Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.075573

ARTICLE

TQKD: A More Efficient QKD Network Based on Homomorphic Encryption Technology

Tianhua Lin[1], Sijiang Xie[1,*], Yalong Yan[2], Jianguo Xie[2] and Ang Liu[2]

[1]Department of Cyberspace Security, Beijing Electronic Science and Technology Institute, Beijing, China
[2]Institute of Information Security, Beijing Electronic Science and Technology Institute, Beijing, China

*Corresponding Author: Sijiang Xie. Email: xiesj@besti.edu.cn

Received: 04 November 2025; Accepted: 16 January 2026; Published: 08 May 2026

ABSTRACT: Quantum key distribution (QKD) provides unconditional security but relies on repeaters to extend coverage, thereby introducing repeater trust risks—compromised repeaters may leak keys. Brakerski/Fan-Vercauteren scheme (BFV)-based QKD addresses this issue through key encryption and quantum attack resistance. However, Fast Fully Homomorphic Encryption over the Torus (TFHE) outperforms BFV in encryption/decryption speed for single-qubit homomorphic XOR operations, which is critical for the real-time requirements of QKD. We propose TFHE-based QKD (TQKD), a quantum key distribution protocol based on public-key TFHE. During key forwarding, it leverages the "usable-but-unobservable" property of homomorphic encryption to prevent key exposure. A reduction proof verifies the scheme's Indistinguishability under Chosen-Plaintext Attack (IND-CPA) security. To validate TQKD's computational speed advantage, we developed code using the Open-Source Fully Homomorphic Encryption Library (OpenFHE) platform and designed single-pass and multi-hop relay experiments. We compared the computational efficiency of TQKD against QKD schemes based on similar homomorphic encryption algorithms, Brakerski-Gentry-Vaikuntanathan scheme (BGV) and BFV. Results demonstrate that our scheme achieves faster key encryption/decryption speeds in both scenarios, significantly reducing processing time compared to similar algorithms. Furthermore, while enhancing the scalability of quantum key distribution networks, the added computational overhead is negligible, indicating higher practical value.

KEYWORDS: Quantum key distribution; untrusted relay; homomorphic encryption; learning with errors; TFHE; OpenFHE

1 Introduction

In 1984, Bennett and Brassard proposed the first quantum key distribution protocol, namely the BB84 protocol [1], opening a new chapter in secure communication based on quantum properties. QKD is a secure communication technology based on the principles of quantum mechanics. Its core function is to establish a connection between two geographically separated users and distribute symmetric encryption keys. Unlike traditional public-key cryptosystems based on computational complexity, the security of QKD is not built on complex mathematical problems but on fundamental physical principles of quantum mechanics, such as Heisenberg's uncertainty principle and the quantum no-cloning theorem. It provides a guarantee of the theoretical security of information rather than computational assumptions, ensuring the unconditional security of the key generation and distribution process. At first, this technology did not attract people's attention. It was not until the emergence of quantum computing that traditional encryption algorithms faced unprecedented challenges. Peter Shor's Shor algorithm, which can be implemented on a quantum

computer, can solve in polynomial time the decomposition and elliptic curve logarithm problems for large numbers [2,3]. It almost destroyed the current mainstream public-key cryptography systems represented by RSA, ECC, DH, and DSA [4]. At the same time, the Grover algorithm [5] proposed by Lov Grover can significantly accelerate brute-force cracking of symmetric encryption. That means it has had a significant impact on brute-force cracking. However, QKD possesses the theoretically unconditional security property, rendering it immune to attacks from these quantum algorithms. It has prompted researchers in the field of information security to actively explore schemes based on QKD and Post-Quantum Cryptography(PQC). Consequently, QKD is regarded as the most important, mainstream, and promising technology in current quantum secure communication research and applications.

Although QKD is theoretically highly secure, it faces significant challenges in practical applications and large-scale deployment that cannot be ignored. The core of QKD lies in using quantum states at the single-photon level, such as photon polarization or phase, to encode information. These quantum states are highly fragile and readily interact with surrounding matter, leading to decoherence or absorption that destroys the quantum information they carry. Therefore, QKD requires a medium that maximizes the protection of quantum signals and enables their stable transmission, and fiber-optic channels precisely meet this core requirement. Although free space, such as the atmosphere or satellites, also represents an important research direction for quantum channels, the existing highly developed fiber-optic communication infrastructure network significantly reduces deployment costs and engineering complexity when implementing QKD. Consequently, fiber-optic channels possess an irreplaceable advantage. However, single-photon propagation in optical fibers is inevitably affected by signal attenuation and background noise. Takeoka et al. [6] were the first to rigorously demonstrate that the key generation rate in optical QKD is bounded by a ceiling determined solely by channel loss—meaning there exists a maximum effective communication distance between two optical nodes without relaying. This limit was later extended to 300 km by Korzh et al. [7]. To address this critical issue, researchers have proposed various schemes over the years, as outlined in Section 2, to extend the effective transmission distance of QKD and build large-scale networks.

After years of practical exploration, researchers worldwide have conducted numerous experiments deploying QKD networks [8–12]. Currently, most QKD experimental projects rely on trusted relays, in which quantum keys from adjacent segments are XORed within the relay. While this design ensures real-time and accurate key delivery, it introduces a significant security vulnerability: if any relay node is compromised, leading to the leakage of one quantum key, the attacker can reconstruct the quantum key for the other segment based on the XOR result, thereby compromising all quantum keys along the entire path. Therefore, the key to resolving this issue lies in ensuring key security even when relays are untrusted.

Our TQKD scheme focuses on relay-based QKD, employing the TFHE homomorphic encryption algorithm to ensure privacy and security during relay-based quantum key forwarding. It eliminates reliance on trusted relays, enabling efficient and secure QKD key transmission in untrusted relay scenarios. By performing bitwise homomorphic XOR operations on quantum keys in relays, TFHE demonstrates significant advantages over BGV and BFV algorithms in terms of ciphertext noise reset and logical gate operations. Through its Fast Bootstrapping mechanism, TFHE can instantly reset the ciphertext noise after homomorphic XOR operations, moving it from the threshold edge to a secure level. This eliminates the need for BGV and BFV algorithms to pre-evaluate the circuit's noise threshold before performing Bootstrapping. Furthermore, TFHE incorporates efficient logic gate circuits. The XOR operation is implemented via $Bootstrap(2(c_1 - c_2))$, whereas BGV and BFV lack native XOR logic gates. Consequently, XOR in these algorithms must be decomposed into AND/OR/NOT combinations, requiring multiple rounds of operations per XOR calculation and incurring substantial redundant overhead.

The structure of this paper is as follows: In Section 2, we outline the development trajectory of QKD networks, categorize them based on different QKD implementation schemes, and discuss previous seminal work. In Section 3, we explain how the advanced mechanisms of TFHE influence our proposed scheme. In Section 4, we introduce the public-key version of the TFHE algorithm and the TQKD network model, presenting a complete workflow for TQKD. In Section 5, we design two experimental scenarios to evaluate the innovative value of TFHE in terms of efficiency and propose potential directions for improvement. Finally, in Section 6, we summarize the advantages of the TFHE scheme and identify areas requiring future effort.

2 Related Work

In this section, we will categorize and present the progress of relevant work based on different implementation approaches, including relay-based QKD.

2.1 XOR Based QKD (XOR-QKD)

XOR-QKD is a QKD scheme based on trusted relays that requires full trust in them. Its core concept involves using the lightweight XOR operation to preprocess the key at the trusted relay node, then relaying it hop-by-hop to adjacent nodes. It enables efficient reconstruction of the shared key at the target QKD node, achieving end-to-end key distribution. As shown in Fig. 1, each pair of adjacent nodes shares the same quantum key based on the BB84 protocol, denoted as K_1, K_2, K_3. Alice needs to share the key K_1 with Bob. Trusted relay A first computes $K_1 \oplus K_2$ and then transmits it to trusted relay B. Trusted relay B computes $(K_1 \oplus K_2) \oplus K_2$ to obtain K_1, then calculates $K_1 \oplus K_3$ and sends it to Bob. Upon receiving it, Bob computes $(K_1 \oplus K_3) \oplus K_3$ to obtain K_1. At this point, Alice and Bob securely share the same quantum key K_1. When transmitting data over classical communication networks, both parties can directly use the quantum key K_1 to encrypt information. Notably, this key employs the OTP algorithm, as validated by Shannon, ensure its theoretical security.

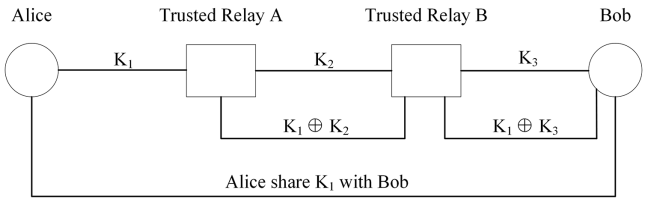

Figure 1: XOR-QKD.

2.2 Measurement Device Independent-QKD (MDI-QKD)

MDI-QKD is a QKD protocol based on untrusted relays, proposed by Hoi Kwong et al. The protocol innovatively installs quantum measurement devices within untrusted relays, ensuring that even if an eavesdropper takes control, the overall security of the system remains unaffected. As illustrated in Fig. 2, its core concept involves delegating quantum measurement to an untrusted third party (typically Charlie). Alice and Bob independently and randomly select BB84-encoded basis vectors (linear polarization basis or diagonal basis) and bit values. They prepare corresponding quantum states using phase-randomized weak light pulses and transmit them to Charlie via a channel. Subsequently, Charlie performs Bell state measurements and publicly discloses the results. Finally, Alice and Bob generate the key based on these measurements.

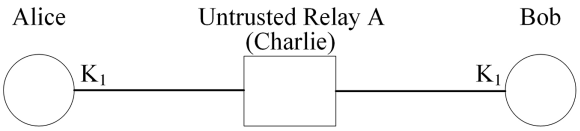

Figure 2: MDI-QKD.

2.3 Multipath-QKD

Multipath-QKD is a QKD scheme based on partially trusted relays, as illustrated in Fig. 3. The communication parties are Alice and Bob. Alice divides the key K to be distributed to Bob into several key fragments $(K_1, K_2, K_3, K_4, K_5)$ and distributes them simultaneously via multiple independent paths, rather than relying on a single path. The relay nodes on these paths are categorized as trusted nodes and untrusted nodes. Trusted nodes possess extremely high security, effectively resisting eavesdropper attacks; untrusted nodes have lower security and may be successfully intercepted. Furthermore, relay nodes along different paths do not overlap—i.e., no relay node appears on more than one path—further reducing the risk of a single point being compromised. Ultimately, upon receiving these key fragments, Bob reconstructs them into a global key through key assembly. Unless an attacker can simultaneously eavesdrop on all paths, they cannot obtain the complete key.

