cryptography

MDPI

Article

Quantum Identity-Based Encryption from the Learning with Errors Problem

Wenhua Gao [1,2,*], Li Yang [1,2,*], Daode Zhang [1,2] and Xia Liu [1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China; zhangdaode0119@gmail.com (D.Z.); liuxia@iie.ac.cn (X.L.)
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: gaowenhua@iie.ac.cn (W.G.); yangli@iie.ac.cn (L.Y.)

Abstract: To prevent eavesdropping and tampering, network security protocols take advantage of asymmetric ciphers to establish session-specific shared keys with which further communication is encrypted using symmetric ciphers. Commonly used asymmetric algorithms include public key encryption, key exchange, and identity-based encryption (IBE). However, network security protocols based on classic identity-based encryption schemes do not have perfect forward secrecy. To solve this problem, we construct the first quantum IBE (QIBE) scheme based on the learning with errors (LWE) problem, which is also the first cryptographic scheme that applies the LWE problem to quantum encryption. We prove that our scheme is fully secure under the random oracle model and highlight the following advantages: (1) Network security protocols with our QIBE scheme provide perfect forward secrecy. The ciphertext is transmitted in the form of a quantum state unknown to the adversary and cannot be copied and stored. Thus, in network security protocols based on QIBE construction, the adversary does not have any previous quantum ciphertext to decrypt for obtaining the previous session key, even if the private identity key is threatened. (2) Classic key generation centre (KGC) systems can still be used in the QIBE scheme to generate and distribute private identity keys, reducing the cost when implementing this scheme. The classic KGC systems can be used because the master public and secret keys of our scheme are both in the form of classic bits. Finally, we present quantum circuits to implement this QIBE scheme and analyse its required quantum resources for given numbers of qubits, Hadamard gates, phase gates, T gates, and CNOT (controlled-NOT) gates. One of our main findings is that the quantum resources required by our scheme increase linearly with the number of plaintext bits to be encrypted.

Keywords: quantum IBE (identity-based encryption); LWE (learning with errors); forward secrecy; quantum circuits

Citation: Gao, W.; Yang, L.; Zhang, D.; Liu, X. Quantum Identity-Based Encryption from the Learning with Errors Problem. *Cryptography* **2022**, *6*, 9. https://doi.org/10.3390/cryptography6010009

Academic Editor: Josef Pieprzyk

Received: 18 January 2022
Accepted: 10 February 2022
Published: 16 February 2022

1. Introduction

Identity-based encryption (IBE) is an advanced form of public key encryption, and the notion of IBE was first proposed by Shamir in 1984 [1]. In the IBE scheme, the public key is directly calculated from the receiver's identity id_R, which may be a phone number, email address, or network address, and the corresponding private identity key sk_{id_R} is generated by a trusted key generation centre (KGC), which owns the master public key mpk and the master secret key msk. When the sender wants to send a message m to the receiver, the sender encrypts the message to obtain a ciphertext $ct = \mathsf{Encrypt}(\mathsf{mpk}, id_R, m; r)$, where r is a random number. After receiving the ciphertext ct, the receiver can decrypt it and obtain the message $m = \mathsf{Decrypt}(sk_R, ct)$. Compared with a public key infrastructure (PKI)-based cryptographic system, an identity-based cryptographic system avoids the high cost of storing and managing public key certificates, simplifies the process of public key management, and reduces the pressure on the system. Therefore, identity-based cryptosystems have been widely developed and applied.

The first practical IBE scheme was proposed by Boneh et al. in 2001 [2], and it was followed by numerous other classic IBE schemes. These classic IBE schemes can be divided into three categories: based on elliptic curve bilinear pairings [2–4], based on quadratic residue [5–8], and based on lattices [9–13]. With the development of quantum computers and quantum algorithms, especially the Shor algorithm [14], the security of IBE schemes based on elliptic curve bilinear pairings and quadratic residue has been seriously threatened. As there is currently no quantum algorithm that can solve lattice-based problems, the construction of lattice-based IBE schemes has become a major research area for cryptographers.

One of the main applications of IBE is the construction of network security protocols, such as the Chinese SSL VPN technology specification [15]. In the IBE-based security protocol, the receiver sends its identity id_R and mpk to the sender, and the sender chooses a *sessionKey* and sends its ciphertext to the receiver. The receiver decrypts the ciphertext to obtain the *sessionKey*. Subsequently, both the sender and the receiver can own this secret session key *sessionKey* with which further communication is encrypted using a symmetric cipher. The entire procedure is briefly described in Figure 1. A security protocol is said to provide perfect forward secrecy [16] if the compromise of long-term keys does not compromise past session keys that were established before the compromise of the long-term key. In security protocols based on classic IBE, all session keys and their ciphertexts are in the form of classic bits. A patient attacker can capture the conversations and store the ciphertexts of the session keys, whose confidentiality is protected by a private identity key (which is the long-term key of the network security protocol based on IBEs) and wait until the long-term key is threatened. Once the patient attacker obtains the long-term key, the attacker can decrypt the ciphertexts of all previous session keys. Ultimately, all encrypted communications and sessions recorded in the past can be retrieved. Therefore, the security protocols based on the classic IBE scheme do not have perfect forward secrecy. This naturally leads to the following question:

> *Can we construct a fully secure IBE scheme with which the network security protocol can provide perfect forward secrecy?*

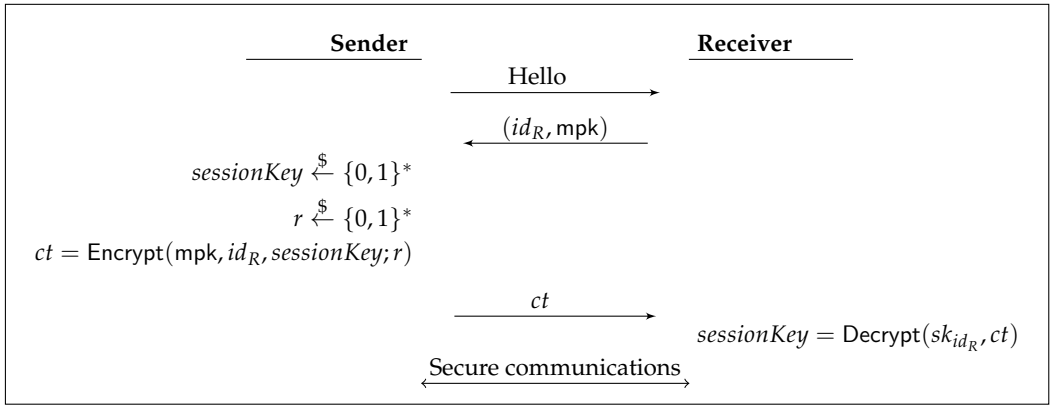

Figure 1. Security protocols based on IBE. The "Hello" message means that the sender wants to communicate with the receiver.

1.1. Our Contributions

To solve this problem, considering that an adversary cannot replicate an unknown quantum state [17], we propose the notion of quantum identity-based encryption (QIBE) and construct the first QIBE scheme based on the learning with errors (LWE) problem. Then, we prove that our scheme is fully secure under the random oracle model. Our scheme possesses the following advantages:

- Network security protocols with our QIBE scheme provide perfect forward secrecy. The ciphertext is transmitted in the form of a quantum state that is unknown to the

adversary, who cannot duplicate the ciphertext. Thus, in security protocols based on QIBE construction, even if the private identity key is threatened, the adversary does not possess the previous ciphertexts of session keys to decrypt, and therefore cannot threaten the security of the previous session keys. Therefore, security protocols based on QIBE construction have perfect forward secrecy.

- The classic KGC system can still be used for QIBE schemes to generate and distribute private identity keys, reducing the cost of implementing this scheme. The classic KGC system can be used because the Setup, KeyGen algorithms of our scheme are classic, and the master public and secret keys of our scheme are both in the form of classic bits.

Finally, we present quantum circuits to implement this QIBE scheme and establish its required quantum resource estimates for given numbers of qubits, Hadamard gates, phase gates, T gates, and CNOT gates. One of our primary findings is that the quantum resources required by our scheme increase linearly with the number of plaintext bits to be encrypted.

1.2. Outline of the Paper

The remainder of this paper is organised as follows: In Section 2, we present the necessary background for the paper, including quantum computation, lattices, and classic IBE. In Section 3, we first present the definition of QIBE; then, we describe the concrete construction of the scheme and analyse its correctness, and finally, prove the security of the scheme and discuss its advantages. In Section 4, we discuss the specific quantum circuit implementation for the QIBE scheme and estimate the quantum resources needed. In Section 5, we summarise our work, discuss the drawbacks of our scheme, and suggest directions for future work.

2. Preliminaries

2.1. Quantum Computation

In this section, we briefly discuss the basic concepts used in this work. Please refer to [18] for a more detailed introduction to quantum computing. "$\overline{(\cdot)}$" is defined as the bit-flip of "$(\cdot)$", such as $|\overline{0}\rangle = |1\rangle$, $|\overline{1}\rangle = |0\rangle$, and $|\overline{101}\rangle = |010\rangle$. We begin by introducing some quantum gates.

Fundamental gates. The fundamental gate set we use comprises the NOT gate, the CNOT gate, the CNOT variant gate, the Toffoli (two-controlled NOT) gate, and the Fredkin gate. These four gates are shown in Figure 2. One CNOT variant gate can be obtained by a CNOT gate and two NOT gates. The Fredkin gate is also known as a three-qubit controlled swap gate, that is, if the control qubit is in state$|1\rangle$, the two target qubits swap their states; otherwise, they remain in their initial states if the control qubit is in state $|0\rangle$. One Fredkin gate can be constructed from one Toffoli gate and two CNOT gates.

