Secure SSL/TLS Communication System Based on Quantum Keys

WANG Jigang[1,2], LU Yuqian[3], WEI Liping[1,2], JIANG Xinzao[3], ZHANG Han[1,2]

(1. State Key Laboratory of Mobile Network and Mobile Multimedia, Shenzhen 518055, China;
2. Department of Cyberspace Security, ZTE Corporation, Shenzhen 518057, China;
3. School of Cyber Science and Engineering, Southeast University, Nanjing 211102, China)

DOI: 10.12142/ZTECOM.202403013

https://kns.cnki.net/kcms/detail/34.1294.TN.20240923.1559.002.html, published online September 23, 2024

Manuscript received: 2023-10-28

Abstract: Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols facilitates a secure framework for identity authentication, data encryption, and message integrity verification. However, with the recent development in quantum computing technology, the security of conventional key-based SSL/TLS protocols faces vulnerabilities. In this paper, we propose a scheme by integrating the quantum key into the SSL/TLS framework. Furthermore, the application of post-quantum algorithms is used to enhance and complement the existing encryption suites. Experimental results show that the proposed SSL/TLS communication system based on quantum keys exhibits high performance in latency and throughput. Moreover, the proposed system showcases good resilience against quantum attacks.

Keywords: SSL/TLS protocols; quantum key; post-quantum cryptography

Citation (Format 1): WANG J G, LU Y Q, WEI L P, et al. Secure SSL/TLS communication system based on quantum keys [J]. ZTE Communications, 2024, 22(3): 106 - 115. DOI: 10.12142/ZTECOM.202403013
Citation (Format 2): J. G. Wang, Y. Q. Lu, L. P. Wei, et al., "Secure SSL/TLS communication system based on quantum keys," *ZTE Communications*, vol. 22, no. 3, pp. 106 - 115, Sept. 2024. doi: 10.12142/ZTECOM.202403013.

1 Introduction

In recent years, the development of quantum computing has posed challenges to the security of traditional encryption algorithms. The Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocol, which is built upon these conventional encryption algorithms, facilitates identity authentication, data encryption, and message integrity verification. During the handshake phase of the SSL/TLS protocol, the client transmits its supported encryption suites to the server. The server, based on its configuration, selects a suite that encompasses authentication algorithms, key exchange algorithms, digest algorithms, and others. These are employed for subsequent processes such as identity authentication, key negotiation, and encrypted communication. Consequently, if the algorithms within the cipher suite harbor known vulnerabilities such as discrete logarithm and prime factorization problems susceptible to the Shor quantum algorithm[1], the cipher suite may be insecure, thus imperiling the SSL/TLS protocol against potential attacks.

In this paper, we address the aforementioned issues by incorporating quantum key distribution into the SSL/TLS protocol. Quantum keys, as supplementary and preferred sources of keys, offer a resilience against quantum attacks. In situations where quantum keys are inaccessible, the system seamlessly transitions to a post-quantum cipher mode. Post-quantum cipher algorithms, similarly fortified against quantum attacks, optimize and complement the original encryption suites of the SSL/TLS protocol. The dynamic switch between quantum keys and post-quantum encryption algorithms ensures constant protection against quantum attacks, enhancing the system's security and reliability.

This paper is structured as follows: Section 2 introduces the background knowledge of the SSL/TLS protocol, quantum key distribution, and post-quantum encryption algorithms. Section 3 provides an overview of the overall system architecture, detailing the handshake protocol and cipher suite employed in this study. The experimental environment and system modules are presented in Section 4. Section 5 conducts testing and analysis of the system's performance with regard to handshake latency and post-establishment data throughput. Finally, a comprehensive conclusion is made in Section 6.

2 Background

2.1 SSL/TLS Protocol

The SSL and TLS protocols are secure transport protocols

This work was supported by ZTE Industry-University-Institute Cooperation Funds under Grant No. HC-CN-20221029003.

that reside between the application layer and the transport layer in the TCP/IP protocol stack. The SSL/TLS protocol encompasses two layers of communication. The record protocol offers fundamental security services to various higher-level protocols and defines the format for data transmission[2]. Moreover, SSL/TLS establishes three higher-level mechanisms involving data encryption, identity authentication, and message integrity verification, thereby ensuring security and data integrity during transmission.

The SSL/TLS protocol negotiates cipher suites and keys through a handshake process between the client and the server. This handshake protocol consists of a series of messages exchanged between the client and server, which can be categorized into four distinct stages.

The first stage initiates the logical connection and establishes relevant security functionalities. It commences with a client "hello" message and concludes with a server "hello" message. During this stage, the client and server negotiate the SSL version for use, session ID, compression methods, and cipher suites. Random numbers are also exchanged. Cipher suites define key exchange algorithms and CipherSpecs, which encompass encryption algorithms, MAC algorithms, and other pertinent information. Supported key exchange methods by the SSL/TLS protocol include Rivest-Shamir-Adleman (RSA), fixed Diffie-Hellman (DH), ephemeral DH, and anonymous DH[3].

The second stage pertains to server authentication and key exchange. If authentication is required, the server sends its certificate at the beginning of this stage. Any agreed-upon key exchange, apart from anonymous DH, necessitates this certificate message. Subsequently, a server key exchange message is sent. This message is not required if RSA key exchange is employed, or if the server sends a certificate with fixed DH parameters. Additionally, non-anonymous servers can request certificates from clients by sending a certificate request message. This stage ends with a server-done message.

In the third stage, client authentication and key exchange are initiated by the client's certificate message. Next, the client sends a client key exchange message to create a premaster secret between the client and server. The content of this message varies based on the key exchange method. The exchanged premaster secret will be used by both parties to derive a shared master key. CipherSpec parameters are generated from the master key using hash techniques. These parameters include a client write MAC, a server write MAC, a client write key, a server write key, a client write Initialization Vector (IV), and a server write IV[4]. Finally, the client sends a certificate verification message to validate its certificate explicitly.

In the fourth stage, the client sending a change-cipher-spec message to transfer the pending CipherSpec state to the current state. Subsequently, a finished message is sent using the new algorithm and key. Finally, the server sends a change-cipher-spec message to transfer the pending messages to the current CipherSpec, and it also sends its own finished message[5].

The record protocol in the SSL/TLS framework is established atop a reliable transport protocol (such as TCP) and provides support for fundamental functionalities like data encapsulation, compression, and encryption. One key advantage of SSL/TLS lies in its independence from specific application layer protocols. Higher-level application layer protocols (e.g., HTTP, FTP, Telnet) can seamlessly operate over the SSL/TLS protocol[6]. The SSL/TLS protocol completes encryption algorithm negotiation, communication key establishment, and server authentication before the communication between application layer protocols begins. As a result, data transmitted by application layer protocols are encrypted, ensuring communication confidentiality.