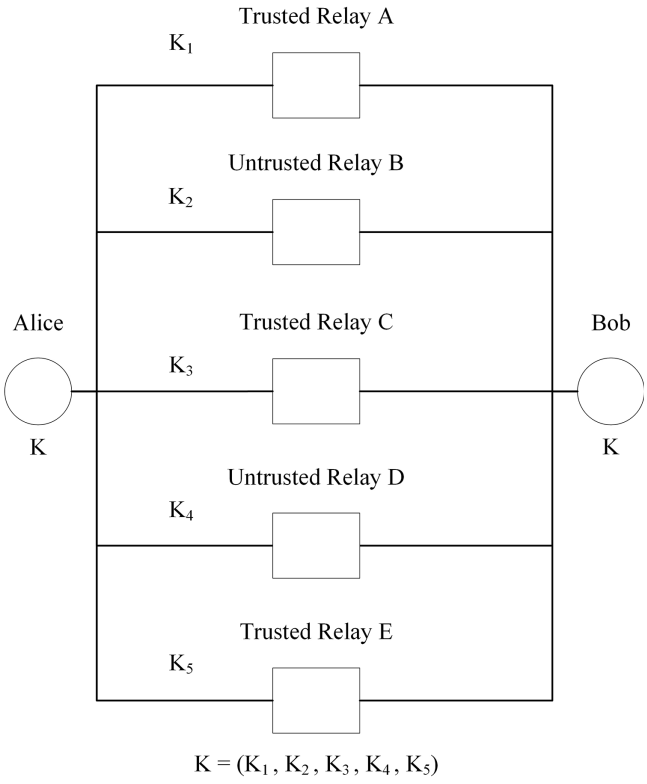

Figure 3: Multipath-QKD.

2.4 Quantum Repeaters-QKD

Briegel et al. first proposed the concept of quantum repeaters [13] and analyzed their role in long-distance quantum communication. Quantum repeaters are key devices for extending the range of quantum

communication. In long-distance quantum communication, quantum repeaters divide the path into multiple short-distance segments. Each relay establishes quantum entangled pairs with its neighboring node. Subsequently, the relay performs Bell state measurements to link the entangled states of adjacent segments. Each node stores the received quantum states in quantum memory, awaiting the completion of entanglement swapping across all segments. Ultimately, the two segments generate a shared key via the entangled state. Due to the nonlocality of entangled states, any eavesdropping attempt is detected, eliminating the need for trust in relay nodes. However, under current technological constraints, quantum repeater technology remains immature. Without employing trusted relay techniques, long-distance QKD cannot be achieved [14]. Future QKD advancements hinge on quantum repeater technology, which is the key solution to overcoming QKD's distance limitations [15,16].

3 Preliminaries

This section introduces the mathematical foundations and key technical features of TFHE used in the public-key TFHE homomorphic encryption QKD scheme, laying the groundwork for the next section Section 4.

3.1 Homomorphic Encryption

Homomorphic encryption is a cryptographic technique that enables specific computations to be performed directly on encrypted data while preserving the functional and structural properties of the data. The decrypted result matches the outcome of operating directly on the plaintext. Taking additive homomorphic encryption as an example, as shown in Eq. (1), for sample messages x and y, encrypting them using the homomorphic encryption function $Encryption(Enc)$ yields $Enc(x)$ and $Enc(y)$. Adding these results produces $Enc(x + y)$, with the entire encryption process requiring no explicit knowledge of x and y.

$$Enc(x) + Enc(y) = Enc(x + y) \tag{1}$$

This differs significantly from traditional encryption schemes. Homomorphic encryption does not require decrypting encrypted data beforehand; instead, it enables mathematical operations and data processing while the plaintext remains unknown. It effectively safeguards user privacy, eliminating concerns about data leaks even when stored in the cloud or on servers. It aligns perfectly with our enhanced solution. Applying homomorphic encryption to QKD effectively prevents key leakage caused by relay attacks.

Looking back at the history of homomorphic encryption, this technology was first named "privacy homomorphism" by Rivest et al. in 1978 [17]. Subsequently, researchers worldwide built upon this foundation to design numerous homomorphic encryption schemes, including: RSA cryptosystem (1978) supporting multiplicative homomorphic operations [18], the encryption algorithm proposed by Goldwasser and Micali (1982) [19], the ElGamal algorithm for randomized encryption (1985) [20], Benaloh's (1994) [21], the widely used additive homomorphic Paillier algorithm (1999) [22], and the encryption algorithm proposed by Boneh et al. (2005) [23], which supports both additive and multiplicative operations. Goh and Nissim's cryptographic scheme (2005) [23], which supports both additive and multiplicative operations, allowing infinite additive homomorphisms but only a single multiplicative homomorphism. However, it was not until 2009 that Gentry constructed the first fully homomorphic encryption scheme supporting arbitrary circuit evaluation in his doctoral dissertation [24]. The scheme is based on ideal lattices and innovatively introduces Bootstrapping technology to reduce ciphertext noise. Building upon this foundation, various novel homomorphic encryption schemes have been proposed. Among the most representative are BGV (2012) [25], BFV (2012) [26,27], GSW (2013) [28], TFHE (2016) [29,30], CKKS (2017) [31]. Among these, the

TFHE algorithm excels at Boolean operations (XOR), aligning with key operations in QKD relays. It enables faster computation of key bits while maintaining resistance to quantum attacks. We attempted to apply it to QKD scenarios and achieved significant results through simulation experiments.

3.2 Learning with Errors (LWE) & Torus Learning with Errors (TLWE)

LWE was proposed by American computer scientist Oded Regev in 2005 [32,33], and remains one of his most significant contributions to date. The LWE problem is a computationally complex problem whose security relies on worst-case lattice problems (such as the Shortest Vector Problem, SVP). It serves as a core tool in post-quantum cryptography. Its fundamental concept is to recover the original secret vector from a set of linear equations deliberately corrupted by errors. It is precisely the presence of these errors that renders the LWE problem computationally challenging.

The formal definition is given below: Dimension n, modulus q, noise distribution χ (typically a Gaussian distribution over $\mathbb{Z}_q$). Noise term $e \leftarrow \chi$. Several samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, each containing a secret vector $s \in \mathbb{Z}_q^n$.

$$b = as + e \mod q \tag{2}$$

Given a genuine sample (a, b) that satisfies equation Eq. (2). Intuitively, when (a, b) is known, it is difficult to determine whether b is the inner product of a and a secret vector s with added noise, or a completely random value. This computational difficulty underlies cryptographic applications of LWE. By exploiting this difficulty, public-key cryptography can be constructed.

Although LWE is secure and versatile, homomorphic operations cause rapid noise expansion and require frequent $mod\ q$ operations, potentially introducing additional noise. It presents a significant flaw in fully homomorphic encryption. The TFHE algorithm ingeniously resolves this issue by incorporating the Torus. The TLWE problem replaces the integer modulus space $\mathbb{Z}_q$ of LWE with the Torus, as detailed in Section 3.3.1. Similar to LWE, the TLWE problem involves samples $(a_i, b_i) \in \mathbb{T}^n \times \mathbb{T}$ satisfying equation Eq. (3).

$$b = as + e \mod 1 \tag{3}$$

3.3 TFHE

TFHE is a Fast Bootstrapping fully homomorphic encryption framework based on the torus, reducing bootstrapping time from seconds to under 0.1 seconds and enabling millisecond-level homomorphic computations. This framework was first proposed by Chillotti et al. in 2016 [29]. In 2020, Chillotti et al. systematically expanded and optimized the 2016 work [30], formalizing the TFHE framework under its official name and open-sourcing it. This scheme represents the third generation of fully homomorphic encryption technology, achieving significant breakthroughs in computational efficiency and practicality compared to previous generations. Next, we will introduce some key features of TFHE.

3.3.1 Torus

The "T" in TFHE stands for "Torus", denoted as $\mathbb{T}$. Its rigorous mathematical definition is the quotient space of the set of real numbers $\mathbb{R}$ by the set of integers $\mathbb{Z}$, as shown in Eq. (4).

$$\mathbb{T} = \mathbb{R}/\mathbb{Z} \tag{4}$$

In layman's terms, it can be visualized by folding the real number line into a torus with a period of 1. Any real number $x \in \mathbb{R}$ corresponds to the element $x \pmod 1$ on the Torus, where every element can be uniquely represented as $x \in [0,1)$ or $(-\frac{1}{2}, \frac{1}{2}]$. To support efficient ring operations (based on Ring Learning with Errors(RLWE)), TFHE further defines the torus over a polynomial ring, denoted as $\mathbb{T}_N[X]$, as shown in Eq. (5). Here, $(X^N + 1)$ is the anti-cyclic polynomial, ensuring compatibility with RLWE.

$$\mathbb{T}_N[X] = \mathbb{R}[X]/(X^N + 1) \quad mod \ 1 \tag{5}$$

The Torus design offers several advantages.

Natural noise compression

In traditional LWE encryption, ciphertext noise expands as the modulus q increases. To ensure correct decryption, the condition $\Delta < \frac{q}{2}$ must be satisfied, where Δ denotes the absolute value of the noise [34]. However, noise after homomorphic operations still needs to be reduced modulo q, which may introduce uncontrolled noise. Torus's "mod 1" property inherently compresses noise. In TFHE, noise e is defined as a small element on the torus, $|e| < \frac{1}{4}$. After homomorphic computation, the noise remains confined within $[0,1)$ or $(-\frac{1}{2}, \frac{1}{2}]$, eliminating the need for additional modulo reduction operations.

Scale invariance

Torus's "mod 1" property grants it scale invariance. Regardless of the original data's length or value range, it can be mapped to a unified space on the Torus for processing without requiring adaptation to data formats or modulus q. It eliminates the complex trade-offs involved in selecting modulus q in traditional schemes (q must simultaneously satisfy security, noise upper bound, and computational efficiency requirements). In other words, for QKD, keys generated across different links may vary in length and format due to device and protocol differences. However, after Torus processing, TFHE homomorphic operations (key XOR operations at relays) can be executed directly within this unified space. It eliminates the need to design encryption parameters individually for each QKD link, thereby reducing adaptation costs for cross-scenario applications.