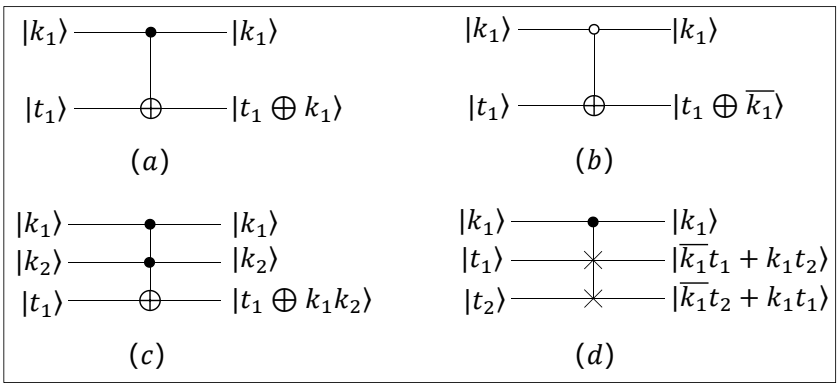

Figure 2. Fundamental gates. The CNOT gate is shown in (**a**); the CNOT variant gate is shown in (**b**); the Toffoli gate is shown in (**c**); and the Fredkin gate is shown in (**d**).

Complex Gates. We use four complex gates (Figure 3): the first gate is the ℓ-controlled-NOT gate, which can be decomposed into $(2\ell - 3)$ Toffoli gates, the second gate is the variant of the ℓ-controlled-NOT gate, which can also be decomposed into $(2\ell - 3)$ Toffoli gates, the third gate is the ℓ-combination-CNOT gate, which is a combination of ℓ CNOT gates, and the last gate is the ℓ-Fredkin gate, which can be constructed from ℓ Toffoli gates and 2ℓ CNOT gates.

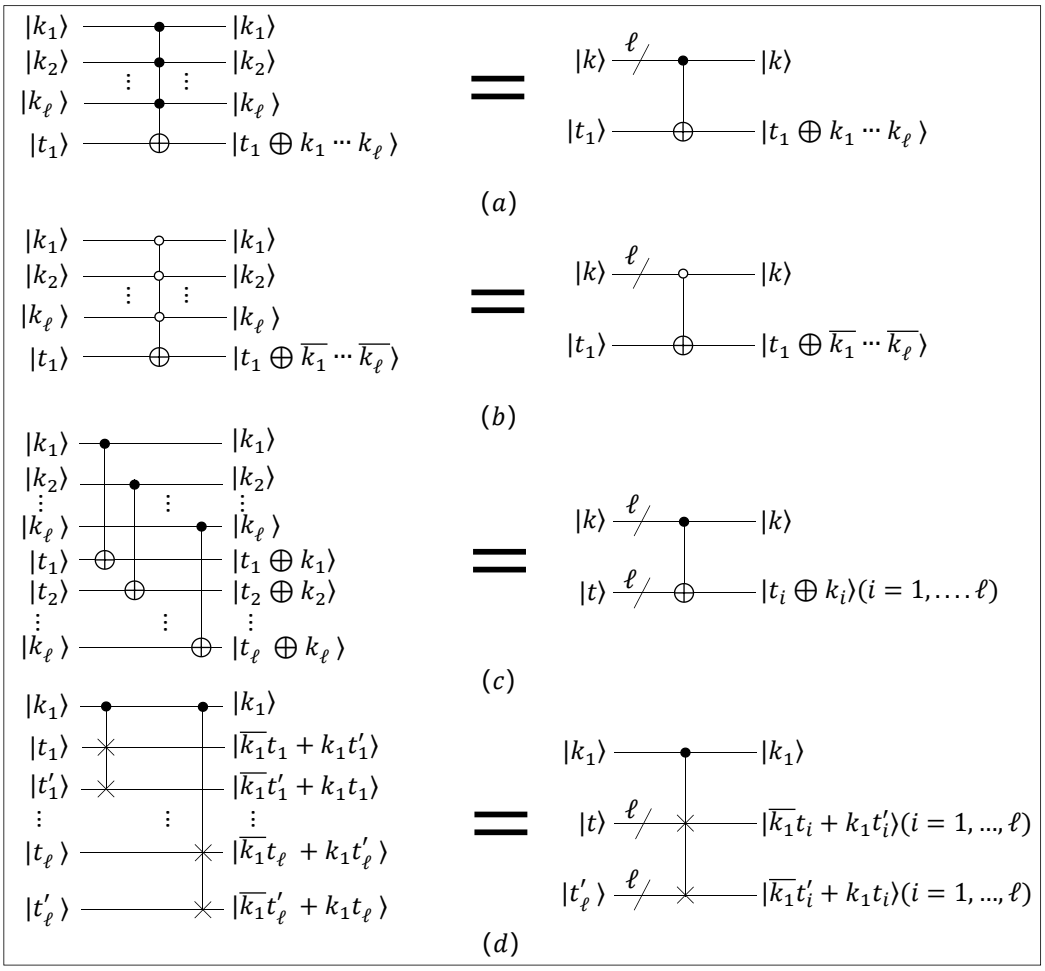

Figure 3. Complex gates. The ℓ-controlled-NOT gate is shown in (**a**); the variant of the ℓ-controlled-NOT gate is shown in (**b**); the ℓ-combination-CNOT gate is shown in (**c**); and the ℓ-Fredkin gate is shown in (**d**).

Quantum Basic Circuits. Figure 4f depicts the controlled copy circuit in which copying the integer d into the quantum register is controlled by $|\text{ctrl}\rangle$. If the controlled bit $|\text{ctrl}\rangle = |0\rangle$, the output of the controlled copy circuit is $(|\text{ctrl}\rangle, |0\rangle)$. Otherwise, the output is $(|\text{ctrl}\rangle, |d\rangle)$. The output of the controlled copy circuit is $(|\text{ctrl}\rangle, |\text{ctrl} \cdot d\rangle)$. We denote the controlled copy circuit by $\text{CCopy}(|\text{ctrl}\rangle, d)$ when this algorithm takes a qubit $|\text{ctrl}\rangle$ and an ℓ-bit number d as input. According to the conclusion in [19], copying an integer uses only NOT gates so that a controlled copy circuit only uses CNOT gates. Suppose that d is a uniform integer in $\{0, 1, \cdots, 2^\ell - 1\}$; then, the controlled copy circuit uses approximately $\ell/2$ CNOT gates, where $\ell = \lfloor \log d \rfloor + 1$.

Quantum Arithmetic Circuits. We introduce some quantum arithmetic circuits, including addition, subtraction, modular addition, modular subtraction, and comparison, and describe the corresponding quantum resources required, including the number of qubits and the number of CNOT gates and Toffoli gates. To simplify the description, we only show the simplified form of these quantum arithmetic circuits.

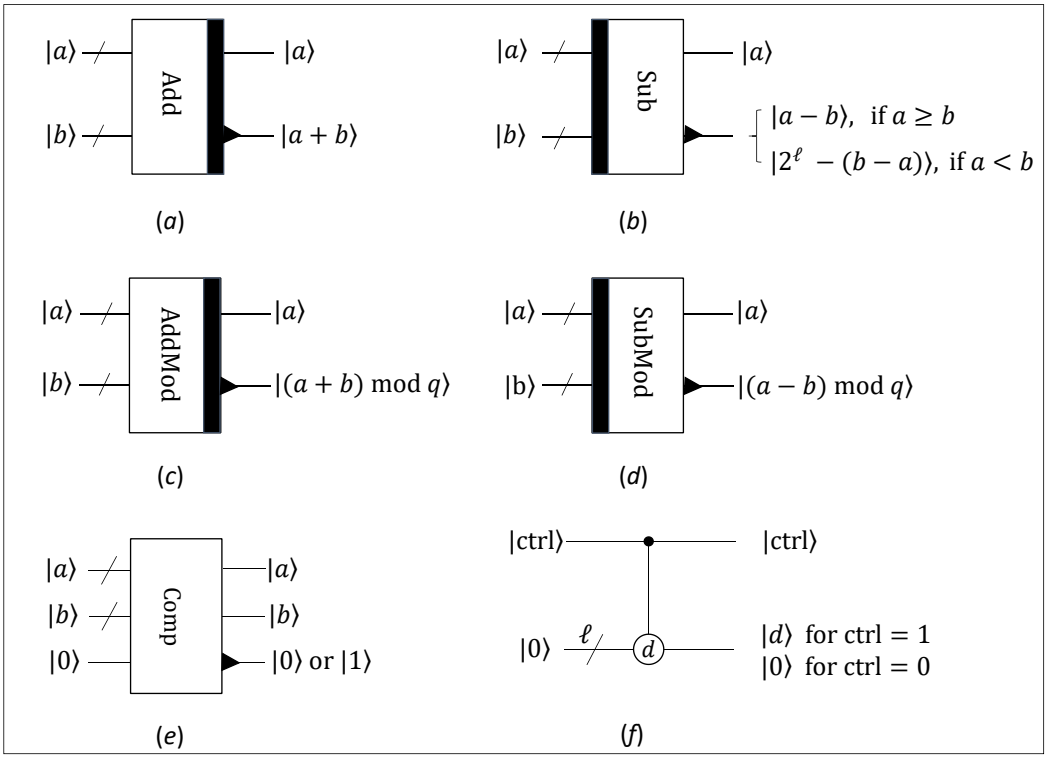

Figure 4. Some quantum basic and arithmetic circuits. In the Figure, (**a**) shows the addition circuit while (**b**) shows the subtraction circuit; (**c**) depicts the modular addition circuit while (**d**) depicts the modular subtraction circuit; (**e**) illustrates the comparison circuit; and the controlled copy circuit is shown in (**f**).