2.2 Quantum Key Distribution

Quantum Key Distribution (QKD) is theoretically proven to be unconditionally secure, with its security guaranteed by the fundamental principles of quantum mechanics[7]. QKD utilizes quantum states to encode and transmit information, providing theoretically unconditional secure shared keys for both communicating parties and establishing secure confidential communication. QKD guarantees the security of point-to-point key distribution. The process involves the exchange of quantum bits (qubits) between a quantum transmitter and a quantum receiver through a quantum channel. They further exchange measurement bases through a public channel, perform key sifting, and subsequently perform error correction. This process is designed to detect the presence of potential attackers and determine the final session key.

In the process of QKD, pairs of photons with different polarization states are randomly emitted by quantum devices. On the receiving side, the photon states sent by the quantum devices are measured by randomly selecting measurement bases[8]. Based on the polarization state of the emitted photon from the quantum device and the orientation of the measurement basis at the receiving side, the information received is determined as either 0 or 1 for each received photon. Moreover, due to the non-cloneability of quantum states, which means they cannot be copied or measured without disrupting their state, any attempt at eavesdropping could potentially alter the quantum states themselves, resulting in a high error rate and thus making eavesdropping detectable. Furthermore, each string of keys is generated randomly, and if intercepted, the communicating parties can detect it and change the password[9], thus rendering quantum keys non-cloneable and reliable.

Consequently, the distinct advantage of quantum keys is their ability to resist quantum computing attacks, achieved through their inherent properties of single quantum indivisibility and unclonable quantum states[10]. The combination

of QKD and the one-time pad enables information-theoretic secure encryption, meaning that it remains secure even against adversaries with unlimited computational resources. QKD's functionality includes symmetric key negotiation and generation, which, when combined with symmetric cipher algorithms, can achieve encryption, decryption, and authentication capabilities.

2.3 Post-Quantum Cryptography

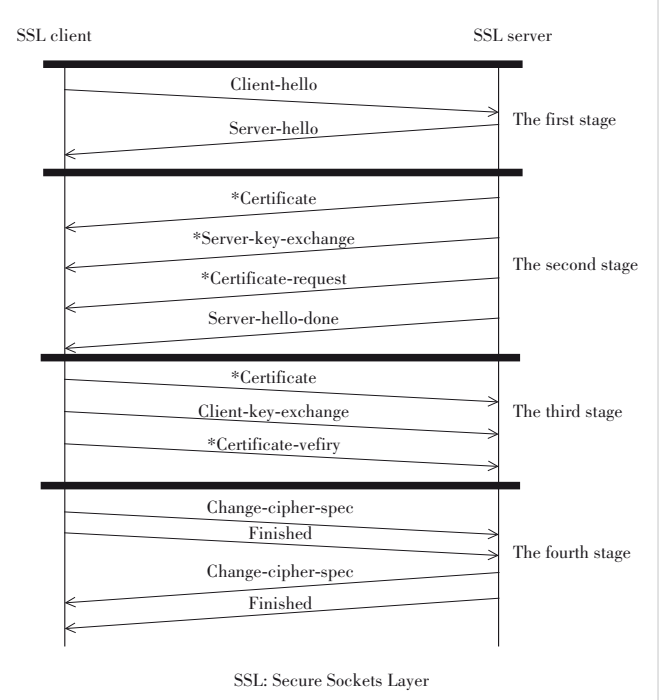
Cryptographic algorithms that can resist attacks from quantum computers are collectively referred to as Post-Quantum Cryptography (PQC). These algorithms have been developed to tackle security threats posed by the emergence of quantum computing[11]. Post-quantum cryptographic algorithms offer computation speeds surpassing those of existing public key algorithms while maintaining the same level of security. They can be used to replace existing algorithms and protocols, including public key encryption, key exchange, digital signatures, and more.

Post-quantum cryptographic algorithms can be categorized into four main classes: lattice-based crypto systems, code-based crypto systems, multivariate crypto systems, and hash-based cryptosystems. Among these, lattice-based post-quantum cryptographic algorithms stand out due to their relative efficiency, versatility, and ability to be highly parallelized[12]. They strike a better balance between security, key sizes, and computation speed compared to traditional number theory-based constructions. In some cases, lattice-based algorithms can even outperform traditional number theory-based cryptographic algorithms in terms of computation speed. Among various post-quantum cryptographic algorithms, lattice-based ones are more suitable for practical applications when considering the same level of security. The security of lattice-based post-quantum cryptographic algorithms is based on the hardness of solving lattice problems. They achieve smaller public and private key sizes, faster computation speeds, and can be used to construct various cryptographic primitives[13], making them more suitable for real-world applications. Compared to number-theory-based cryptographic algorithms, lattice-based algorithms offer significantly improved computation speeds and higher security levels.

On July 5th, 2022, NIST announced a selection of algorithms for standardization, which includes CRYSTALS-KYBER for asymmetric encryption and key encapsulation mechanisms, CRYSTALS-Dilithium, FALCON, and SPHINCS+ for digital signatures. Among them, NIST recommends the CRYSTALS-Kyber algorithm for general-purpose encryption of information exchanged over public networks and the other three algorithms for identity authentication. CRYSTALS-Kyber is a lattice-based post-quantum cryptographic algorithm that provides an IND-CCA2 secure key encapsulation mechanism. Its security relies on the difficulty of solving the Module Learning With Errors problem on lattices[14]. The module is an extension of the ideal lattice and general lattice, while Module Learning with Errors is an extension of Ring Learning with Errors (RLWE). When appropriate parameters are chosen, cryptographic schemes constructed based on Module Learning with Errors (MLWE) provide a good balance between efficiency and security. Therefore, our scheme employs the CRYSTALS-Kyber algorithm based on MLWE to guarantee strong security. CRYSTALS-Kyber includes algorithms for public-private key pair generation, key encapsulation, and ciphertext generation. CRYSTALS-Kyber, as a public key algorithm, can be used in the key negotiation part of the handshake process. It offers the advantages of relatively small encryption keys, small data exchange volume, and fast operation speed while ensuring security. Kyber defines three parameter sets: Kyber512, Kyber768, and Kyber1024. By utilizing post-quantum cryptographic algorithms like Kyber for key exchange and negotiation, systems can more effectively address potential attacks from future quantum computing technologies[15].