High-efficiency ring

The polynomial extension of the torus, denoted as $\mathbb{T}_N[X]$, forms the foundation for realizing Torus Ring Learning with Errors(TRLWE). Within $\mathbb{T}_N[X]$, ring operations can be accelerated using Fast Fourier Transform (FFT) operations, reducing computational complexity from $O(N^2)$ to $O(N \log^N)$ [35]. Furthermore, the external product operation $\boxdot$ in TRGSW requires multiplying polynomials by vectors over $\mathbb{T}_N[X]$. The periodicity of the torus ensures that the result remains within the controlled space, thereby preventing coefficient overflow.

3.3.2 Fast Bootstrapping

The core challenge of fully homomorphic encryption is noise accumulation. After each round of homomorphic computation, noise within the ciphertext is amplified. Following a finite number of homomorphic operations, the noise exceeds the decryption threshold, rendering plaintext recovery impossible. To overcome the limitation, Gentry introduced a novel technique called Bootstrapping [24,36], which addresses the issue by refreshing noise. Early algorithms like BGV and BFV achieved full homogeneity using this technique, but it imposed significant computational overhead, reducing efficiency and hindering the practical implementation of FHE. The Fast Bootstrapping technique proposed in TFHE drastically reduces the noise refresh time, enabling real-time applications such as QKD key relaying.

The breakthrough of Fast Bootstrapping technology lies in its mathematical structure, based on the torus, which replaces the traditional discrete integer ring with a continuous torus to simplify noise quantization and computational rules. Simultaneously, through Noiseless Blind Rotation and a Look-Up Table (LUT), the core bootstrapping step is simplified from polynomial multiplication to vector operations. Its essence lies in performing the decryption-re-encryption process under encryption, achieving noise reset via homomorphic computation of symbolic functions. Let the ciphertext be defined as $c = (a, b)$, the phase as φ, and the private key as s. The bootstrapping key for the private key component encrypted by $Torus\ GSW(TGSW)$ is $bk = \{TGSWEnc(s_0), TGSWEnc(s_1), ..., TGSWEnc(s_{n-1})\}$. The homomorphic computation symbol function $Sign(\varphi)$. enumerates all possible values of φ in the plaintext domain, computes the corresponding $Sign(\varphi)$ results, and stores them as the LUT. During homomorphic computation, no real-time derivation is required; results are obtained directly by matching φ via the LUT, significantly enhancing computational efficiency and enabling real-time processing of QKD keys. The core steps are as follows:

1. **Homomorphic decryption:** Perform homomorphic computation on the ciphertext $c = (a, b)$ using bk without decryption to obtain the encrypted phase $Hom(\varphi)$.
2. **Symbolic functions mapping:** Calculate $Sign(\varphi)$ via LUT. While preserving the plaintext information m in the output, it also compresses the noise back to the initial security level ($e_{total} < \Delta/4$) to ensure that subsequent homomorphic operations still satisfy the decryption threshold requirements.
3. **Output new ciphertext:** Convert the output of $Sign(\varphi)$ into a standard TLWE ciphertext to complete noise reset.

In our TQKD solution, Fast Bootstrapping serves as the core technology for implementing cryptographic key processing on untrusted relay nodes. It reduces processing time to under 0.1 s, meeting the real-time requirements of QKD systems. Since QKD keys undergo multi-hop relaying, each hop introduces additional noise. Fast Bootstrapping resets this noise from the threshold edge back to its initial level. Consequently, ciphertexts remain decryptable even after multiple homomorphic processing steps at relays, eliminating limitations imposed by the number of homomorphic operations. The security of Fast Bootstrapping relies on the LWE problem over a Torus, providing quantum-resistant properties that complement QKD's unconditional security. The entire process operates in an encrypted state, preventing relay nodes from accessing key information and ensuring key security.

3.3.3 Efficient Boolean Operation

For performing Boolean operations on encrypted bit strings, TFHE employs an efficient processing mode. Its core mechanism involves applying external multiplication and bootstrapping to achieve efficient XOR operations. The specific principle is as follows:

In TFHE, reference [29] proposes combining TLWE and TGSW, followed by noise reduction via Fast Bootstrapping. Specifically, TLWE samples are used to encrypt data, while TGSW samples encrypt control bits. These two can be multiplied to form new TLWE samples. Applying this method to logic gate circuits yields a series of basic logic gates, including $Homomorphic\ NOT(c)(HomNOT(c))$, $Homomorphic\ AND(c_1, c_2)HomAND(c_1, c_2))$, $Homomorphic\ NAND(c_1, c_2)(HomNAND(c_1, c_2))$, $Homomorphic\ OR(c_1, c_2)(HomOR(c_1, c_2))$, and $HomXOR(c_1, c_2)$, where $c_1 = (a_1, b_1)$ and $c_2 = (a_2, b_2)$ are single-bit TLWE ciphers under TFHE. For our homomorphic encryption QKD scheme, $HomXOR(c_1, c_2)$ requires detailed elaboration. We will briefly verify it next. As shown in Eq. (6), it can also be implemented via $Bootstrap(2(c_1 + c_2))$.

$$HomXOR(c_1, c_2) = Bootstrap(2(c_1 - c_2)) \tag{6}$$

As shown by the specific algorithm in Section 4.1, the homomorphic addition of two ciphers results in $2(c_1 - c_2)$. After decryption, the phase result is φ, as detailed in Eq. (7).

$$\varphi = 2\Delta(m_1 - m_2) + 2(e_1 - e_2) \quad mod \ \ q \tag{7}$$

Fast Bootstrapping refreshes the noise after HomXOR, resetting it to its initial level. The results are shown in Table 1, where (m_1, m_2) represents the four plaintext combinations of bits, $XOR(m_1, m_2)$ denotes the XOR result of the plaintext combination, and φ is the core information obtained after eliminating the private key component during decryption. $Bootstrap(*)$ denotes the offset from zero after Fast Bootstrapping the phase $\varphi = 2 \cdot (c_1 - c_2)$. If the phase is close to zero, it maps to zero; if the phase is far from zero, it maps to one, yielding the result of $HomXOR(c_1, c_2)$.

Table 1: HomXOR.

(m_1, m_2)	$XOR(m_1, m_2)$	φ	$Bootstrap(*)$	$HomXOR$
(0, 0)	0	e	Close 0	0
(0, 1)	1	2Δ	Apart 0	1
(1, 0)	1	2Δ	Apart 0	1
(1, 1)	0	e	Close 0	0

Through linear operations of addition, subtraction, and scaling, the difference (c_1, c_2) between ciphertexts is transformed into a distinguishable phase state. Subsequently, a single Fast Bootstrapping step eliminates noise and maps the result to 0/1, ultimately achieving the HomXOR operation. The entire process avoids nonlinear operations or complex gate circuits (e.g., AND, MUX), with noise linearly superimposed. Subsequent processing requires only a single lightweight Fast Bootstrapping step. It represents a lightweight approach that bypasses complex gate combinations by directly leveraging the phase characteristics of TLWE.

4 TQKD

We designed the TQKD scheme, whose core leverages the asymmetric encryption properties of public-key TFHE and its efficient Boolean homomorphic computation capabilities to enhance QKD efficiency. Simultaneously, based on the Zero-Trust architecture proposed by Brazaola-Vicario et al. [37], we addressed the trust bottleneck in traditional QKD networks caused by relay nodes, thereby reducing trust dependencies and mitigating single-point risks. The approach also aligns with NIST requirements, Specifically, the proposed Zero-Trust Architecture (ZTA) [38], which aims to minimize trust boundaries within systems and avoid reliance on broad trust domains.

4.1 Public-Key TFHE

The TFHE algorithm stands as a quintessential example of efficient fully homomorphic encryption technology. However, it is noteworthy that its encryption and decryption processes rely on the same set of keys, fundamentally constituting a symmetric-key-based encryption system. This characteristic limits TFHE's applicability in scenarios requiring shared keys and identity authentication, such as multi-party secure communication and cloud storage. To address this issue, Marc Joye proposed a public-key version of TFHE [39], which overcomes this limitation and is particularly suited for untrusted relay scenarios in QKD networks. Public-key TFHE transforms TLWE's symmetric-key architecture into an asymmetric form via a key-separation mechanism, while preserving TFHE's torus structure and noise-control advantages to

maximize efficient fully homomorphic computation. To facilitate the description of the TQKD scheme, we define the algorithm below.

Definition 1: *TQKD comprises five algorithms: Setup, SecretKeyGen, Encreption, Decryption, and HomXOR.*

- **Setup:** Generate the public parameter set (pp) for the QKD scenario (Algorithm 1).
- **SecretKeyGen:** Generate a public-private key pair for encryption and decryption (Algorithm 2).
- **Encreption:** Input message m, output ciphertext c (Algorithm 3).
- **Decryption(Dec):** Input ciphertext c, output message m (Algorithm 4).
- **HomXOR:** Input two ciphertexts (c_1, c_2), obtain the ciphertext after homomorphic computation (Algorithm 5).