- **Addition and subtraction:** Cuccaro et al. [20] proposed a quantum addition circuit. The quantum addition achieves the addition of two registers, that is,

$$|a, b\rangle \rightarrow |a, a + b\rangle.$$

To prevent overflows caused by the carry, the second register (initially loaded in state $|b\rangle$) should be sufficiently large, i.e., if both a and b are encoded on ℓ qubits, the second register should be of size $\ell + 1$. In the addition circuit, the last carry is the most significant bit of the result and is written in the $\ell + 1$-th qubit of the second register. As a result of the reversibility of unitary operations, by reversing the addition circuit, i.e., applying each gate of the circuit in the reversed order, the subtraction circuit is obtained. The addition and subtraction circuits are shown in (a) and (b) of Figure 4, respectively. In this paper, a circuit with a bar on the left side represents the reversed sequence of elementary gates embedded in the same circuit with the bar on the right side.
In the subtraction circuit, with the input $(|a\rangle, |b\rangle)$, the output produces $(|a\rangle, |a - b\rangle)$ when $a \geq b$. When $a < b$, the output is $(|2^\ell - (b - a)\rangle)$, where the size of the second register is $\ell + 1$. i.e.,

$$\begin{cases} |a, b\rangle \rightarrow |a, a - b\rangle, & \text{for } a \geq b. \\ |a, b\rangle \rightarrow |a, 2^\ell - (b - a)\rangle, & \text{for } a < b. \end{cases}$$

When $a < b$, the significant qubit, that is, the $\ell + 1$-th qubit of the second register, which indicates whether an overflow occurred in the subtraction, always contains 1. We denote the addition circuit by $\mathsf{Add}(|a\rangle, |b\rangle)$ when this algorithm takes two ℓ-qubit states $|a\rangle$ and $|b\rangle$ as input. Similarly, we denote the subtraction circuit by $\mathsf{Sub}(|a\rangle, |b\rangle)$ when this algorithm takes two ℓ-qubit states $|a\rangle$ and $|b\rangle$ as input. To calculate the

addition or subtraction of two ℓ-bit inputs, a total of $2\ell + 2$ qubits, 2ℓ Toffoli gates, and $4\ell + 1$ CNOT gates are required.

- **Addition and subtraction module q:** Liu et al. [21] improved Roetteler's [22] quantum modular addition circuit and reduced the number of quantum gates required. This quantum circuit produces

$$|a, b\rangle \rightarrow |a, (a + b) \bmod q\rangle,$$

where $0 \leq a, b < q$. The simplified form of the addition module q circuit is shown in Figure 4c. The modular subtraction can be obtained by reversing the modular addition circuit, and its bar is on the left-hand side. We denote the modular addition circuit by $\mathsf{AddMod}(|a\rangle, |b\rangle)$ when this algorithm takes two $\lceil \log q \rceil$-qubit states $|a\rangle$ and $|b\rangle$ as input. Similarly, we denote the modular subtraction circuit by $\mathsf{SubMod}(|a\rangle, |b\rangle)$ when this algorithm takes two $\lceil \log q \rceil$-qubit states $|a\rangle$ and $|b\rangle$ as input. To calculate the addition or subtraction module q of two $\lceil \log q \rceil$-bit inputs, a total of $3 \cdot \lceil \log q \rceil + 3$ qubits, $8 \cdot \lceil \log q \rceil$ Toffoli gates, $13 \cdot \lceil \log q \rceil + 6$ CNOT gates are required.
- **Comparison:** Markov et al. [23] constructed a quantum comparison circuit by comparing $|a\rangle$ and $|b\rangle$ based on whether the highest bit of $|a - b\rangle$ is $|0\rangle$ or $|1\rangle$. This circuit is obtained by modifying the previous subtraction circuit so that it outputs only the highest bit of $|a - b\rangle$. The comparison circuit achieves the comparison of two registers, that is

$$\begin{cases} |a, b\rangle|0\rangle \rightarrow |a, b\rangle|0\rangle, & \text{for } a \geq b. \\ |a, b\rangle|0\rangle \rightarrow |a, b\rangle|1\rangle, & \text{for } a < b. \end{cases}$$

The simplified form of the quantum comparison circuit is shown in Figure 4e. We denote the comparison circuit by $\mathsf{Comp}(|a\rangle, |b\rangle)$ when this algorithm takes two ℓ-qubit states $|a\rangle$ and $|b\rangle$ as input. To compare two ℓ-bit inputs $|a\rangle$ and $|b\rangle$, a total of $2\ell + 2$ qubits, 2ℓ Toffoli gates, and $4\ell + 1$ CNOT gates are required.

2.2. Lattices

Let X and Y be two random variables over some finite set S_X, S_Y, respectively. The statistical distance $\Delta(X, Y)$ between X and Y is defined as

$$\Delta(X, Y) = \frac{1}{2} \sum_{s \in S_X \cup S_Y} |\Pr[X = s] - \Pr[Y = s]|.$$

For integer $q \geq 2$, $\mathbb{Z}_q$ denotes the quotient ring of integer modulo q. We use bold capital letters to denote matrices, such as $\mathbf{A}$, $\mathbf{B}$, and bold lowercase letters to denote column vectors, such as $\mathbf{x}$, $\mathbf{y}$. The notation $\mathbf{A}^\top$ denotes the transpose of the matrix $\mathbf{A}$.

Let $\mathbf{S}$ be a set of vectors, $\mathbf{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_n\}$ in $\mathbb{R}^{\widetilde{m}}$. We use $\widetilde{\mathbf{S}} = \{\widetilde{\mathbf{s}}_1, \cdots, \widetilde{\mathbf{s}}_n\}$ to denote the Gram–Schmidt orthogonalisation of the vectors $\mathbf{s}_1, \cdots, \mathbf{s}_n$ in that order and $\|\mathbf{S}\|$ to denote the length of the longest vector in $\mathbf{S}$. For positive integers $q, n, \widetilde{m}$ with q prime, and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \widetilde{m}}$, the $\widetilde{m}$-dimensional integer lattices are defined as follows: $\Lambda_q(\mathbf{A}) = \{\mathbf{y} : \mathbf{y} = \mathbf{A}^\top \mathbf{s} \text{ for some } \mathbf{s} \in \mathbb{Z}^n\}$ and $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{y} : \mathbf{A}\mathbf{y} = \mathbf{0} \bmod q\}$. Moreover, for $\mathbf{u} \in \mathbb{Z}_q^n$, the set of syndromes is defined as $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{y} : \mathbf{u} = \mathbf{A}\mathbf{y} \bmod q\}$.

For $\mathbf{x} \in \Lambda$, define the Gaussian function $\rho_{s,\mathbf{c}}(\mathbf{x})$ over $\Lambda \subseteq \mathbb{Z}^{\widetilde{m}}$ centred at $\mathbf{c} \in \mathbb{R}^{\widetilde{m}}$ with parameter $s > 0$ as $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi||\mathbf{x} - \mathbf{c}||/s^2)$. Let $\rho_{s,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_{s,\mathbf{c}}(\mathbf{x})$, and define the discrete Gaussian distribution over Λ as $\mathcal{D}_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}$, where $\mathbf{x} \in \Lambda$. For simplicity, $\rho_{s,\mathbf{0}}$ and $\mathcal{D}_{\Lambda,s,\mathbf{0}}$ are abbreviated as ρ_s and $\mathcal{D}_{\Lambda,s}$, respectively.

Lemma 1 (Adopted from [9,24,25])**.** *Let $q, n, \widetilde{m}$ be positive integers with $q \geq 2$ and q prime. There exist the following PPT (probabilistic polynomial-time) algorithms:*

- $\mathsf{TrapGen}(\widetilde{m}, n, q)$: *a randomised algorithm that, when* $\widetilde{m} \geq 6n\lceil \log q \rceil$, *outputs a pair* $(\mathbf{A}, \mathbf{T_A}) \in \mathbb{Z}_q^{n \times \widetilde{m}} \times \mathbb{Z}^{\widetilde{m} \times \widetilde{m}}$ *such that* $\mathbf{A}$ *is* $2^{-\Omega(n)}$*−close to uniform in* $\mathbb{Z}_q^{n \times \widetilde{m}}$ *and* $\mathbf{T_A}$ *is a basis of* $\Lambda_q^{\perp}(\mathbf{A})$, *satisfying* $\|\widetilde{\mathbf{T_A}}\| \leq \mathcal{O}(\sqrt{n \log q})$ *with overwhelming probability.*
- $\mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$: *a randomised algorithm that, given a full rank matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times \widetilde{m}}$,*a basis* $\mathbf{T_A}$ *of* $\Lambda_q^{\perp}(\mathbf{A})$, *a vector* $\mathbf{u} \in \mathbb{Z}_q^n$ *and* $\sigma \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log \widetilde{m}})$, *outputs a vector* $\mathbf{r} \in \mathbb{Z}_q^{\widetilde{m}}$ *sampled from a distribution* $2^{-\Omega(n)}$*−close to* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\sigma}$.

Discrete Gaussian Lemmas. The following lemmas are used to manipulate and obtain meaningful bounds on discrete Gaussian vectors.

Lemma 2 (Adopted from [9], Lemma 5.2)**.** *Let* n, $\widetilde{m}$, q *be positive integers such that* $\widetilde{m} \geq 2n \log q$ *and* q *is prime. Let* σ *be any positive real number such that* $\sigma \geq \sqrt{n + \log \widetilde{m}}$. *Then, for all but a* $2^{-\Omega(n)}$ *fraction of* $\mathbf{A} \in \mathbb{Z}_q^{n \times \widetilde{m}}$, *the distribution of* $\mathbf{u} = \mathbf{Ar}$ *mod* q *for* $\mathbf{r} \leftarrow D_{\mathbb{Z}^{\widetilde{m}},\sigma}$ *is* $2^{-\Omega(n)}$*-close to uniform distribution over* $\mathbb{Z}_q^n$. *Furthermore, for a fixed* $\mathbf{u} \in \mathbb{Z}_q^n$, *the conditional distribution of* $\mathbf{r} \leftarrow D_{\mathbb{Z}^{\widetilde{m}},\sigma}$, *given* $\mathbf{Ar} = \mathbf{u}$ *mod* q *is* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\sigma}$.

The Learing with Errors Problem. The security of our construction is based on the LWE problem. The LWE problem is a hard problem based on lattices defined by Regev [26]. It is stated as follows: given an input $(\mathbf{A}, \mathbf{d})$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times \widetilde{m}}$ for any $\widetilde{m} = poly(n)$, integer $q \geq 2$ is prime, and $\mathbf{d} \in \mathbb{Z}_q^{\widetilde{m}}$ is either of the form $\mathbf{d} = (\mathbf{A}^{\top}\mathbf{s} + \mathbf{e}) \bmod q$ for $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathcal{D}_{\mathbb{Z}^{\widetilde{m}},\sigma}$ or is uniformly random (and independent of $\mathbf{A}$), distinguish which is the case, with non-negligible advantage. Regev proved that the LWE problem is as hard as approximating standard lattice problems in the worst case using a quantum algorithm.