3 System Architecture

The system framework designed in this paper is depicted in Fig. 1. The quantum key storage devices are abstractly represented as a Quantum Key Pool (QKP). When the remaining quantity of quantum keys in the pool meets the demand, the proposed approach utilizes the quantum keys negotiated through the QKD system as the session key for the SSL/TLS protocol. Subsequently, during the encryption of communication data using encryption algorithms, the quantum key serves



▲Figure 1. SSL/Transport Layer Security (TLS) handshake protocol

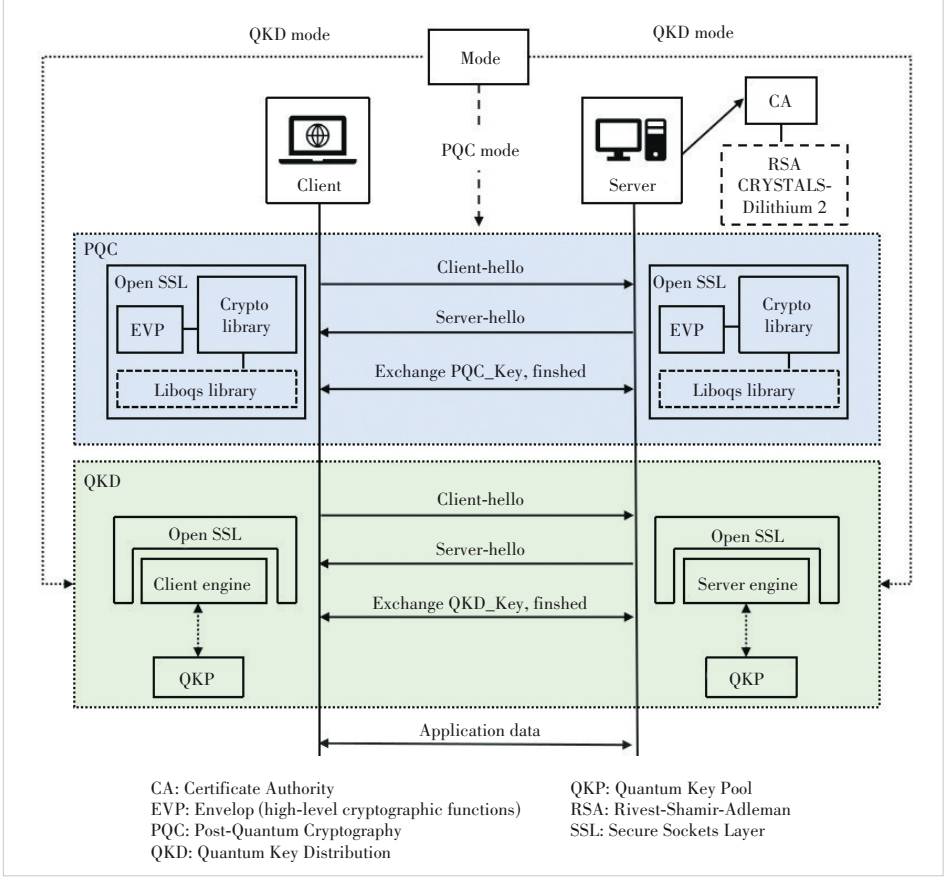
as the symmetric key for encryption. In cases where the quantity of quantum keys in the pool falls short of the demand, the application employs post-quantum cryptographic algorithms to optimize and supplement the corresponding cipher suites of the original SSL/TLS protocol. This approach enables resistance against quantum attacks in the post-quantum cryptographic mode, ensuring quantum security in the entire mode.

3.1 Handshake Protocol

Keys are stored in pairs between any two QKD nodes. The key pool is divided into multiple virtual spaces based on the source and destination nodes of communication. Keys are placed in the corresponding index-numbered key pool based on the source and destination node identifiers of communication requests. Quantum key distribution devices generate quantum keys and their corresponding identifiers, known as quantum key identifiers, through quantum key negotiation protocols (such as BB84/B92/TF-QKD). These keys and identifiers are then stored in the key pool. The key pool performs unified authentication and management of quantum keys and their corresponding identifiers. Quantum keys are used as session keys for SSL/TLS communication, while key identifiers play a crucial role in obtaining quantum key credentials and ensuring consistency during the SSL/TLS handshake process. The handshake procedure in the quantum key mode is illustrated in Fig. 2.

In the handshake process of the quantum key mode, the client initiates communication by sending an encrypted communication request "Client Hello" to the server. During this process, the client primarily provides information such as the supported protocol version (e. g., TLS 1.3), a randomly generated client nonce, encryption algorithm suites, supported compression methods, and other relevant configurations to the server. Upon receiving the client's request, the server responds with a message "Server Hello" confirming the use of the encryption communication protocol version (TLS 1.3), a randomly generated server nonce, configurations for server encryption algorithms, and the chosen encryption algorithm suite. The server encrypts the "Server Hello" information using its private key and sends it back to the client, along with its certificate containing the public key. The client verifies the server's certificate upon receiving the response. If the certificate is not issued by a trusted authority, the domain name does not match, or the certificate has expired, a warning is displayed to the user, who can decide whether to continue communication. If the certificate is valid, the client extracts the server's public key from the certificate and decrypts the "Server Hello" message, which was encrypted by the server's private key. Failure to decrypt indicates a false "Server Hello" message, resulting in the termination of communication.

After the initial authentication, the client sends a request for quantum key allocation to the key pool of the quantum key distribution device. The key pool sends quantum keys and their corresponding key identifiers to the client, providing the client with a pair of quantum keys and their identifiers. The client encrypts the quantum key identifier using a premaster secret and sends it to the server. Upon decryption, the server obtains the quantum key identifier. Following verification of the client's identity, the server calculates the premaster secret for the current session by combining the previously shared random numbers (client nonce and server nonce). The server then decrypts the quantum key identifier to obtain it. Using the quantum key identifier, the server sends a quantum key extraction request to the key pool, which responds by sending the corresponding quantum key.



▲**Figure 2. Overall system framework**

The received quantum key undergoes a hash operation, and its hash is compared to the hash of the quantum key in the client's message to verify consistency. In case of inconsistency, an error message is sent to the server, triggering a quantum key retrieval process.

Once the server notifies the successful negotiation of the quantum key agreement, both parties send change cipher spec notifications to indicate that subsequent message encryption will involve the symmetric encryption method agreed upon and the quantum key. A handshake completion notification is sent, and at this point, both the client and server process the quantum key to obtain a compatible format for the session key.