Algorithm 1: Setup

Input: λ

define positive integers t, q, M
let $\Delta = q/t$
$n = \phi(M)$ where φ donates Euler's totient function.
define $\hat{\chi}_1, \hat{\chi}_2$ discretized error distributions over $\mathbb{Z}$
let $m = \{0,1\}^t$ be the massage space
computing LUT

Output: $pp = (\Delta, t, q, n, \hat{\chi}_1, \hat{\chi}_1, m, LUT)$

Algorithm 2: SecretKeyGen

Input: pp

sample $\boldsymbol{s} = (s_1, ..., s_n) \xleftarrow{\$} \{0,1\}^n$
sample $\mathfrak{a} \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^n$
sample $\boldsymbol{e} \leftarrow \hat{\chi}_1$
$\mathfrak{b} = \mathfrak{a} \circledast \boldsymbol{s} + \boldsymbol{e} \in (\mathbb{Z}/q\mathbb{Z})^n$
sk = $\boldsymbol{s}$
pk = $(\mathfrak{b}, \mathfrak{a})$
bk = $\{TGSWEnc_{pk}(s_i) | i = 1, ..., n\}$

Output: pk, sk, bk

Algorithm 3: Encreption

Input: pk = $(\mathfrak{b}, \mathfrak{a}) \in (\mathbb{Z}/q\mathbb{Z})^n (\mathbb{Z}/q\mathbb{Z})^n$, $m \in \{0,1\}^t$

sample $\boldsymbol{r} \xleftarrow{\$} \{0,1\}^n$
sample $\boldsymbol{e}_1 \leftarrow \hat{X}_1, e_2 \leftarrow \hat{X}_2$
$\boldsymbol{a} = \mathfrak{a} \circledast r + e_1$
$b = \langle \mathfrak{b}, \boldsymbol{r} \rangle + \Delta m + e_2$
$\boldsymbol{c} = (\boldsymbol{a}, b)$

Output: $\boldsymbol{c} \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$

Algorithm 4: Decryption

Input: $\boldsymbol{c} = (\boldsymbol{a}, b) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$, $\boldsymbol{s} \in \{0,1\}^n$

1: $\varphi = (b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle)\ mod\ q$

2: $m = \lceil \varphi/\Delta \rfloor\ mod\ t$

Output: $m \in \{0,1\}^t$

Algorithm 5: HomXOR

Input: $\boldsymbol{c}_1 = (\boldsymbol{a}_1, b_1) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}, \boldsymbol{c}_2 = (\boldsymbol{a}_2, b_2) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$

1: $\boldsymbol{a}' = \boldsymbol{a}_1 - \boldsymbol{a}_2 \quad mod\ q$

2: $b' = b_1 - b_2 \quad mod\ q$

3: $\boldsymbol{c}' = (\boldsymbol{a}', b')$

4: $\boldsymbol{c} = Bootstrap(2\boldsymbol{c}')$

Output: $\boldsymbol{c} \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$

These algorithms are based on the work of Chillotti, Joye, and others [29,30,39], and we do not elaborate on the specific proofs and details here. Throughout the TQKD scheme, both the public-private key pairs and ciphertexts are derived from the integer modulus field $(\mathbb{Z}/q\mathbb{Z})$. It does not conflict with Torus but rather represents a complementary relationship between theoretical models and engineering applications. Torus and TLWE provide the theoretical foundation for cryptographic security and noise control, while $(\mathbb{Z}/q\mathbb{Z})$ serves as the carrier for discretizing the Torus, enabling data storage and computation. For any Torus element $x \in \mathbb{T}$, it can be discretized in the integer modulus as $\lfloor xq \rfloor\ mod\ q$. This mapping quantizes the continuous Torus space into an integer space, enabling computers to process Torus elements through integer operations while avoiding precision issues associated with computing continuous real numbers.

4.2 TQKD Network Model

We divide the QKD network into three layers: Quantum Layer at the bottom, Key Network Layer in the middle, and Key Service Layer at the top [40,41], as shown in Fig. 4. The network is described in detail as follows:

1. **Quantum Layer:** It serves as the physical foundation and primary source of TQKD networks, composed of optical nodes interconnected via optical links (fiber-optic cables). Based on quantum mechanical principles (the no-cloning theorem and the uncertainty principle), as well as QKD protocols such as BB84, E91, and B92 [1,42,43], it primarily involves the preparation, detection, and post-processing of quantum states to achieve secure key agreement between adjacent nodes. Its core function is to generate unconditionally secure quantum keys between two directly connected nodes.
2. **Key Management Layer:** As the core control layer of the TQKD network, it validates and manages previously generated keys, serving as a bridge between layers. It interfaces downward with the point-to-point keys from the Quantum Layer, extending the reach of key distribution through hop-by-hop relaying. Upward, it provides schedulable key resources to the Key Service Layer while maintaining the network-wide key state. Its core function is to resolve the challenge of converting short-range keys generated at the quantum layer into wide-area, routable key resources.
3. **Key Service Layer:** As the user interface layer of the TQKD network, it is responsible for converting the raw quantum keys provided by the Key Network Layer into usable keys suitable for various application scenarios. Its core function is to ensure application systems can conveniently and securely utilize QKD keys.

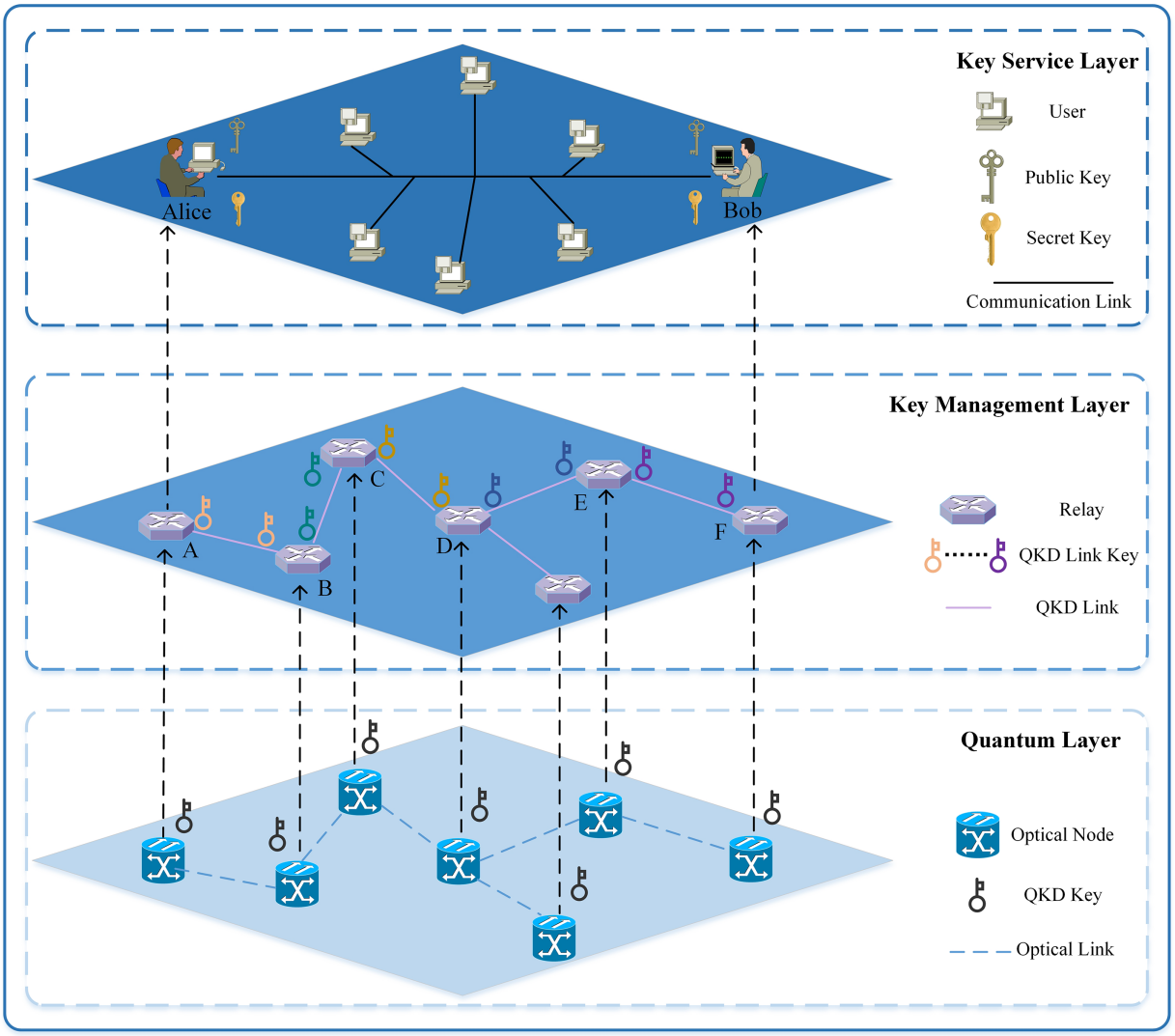

Figure 4: TQKD network model based on relay.

During the key generation phase, the Optical Node in the Quantum layer generates quantum keys using a quantum random number generator (QRNG) [44], which exploits the inherent randomness of quantum mechanics (e.g., superposition states, measurement collapse). This randomness stems from the depletion of quantum coherence, enabling the generation of truly random numbers—a capability unavailable in classical physical schemes. Leveraging the uncertainty principle and the no-cloning theorem, eavesdropping during key exchange is prevented, ensuring attackers cannot intercept or alter information. When the key is transmitted to the Key Network Layer, each Relay (labeled A through F) connects via QKD links. Each QKD link shares a QKD key, termed the QKD Link Key, represented by the same color in the diagram. At this layer, keys are transmitted via Relays using homomorphic encryption. Intermediate Relays remain unaware of the key information, enhancing key privacy. Within the Key Service Layer, each user possesses a corresponding public-private key pair (the diagram illustrates two users, Alice and Bob). These pairs are used to encrypt the QKD link keys from the Key Network Layer, ensuring the smooth operation of homomorphic encryption. After obtaining the recipient's QKD Key, the sender encrypts the message using OTP (a theoretically

unbreakable encryption method proposed by information theory founder Shannon [45]), thereby enabling secure communication.

Next, we will demonstrate a complete encrypted communication process using this scheme to understand how public-key TFHE works in conjunction with QKD and the role of homomorphic encryption in this process. As shown in Fig. 5, Alice wishes to communicate with Bob via QKD and plans to send the generated quantum key $qkey$ to Bob. The specific steps are as follows:

1. During the initialization phase, each user generates a TFHE public-private key pair (pk, sk) and a bootstrapping key bk (Algorithm 1). Simultaneously, adjacent relay nodes share a quantum key $key_{ab}, key_{bc}...$ via QKD, preparing for peer-to-peer QKD.
2. Alice sends a QKD request to Bob via the communication link, intending to distribute a quantum key to Bob through the QKD link. Each relay node obtains Bob's public key pk and encrypts the quantum key using algorithm Algorithm 3, yielding $Enc(key), Enc(key_{ab}), Enc(key_{bc}), \ldots, Enc(key_{ef})$.
3. Relay A performs homomorphic encryption on $Enc(key)$ and $Enc(key_{ab})$ using Algorithm 5 (Eq. (8)), yielding $Enc(key \oplus key_{ab})$, then transmits the result to Relay B.

$$Enc(key) \ \oplus \ Enc(key_{ab}) = Enc(key \oplus key_{ab}) \tag{8}$$

4. Relay B receives $Enc(key \oplus key_{ab})$ and similarly performs homomorphic computation on $Enc(key_{ab})$ and $Enc(key_{bc})$ via Algorithm 5.

$$Enc(key \oplus key_{ab}) \ \oplus \ Enc(key_{ab}) = Enc(key) \tag{9}$$

yielding $Enc(key \oplus key_{bc})$ (Eqs. (9) and (10)), then sends the result to Relay C.

$$Enc(key) \ \oplus \ Enc(key_{bc}) = Enc(key \oplus key_{bc}) \tag{10}$$

5. Subsequent relays receive the homomorphic encryption result and perform the same steps until the final relay F receives the ciphertext $Enc(key \oplus key_{ef})$. Then, using homomorphic operations, the quantum key key encrypted with the public key is recovered (Eq. (11)).

$$Enc(key \oplus key_{ef}) \ \oplus \ Enc(key_{ef}) = Enc(key) \tag{11}$$

6. Bob uses the private key sk of the TFHE algorithm to decrypt $Enc(key)$ via Algorithm 4 (Eq. (12)), thereby obtaining the quantum key Alice intends to distribute.

$$key = Dec(Enc(key)) \tag{12}$$

In this example, we have not described the Fast Bootstrapping process nor depicted it in the diagram. The process occurs during the HomXOR procedure. It is worth noting that Alice and Bob merely represent the communicating parties. For explanatory purposes, only these two users are highlighted in the network. However, each user corresponds to a relay and a QRNG. These nodes can not only generate quantum keys but also act as relays to forward quantum keys from other users, regardless of their position within the network topology.