2.3. Classic Identity-Based Encryption

A classic identity-based encryption scheme consists of the following four algorithms:

- $\mathsf{KeyGen}(1^{\lambda}) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm takes a security parameter 1^{λ} as input. It outputs a master public key mpk and a master secret key msk.
- $\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, id) \rightarrow sk_{id}$. The key extraction algorithm takes a master public key mpk, a master secret key msk, and an identity id as input. It outputs a private identity key sk_{id}. We assume that id is implicitly included in sk_{id}.
- $\mathsf{Encrypt}(\mathsf{mpk}, id, m; r) \rightarrow ct$. The encryption algorithm takes a master public key mpk, an identity id, and the message m as input. It outputs a ciphertext ct.
- $\mathsf{Decrypt}(sk_{id}, ct) \rightarrow m$. The decryption algorithm takes the master public key mpk, the private identity key sk_{id}, and the ciphertext ct as input. It outputs the message m.

  Correctness. For all $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\lambda})$, all identities $id \in ID$, all messages m, and all $c \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, id, m; r)$, we have

$$\Pr[\mathsf{Decrypt}(\mathsf{mpk}, sk_{id}, ct) = m] = 1 - \mathsf{negl}(\lambda).$$

  Security. The security game is defined by the following experiment, which is played by a challenger and an adversary $\mathcal{A}$:

- The challenger runs KeyGen to generate $(\mathsf{mpk}, \mathsf{msk})$. It gives mpk to the adversary $\mathcal{A}$.
- The adversary $\mathcal{A}$ adaptively requests keys for any identity id_i of its choice. The challenger responds with the corresponding secret key sk_{id_i}, which it generates by running $\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, id_i)$.
- The adversary $\mathcal{A}$ submits two messages of equal length, m_0 and m_1, and a challenge identity id^* with the restriction that id^* is not equal to any identity requested in the previous phase. The challenger picks $\beta \xleftarrow{\$} \{0, 1\}$, encrypts m_{β} under id^* by running the encryption algorithm, and sends the ciphertext to the adversary $\mathcal{A}$.
- $\mathcal{A}$ continues to issue key queries for any identity id_i as in step (2) with the restriction that $id_i \neq id^*$.

- The adversary $\mathcal{A}$ outputs a guess β' for β.

  The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{IBE}}(\lambda)$ of an adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{IBE}}(\lambda) = \left|\Pr[\beta' = \beta] - 1/2\right|.$$

Definition 1. *An IBE scheme is fully secure if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{IBE}}(\lambda)$ *is a negligible function in* λ.

3. QIBE and Its Construction

3.1. Definition of QIBE

We begin by defining a primitive for identity-based encryption schemes in which some elements may belong to a quantum space.

Definition 2. *We say that an identity-based encryption scheme* IBE *is a quantum identity-based encryption (QIBE) scheme if there exists at least one quantum algorithm in its algorithms that include* KeyGen, Extract, Encrypt, *and* Decrypt.

Note that the classification of QIBE can be analogous to that of quantum public key encryption [27] and a quantum symmetric-encryption scheme [28]. Since each algorithm in the QIBE schemes could be either classic or quantum, there exist only fifteen types of QIBE schemes in total.

3.2. Our Construction

Then, we design a QIBE scheme by utilising the proposed classic IBE scheme [9] and prove its security based on the LWE problem.

In our QIBE scheme, KeyGen and Extract are classic algorithms while Encrypt and Decrypt are quantum algorithms. To make it easier to distinguish between classic IBE and QIBE, we denote our scheme as QIBE, and our QIBE scheme consists of one tuple $(\mathsf{QKeyGen}, \mathsf{QExtract}, \mathsf{QEncrypt}, \mathsf{QDecrypt})$. Let integer parameters $n = \mathcal{O}(\lambda), \widetilde{m} = \mathcal{O}(n), \sigma = \mathcal{O}(n^{0.5}), q = \mathcal{O}(\widetilde{m}^{3.5})$ as specified in [9], where λ is a security parameter.

- $\mathsf{QKeyGen}$: (1) It runs $\mathsf{TrapGen}(\widetilde{m}, n, q)$ to obtain a uniformly random $n \times \widetilde{m}-$matric $\mathbf{A} \in \mathbb{Z}_q^{n\times\widetilde{m}}$ and $\mathbf{T_A} \in \mathbb{Z}_q^{\widetilde{m}\times\widetilde{m}}$ which is a good basis for $\Lambda_q^{\perp}(\mathbf{A})$. (2) Then, it selects a hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{Z}_q^n$, which maps an identity to a vector. (3) Finally, it outputs $\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H})$ and $\mathsf{msk} = (\mathbf{T_A})$. (4) In summary, $\mathsf{QKeyGen}(\lambda, q, \widetilde{m}, n) \to (\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H}), \mathsf{msk} = (\mathbf{T_A}))$.
- $\mathsf{QExtract}$: (1) On input mpk, msk and an identity $id \in \{0,1\}^n$, it computes $\mathbf{u} = \mathsf{H}(id)$ and generates $sk_{id} = \mathbf{r}$ such that $\mathbf{r} = \mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$. It is clear that $\mathbf{u} = \mathbf{Ar} \bmod q$. (2) In summary, $\mathsf{QExtract}(\mathsf{msk}, \mathsf{mpk}, id) \to sk_{id} = \mathbf{r}$.
- $\mathsf{QEncrypt}$: (1) On input of an identity id, mpk, and a bit quantum superposition state $|\phi\rangle = \sum_{m\in\{0,1\}} \alpha_m|m\rangle$, it first computes $\mathbf{u} = \mathsf{H}(id)$ and chooses a uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_0 \in \mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathbf{e} \in \mathcal{D}_{\mathbb{Z}^{\widetilde{m}},\sigma}$. (2) Then, it sets $x = (\mathbf{u}^\top\mathbf{s} + e_0) \bmod q$ and $\mathbf{c}_1 = (\mathbf{A}^\top\mathbf{s} + \mathbf{e}) \bmod q$. More procedures are performed as follows:
  - ⋆ Step 1: Taking $|\phi\rangle = \sum_{m\in\{0,1\}} \alpha_m|m\rangle$ and $\lfloor\frac{q}{2}\rfloor$ as input, it runs $\mathsf{CCopy}(|\phi\rangle, \lfloor\frac{q}{2}\rfloor)$ and gets
$$\sum_{m\in\{0,1\}} \alpha_m|m\rangle|m\cdot\lfloor\frac{q}{2}\rfloor\rangle.$$
  - ⋆ Step 2: Taking the second register of the above result and x as input, it runs AddMod and obtains
$$\sum_{m\in\{0,1\}} \alpha_m|m\rangle|(m\cdot\lfloor\frac{q}{2}\rfloor + x) \bmod q\rangle.$$

⋆ Step 3: Finally, taking the two registers of the above result as input, it runs the unentanglement quantum circuit (which is described in Section 4, and we denote it by Unentangle) to obtain

$$|\psi\rangle = \sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle.$$

(3) In summary, $\mathsf{QEncrypt}(id, |\phi\rangle) \rightarrow (ct = (|\psi\rangle, \mathbf{c}_1))$.

- QDecrypt: To decrypt a ciphertext $(|\psi\rangle, \mathbf{c}_1)$ using an identity secret key $sk_{id} = \mathbf{r}$, it computes $y = \mathbf{r}^\top \mathbf{c}_1 \bmod q \in \mathbb{Z}_q$. Then, more processes are performed as follows:

⋆ Step 1: Taking $|\psi\rangle = \sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $|y\rangle$ as input, it runs $\mathsf{SubMod}(|\psi\rangle, |y\rangle)$ to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x - y) \bmod q\rangle.$$

⋆ Step 2: Taking the above result and $\lfloor \frac{q}{2} \rfloor$ as input, it runs SubAbs (SubAbs is a quantum circuit, which takes two ℓ-qubit quantum states $|a\rangle$ and $|b\rangle$ as input and outputs the absolute value of their subtraction, i.e., $|\mathsf{Abs}(a-b)\rangle$) to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |\mathsf{Abs}((m \cdot \lfloor \frac{q}{2} \rfloor + x - y) \bmod q - \lfloor \frac{q}{2} \rfloor)\rangle.$$

Please refer to Section 4.1 for more information about SubAbs.

⋆ Step 3: Taking the above result and $\lfloor \frac{q}{4} \rfloor$ as input, it runs Comp to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |\mathsf{Abs}((m \cdot \lfloor \frac{q}{2} \rfloor + x - y) \bmod q - \lfloor \frac{q}{2} \rfloor)\rangle |m\rangle.$$

Next, the algorithm QDecrypt will unentangle the first and second registers of this quantum state.

⋆ Step 4: Taking the first register of the above result and $\lfloor \frac{q}{2} \rfloor$ as input, it runs InvSubAbs (The InvSubAbs (which is the inverse of SubAbs) is a quantum circuit which takes in two ℓ-qubit quantum states $|\mathsf{Abs}(a-b)\rangle$ and $|b\rangle$ as input and outputs one ℓ-qubit quantum state, i.e., $|a\rangle$) to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x - y) \bmod q\rangle |m\rangle.$$

Please refer to Section 4.1 for more information about the inverse of SubAbs.

⋆ Step 5: Taking the first register of the above result and $|y\rangle$ as input, it runs AddMod to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle |m\rangle.$$

⋆ Step 6: Taking the second register of the above result and $\lfloor \frac{q}{2} \rfloor$ as the input, it runs CCopy to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle |m\rangle |m \cdot \lfloor \frac{q}{2} \rfloor\rangle.$$

⋆ Step 7: Taking the first register and the third register of the above result as input, it runs SubMod to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |x \bmod q\rangle |m\rangle |m \cdot \lfloor \frac{q}{2} \rfloor\rangle.$$

Finally, the algorithm QDecrypt unentangles the second and third registers of the above result.