Considering the possibility of quantum devices experiencing emergencies such as downtime, our system's key pool features a backup function. Periodically, we securely store and backup keys, allowing us to retrieve quantum keys from the key pool's backup. If the client and server cannot obtain the same quantum key due to key pool asynchrony or an insufficient quantity of remaining keys in the pool, we will switch to the classic mode of the SSL/TLS protocol using PQC algorithms. PQC algorithms optimize and supplement the encryption cipher suites of the original SSL/TLS protocol. In this context, a post-quantum cryptographic algorithm, such as CRYSTALS-Kyber, is used to modify the existing SSL/TLS protocol. The SSL/TLS handshake process using post-quantum cryptographic algorithms is illustrated in Fig. 3.

In the handshake procedure of the post-quantum cryptographic mode, the client initially dispatches a "Client Hello" request to initiate encrypted communication with the server. In response, the server reciprocates by sending a "Server Hello" message back to the client, along with its own certificate and static public key (Public_Key_02). Subsequently, the client generates a temporary public-private key pair (Public_Key_01, Secret_Key_01) via the Kyber key generation algorithm. Utilizing Public_Key_02 provided by the server, the client generates a random number K_1. This number is then subjected to the Kyber encryption algorithm, resulting in the ciphertext C_1. The client then forwards the temporary public key (Public_Key_01) and ciphertext C_1 to the server. Upon receipt of Public_Key_01 and ciphertext C_1 from the client, the server encrypts the random number K_2 using Public_Key_01 to derive the ciphertext C_2. Concurrently, it decrypts ciphertext C_1 using its static private key to obtain pre master secret K_1'. Then, the server sends ciphertext C_2 to the client. Subsequently, the client receives ciphertext C_2 from the server and decrypts it using its temporary private key (Secret_Key_01) to extract the pre master secret K_2'. Then, both the server and client perform hash operations on K_1' and K_2' respectively, utilizing the resultant hash value as the session key.

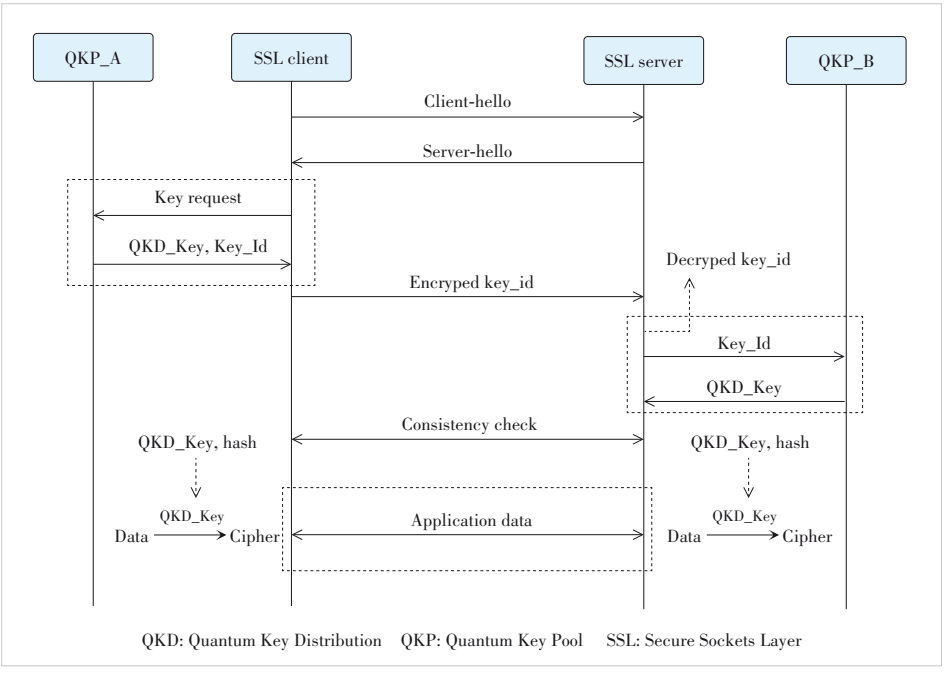
During the handshake process, the generation of public-private key pairs, encryption and decryption of random numbers are performed using the CRYSTALS-Kyber algorithm. Through the Kyber encryption and decryption algorithms, a secure exchange of random numbers used in the SSL/TLS protocol are accomplished between the client and the server. Due to the quantum-resistant properties of the Kyber algorithm, this process is impervious to decryption by quantum attacks.

Finally, both the client and server send change cipher spec notifications to indicate that subsequent message encryption will involve the symmetric encryption method agreed upon and the key derived from the post-quantum cryptographic process. A handshake completion notification is sent, marking the completion of the handshake process in the post-quantum cryptographic mode.

3.2 Encryption Suite

Within the SSL/TLS protocol, the SSL_CIPHER data structure is used to describe a cipher suite, which is composed of a set of cryptographic algorithms including key exchange, authorization, communication encryption, and digest algorithms.

During the communication process between a client and a server, the client initiates the handshake by sending a "Client Hello" packet, containing a list of supported cipher suites. Multiple suites can be separated by symbols, such as



▲Figure 3. Handshake flow in quantum key mode

TLS_RSA_WITH_AES_128_CBC_SHA(0x002f) and others. The server, based on the cipher suite identifiers and their priority sent by the client, selects a supported cipher suite and includes it in the "Server Hello" response. This agreed cipher suite is ultimately used for the session negotiation between the two parties.

In the quantum cryptographic mode, the following encryption parameters are generated: client MAC, server MAC, client key, server key, client IV, server IV, and pre-master secret. MAC is used to generate message digests. IV is generated only when traditional block cipher encryption is applied to application traffic. When secure encryption and authentication are used, the size of the required write key and MAC must be negotiated during the SSL/TLS protocol handshake.

Each quantum encryption suite must include a key exchange algorithm, a symmetric encryption algorithm, and an authentication algorithm, with the key exchange algorithm employing a quantum-key-based key negotiation algorithm. The server selects a supported cipher suite from the list of available cipher suites. If no compatible cipher suite is found, the server returns a handshake failure alert message and terminates the connection. Cipher suites in the quantum cipher mode are shown in Table 1.

Each post-quantum encryption suite comprises a key exchange algorithm, symmetric encryption algorithm, and message digest algorithm. The key exchange algorithm in these suites employs post-quantum encryption techniques, such as CRYSTALS-Kyber. The message digest algorithm is used for verifying server signatures and can include Rivest-Shamir-Adleman (RSA), Digital Signature Standard (DSS), Ellipse Curve Cryptography (ECC), or their corresponding variant algorithms. The server's authentication relies on a robust PKI mechanism, encompassing certificate issuance, certificate management, and certificate validity verification. Only after validating the certificate's legitimacy can the verification of the server's own signature take place. Cipher suites in the post-quantum cipher mode are shown in Table 2.