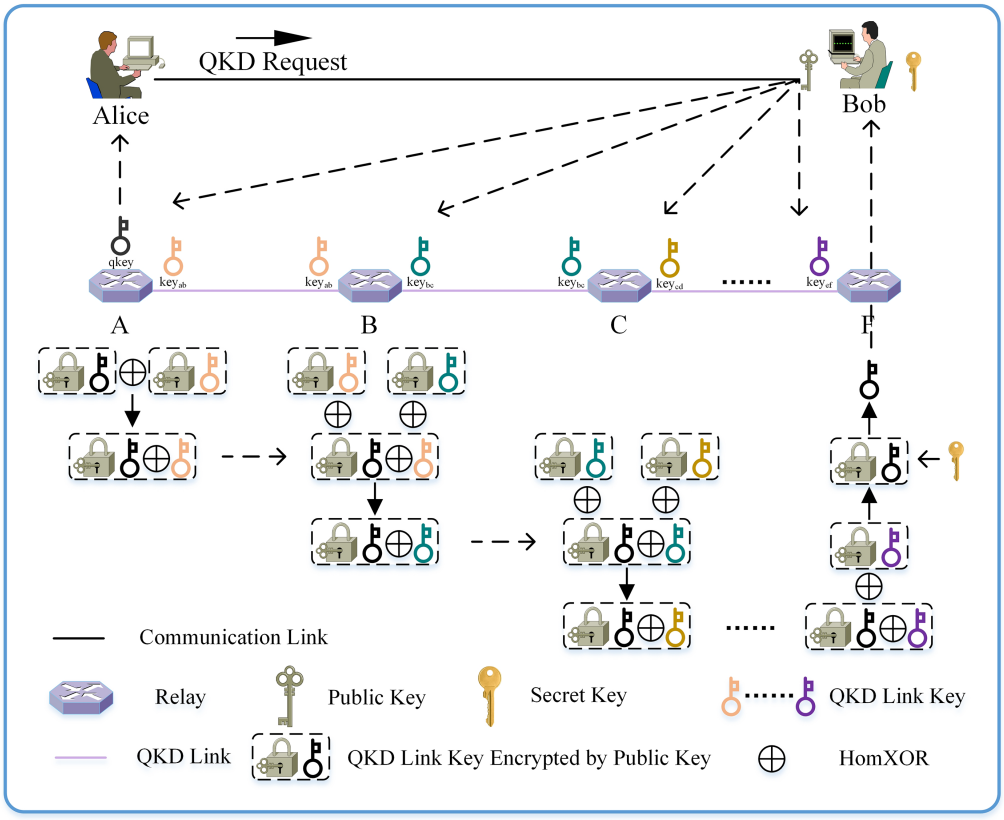

Figure 5: TQKD communication.

4.3 TQKD Security Proof

The network model of TQKD is based on a quantum-classical hybrid channel architecture, whose security proof encompasses both the information-theoretic security at the quantum layer and the computational security at the key management layer. Since the security at the quantum layer is grounded in fundamental principles of quantum mechanics, independent of any computational assumptions, and is widely regarded as unconditional, this paper does not delve into its security proof in detail. Instead, it focuses on demonstrating the computational security involving the homomorphic encryption algorithm. Below, we will prove that the TQKD scheme is IND-CPA secure based on Definition 2.

Definition 2: *We define an honest but curious adversary $\mathcal{A}$, a challenger $\mathcal{C}$, and the rules governing their behavior.*

$\mathcal{A}$:

- Possessing probabilistic polynomial-time(PPT) computational capabilities, it cannot resolve fundamental problems in quantum mechanics or those that are hard on lattices.
- Control relay nodes to strictly adhere to the public-key TFHE algorithm process, collecting publicly available information $(pp, pk, bk...)$ without tampering with data or terminal transmission.
- Recover quantum keys from intercepted information or distinguish plaintexts corresponding to homomorphic encrypted ciphers.

$\mathcal{C}$:

- Generate $pp \leftarrow Setup(\lambda)$ and provide it to $\mathcal{A}$.

- Generate $(pk, sk, bk) \leftarrow SecretKeyGen(pp)$. When $\mathcal{A}$ issues a request, pk, bk are distributed. $\mathcal{A}$ may request a polynomial number of such keys.

Definition 3: *For any PPT adversary $\mathcal{A}$ and two challenge plaintexts $m_0, m_1 \in M$, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The advantage of game G_i is defined as: $Adv_{G_i} = |Pr[\mathcal{A}(pk, c) = 1 | c = Enc(pk, m_0)] - Pr[\mathcal{A}(pk, c) = 1 | c = Enc(pk, m_1)]|$, where the function $negl(\lambda)$ is negligible: for any polynomial $P(\lambda)$, when λ is sufficiently large, $negl(\lambda) < 1/P(\lambda)$.*

We have demonstrated that TQKD is IND-CPA secure through the following series of Games. Game_0 is the baseline game, fully reproducing the original scheme's encryption logic. Both the public key and ciphertext components are TLWE samples. The advantages of Game_0 are shown in Eq. (13).

$$Adv_{G_0} = |Pr[\mathcal{A}(pk, Enc(pk, m_0)) = 1] - Pr[\mathcal{A}(pk, Enc(pk, m_1)) = 1]| \tag{13}$$

Game $_1$ substitutes the public key with a randomly selected pk' from the TLWE sample, rather than using the actual public key from the original scheme. The advantage of Game $_1$ is shown in Eq. (14).

$$Adv_{G_1} = |Pr[\mathcal{A}(pk', Enc(pk', m_0)) = 1] - Pr[\mathcal{A}(pk', Enc(pk', m_1)) = 1]| \tag{14}$$

Since the public keys of Game_0 and Game_1 are distribution-equivalent, any PPT attacker cannot distinguish between their public keys. $|Adv_{G_0} - Adv_{G_1}| \leqslant negl(\lambda)$, $G_0 \leftrightarrow G_1$ are indistinguishable.

Game $_2$ samples a public key pk'' from a uniform random distribution, with Game $_2$'s advantage as shown in Eq. (15).

$$Adv_{G_2} = |Pr[\mathcal{A}(pk'', Enc(pk'', m_0)) = 1] - Pr[\mathcal{A}(pk'', Enc(pk'', m_1)) = 1]| \tag{15}$$

Given the TLWE assumption that $pk^{'}$ sampled from a uniform random distribution is indistinguishable from pk' sampled from the TLWE distribution, then $|Adv_{G_1} - Adv_{G_2}| \leqslant negl(\lambda)$, and $G_1 \leftrightarrow G_2$ are indistinguishable.

Finally, using the triangle inequality, the advantage differences across games are combined, as shown in Eq. (16):

$$Adv_{G_0} \leqslant |Adv_{G_0} - Adv_{G_1}| + |Adv_{G_1} - Adv_{G_2}| \leqslant negl(\lambda) + negl(\lambda) = negl(\lambda) \tag{16}$$

Through this series of games, the security of the TQKD scheme is reduced to the computational hardness of the TLWE problem, thereby proving the IND-CPA security of TQKD.

5 Experiment

To validate the feasibility and superiority of the QKD scheme based on the public-key TFHE algorithm, we designed a series of experimental scenarios in Python. These experiments evaluate the TQKD scheme's performance metrics for homomorphic operation latency, ensuring it meets the real-time requirements of QKD networks. Simultaneously, by comparing it with QKD schemes based on homomorphic encryption algorithms such as BGV and BFV, we highlight the TQKD scheme's unique performance advantages and provide data to support its engineering implementation.

5.1 Experiment Design

Currently, the open-source community has developed multiple mature toolkits with distinct functionalities. Among them, the most prominent and active open-source homomorphic encryption libraries

include: Simple Encrypted Arithmetic Library(SEAL) [46] developed by Microsoft, OpenFHE [47] released by the open-source community, Homomorphic Encryption library(HElib) [48] developed and maintained by IBM, and concreteML [49] developed by Zama. However, our approach involves bit-level homomorphic encryption operations and requires support for BGV, BFV, and TFHE algorithms. OpenFHE supports most mainstream algorithms, including BGV, BFV, CKKS, FHEW, and TFHE, making it suitable for our experimental requirements.

OpenFHE is an open-source, fully homomorphic encryption library written in C++. Maintained by a community comprising experts from multiple universities and industry, it can be regarded as the official continuation and significant evolution of the early renowned HE library PALISADE [50]. A standout feature of OpenFHE is its comprehensive support for homomorphic encryption schemes, encompassing nearly all mainstream fully homomorphic encryption protocols. Furthermore, it provides comprehensive Application Programming Interface(API) documentation [51] and extensive code examples, significantly lowering the learning curve. OpenFHE delivers a modern, fully-featured, modular, high-performance, and user-friendly homomorphic encryption development platform suitable for both academic research and industrial applications.