⋆ Step 8: Taking the third register of the above result and $\lfloor \frac{q}{2} \rfloor$ as input, it performs the inverse of the controlled copy circuit to obtain

$$\sum_{m\in\{0,1\}} \alpha_m |x \bmod q\rangle |m\rangle |0\rangle.$$

The quantum state $\sum_{m\in\{0,1\}} \alpha_m |m\rangle$ is no longer entangled with other registers and the decryption procedure is completed. In summary,

$$\mathsf{QDecrypt}(id, \mathsf{mpk}, \mathbf{r}, (|\psi\rangle, \mathbf{c}_1)) \to |\phi\rangle = \sum_{m\in\{0,1\}} \alpha_m |m\rangle.$$

3.3. Correctness

Considering a ciphertext

$$\begin{aligned}&(|\psi\rangle, \mathbf{c}_1)\\ =&\left(\sum_{m\in\{0,1\}} \alpha_m |(m\cdot\lfloor\frac{q}{2}\rfloor + x) \bmod q\rangle, (\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \bmod q\right)\end{aligned}$$

of a qubit quantum superposition state $|\phi\rangle = \sum_{m\in\{0,1\}} \alpha_m |m\rangle$, it can be observed that $|\psi\rangle = \sum_{m\in\{0,1\}} \alpha_m |c_0\rangle$, where $c_0 = (x + m\cdot\lfloor\frac{q}{2}\rfloor) \bmod q = \mathbf{u}^\top \mathbf{s} + e_0 + m\cdot\lfloor\frac{q}{2}\rfloor \bmod q$. In step 2 of QDecrypt, it is clear that

$$\begin{aligned}&\mathsf{Abs}\Big(\Big(\Big(x + \lfloor\frac{q}{2}\rfloor\cdot m\Big) \bmod q - y\Big) \bmod q - \lfloor\frac{q}{2}\rfloor\Big)\\ =&\mathsf{Abs}\Big((c_0 - y) \bmod q - \lfloor\frac{q}{2}\rfloor\Big),\end{aligned}$$

which is equal to the absolute value of b in the Decrypt of Theorem A1, i.e., $\mathsf{Abs}(b)$. Finally, in step 3 of QDecrypt, we compare $\mathsf{Abs}(b)$ with $\lfloor\frac{q}{4}\rfloor$ and obtain m. According to Theorem A1, the decryption algorithm Decrypt with the identity secret key $sk_{id} = \mathbf{r}_{id}$ can decrypt the ciphertext $(c_0, \mathbf{c}_1)$ correctly with a probability of $1 - \mathsf{negl}(\lambda)$. Therefore, the decryption algorithm QDecrypt with the identity secret key $sk_{id} = \mathbf{r}_{id}$ can decrypt the ciphertext $ct = (|\psi\rangle, \mathbf{c}_1)$ correctly with a probability of $1 - \mathsf{negl}(\lambda)$.

3.4. Security Proof

Theorem 1. *The above QIBE scheme is fully secure in the random oracle model under the* LWE *assumption, namely, for any classical PPT adversary $\mathcal{A}$ making at most Q_{H} random oracle queries to* H *and Q_{ID} identity secret key queries, there exists a classical PPT algorithm $\mathcal{B}$ such that*

$$\mathrm{Adv}^{\mathrm{QIBE}}_{\mathcal{A}}(\lambda) \le Q_{\mathsf{H}} \cdot \mathrm{Adv}^{\mathrm{LWE}}_{\mathcal{B}}(\lambda) + (Q_{\mathsf{H}} + Q_{\mathsf{ID}} + 1)\cdot 2^{-\Omega(n)}. \quad (1)$$

Proof of Theorem 1. Without loss of generality, we make some simplifying assumptions about $\mathcal{A}$. First, we assume that whenever $\mathcal{A}$ queries a secret key or asks for a challenge ciphertext, the corresponding *id* has already been queried to the random oracle H. Second, we assume that $\mathcal{A}$ makes the same query for the same random oracle at most once. Third, we assume that $\mathcal{A}$ does not repeat secret key queries for the same identity more than once. We show the security of the scheme via the following games. In each game, we define X_i as the event that the adversary $\mathcal{A}$ wins in $\mathbf{Game}_i$.

$\mathbf{Game_0}$: This is a real security game. At the beginning, $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ is run, and the adversary $\mathcal{A}$ is given $\mathbf{A}$. Then, the challenger samples $\beta \leftarrow \{0,1\}$ and keeps it secret. During the game, $\mathcal{A}$ may make random oracle queries, secret key queries, and the challenge query. These queries are handled as follows:

- Hash queries: When $\mathcal{A}$ makes a random oracle query to H on id, the challenger chooses a random vector $\mathbf{u}_{id} \leftarrow \mathbb{Z}_q^n$ and locally stores the tuple $(id, \mathbf{u}_{id}, \perp)$, and returns $\mathbf{u}_{id}$ to $\mathcal{A}$.
- Identity secret key queries: When $\mathcal{A}$ queries an identity secret key for id, the challenger uses the algorithm SampleD, which takes $\mathbf{A}, \mathbf{T}_A, \sigma, \mathbf{u}_{id}$ as input to compute $\mathbf{r}_{id}$ and returns $\mathbf{r}_{id}$ to $\mathcal{A}$.
- Challenge ciphertext: When the adversary $\mathcal{A}$ submits two messages $\sum_{m_0 \in 0,1} \alpha_{m_0} |m_0\rangle$ and $\sum_{m_1 \in 0,1} \alpha_{m_1} |m_1\rangle$ of equal length and a challenge identity id^* with the restriction that id^* is not equal to any identity requested in the previous phase, the challenger picks $\beta \xleftarrow{\$} \{0,1\}$, and encrypts $\sum_{m_\beta \in \{0,1\}} \alpha_{m_\beta} |m_\beta\rangle$ under id^* by running the encryption algorithm QEncrypt to get $ct^* = (|\psi\rangle, \mathbf{c}_1)$, where $|\psi\rangle = \sum_{m_\beta \in \{0,1\}} \alpha_{m_\beta} |(m_\beta \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $\mathbf{c}_1 = (\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \bmod q$ and $x = \mathbf{u}^\top \mathbf{s} + e_0 \bmod q$. Then, the ciphertext ct^* is sent to $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs a guess β' for β. Finally, the challenger outputs β. By definition, we have

$$|\Pr[X_0] - \frac{1}{2}| = |\Pr[\beta' = \beta] - \frac{1}{2}| = \mathrm{Adv}_{\mathcal{A}}^{\mathrm{QIBE}}(\lambda). \tag{2}$$

Game$_1$: In this game, we change the way the random oracle queries to H are answered.

- Hash queries: When $\mathcal{A}$ queries the random oracle H on id, the challenger generates a pair $(\mathbf{u}_{id}, \mathbf{r}_{id})$ by first sampling $\mathbf{r}_{id} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{\widetilde{m}}, \sigma}$ and setting $\mathbf{u}_{id} = \mathbf{A} \cdot \mathbf{r}_{id}$. Then, it locally stores the tuple $(id, \mathbf{u}_{id}, \perp)$ and returns $\mathbf{u}_{id}$ to $\mathcal{A}$.
- Identity secret key queries: When $\mathcal{A}$ makes an identity secret key query for id, the challenger uses the algorithm SampleD, which takes $\mathbf{A}, \mathbf{T}_A, \sigma, \mathbf{u}_{id}$ as input to compute $\mathbf{r}'_{id}$ and returns $\mathbf{r}'_{id}$ to $\mathcal{A}$.
- Challenge ciphertext: The same as that in the **Game$_0$**.

Note that $\mathbf{r}'_{id}$ is independent of $\mathbf{r}_{id}$, which was generated in the simulation of the random oracle H on input id. Due to Lemma 2, the distribution of $\mathbf{u}_{id}$ in **Game$_1$** is $2^{-\Omega(n)}$-close to that of **Game$_0$** except for $2^{-\Omega(n)}$ fraction of $\mathbf{A}$ as we choose $\sigma > \sqrt{n + \log \widetilde{m}}$. Therefore, we have

$$|\Pr[X_1] - \Pr[X_0]| = Q_{\mathsf{H}} \cdot 2^{-\Omega(n)}. \tag{3}$$

Game$_2$: In this game, we change the way identity secret key queries are answered. By the end of this game, the challenger will no longer require the trapdoor $\mathbf{T_A}$ to generate the identity secret keys.

- Hash queries: When $\mathcal{A}$ queries the random oracle on id, the challenger generates a pair $(\mathbf{u}_{id}, \mathbf{r}_{id})$ as in the previous game. Then, it locally stores the tuple $(id, \mathbf{u}_{id}, \mathbf{r}_{id})$ and returns $\mathbf{u}_{id}$ to $\mathcal{A}$.
- Identity secret key queries: When $\mathcal{A}$ queries an identity secret key for id, the challenger retrieves the unique tuple $(id, \mathbf{u}_{id}, \mathbf{r}_{id})$ from the local storage and returns $\mathbf{r}_{id}$.
- Challenge ciphertext: The same as that in the **Game$_1$**.

For any fixed $\mathbf{u}_{id}$, let $\mathbf{r}_{id,1}$ and $\mathbf{r}_{id,2}$ be random variables that are distributed according to the distributions of sk_{id} conditional on $\mathsf{H}(id) = \mathbf{u}_{id}$ in **Game$_1$** and **Game$_2$**, respectively. Owing to Lemma 1, we have $\Delta(\mathbf{r}_{id,1}, \mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}), \sigma}) \leq 2^{-\Omega(n)}$. Owing to Lemma 2, we have $\Delta(\mathbf{r}_{id,2}, \mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}), \sigma}) \leq 2^{-\Omega(n)}$. Then, we obtain $\Delta(\mathbf{r}_{id,1}, \mathbf{r}_{id,2}) \leq 2^{-\Omega(n)}$. Therefore, we have

$$|\Pr[X_2] - \Pr[X_1]| = Q_{\mathsf{ID}} \cdot 2^{-\Omega(n)}. \tag{4}$$

Game$_3$: In this game, we change the way the matrix $\mathbf{A}$ is generated. Specifically, the challenger chooses $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times \widetilde{m}}$ without generating the associated trapdoor $\mathbf{T_A}$.