▼Table 1. Cipher suites in quantum cipher mode

Grade	Encryption Suite
1	TLS_QKD_SHA_RSA_WITH_OTP_MD5
2	TLS_QKD_DHE_DSS_WITH_DES_CBC_UHAH1
3	TLS_QKD_UHASH1_PSK_WITH_OPT_ UHAH2
4	TLS_QKD_MD5_DH_RSA_WITH_AES_128_CBC_SHA

▼Table 2. Cipher suites in post-quantum cipher mode

Grade	Encryption Suite
1	TLS_Kyber_SHA_RSA_WITH_OTP_MD5
2	TLS_Kyber_DHE_DSS_WITH_DES_CBC_UHAH1
3	TLS_Kyber_UHASH1_PSK_WITH_OPT_ UHAH2
4	TLS_Kyber_MD5_DH_RSA_WITH_AES_128_CBC_SHA

4 Experiment

4.1 Experiment Environment

Our experimental setup involves two host machines equipped with Core i5-13600 processors, 8 GB of RAM, and 100 GB of disk storage each. Additionally, we have two quantum communication devices employing BB84 protocols. The length of the quantum key generated by quantum communication devices is 256 bit, while the key identifier stored in the quantum key pool is 12 bit. The attenuation in the photon transmission process and finite-size effects may lead to communication delays. Therefore, in our experimental conditions, the quantum channel distance is set to be short-range (within 50 km) with a key rate of 20 kbit/s. The experimental testing is facilitated using the OpenSSL 1.1.1t toolkit.

The testing environment consists of two host machines: the server employs an Ubuntu Server 22.10 operating system, while the client is based on a Windows 11 environment. To capture and analyze network traffic during testing, we utilize the Tshark network capture software.

4.2 Quantum Key Based SSL Communication Module

We leverage the OpenSSL engine mechanism to integrate QKD with the SSL/TLS protocol. This engine mechanism enables third parties to augment OpenSSL with extensions, which can be implemented as dynamic libraries dynamically loaded into OpenSSL[16]. The engine mechanism seamlessly facilitates encryption using software cryptographic libraries or hardware encryption devices. By overloading the callback functions used for hardware-accelerated DH key exchange, we realize the sharing of quantum keys between communicating parties. Due to the inability of the DH callback function within the engine to distinguish between client and server invocations, we separately implement two independent engines for dynamic loading: one on the client side and the other on the server side. This duality serves to adapt QKD to the SSL/TLS protocol. The callback functions that are overloaded for hardware acceleration encompass the following:

1) init(): This function is formerly employed for initializing the engine; post-overloading, it initializes the QKD device.

2) generate_key(): This function is used for generating DH private keys and computing public keys using negotiated parameters; upon overloading, the server utilizes a fixed value as the DH private key, invokes QKD_START() to acquire the quantum key identifier from the QKD device or USB key as the DH public key, which is subsequently transmitted to the client.

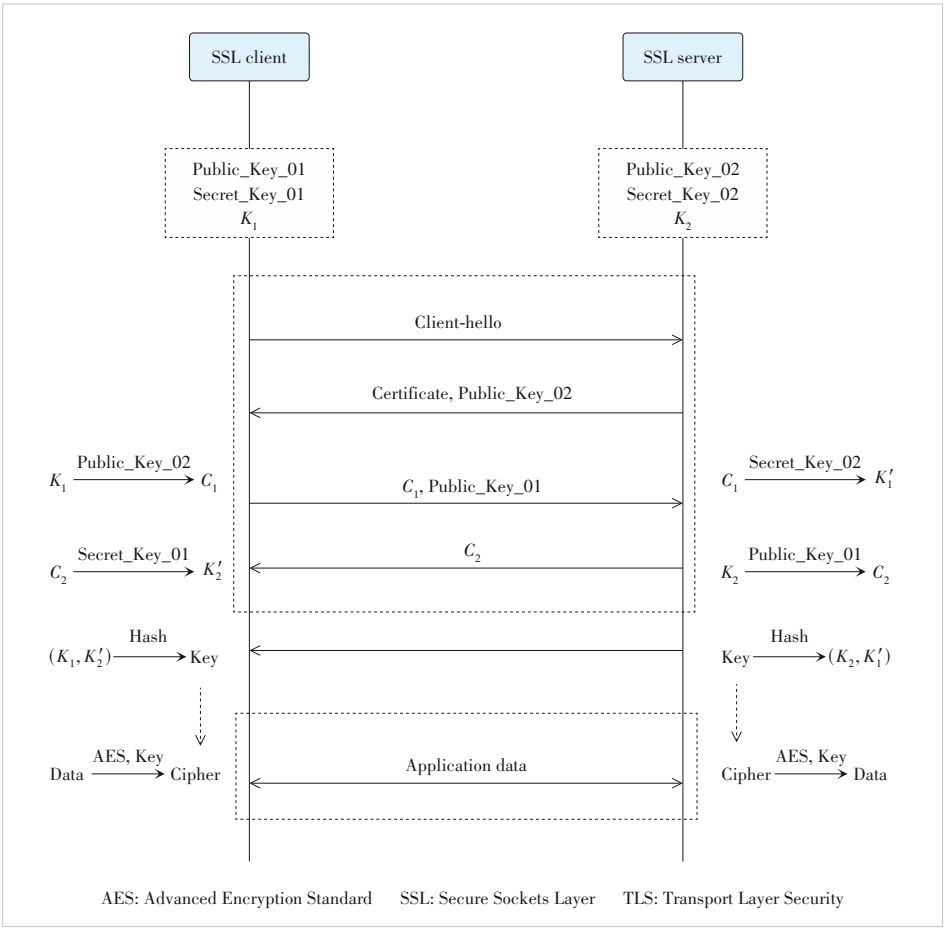
3) compute_key(): This function is initially employed for calculating DH shared keys; post-overloading, the client utilizes the DH public key received from the server as the quantum key identifier, passing it as a parameter to QKD_START (). Following this, both the server and client individually invoke QKD_CONNECT() with the QKD device and input the

quantum key identifier into QKD_GET_KEY(). This procedure retrieves the corresponding quantum key from the QKD device as the DH shared key, culminating with a QKD_CLOSE() invocation.

The QKD API invocation process for both the client and server is depicted in Fig. 4.

During the process of acquiring quantum keys using the QKD API on both the client and server sides, the SSL server initiates the procedure by invoking QKD_START() with a null value. Consequently, the QKD system returns a new available quantum key identifier to the server. This identifier is stored by the server and then encrypted before being transmitted to the client. Subsequently, the client employs the QKD_START() function with the key identifier as its parameter.

