ as the secret anchor to authenticate each other. We complement it by integrating $tpsk$ into three QKEM procedures. If any party does not have access to $tpsk$, the AEAD ciphertext cannot be decrypted, and the mac1 field verification also fails.

Key Compromise Impersonation resistance. KCI resistance requires that the adversary cannot impersonate the parties when the $tpsk$ is leaked, which is protected by the confidentiality of QKD keys. The adversary cannot obtain the QKD keys without being detected due to the quantum no-cloning theorem, and therefore still cannot complete a successful handshake without the correct QKD keys.

V. PERFORMANCE EVALUATION

We implement a demonstration WireGuard-QKD platform based on the cross-platform WireGuard-go [10] and conduct a peer-to-peer experiment to illustrate the performance. The environment involves two Linux machines connected via a 1Gbps switch. Then, we use WireGuard-QKD and WireGuard-go to establish a secure VPN tunnel and evaluate the performance. We also evaluate the direct transmission without a VPN tunnel. The results are shown in Fig. 5.

We observe that our WireGuard-QKD achieves a similar performance compared to the original WireGuard-go. The

delay is about 1.024 ± 0.071 milliseconds in WireGuard-QKD while it is 1.047 ± 0.049 milliseconds in WireGuard-go. No drop packet is observed during a 10-minute transmission. Meanwhile, we use iPerf3 to evaluate the throughput, and the result is 904 Mbps in WireGuard-QKD and 901 Mbps in WireGuard-go. It indicates that the integration of QKD keys has little influence on the transmission. Our scheme required that at least 128 bytes be generated for 90 seconds to achieve the same PFS, which is achievable for the current QKD networks. With more available QKD keys, the dynamic rekey period effectively updates the session keys in a timely manner to ensure security.

We also evaluate the performance of the direct transmission. The results show that both WireGuard-QKD and WireGuard-go have little performance downgrading. For example, the delay decreases slightly from about 0.3 millisecond to about 1 millisecond mainly due to the encryption/decryption procedure. Overall, the WireGuard-QKD can still make full use of the 1Gbps network.

VI. CONCLUSION

With the development of quantum information technology, QKD is viewed as an innovative technology for establishing secret keys between remote parties and providing a new approach to address the vulnerabilities of the classic security protocols posed by quantum computing attacks. Its advantage includes not only a larger amount of usable keys compared to the classic key agreement methods but also providing beneficial security properties. This paper leverages QKD to design a post-quantum WireGuard-QKD protocol. Our approach incorporates a tunnel-based Pre-Shared Key scheme as a secure anchor for authentication, a QKD-based KEM to replace all insecure ECDH operations, and a QKD-integrated key derivation procedure with a dynamic rekey period to fully utilize QKD keys. We provide a comprehensive security analysis to demonstrate that the proposed WireGuard-QKD protocol satisfies both the fundamental security properties such as PFS, KCI resistance, and Identity Hiding. Furthermore, we implement a cross-platform WireGuard-QKD to show its feasibility and efficiency.

ACKNOWLEDGEMENT

This work is supported in part by the Innovation Program for Quantum Science and Technology under Grant No. 2021ZD0301301, Anhui Initiative in Quantum Information Technologies under Grant No. AHY150300, National Natural Science Foundation of China under Grant No. 62201540, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