- Hash queries: The same as that in the **Game$_2$**.
- Identity secret key queries: The same as that in the **Game$_2$**.

- Challenge ciphertext: The same as that in the $\mathbf{Game_2}$.

By Lemma 1, this makes only $2^{-\Omega(n)}$-statistical difference. As the challenger can answer all the secret key queries without the trapdoor owing to the change made in the previous game, the view of $\mathcal{A}$ is altered only by $2^{-\Omega(n)}$. Therefore, we have

$$|\Pr[X_3] - \Pr[X_2]| = 2^{-\Omega(n)}. \tag{5}$$

$\mathbf{Game_4}$: In this game, we change the way the random oracle queries to H are answered and the challenge ciphertext is created. The challenger chooses an index $i^* \xleftarrow{\$} [Q_{\mathsf{H}}]$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$ uniformly at random.

- Hash queries: On $\mathcal{A}$'s $j-$th distinct query id_j to H, the challenger does the following: if $j = i^*$, then locally stores the tuple $(id_j, \mathbf{u}, \perp)$ and returns $\mathbf{u}$ to $\mathcal{A}$. Otherwise, for $j \neq i^*$, the challenger selects $\mathbf{r}_{id_j}$ and computes $\mathbf{u}_{id_j} = \mathbf{A}\mathbf{r}_{id_j}$; then, it locally stores the tuple $(id_j, \mathbf{u}_{id_j}, \mathbf{r}_{id_j})$ and returns $\mathbf{u}_{id_j}$ to $\mathcal{A}$.
- Identity secret key queries: The same as that in the $\mathbf{Game_3}$.
- Challenge ciphertext: When $\mathcal{A}$ produces a challenge identity id^* (distinct from all its identity secret key queries) and messages $\sum_{m_0 \in \{0,1\}} \alpha_{m_0} |m_0\rangle$, $\sum_{m_1 \in \{0,1\}} \alpha_{m_1} |m_1\rangle$, assume without loss of generality that $\mathcal{A}$ already queried H on id^*. If $id^* \neq id_{i^*}$, i.e., if the tuple $(id_{i^*}, \mathbf{u}, \perp)$ is not in the local storage, then the challenger ignores the output of $\mathcal{A}$ and aborts the game (we denote this event as abort). Otherwise, i.e., if the abort does not occur (we denote this event as $\overline{\mathsf{abort}}$), the challenger picks $\beta \xleftarrow{\$} \{0,1\}$ and encrypts $\sum_{m_\beta \in \{0,1\}} \alpha_{m_\beta} |m_\beta\rangle$ under id^* by running the encryption algorithm $\mathsf{QEncrypt}$ to obtain $ct^* = (|\psi\rangle, \mathbf{c}_1)$, where $|\psi\rangle = \sum_{m_\beta \in \{0,1\}} \alpha_{m_\beta} |(m_\beta \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $\mathbf{c}_1 = (\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \bmod q$, and $x = \mathbf{u}^\top \mathbf{s} + e_0 \bmod q$. Then, the ciphertext ct^* is sent to the adversary $\mathcal{A}$.

Conditional on the challenger not aborting, we affirm that the view it provides to $\mathcal{A}$ in $\mathbf{Game_4}$ is statistically close to that in $\mathbf{Game_3}$. Therefore, we have

$$\Pr[X_4 \mid \overline{\mathsf{abort}}] = \Pr[X_3 \mid \overline{\mathsf{abort}}]. \tag{6}$$

By a standard argument, the probability that the challenger does not abort during the simulation is $\frac{1}{Q_{\mathsf{H}}}$ (this is proved by considering a game in which the challenger can answer all identity secret key queries, so that the value of i^* is perfectly hidden from $\mathcal{A}$). Therefore, we have

$$\Pr[\overline{\mathsf{abort}}] = \frac{1}{Q_{\mathsf{H}}}. \tag{7}$$

$\mathbf{Game_5}$: In this game, we change the way the challenge ciphertext is created.

- Hash queries: The same as that in the $\mathbf{Game_4}$.
- Identity secret key queries: The same as that in the $\mathbf{Game_4}$.
- Challenge ciphertext: When $\mathcal{A}$ produces a challenge identity id^* (distinct from all its identity secret key queries) and messages $\sum_{m_0 \in \{0,1\}} \alpha_{m_0} |m_0\rangle$, $\sum_{m_1 \in \{0,1\}} \alpha_{m_1} |m_1\rangle$, assume without loss of generality that $\mathcal{A}$ already queried H on id^*. If $id^* \neq id_i$, i.e., if the tuple $(id_i, \mathbf{u}, \perp)$ is not in the local storage, then the challenger ignores the output of $\mathcal{A}$ and aborts the game. Otherwise, i.e., if the abort does not occur, the challenger picks $\beta \xleftarrow{\$} \{0,1\}$ and encrypts $\sum_{m_\beta \in \{0,1\}} \alpha_{m_\beta} |m_\beta\rangle$ under id^* using two random vectors $b' \xleftarrow{\$} \mathbb{Z}_q, \mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m$ to obtain $ct^* = (|\psi\rangle, \mathbf{c}_1)$, where $|\psi\rangle = \sum_{m_\beta \in \{0,1\}} \alpha_{m_\beta} |(m_\beta \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $\mathbf{c}_1 = \mathbf{b}$, and $x = b'$. Then, the ciphertext ct^* is sent to the adversary $\mathcal{A}$.

It can be seen that if $(\mathbf{A}, \mathbf{u}, \mathbf{c}_1, x)$ are valid LWE samples (i.e., $\mathbf{c}_1 = (\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \bmod q$ and $x = \mathbf{u}^\top \mathbf{s} + e_0 \bmod q$), the view of the adversary corresponds to $\mathbf{Game_4}$. Otherwise (i.e., $\mathbf{c}_1 \xleftarrow{\$} \mathbb{Z}_q^{\widetilde{m}}, x \xleftarrow{\$} \mathbb{Z}_q$), it corresponds to $\mathbf{Game_5}$. Therefore, we have

$$\left|\Pr[X_5 \wedge \overline{\mathsf{abort}}] - \Pr[X_4 \wedge \overline{\mathsf{abort}}]\right| \leq \mathrm{Adv}_{\mathcal{B}}^{\mathrm{LWE}}(\lambda). \tag{8}$$

Note that $\mathbf{c}_1, x$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{\widetilde{m}} \times \mathbb{Z}_q$, so that

$$\Pr[X_5 \mid \overline{\mathsf{abort}}] = \frac{1}{2}. \tag{9}$$

According to Equations (6)–(9), we can obtain

$$\left|\Pr[X_3 \mid \overline{\mathsf{abort}}] - \frac{1}{2}\right| \leq Q_{\mathsf{H}} \cdot \mathrm{Adv}_{\mathcal{B}}^{\mathrm{LWE}}(\lambda).$$

Then, because $\overline{\mathsf{abort}}$ is independent of X_3, we get

$$\left|\Pr[X_3] - \frac{1}{2}\right| \leq Q_{\mathsf{H}} \cdot \mathrm{Adv}_{\mathcal{B}}^{\mathrm{LWE}}(\lambda). \tag{10}$$

Finally, according to Equations (2)–(5) together with Equation (10), we obtain Equation (1). □

3.5. Advantages of Our QIBE

The proposed QIBE scheme has two main advantages. One is that the network security protocol based on our QIBE scheme has perfect forward secrecy, and the other is that the QIBE scheme can still use the KGC system to generate and distribute private identity keys, thus reducing the cost when this scheme is implemented.

- A fundamental fact in quantum information theory is that unknown or random quantum states cannot be replicated [17]. The probability amplitude and corresponding basis state of ciphertext $|\psi\rangle = \sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ are unknown to the adversary, so the ciphertext $|\psi\rangle$ is an unknown quantum state and cannot be replicated during transmission. When attempting to attack our QIBE-based network security protocol, an attacker cannot copy and store the ciphertext of the session key, which is encrypted by a private identity key (called a long-term key). Thus, the attacker does not have the quantum ciphertext of the previous session key to decrypt, and the security of the previous session key will not be threatened even if the attacker obtains the long-term key. In other words, all encrypted communications and sessions recorded in the past cannot be retrieved. Therefore, the security protocol based on our QIBE has perfect forward secrecy, which cannot be achieved by the security protocol based on classic IBE.
- In our QIBE scheme, the KGC uses the algorithms $\mathsf{QKeyGen}$ and $\mathsf{QExtract}$ to generate the private identity key sk_{id} when it takes as input a master public key mpk, a master secret key msk, and an identity id. The key observation is that both the input $(\lambda, q, \widetilde{m}, n)$ and the output $(\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H}), \mathsf{msk} = (\mathbf{T_A}))$ of $\mathsf{QKeyGen}$ are in the form of classic bits, and so are the input $(\mathsf{msk}, \mathsf{mpk}, id)$ and the output $(sk_{id} = \mathbf{r})$ of $\mathsf{QExtract}$. Therefore, the classic KGC system can still be used in our QIBE scheme to generate and distribute private identity keys, reducing the cost when this scheme is implemented.

4. Quantum Circuit Realisation

The realisation of a quantum scheme requires describing the corresponding quantum circuit. To analyse the realisability of our QIBE scheme, we begin by providing details for the quantum circuits of $\mathsf{QEncrypt}$ and $\mathsf{QDecrypt}$, and then, we analyse the complexity of these two quantum circuits.

4.1. Quantum Circuit

Note that the algorithms QKeyGen and QExtract of QIBE are classic, so there is no need to construct quantum circuits for these two algorithms.

The Quantum Circuit of QEncrypt**.** The main part of QEncrypt is the unentangled quantum circuit Unentangle. Taking an additional quantum state $|(\lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ as input, Unentangle can transform $\sum_{m\in\{0,1\}} \alpha_m |m\rangle|(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ to a non-entangled quantum state $|\psi\rangle = \sum_{m\in\{0,1\}} \alpha_m |(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$, as shown in Figure 5a. The correctness of Unentangle can be proven by the following facts.