Following these initial steps, both the server and client independently execute QKD_CONNECT() to establish a QKD connection. This connection facilitates the exchange of quantum key identifiers sent by the client and server. This step serves to verify that the client possesses an identical quantum key identifier to that of the server. Ultimately, to conclude the process, both the server and client separately invoke QKD_CLOSE() to terminate the QKD connection.



▲**Figure 4. Handshake flow of the SSL/TLS protocol employing post-quantum cryptographic algorithms**

4.3 Post-Quantum Key Based SSL Communication Module

We extend the encryption module by building upon the OpenSSL Crypto Library and integrating algorithms from the liboqs library. The liboqs library, an open-source C library designed for post-quantum encryption algorithms, is incorporated into the OpenSSL framework as a novel branch. This integration allows the post-quantum cryptographic algorithms from the liboqs library to complement the existing functionality of OpenSSL's Crypto Library. As a result, these post-quantum cryptographic algorithms fortify the SSL/TLS protocol with quantum-resistant capabilities. The communication framework of the SSL/TLS protocol, rooted in post-quantum cryptographic algorithms, is illustrated in Fig. 5.

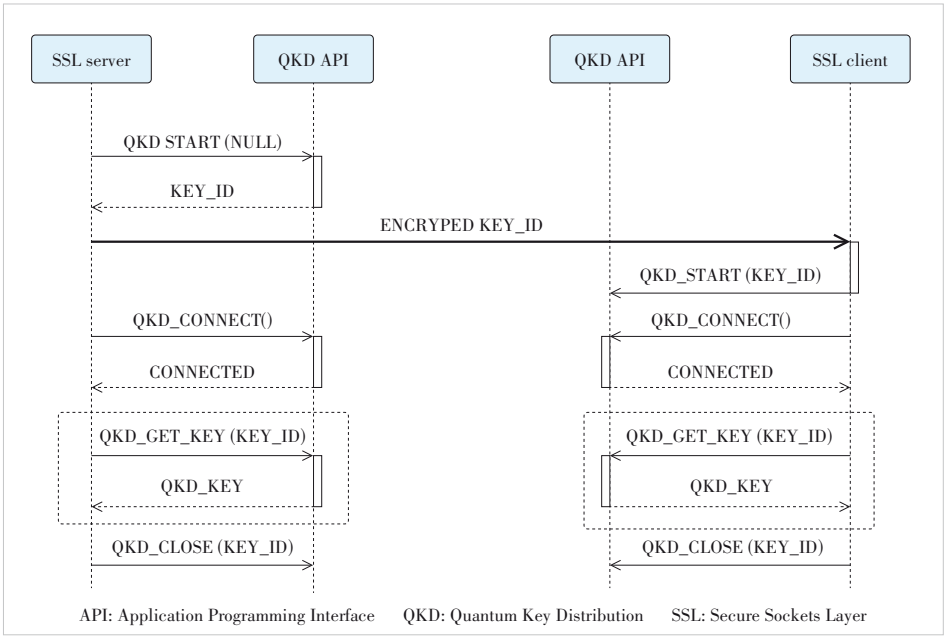
When higher-level applications invoke the OpenSSL library to facilitate secure encrypted communication via the SSL/TLS protocol, they do not directly engage with the low-level specifics of individual cryptographic algorithms[17]. Instead, these applications interact with the EVP interface provided by the OpenSSL library. The EVP module encapsulates the intricate details of cryptographic algorithm implementations and offers abstract methods and data types to manage cryptographic operations. The foundational encryption functions required for network protocols are realized through the libcrypto library. This library encompasses the concrete implementations of encryption.

In the context of the SSL/TLS communication module based on post-quantum cryptography, digital signature certificates are generated by a custom certificate authority (CA). The digital signature algorithm utilized is CRYSTALS-Dilithium2. These certificates contain their own public key as well as a random value, serving the purpose of identity authentication.

5 Results and Analysis

We conducted performance testing on the SSL/TLS communication system proposed in this paper. In the tests, the client initiates an SSL/TLS connection request to the server. Both parties establish a secure and reliable SSL/TLS session through the handshake process.

We created an SSL server and an SSL client using the developed communication system tools based on OpenSSL. The



▲**Figure 5. Process of invoking QKD API for the client-side and server-side**

server dynamically loaded the server-side engine and awaited connection requests from SSL clients. Subsequently, the client dynamically loaded the client-side engine and initiated an SSL connection to the server.

We employed network packet capture software Tshark to monitor the interaction between the client and server. This allowed us to capture encrypted packets and analyze relevant information such as packet size and timestamps. Through this analysis, we evaluated the system's performance.

As a point of comparison, we also subjected an SSL/TLS communication system using classical cryptographic algorithms to testing. This comparative approach enabled a quantitative assessment of the impact of the quantum key application on the performance of the original protocol. The primary evaluation metrics encompassed the handshake latency between communication parties and the data throughput following the establishment of a secure session between the parties.

5.1 Handshake Delay

We conducted multiple experiments by varying the key negotiation method and the key size to measure handshake latency. Table 3 provides a description of the SSL/TLS communication handshake latency data when employing quantum key distribution as the key negotiation method.

▼**Table 3. SSL/TLS communication handshake delay under different key negotiation methods**

Key Negotiation Method	Key Size	Handshake Delay/ms
Quantum key distribution	Quantum key 512 kB	16.2
Quantum key distribution	Quantum key 16 kB	15.7
Quantum key distribution	Quantum key 160 B	15.6
Classical public key algorithm	RSA 4 096 bit	15.3
Classical public key algorithm	RSA 2 048 bit	3.7
Classical public key algorithm	RSA 1 024 bit	2.1

SSL: Secure Sockets Layer TLS: Transport Layer Security

According to the findings presented in Table 3, it is evident that when utilizing quantum key distribution as the key negotiation method for the SSL/TLS protocol, the resulting handshake latency remains consistently around 16 ms. This value is nearly identical to the handshake latency observed when employing classical public key algorithms (4 096 bit RSA public keys) as the key negotiation method.

Furthermore, as indicated by the results in Table 3, the handshake latency of SSL/TLS communication generated by classical public key algorithms increases with larger RSA key sizes. To maintain higher security levels, systems or users are required to continually escalate the size of the RSA public keys they employ. Consequently, in communication environments demanding elevated security standards, the disparity in handshake latency between utilizing quantum key distribution and classical public key algorithms as the key negotiation methods for the SSL/TLS protocol will progressively diminish.