REFERENCES

- J. A. Donenfeld, "Wireguard: next generation kernel network tunnel," in *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS)*, 2017, pp. 1–12.
- T. Perrin, "The Noise protocol framework," <http://noiseprotocol.org/noise.pdf>, 2016.
- M. Prince, "Introducing WARP: fixing mobile internet performance and security," <https://blog.cloudflare.com/1111-warp-better-vpn/>, 2019.
- Mozilla, "Features that protect your life online," <https://www.mozilla.org/en-US/products/vpn/features/>, 2020.
- F. V. Massoli, L. Vadicamo, G. Amato, and F. Falchi, "A leap among quantum computing and quantum neural networks: A survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.
- A. Hülsing, K.-C. Ning, P. Schwabe, F. Weber, and P. R. Zimmermann, "Post-quantum wireguard," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (S&P)*, 2021, pp. 304–321.
- J. Bos, L. Ducas, E. Kiltz *et al.*, "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM," in *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018, pp. 353–367.
- Y. Cao, Y. Zhao, Q. Wang *et al.*, "The evolution of quantum key distribution networks: On the road to the qinternet," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 839–894, 2022.
- J. Viksna, S. Kozlovics, and E. Rencis, "POSTER: Integrating Quantum Key Distribution into Hybrid Quantum-Classical Networks," in *Proceedings of the 2023 International Conference on Applied Cryptography and Network Security (ACNS)*, 2023, pp. 695–699.
- WireGuard, "WireGuard-Go," <https://git.zx2c4.com/wireguard-go/>, 2018.
- M. Wang, J. Li, K. Xue, R. Li, N. Yu, Y. Li, Y. Liu, Q. Sun, and J. Lu, "A segment-based multipath distribution method in partially-trusted relay quantum networks," *IEEE Communications Magazine*, 2023.
- Z. Tang, Z. Liao, F. Xu *et al.*, "Experimental demonstration of polarization encoding measurement-device-independent quantum key distribution," *Physical Review Letters*, vol. 112, no. 19, p. 190503, 2014.
- Y. Liu, Z.-W. Yu, W. Zhang, J.-Y. Guan, J.-P. Chen, C. Zhang, X.-L. Hu, H. Li, C. Jiang, J. Lin *et al.*, "Experimental twin-field quantum key distribution through sending or not sending," *Physical Review Letters*, vol. 123, no. 10, p. 100505, 2019.
- M. Sasaki, M. Fujiwara, H. Ishizuka *et al.*, "Field test of quantum key distribution in the Tokyo QKD Network," *Optics Express*, vol. 19, no. 11, pp. 10 387–10 409, 2011.
- Y.-A. Chen, Q. Zhang, T.-Y. Chen *et al.*, "An integrated space-to-ground quantum communication network over 4,600 kilometres," *Nature*, vol. 589, no. 7841, pp. 214–219, 2021.
- W. Li, L. Zhang, H. Tan, Y. Lu, S.-K. Liao, J. Huang, H. Li, Z. Wang, H.-K. Mao, B. Yan *et al.*, "High-rate quantum key distribution exceeding 110 Mb s^{-1} ," *Nature Photonics*, vol. 17, no. 5, pp. 416–421, 2023.
- M. Mehic, M. Niemiec, S. Rass *et al.*, "Quantum key distribution: a networking perspective," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–41, 2020.
- M. Mosca, D. Stebila, and B. Ustaoglu, "Quantum key distribution in the classical authenticated key exchange framework," in *Proceedings of the 2013 Post-Quantum Cryptography: 5th International Workshop (PQCrypto)*, 2013, pp. 136–154.
- D. J. Bernstein, "Curve25519: new Diffie-Hellman speed records," in *Proceedings of the 9th International Conference on Theory and Practice in Public-Key Cryptography (PKC)*, 2006, pp. 207–228.
- J.-P. Aumasson, S. Neves *et al.*, "BLAKE2: simpler, smaller, fast as MD5," in *Proceedings of the 11th Applied Cryptography and Network Security (ACNS)*, 2013, pp. 119–135.
- H. Krawczyk and P. Eronen, "RFC5869: HMAC-based extract-and-expand key derivation function (HKDF)," 2010.
- N. H. Walfield and W. Koch, "TOFU for OpenPGP," in *Proceedings of the 9th European Workshop on System Security*, 2016, pp. 1–6.
- H. Jiang, Z. Zhang *et al.*, "IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited," in *Proceedings of the 38th Advances in Cryptology (CRYPTO)*, 2018, pp. 96–125.
- H. Krawczyk and P. Eronen, "RFC2104: HMAC: Keyed-hashing for message authentication," 1997.
- M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," in *Proceedings of the 26th Advances in Cryptology (CRYPTO)*, 2006, pp. 602–619.
- D. Basin, C. Cremers, J. Dreier, and R. Sasse, "Tamarin: verification of large-scale, real-world, cryptographic protocols," *IEEE Security & Privacy*, vol. 20, no. 3, pp. 24–32, 2022.
- C. Cremers, M. Horvat, J. Hoyland *et al.*, "A comprehensive symbolic analysis of TLS 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1773–1788.