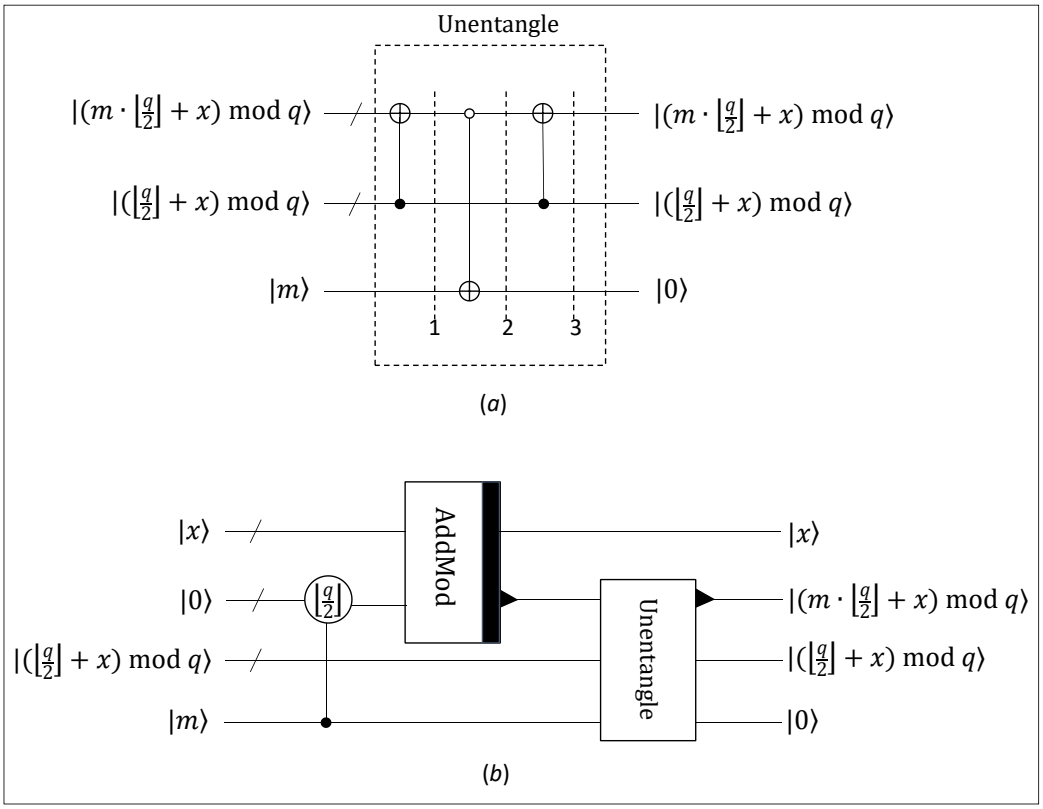

Figure 5. The quantum circuits of Unentangle and QEncrypt. In this figure, (**a**) shows the Unentangle circuit, and (**b**) depicts the QEncrypt circuit.

- On input $|(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$, $|(\lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $|m\rangle$, the initial output where the number 1 is located is $|((m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q) \oplus (\lfloor \frac{q}{2} \rfloor + x) \bmod q)\rangle$, $|(\lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $|m\rangle$. This is because only one $\lceil \log q \rceil$-combination-CNOT operation is performed.
- The output where the number 2 is located is $|((m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q) \oplus (\lfloor \frac{q}{2} \rfloor + x) \bmod q)\rangle$, $|(\lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $|0\rangle$. This is because only one $\lceil \log q \rceil$-controlled-NOT variant operation is performed, in which the control bits are $|((m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q) \oplus (\lfloor \frac{q}{2} \rfloor + x) \bmod q)\rangle$ and the target bit is $|m\rangle$. If $|m\rangle = |1\rangle$, the control bits are $\lceil \log q \rceil$-qubit $|0\rangle$, and the target qubit $|m\rangle$ will change to $|0\rangle$. Otherwise, the control bits are not equal to $\lceil \log q \rceil$-qubit $|0\rangle$, and the target qubit $|m\rangle$ is $|0\rangle$ all the time.
- The final output where the number 3 is located is $|(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$, $|(\lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and $|0\rangle$. Only one $\lceil \log q \rceil$-combination-CNOT operation is performed, in which the control bits are $|(\lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$ and the target bits are $|((m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q) \oplus (\lfloor \frac{q}{2} \rfloor + x) \bmod q)\rangle$; therefore, it can obtain $|(m \cdot \lfloor \frac{q}{2} \rfloor + x) \bmod q\rangle$.

Note that one unentangled quantum circuit is composed of two $\lceil \log q \rceil$-combination-CNOT gates and one $\lceil \log q \rceil$-controlled-NOT variant gate. Thus, one unentangled quantum circuit is composed of $2 \cdot \lceil \log q \rceil$ CNOT gates and $2\lceil \log q \rceil - 3$ Toffoli gates.

Using the unentangled quantum circuit Unentangle, we can provide details for the quantum circuit of QEncrypt, which is depicted in Figure 5b. Note that one QEncrypt is composed of one controlled copy circuit, one modular addition circuit, and one unentangled quantum circuit. To encrypt one qubit, a total of $4 \cdot \lceil \log q \rceil + 4$ qubits, $10 \cdot \lceil \log q \rceil - 3$ Toffoli gates and $15.5 \cdot \lceil \log q \rceil + 6$ CNOT gates are required.

The Quantum Circuit of QDecrypt**.** The main part of QDecrypt is SubAbs and the inverse of SubAbs. In the SubAbs circuit, with the input $(|a\rangle, |b\rangle, |0\rangle)$, the output will produce $(|a\rangle, |\mathsf{Abs}(a-b)\rangle, |0\rangle)$ when $a \geq b$. When $a < b$, the output is $(|b\rangle, |\mathsf{Abs}(a-b)\rangle, |1\rangle)$.

$$\begin{cases} |a, b, 0\rangle \rightarrow |a, a-b, 0\rangle, & \text{for } a \geq b. \\ |a, b, 0\rangle \rightarrow |b, b-a, 1\rangle, & \text{for } a < b. \end{cases}$$

On the inverse of the SubAbs circuit, i.e., InvSubAbs, with the input $(|a\rangle, |a-b\rangle, |0\rangle)$, the output will produce $(|a\rangle, |b\rangle, |0\rangle)$ when $a \geq b$. When $a < b$, the input is $(|b\rangle, |b-a\rangle, |1\rangle)$ and the output is $(|a\rangle, |b\rangle, |0\rangle)$.

$$\begin{cases} |a, a-b, 0\rangle \rightarrow |a, b, 0\rangle, & \text{for } a \geq b. \\ |b, b-a, 1\rangle \rightarrow |a, b, 0\rangle, & \text{for } a < b. \end{cases}$$

The quantum circuit of SubAbs is depicted in Figure 6, and the correctness of the quantum circuit SubAbs can be easily verified. As a result of the reversibility of unitary operations, by reversing the circuit of SubAbs, that is, by applying each gate of the circuit in the reversed order, the quantum circuit of InvSubAbs can be obtained.

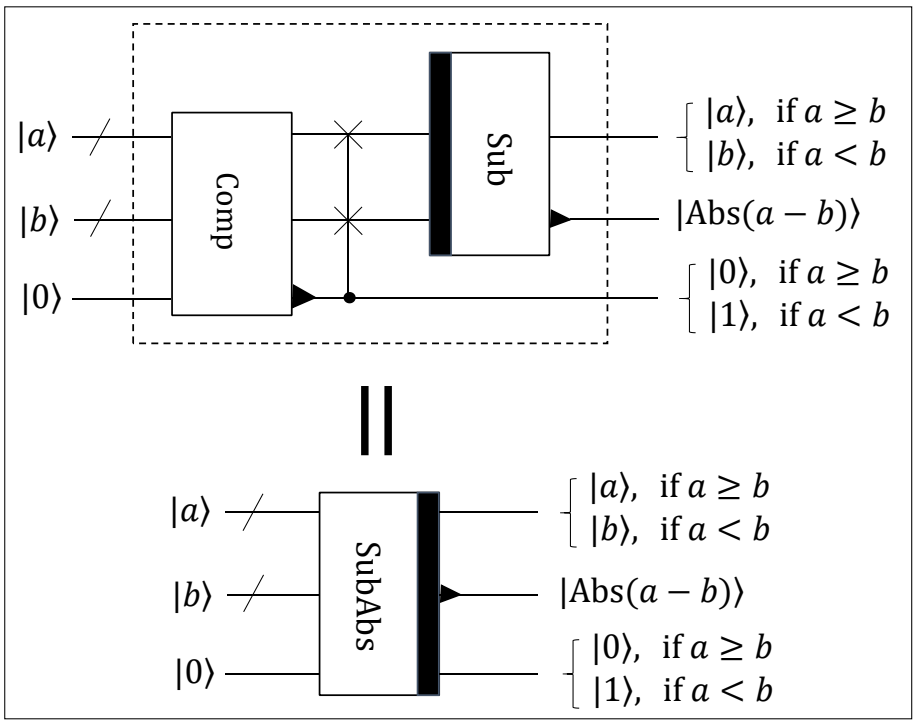

Figure 6. The quantum circuit of SubAbs.

Note that one SubAbs is composed of one comparison circuit and one subtraction quantum circuit. To compute the absolute value of the subtraction of two ℓ-bit inputs $|a\rangle$ and $|b\rangle$, a total of $2\ell + 4$ qubits, 5ℓ Toffoli gates, and $10\ell + 2$ CNOT gates are required. The conclusion is also applicable to the inverse of SubAbs, that is, InvSubAbs.

Using the quantum circuit SubAbs and its reverse InvSubAbs, we can provide particulars for the quantum circuit of QDecrypt which is depicted in Figure 7. Note that one QDecrypt is composed of two subtraction modular circuits, one quantum circuit of SubAbs,

one comparison quantum circuit, one quantum circuit of InvSubAbs, one modular addition circuit, and two controlled copy quantum circuits. To decrypt one ciphertext that encrypts one qubit, a total of $6 \cdot \lceil \log q \rceil + 5$ qubits, $36 \cdot \lceil \log q \rceil$ Toffoli gates, and $64 \cdot \lceil \log q \rceil + 23$ CNOT gates are required.