5.2 Data Throughput

When both communicating parties employ the AES algorithm as the encryption method, a higher frequency of key updates enhances communication security. However, the trade-off is that a higher frequency of key updates can lead to a decrease in data throughput for the session between the parties.

The TCP throughput of the network used in our tests was measured at 985 Mbit/s. In the SSL/TLS protocol, we performed key negotiations using both quantum key distribution technology and classical public key algorithms. For quantum key distribution, a key size of 160 B was utilized, while for classical public key algorithms, an RSA key size of 1 024 bits was used. The data throughput performance under different key update frequencies, using these two different key negotiation methods, is depicted in Fig. 6.

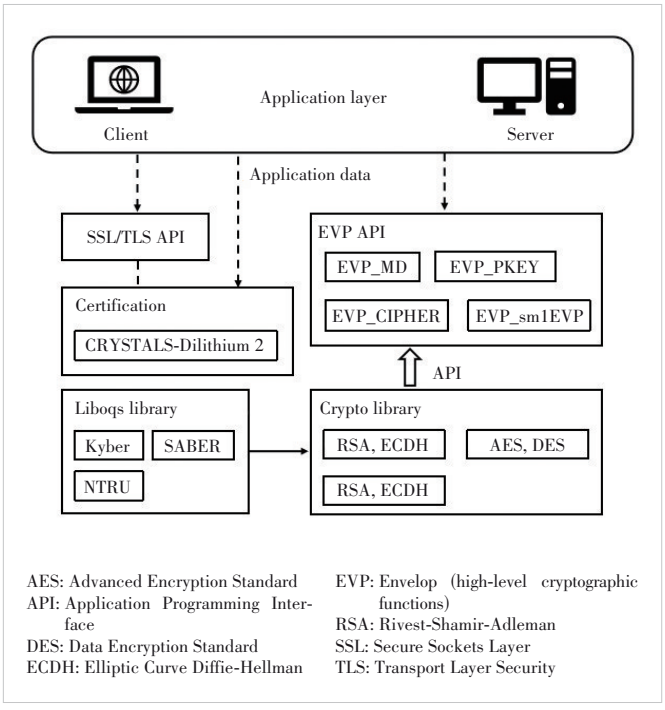
For each transmission of a fixed amount of data by the client (such as 1 MB), a new SSL/TLS session connection is initiated, leading to an update of the AES key.

From Fig. 6, it becomes evident that when transmitting 1 MB of data and utilizing quantum key distribution for key negotiation, a higher frequency of key updates exerts a notable impact on subsequent communication data throughput. However, as the key update frequency reaches 100 MB per update,

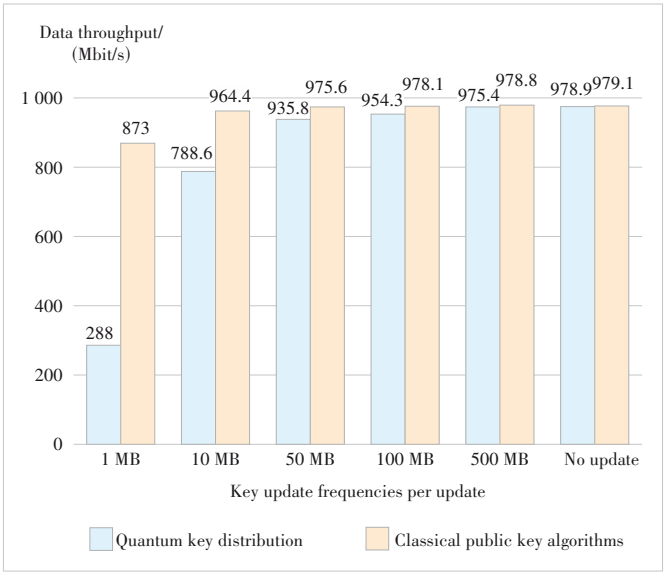
the difference in data throughput between the two different key negotiation methods (quantum key distribution and classical public key algorithms) is approximately 3%. When the key update frequency further increases to 500 MB per update, the impact of quantum key distribution on communication data throughput becomes negligible.

5.3 Security Analysis

In the information exchange process of the SSL/TLS communication system described in this paper, the use of the unclonability and tamper-resistance of quantum states ensures the security of information transmission. Currently, several studies[18 - 20] have shown that combining QKD with PQC can enhance a network's resilience to potential quantum computing attacks. In the quantum key distribution process, the key pool stores quantum keys along with key identifiers. These identifiers uniquely represent the quantum keys, which the SSL/TLS protocol's client and server use to obtain quantum keys as session keys from the quantum key distribution device. In the designed SSL/TLS communication system based on quantum keys, both communication parties transmit encrypted quantum key identifiers during the SSL/TLS handshake, rather than directly transmitting the quantum keys over the channel. Subsequent key consistency checks are then performed. As a result, attackers cannot eavesdrop on the SSL/TLS handshake process to steal or tamper with the quantum keys being used by both parties. Therefore, the SSL/TLS protocol based on quantum keys exhibits the ability to resist quantum attacks on the physical level. At the same time, we recognize that any eavesdropping on the QKD process will alter the quantum states, potentially subjecting QKD-based networks to distributed denial-of-service (DDoS) attacks. In future work, we will implement access control measures to prevent DDoS attacks targeting the QKD process within the network.



▲Figure 6. Communication framework diagram of SSL/TLS protocol based on post-quantum cipher algorithm



▲Figure 7. Data throughput under various key update frequencies

The security of post-quantum cryptographic algorithms relies on mathematical problems that current quantum computing cannot efficiently solve. The kyber768 algorithm used in this paper is a lattice-based post-quantum cryptographic algorithm, and its security hinges on the hardness of the MLWE problem on lattices. When appropriate parameters are chosen, there are currently no known classical or quantum algorithms capable of rapidly solving this problem. Consequently, this algorithm offers high security against quantum attacks. Therefore, the SSL/TLS protocol that integrates post-quantum cryptography resists quantum attacks on the mathematical level.

In summary, whether utilizing quantum key distribution technology or employing post-quantum cryptographic algorithms for key negotiation, both approaches guarantee quantum-resistant security performance for the SSL/TLS communication process.