To highlight the speed advantage of the TFHE algorithm in QKD, we designed two experimental scenarios for verification:

1. **Single-Pass Homomorphic Encryption:** This experimental scenario simulates a HomXOR operation performed on encrypted keys within a single relay node, without forwarding to subsequent nodes or the recipient. Specifically, it involves a single HomXOR operation between two encrypted keys. Its core objective is to validate fundamental performance differences among BGV and BFV technical solutions on the shortest path by comparing their key performance metrics. The process aims to eliminate path-length-related performance interference (such as the number of relay nodes and physical distance), ensuring that performance differences stem solely from the inherent design of the technical solutions rather than topological complexity. It establishes a baseline for subsequent multi-relay, multi-hop performance analysis.
2. **Multi-Hop Relay Homomorphic Encryption:** Simulating quantum key distribution in practical scenarios, where the sender forwards the key to the receiver through multiple relay nodes. This tests the scheme's noise control characteristics during long-distance transmission and verifies the performance differences between TFHE and BGV/BFV algorithms after multi-hop relay forwarding and repeated homomorphic encryption. In multi-hop scenarios, ciphertexts undergo multiple HomXOR operations, causing noise accumulation. To ensure accurate decryption of the ciphertext, noise reduction is required. TFHE employs Fast Bootstrapping technology to reduce noise-reduction time, thereby accelerating key distribution. It provides theoretical data support for the practical application of the technical solution.

The specific experimental hardware environment is shown in Table 2:

Table 2: Hardware environment configuration.

Hardware Category	Parameter
Central Processing Unit (CPU)	Intel (R) Core (TM) i7-1065G7 CPU @ 1.30 GHz
Operating System (OS)	Ubuntu 24.04.2 LTS
Cache Size	8 GB DDR4 Dual Channel
Memory	30 GB SSD

(Continued)

Table 2 (continued)

Hardware Category	Parameter
Python	Python 3.8.20
OpenFHE	1.4.0.1.20.4

Note: All hardware is in its default factory state, with no overclocking or modification of hardware parameters.

In the experiments, we disregarded the time spent on model initialization and key initialization, recording only the duration from the start to the completion of the bit-homomorphic computation in the code. To ensure the accuracy and scientific rigor of the experimental data, we repeated the experiments multiple times and filtered the data from each run. Based on the variance and mean, we discarded data points that deviated excessively from the mean, thereby approximating a normal distribution. For the TFHE algorithm, a dedicated API for XOR logic gates is available. However, for the BFV and BGV algorithms, no specialized XOR logic gates exist. Consequently, the XOR functionality must be implemented through additive and multiplicative operations (Eq. (17)).

$$a \oplus b = (a + b) - 2 * a * b \tag{17}$$

Furthermore, in practical engineering applications, quantum bits in QKD are continuously generated and encrypted. This means that the sender or relay typically does not wait for a certain number of quantum bits to accumulate before performing operations; instead, operations are conducted bit by bit. Therefore, we also simulated this scenario without using the batch-processing capabilities of the BFV and BGV algorithms in OpenFHE (specifically, acceleration via Python's list data structure). Both algorithms process only one quantum bit at a time, rather than a string of quantum bits, and TFHE operates similarly. Consequently, the experimental data in Section 5.2 are based on single-quantum-bit operations, not the computation time for entire strings of quantum bits.

5.2 Result and Analysis

We conducted simulation experiments with OpenFHE to demonstrate the feasibility of the proposed TQKD scheme. TQKD not only achieves faster HomXOR computation speeds than BGV and BFV algorithms but also supports accommodating more relays in QKD network while maintaining the original encryption efficiency compared to the other two algorithms. Notably, we also observed an interesting phenomenon: under specific parameter settings, the XOR operation efficiency of TFHE algorithms is lower than that of BGV and BFV algorithms. However, the special case above is uncommon in practical applications and therefore does not affect our conclusions.

First, we analyze the first experimental scenario: performing a single homomorphic encryption operation on two distinct ciphers. The algorithms used in the experiment are TFHE, BFV, and BGV. Due to the construction principles of the BFV and BGV algorithms, it is necessary to set the maximum number of consecutive multiplications allowed in homomorphic operations, known as the multiplicative depth. In other words, it represents the maximum number of consecutive homomorphic operations (including multiplication, where addition does not consume a count) that can be performed on the same ciphertext while still guaranteeing successful decryption. As shown in Eq. (17), BFV and BGV require two multiplications to perform a single XOR operation. Therefore, we set the multiplicative depth of BFV and BGV to 2. Interestingly, at this point, the computational efficiency of BFV and BGV surpasses that of TFHE, requiring less time—contrary to previous theoretical analysis. Subsequently, we progressively increased the

multiplicative depth of BFV and BGV. The experimental results, shown in Fig. 6, reveal that the computation time is positively correlated with the multiplicative depth. When the multiplicative depth reaches 7, the TFHE algorithm achieves higher computational efficiency than BFV and BGV, consuming less time—a result consistent with the theoretical analysis. Subsequently, we conducted comparative experiments on CPU and RAM usage across the three algorithms at different multiplicative depths, with results shown in Figs. 7 and 8.

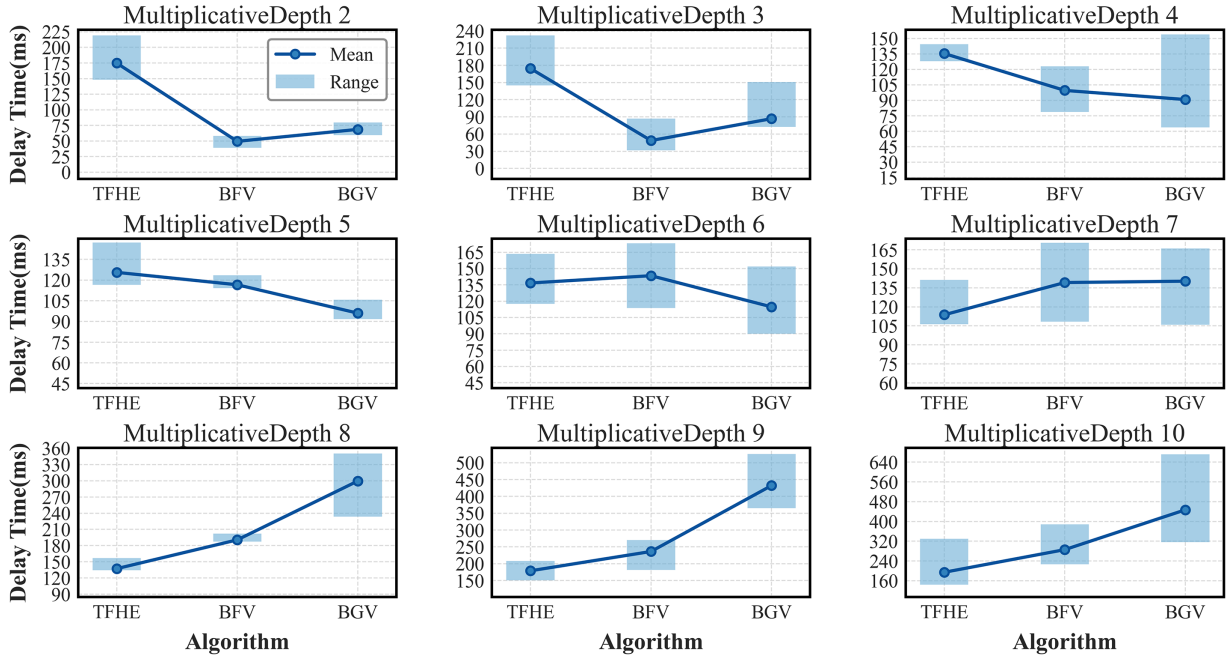

Figure 6: Comparison of computational time for TFHE, BFV, and BGV at different multiplication depths.

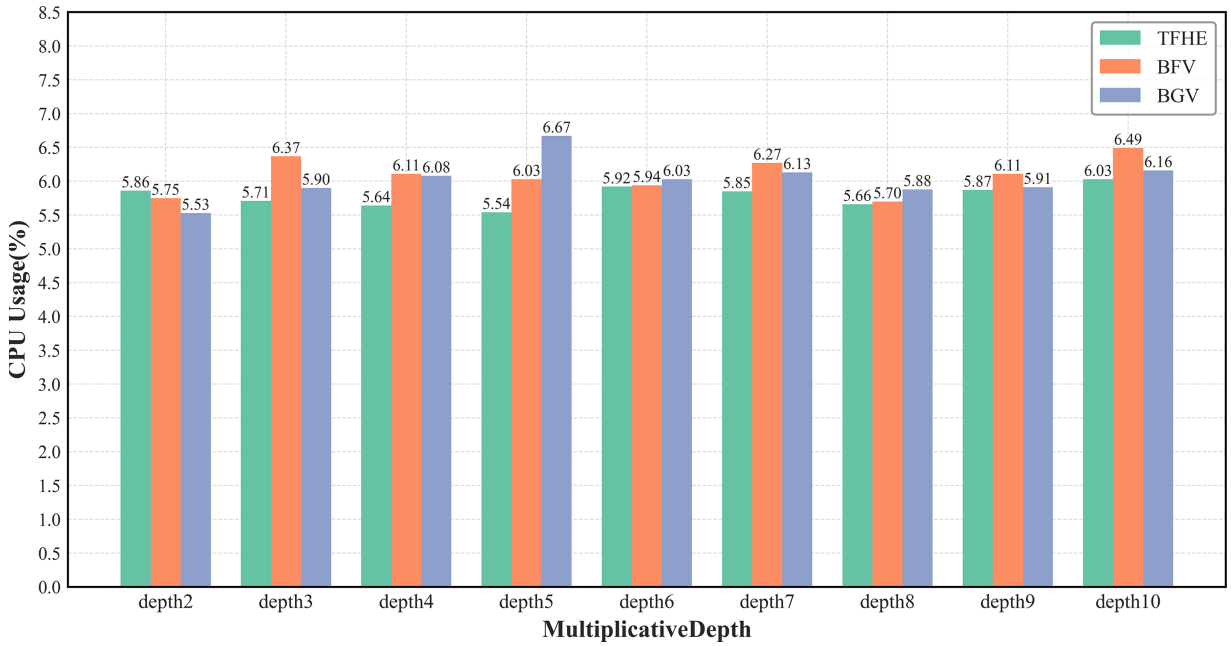

Figure 7: Comparison of CPU usage for TFHE, BFV, and BGV algorithms at different multiplicative depths.