6 Conclusions

This paper presents a novel approach that combines quantum key distribution with post-quantum cryptography in an SSL/TLS protocol secure communication system. By dynamically loading an engine, the integration of key exchange and quantum keys within the SSL/TLS protocol is achieved. Additionally, post-quantum cryptographic algorithms are embedded into the cryptographic suite of the SSL/TLS protocol, thereby expanding its underlying algorithmic capabilities. This extension builds upon the existing SSL/TLS protocol to create a quantum-resistant SSL/TLS communication system,

while maintaining transparency to upper-layer applications. The significance of this work lies in its potential to advance the adoption of quantum technologies within the SSL/TLS protocol. Through packet analysis of communication data within the test environment, the proposed system demonstrates high performance in handshake latency and throughput.

References

[1] JOSEPH D, MISOCZKI R, MANZANO M, et al. Transitioning organizations to post-quantum cryptography [J]. Nature, 2022, 605(7909): 237 – 243. DOI: 10.1038/s41586-022-04623-2
[2] DASTRES R, SOORI M. Secure socket layer (SSL) in the network and web security [J]. International journal of computer and information engineering, 2020, 14(10): 330 – 333
[3] DUDDU S, RISHITA SAI A, SOWJANYA C L S, et al. Secure socket layer stripping attack using address resolution protocol spoofing [C]// Proc. 4th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2020: 973 – 978. DOI: 10.1109/ICICCS48265.2020.9120993
[4] IBRAHIM A. Secure socket layer: fundamentals and certificate verification [EB/OL]. [2023-08-20]. https://engrxiv.org/preprint/view/3532/6315
[5] AHMAD KHAN N, KHAN A S, KAR H A, et al. Employing public key infrastructure to encapsulate messages during transport layer security handshake procedure [C]//Proc. Applied Informatics International Conference (AiIC). IEEE, 2022: 126 – 130. DOI: 10.1109/AiIC54368.2022.9914605
[6] SHAKYA S. An efficient security framework for data migration in a cloud computing environment [J]. Journal of artificial intelligence and capsule networks, 2019, 1(1): 45 – 53. DOI: 10.36548/jaicn.2019.1.006
[7] ZENG P, ZHOU H Y, WU W J, et al. Mode-pairing quantum key distribution [J]. Nature communications, 2022, 13(1): 3903. DOI: 10.1038/s41467-022-31534-7
[8] XU F H, MA X F, ZHANG Q, et al. Secure quantum key distribution with realistic devices [J]. Reviews of modern physics, 2020, 92(2): 025002. DOI: 10.1103/RevModPhys.92.025002
[9] MEHIC M, NIEMIEC M, RASS S, et al. Quantum key distribution [J]. ACM computing surveys, 2021, 53(5): 1 – 41. DOI: 10.1145/3402192
[10] MA W K, CHEN B W, LIU L, et al. Equilibrium allocation approaches of quantum key resources with security levels in QKD-enabled optical data center networks [J]. IEEE Internet of Things journal, 2022, 9(24): 25660 – 25672. DOI: 10.1109/JIOT.2022.3195104
[11] EMURA K, MORIAI S, NAKAJIMA T, et al. Cache-22: a highly deployable end-to-end encrypted cache system with post-quantum security [EB/OL]. [2023-08-20]. https://eprint.iacr.org/2022/220.pdf
[12] ALAGIC G, ALPERIN-SHERIFF J, APON D, et al. Status report on the first round of the NIST post-quantum cryptography standardization process [EB/OL]. [2023-08-20]. https://nvlpubs. nist. gov/nistpubs/ir/2019/NIST.IR.8240.pdf
[13] ASIF R. Post-quantum cryptosystems for Internet-of-Things: a survey on lattice-based algorithms [J]. IoT, 2021, 2(1): 71 – 91. DOI: 10.3390/iot2010005
[14] DANG V B, MOHAJERANI K, GAJ K. High-speed hardware architectures and FPGA benchmarking of CRYSTALS-kyber, NTRU, and saber [J]. IEEE transactions on computers, 2023, 72(2): 306 – 320. DOI: 10.1109/TC.2022.3222954
[15] HUANG Y M, HUANG M Q, LEI Z K, et al. A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse [J]. IEICE electronics express, 2020, 17(17): 20200234. DOI: 10.1587/elex.17.20200234
[16] JATI A, GUPTA N, CHATTOPADHYAY A, et al. A configurable CRYSTALS-kyber hardware implementation with side-channel protection [J]. ACM transactions on embedded computing systems, 2024, 23(2): 1 – 25. DOI: 10.1145/3587037
[17] WALDEN J. OpenSSL 3.0.0: an exploratory case study [C]//Proc. 19th International Conference on Mining Software Repositories. ACM, 2022: 735 – 737. DOI: 10.1145/3524842.3528035
[18] AHN J, KWON H Y, AHN B, et al. Toward quantum secured distributed energy resources: adoption of post-quantum cryptography (PQC) and quantum key distribution (QKD) [J]. Energies, 2022, 15(3): 714. DOI: 10.3390/en15030714
[19] YANG Y H, LI P Y, MA S Z, et al. All optical metropolitan quantum key distribution network with post-quantum cryptography authentication [J]. Optics express, 2021, 29(16): 25859 – 25867. DOI: 10.1364/OE.432944
[20] WANG L J, ZHANG K Y, WANG J Y, et al. Experimental authentication of quantum key distribution with post-quantum cryptography [J]. NPJ quantum information, 2021, 7(1): 67. DOI: 10.1038/s41534-021-00400-7

Biographies

WANG Jigang received his PhD degree in computer science from Harbin Engineering University, China in 2007. From May 2007 to June 2009, he held a postdoctoral position in Institute of Computer Science, Tsinghua University, China. From August 2009, He has been with Cyber Security Product Line, ZTE Corporation as the general manager. His recent research interests include operating systems, network and information security, and artificial intelligence.

LU Yuqian is a master student at School of Cyber Science and Engineering, Southeast University, China. Her research interests include IoT and information security.

WEI Liping received his bachelor's degree in electronic information engineering from Zhejiang University, China in 2005. Since 2020, he has been a product manager of cyber security products at ZTE Corporation. His recent research interests include mobile communication, network and information security, and artificial intelligence.

JIANG Xinzao (230229200@seu.edu.cn) is a PhD student at School of Cyber Science and Engineering, Southeast University, China. His research interests include privacy-preserving computation and IoT.

ZHANG Han received his MSc degree in information technology and management from ITC Institute of Twente University, the Netherlands. He is currently pursuing his PhD degree in School of Cyber Science and Engineering, Southeast University, China. He is a senior system architect of ZTE Corporation, China and the director of Jiangsu Provincial Key Laboratory of Big data Storage and Application, China. He is also an associate researcher at the State Key Laboratory of Mobile Network and Mobile Multimedia Technology, China. His research interests include information security, cloud computing, big data, IoT, 5G technology, and human computer interactions.