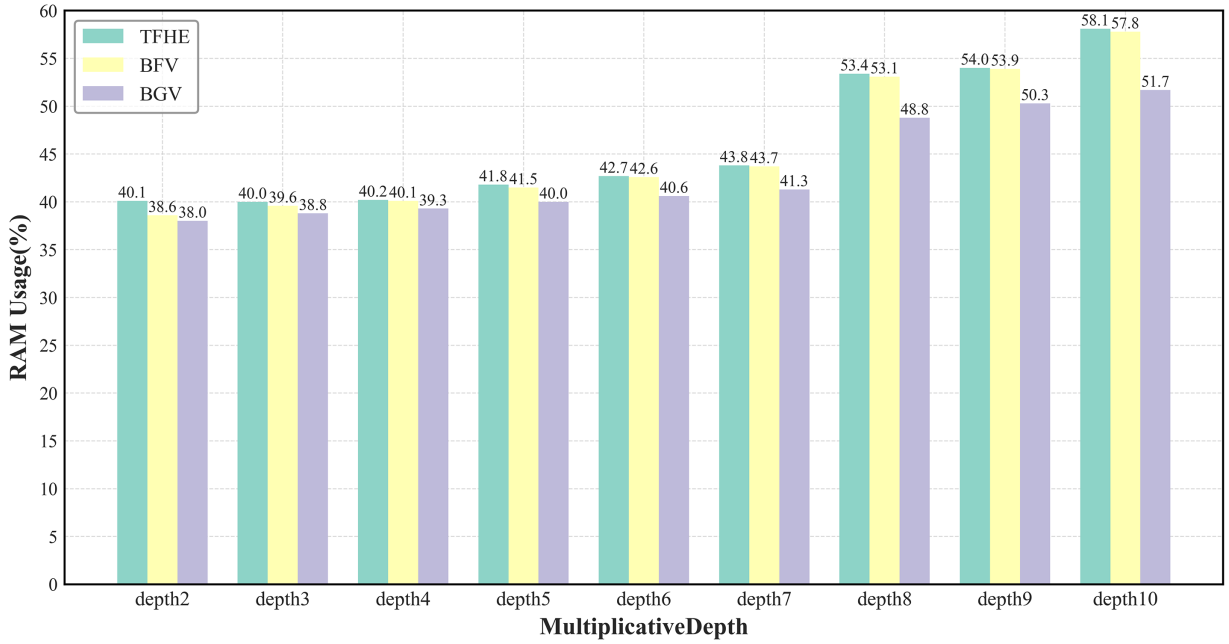

Figure 8: Comparison of RAM usage for TFHE, BFV, and BGV algorithms at different multiplicative depths.

Regarding CPU usage, TFHE remained consistently around 5.8%. When the multiplicative depth was set to 2, BGV and BFV exhibited slightly lower CPU usage than TFHE, indicating higher computational efficiency. However, as the multiplicative depth increased, their CPU usage exceeded TFHE's, while TFHE maintained superior performance. Regarding RAM usage, although the optimized XOR operation in TFHE increases memory consumption, it only requires slightly more RAM compared to BGV and BFV algorithms. This trade-off yields a significant reduction in computation time, making homomorphic XOR more efficient. This approach, sacrificing a small amount of space for substantial time savings, is worthwhile.

Through analysis, we identified discrepancies between the experimental results and the theoretical analysis. These discrepancies stem not from flaws in the theoretical framework itself, but from the strong correlation between the computational overhead of the BFV and BGV algorithms and their multiplicative depth. The multiplicative depth of both algorithms directly determines the complexity of the cryptographic context, thereby impacting computational speed. Within the parameter generation logic, a greater multiplicative depth necessitates a larger Ring Dimension (RingDim), increasing the number of polynomials involved in the computation and consequently elevating computational overhead. Simultaneously, the multiplicative depth impacts the allocation of the initial noise budget. With a smaller multiplicative depth, there is no need to reserve noise space for subsequent rounds of multiplication or perform additional noise control. Conversely, a larger multiplicative depth necessitates reserving a substantial noise budget for future rounds, thereby increasing the overhead of noise control and slowing down computation speed. However, the BFV and BGV algorithms exhibit faster performance because their parameters are over-optimized, operating within an optimal performance range. At this point, both the RingDim of the algorithm model and noise control operate at minimal overhead. It effectively streamlines BFV and BGV algorithms to support only the current task, thereby maximizing computational speed.

In practical applications, sender encryption and receiver decryption each require one HomXOR operation, consuming a total of four multiplicative depths. Each relay node performs two HomXOR operations; if there are n relay nodes, the multiplicative depth for the BFV and BGV algorithms is $4 + 4n$. A QKD

network includes multiple relay nodes, and once the number of nodes exceeds 1, the computational efficiency of BFV and BGV falls behind that of TFHE. Therefore, the advantages of these two algorithms under lightweight implementations are not significant in practical applications. It does not affect the conclusion that our proposed TQKD scheme achieves higher computational efficiency. In Table 3, we summarize the overall performance of TFHE algorithms vs. BFV/BGV for the variable multiplicative depth. When the multiplicative depth is small, BFV/BGV algorithms perform better because shallower depths require fewer initialization parameters. However, in actual HomXOR, the multiplicative depth increases with the number of relays, rendering shallow depths impractical. When the multiplicative depth is large, TFHE demonstrates superior overall performance, enabling the reliable operation of QKD networks with a certain number of relays.

Table 3: Overall algorithm performance comparison.

Overall Algorithm Performance	Index	Algorithms	
		TFHE	BGV/BFV
Multiplicative depth	2		✓
	3		✓
	4		✓
	5		✓
	6		✓
	7	✓	
	8	✓	
	9	✓	
	10	✓	
	...	✓	

In the second experiment, due to limitations in the experimental environment, we simulated only the scenario in which ciphertexts undergo homomorphic operations and are forwarded across two relays. The experimental results are shown in Fig. 9, where the BFV and BGV algorithms have a multiplicative depth of 12 (each relay performs two homomorphic XOR operations on each ciphertext, thus requiring 4 multiplicative depth operations). It can be observed that TFHE demonstrates a significant speed advantage in practical applications. Furthermore, due to its unique algorithmic design, Fast Bootstrapping technique, and optimized XOR logic gates, TFHE eliminate the limitation on multiplication depth, accommodating a certain number of HomXOR operations. It allows TFHE to accommodate more relay nodes joining the key distribution process. In contrast, QKD based on BFV and BGV algorithms requires pre-calculating the necessary multiplicative depth based on the number of relay nodes in the QKD network. It not only reduces computational efficiency but also limits the addition of relay nodes, thereby diminishing the node scalability of the QKD network.

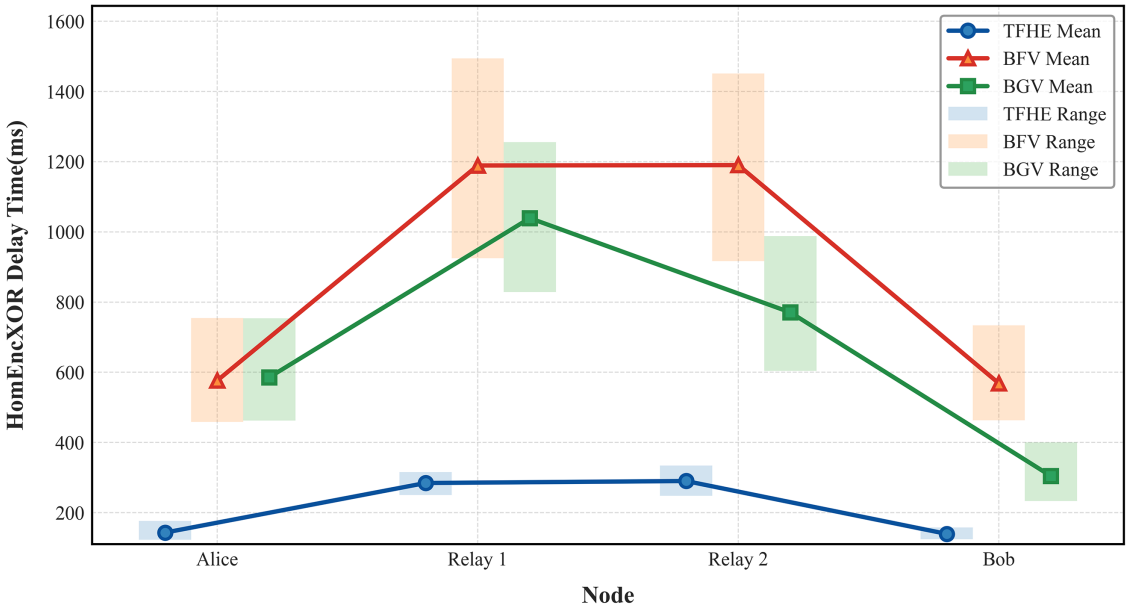

Figure 9: Comparison of computational time for TFHE, BFV, and BGV Algorithms in multi-relay scenarios.

6 Conclusion

QKD achieves unconditional security based on quantum mechanical principles. However, traditional QKD networks rely on trusted relays, which introduce trust issues and single-point vulnerabilities. Furthermore, although previously proposed QKD networks based on the BFV algorithm transmit keys in encrypted form during distribution, effectively ensuring key confidentiality even if relays are compromised, thereby addressing the relay-trust issue, the BFV algorithm itself has limitations. Its computational efficiency degrades as the number of nodes in the QKD network increases, and it cannot scale nodes wirelessly, necessitating optimization in speed and scalability. This study proposes a QKD scheme based on the public-key TFHE algorithm, termed TQKD. By leveraging TFHE's unique Fast Bootstrapping mechanism and an optimized XOR logic gate design, this scheme achieves secure transmission and efficient processing of QKD keys in untrusted relay scenarios.

Furthermore, we conducted two comparative experiments using OpenFHE for validation. This approach provides technical support for the large-scale deployment and application of QKD networks. Although the scheme demonstrated feasibility in experiments, limitations remain. Currently, no TRLWE ciphertext version exists for public-key TFHE algorithms, preventing bit-string encoding into a single TRLWE ciphertext. Furthermore, single-operation processing is restricted to one bit, precluding batch HomXOR operations on ciphertexts—a key focus for future research.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by the National Key Science and Technology Project: "Quantum Science and Technology-National Science and TechnologyMajor Project (QNMP)" (Grant No. 2021ZD0301301) and "the Fundamental Research Funds for the Central Universities" (Grant No. 3282025009);.

Author Contributions: Conceptualization, Tianhua Lin and Jianguo Xie; methodology, Tianhua Lin and Sijiang Xie; software, Yalong Yan; validation, Tianhua Lin, Sijiang Xie, and Ang Liu; formal analysis, Jianguo Xie; investigation, Tianhua Lin; resources, Sijiang Xie; data curation, Tianhua Lin; writing—original draft preparation, Tianhua Lin;

writing—review and editing, Yalong Yan; visualization, Tianhua Lin; supervision, Sijiang Xie; project administration, Tianhua Lin; funding acquisition, Sijiang Xie and Yalong Yan. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Data supporting the findings of Section 5 are available from the corresponding author, Sijiang Xie, upon reasonable request.

Ethics Approval: This study did not involve any human or animal subjects, and therefore, ethical approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bennett CH, Brassard G. Quantum cryptography: public key distribution and coin tossing. Theor Comput Sci. 2014;560:7–11. doi:10.1016/j.tcs.2014.05.025.
2. Ugwuishiwu C, Orji U, Ugwu C, Asogwa C. An overview of quantum cryptography and Shor's algorithm. Int J Adv Trends Comput Sci Eng. 2021;9(5):7487– 7495. doi:10.30534/ijatcse/2020/82952020.
3. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. 1999 Jun;41(2):303–32. doi:10.1137/s0036144598347011.
4. Yang Z, Zolanvari M, Jain R. A survey of important issues in quantum computing and communications. IEEE Commun Surv Tuts. 2023;25(2):1059–94. doi:10.1109/comst.2023.3254481.
5. Szabłowski PJ. Understanding mathematics of Grover's algorithm. Quantum Inf Process. 2021;20(5):191. doi:10.1007/s11128-021-03125-w.
6. Takeoka M, Guha S, Wilde MM. Fundamental rate-loss tradeoff for optical quantum key distribution. Nat Commun. 2014 Oct;5(1):5235. doi:10.1038/ncomms6235.
7. Korzh B, Lim CCW, Houlmann R, Gisin N, Li MJ, Nolan D, et al. Provably secure and practical quantum key distribution over 307 km of optical fibre. Nat Photonics. 2015 Feb;9(3):163–8. doi:10.1038/nphoton.2014.327.
8. Zhang Q, Xu F, Li L, Liu NL, Pan JW. Quantum information research in China. Quantum Sci Technol. 2019 Nov;4(4):040503. doi:10.1088/2058-9565/ab4bea.
9. Chen YA, Zhang Q, Chen TY, Cai WQ, Liao SK, Zhang J, et al. An integrated space-to-ground quantum communication network over 4600 kilometres. Nature. 2021;589(7841):214–9. doi:10.1038/s41586-020-03093-8.
10. Wang S, Yin Z-Q, He D-Y, Chen W, Wang R-Q, Ye P, et al. Twin-field quantum key distribution over 830-km fibre. Nat Photon. 2022;16(2):154–61. doi:10.1038/s41566-021-00928-2.
11. Xu HQ, Li GC, Hong XS, Chen L, Zhang SQ, Liu Y, et al. Informationally complete distributed metrology without a shared reference frame. Nat Commun. 2026;17(1):1025. doi:10.1038/s41467-025-67771-9.
12. Li Y, Cai WQ, Ren JG, Wang CZ, Yang M, Zhang L, et al. Microsatellite-based real-time quantum key distribution. Nature. 2025 Mar;640(8057):47–54. doi:10.1038/s41586-025-08739-z.
13. Briegel HJ, Dür W, Cirac JI, Zoller P. Quantum repeaters: the role of imperfect local operations in quantum communication. Phys Rev Lett. 1998 Dec;81:5932–5. doi:10.1103/physrevlett.81.5932.
14. Huttner B, Alléaume R, Diamanti E, Fröwis F, Grangier P, Hübel H, et al. Long-range QKD without trusted nodes is not possible with current technology. npj Quantum Inf. 2022;8(1):108. doi:10.1038/s41534-022-00613-4.
15. Sangouard N, Simon C, de Riedmatten H, Gisin N. Quantum repeaters based on atomic ensembles and linear optics. Rev Mod Phys. 2011 Mar;83(1):33–80. doi:10.1103/revmodphys.83.33.
16. Singh A, Dev K, Siljak H, Joshi HD, Magarini M. Quantum internet—applications, functionalities, enabling technologies, challenges, and research directions. IEEE Commun Surv Tutor. 2021;23(4):2218–47. doi:10.1109/comst.2021.3109944.
17. Rivest RL, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. Found Secur Comput. 1978;4(11):169–80.
18. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM. 1978 Feb;21(2):120–6. doi:10.1145/359340.359342.

19. Goldwasser S, Micali S. Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the fourteenth annual ACM symposium on Theory of computing; 1982 May 5–7; San Francisco, CA, USA. New York, NY, USA: ACM; 1982. p. 365–77.
20. Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory. 1985;31(4):469–72.
21. Benaloh J. Dense probabilistic encryption. In: Proceedings of the Workshop on Selected Areas of Cryptography; 2026 Aug 24–28; Ottawa, ON, Canada. p. 120–128.
22. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Stern J, editor. Advances in cryptology—EUROCRYPT '99. Berlin/Heidelberg, Germany: Springer; 1999. p. 223–38. doi: 10.1007/3-540-48910-x_16.
23. Boneh D, Goh EJ, Nissim K. Evaluating 2-DNF formulas on ciphertexts. In: Kilian J, editor. Theory of cryptography. Berlin/Heidelberg, Germany: Springer; 2005. p. 325–41. doi: 10.1007/978-3-540-30576-7_18.
24. Gentry C. A fully homomorphic encryption scheme. Stanford, CA, USA: Stanford University; 2009.
25. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference; 212 Jan 8–10; Cambridge, MA, USA. New York, NY, USA: ACM; 2012. p. 309–25.
26. Brakerski Z. Fully homomorphic encryption without modulus switching from classical gapSVP. In: Safavi-Naini R, Canetti R, editor. Advances in Cryptology—CRYPTO 2012. Berlin/Heidelberg: Springer; 2012. p. 868–86. doi: 10.1007/978-3-642-32009-5_50.
27. Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive. 2012.
28. Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. Cryptology ePrint Archive. 2013.
29. Chillotti I, Gama N, Georgieva M, Izabachène M. Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. Cryptology ePrint Archive. 2016.
30. Chillotti I, Gama N, Georgieva M, Izabachène M. TFHE: fast fully homomorphic encryption over the torus. Cryptology ePrint Archive. 2018.
31. Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. Cham, Switzerland: Springer International Publishing; 2016.
32. Regev O. Lattice-based cryptography. In: Proceedings of the 26th Annual International Conference on Advances in Cryptology; 2006 Aug 20–24; Santa Barbara, CA, USA. Berlin, Heidelberg/Germany: Springer; 2006. p. 131–41.
33. Regev O. On lattices, learning with errors, random linear codes, and cryptography. J ACM. 2009 Sep;56(6):34. doi:10.1145/1568318.1568324.
34. Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. SIAM J Comput. 2014;43(2):831–71. doi:10.1137/120868669.
35. Cooley JW, Tukey JW. An algorithm for the machine calculation of complex Fourier series. Math Comput. 1965;19:297–301. doi:10.1090/s0025-5718-1965-0178586-1.
36. Gentry C. Computing arbitrary functions of encrypted data. Commun ACM. 2010 Mar;53(3):97–105. doi:10.1145/1666420.1666444.
37. Brazaola-Vicario A, Lage O, Bernabé-Rodríguez J, Jacob E, Astorga J. Privacy enhanced QKD networks: zero trust relay architecture based on homomorphic encryption. Comput Netw. 2026;280:112172. doi:10.1016/j.comnet.2026.112172.
38. Rose SW, Borchert O, Mitchell S, Connelly S. Zero trust architecture. Gaithersburg, Maryland: National Institute of Standards and Technology (NIST); 2020. doi:10.6028/NIST.SP.800-207.
39. Joye M. TFHE public-key encryption revisited. Berlin/Heidelberg, Germany: Springer-Verlag; 2024. p. 277–91.
40. Tysowski PK, Ling X, Lütkenhaus N, Mosca M. The engineering of a scalable multi-site communications system utilizing quantum key distribution (QKD). Quantum Sci Technol. 2018 Jan;3(2):024001.
41. Mehic M, Niemiec M, Rass S, Ma J, Peev M, Aguado A, et al. Quantum key distribution: a networking perspective. ACM Comput Surv. 2020;53:41. doi:10.1145/3402192.

42. Bennett CH. Quantum cryptography using any two nonorthogonal states. Phys Rev Lett. 1992 May;68(21):3121–4. doi:10.1103/physrevlett.68.3121.
43. Ekert AK. Quantum cryptography based on Bell's theorem. Phys Rev Lett. 1991 Aug;67(6):661–3. doi:10.1103/physrevlett.67.661.
44. Ma X, Yuan X, Cao Z, Qi B, Zhang Z. Quantum random number generation. npj Quantum Inf. 2016;2(1):1–9. doi:10.1038/npjqi.2016.21.
45. Shannon CE. Communication theory of secrecy systems. Bell Syst Tech J. 1949;28(4):656–715. doi:10.1002/j.1538-7305.1949.tb00928.x.
46. Chen H, Laine K, Player R. Simple encrypted arithmetic library – SEAL v2.1. In: Brenner M, Rohloff K, Bonneau J, Miller A, Ryan PYA, Teague V et al., editors. Financial cryptography and data security. Cham, Switzerland: Springer International Publishing; 2017. p. 3–18.
47. Al Badawi A, Bates J, Bergamaschi F, Cousins DB, Erabelli S, Genise N, et al. OpenFHE: open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. WAHC'22; 2022 Nov 7; Los Angeles, CA, USA. New York, NY, USA: ACM; 2022. p. 53–63.
48. Halevi S, Shoup V. Design and implementation of a homomorphic-encryption library. IBM Research. 2013;6(12–15):8–36.
49. Zama. Concrete ML: privacy-preserving machine learning library. Version 1.5.0 [Internet]. 2024 [cited 2026 Jan 14]. Available from: https://github.com/zama-ai/concrete-ml.
50. Polyakov Y, Rohloff K, Ryan GW. PALISADE lattice cryptography library [Internet]. 2018 [cited 2026 Jan 14]. Available from: https://github.com/fuz-woo/PALISADE.
51. Polyakov Y, Quah I, Oliveira R, Triplett M. The OpenFHE Development Team. OpenFHE-python documentation. Version 0.8.0 [Internet]. 2024 [cited 2024 Aug 6]. Available from: https://openfhe-python.readthedocs.io/.