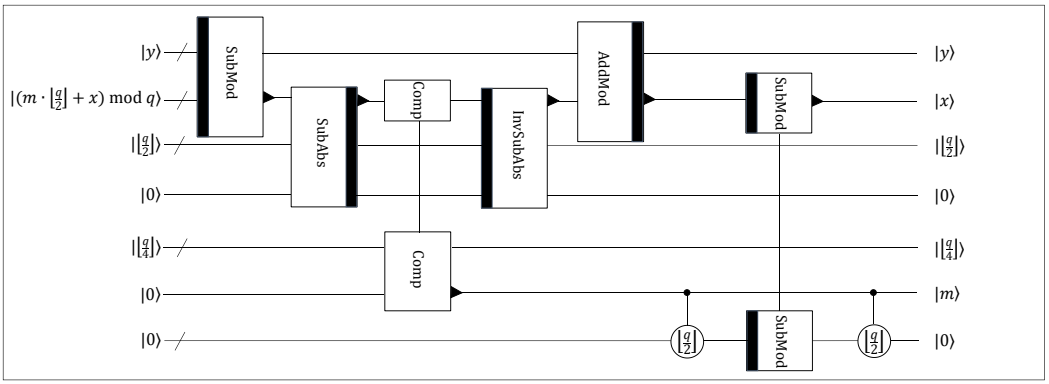

Figure 7. The quantum circuit of QDecrypt.

4.2. Complexity Analysis

To measure the complexity of a quantum circuit, we should consider the quantum resources required by the circuit.

Quantum resources. When analysing the complexity of a quantum circuit, a gate set that arises frequently and that has been studied often in the literature, but by no means the only conceivable gate set, is the so-called Clifford+T gate set. This gate set consists of the Hadamard gate, the phase gate, and the controlled NOT (CNOT) gate, along with the T gate. The Clifford+T gate set is known to be universal [18]. This comes from that any unitary operator can be accurately expressed using single-qubit gates and CNOT gates [29], and Hadamard gates, phase gates, and T gates can used to approximate any single-qubit gates with arbitrary precision [18]. In conclusion, these four types of quantum gates form a universal quantum gate group. As a result, when assessing the complexity of a quantum circuit, we need to compute the number of needed Hadamard gates, phase gates, CNOT gates, and T gates.

Quantum resources needed by QEncrypt **and** QDecrypt**.** In Section 4.1, we have concluded the following:

- QEncrypt : To encrypt one qubit, a total of $4 \cdot \lceil \log q \rceil + 4$ qubits, $10 \cdot \lceil \log q \rceil - 3$ Toffoli gates, and $15.5 \cdot \lceil \log q \rceil + 6$ CNOT gates are required.
- QDecrypt : To decrypt one ciphertext that encrypts one qubit, a total of $6 \cdot \lceil \log q \rceil + 5$ qubits, $36 \cdot \lceil \log q \rceil$ Toffoli gates, and $64 \cdot \lceil \log q \rceil + 23$ CNOT gates are required.

According to [30], one Toffoli gate can be broken down into two Hadamard gates, one phase gate, seven T gates, and six CNOT gates. Furthermore, to save quantum resources, auxiliary bits can be reused according to the sequence of calculations in each circuit [19]. Based on the previous results and analysis, we estimate the quantum resources needed when encrypting one qubit quantum state $\sum_{m \in \{0,1\}} \alpha_m |m\rangle$ with the algorithm QEncrypt and decrypting the corresponding ciphertext with the algorithm QDecrypt in our QIBE scheme, including the number of qubits and the number of Hadamard, phase, T, and CNOT gates. The result is shown in Table 1.

Table 1. Quantum resource.

Quantum Resource	QEncrypt	QDecrypt
Qubit	$4 \cdot \lceil \log q \rceil + 4$	$6 \cdot \lceil \log q \rceil + 5$
Hadamard gate	$20 \cdot \lceil \log q \rceil - 6$	$72 \cdot \lceil \log q \rceil$
phase gate	$10 \cdot \lceil \log q \rceil - 3$	$36 \cdot \lceil \log q \rceil$
T gate	$70 \cdot \lceil \log q \rceil - 21$	$252 \cdot \lceil \log q \rceil$
CNOT gate	$75.5 \cdot \lceil \log q \rceil - 12$	$280 \cdot \lceil \log q \rceil + 23$

Note that our QIBE scheme can encrypt one qubit at a time. By integrating our scheme and the IBE scheme encrypting n-bit message (The IBE scheme is described in Appendix A), we can easily construct a new QIBE that can encrypt n-qubit at a time. It is evident that the quantum resources required by this new QIBE scheme increase linearly with the number of plaintext bits to be encrypted.

5. Conclusions and Future Work

In this paper, we proposed the first QIBE scheme based on the learning with errors problem. Then, we proved that our scheme is fully secure under the random oracle model. Furthermore, we explained that our scheme possesses the following advantages:

- The network security protocol with our QIBE scheme provides perfect forward secrecy. The ciphertext is transmitted in the form of a quantum state that is unknown to the adversary and cannot be copied and stored. Thus, in network security protocols based on our QIBE construction, the adversary cannot have access to any previous quantum ciphertext to decrypt and obtain the previous session key, even if the private identity key is threatened.
- Classic KGC systems still can be used in our QIBE scheme to generate and distribute private identity keys, thus reducing the cost when this scheme is implemented. The classic KGC systems can be used because the master public and secret keys of our scheme are both in the form of classic bits.

Finally, to analyse the realisability of this QIBE scheme, we provided particulars for the quantum circuits of QEncrypt and QDecrypt, and we analysed their required quantum resources for given numbers of qubits, Hadamard gates, phase gates, T gates, and CNOT gates. We concluded that the quantum resources required by our scheme increase linearly with the number of plaintext bits to be encrypted.

As aforementioned, network security protocols based on our QIBE scheme have forward secrecy, unlike the classic IBE. However, our QIBE scheme has certain drawbacks. In terms of quantum circuit realisation, we do not yet have a method to obtain the optimal circuit and find the lower bound of the quantum resources required by this scheme. Furthermore, this construction is a theoretical achievement, and any practical application remains a distant goal before the advent of universal quantum computers.

Our scheme is one of fifteen types of QIBE schemes described in Section 3.1, and the other fourteen types are yet to be studied. Therefore, the focus of our next work is to study and design the other fourteen types of QIBE schemes and their quantum circuit implementations, and to explore ways to find the lower bound of the quantum resources required by these schemes. Moreover, to make our scheme more practical, our next work includes making an implementation in Q# and providing a Github repository for it.

Author Contributions: Conceptualisation, W.G. and L.Y.; methodology, W.G.; formal analysis, D.Z; writing—original draft preparation, W.G; writing—review and editing, W.G., D.Z. and X.L.; supervision, L.Y.; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 61672517), National Natural Science Foundation of China (Key Program, Grant No. 61732021).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Appendix A. Classic IBE Scheme

IBE encrypting one-bit message. In [9], Gentry et al. proposed the first lattice-based IBE scheme $\mathsf{IBE} = (\mathsf{KeyGen}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ from the learning with errors problem. In this scheme, let integer parameters $n = \mathcal{O}(\lambda), \widetilde{m} = \mathcal{O}(n), \sigma = \mathcal{O}(n^{0.5}), q = \mathcal{O}(\widetilde{m}^{3.5})$, where λ is a security parameter. This scheme can encrypt one bit once, and it can also encrypt n bits at a time. At first, we show the scheme which can encrypt one bit.

- KeyGen : (1) It runs $\mathsf{TrapGen}(\widetilde{m}, n, q)$ to obtain a uniformly random $n \times \widetilde{m}-$matric $\mathbf{A} \in \mathbb{Z}_q^{n \times \widetilde{m}}$ and $\mathbf{T_A} \in \mathbb{Z}_q^{\widetilde{m} \times \widetilde{m}}$ which is a good basis for $\Lambda_q^{\perp}(\mathbf{A})$. (2) Then, it selects a hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{Z}_q^n$ which maps an identity to a vector. (3) Finally, it outputs $\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H})$ and $\mathsf{msk} = (\mathbf{T_A})$. (4) In summary, $\mathsf{KeyGen}(\lambda, q, \widetilde{m}, n) \to (\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H}), \mathsf{msk} = (\mathbf{T_A}))$.
- $\mathsf{Extract}$: (1) On input mpk, msk and an identity $id \in \{0,1\}^n$, it computes $\mathbf{u} = \mathsf{H}(id)$ and generates $sk_{id} = \mathbf{r}$ such that $\mathbf{r} = \mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$. It is clear that $\mathbf{u} = \mathbf{Ar} \bmod q$. (2) In summary, $\mathsf{Extract}(\mathsf{msk}, \mathsf{mpk}, id) \to sk_{id} = \mathbf{r}$.
- $\mathsf{Encrypt}$: (1) On input of an identity id, mpk, and a bit message $m \in \{0,1\}$, it first computes $\mathbf{u} = \mathsf{H}(id)$ and chooses a uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_0 \in \mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathbf{e} \in \mathcal{D}_{\mathbb{Z}^{\widetilde{m}},\sigma}$. (2) Then, it computes $x = (\mathbf{u}^\top \mathbf{s} + e_0) \bmod q$ and $\mathbf{c}_1 = (\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \bmod q$, and sets $ct = (c_0, \mathbf{c}_1)$, where $c_0 = x + \lfloor \frac{q}{2} \rfloor \cdot m \bmod q$. (3) In summary, $\mathsf{Encrypt}(id, \mathsf{mpk}, m) \to ct = (c_0, \mathbf{c}_1)$.
- $\mathsf{Decrypt}$: (1) To decrypt a ciphertext $ct = (c_0, \mathbf{c}_1)$ using an identity secret key $sk_{id} = \mathbf{r}$, it computes $y = \mathbf{r}^\top \mathbf{c}_1 \bmod q$. (2) Then, it computes

$$b = (c_0 - y) \bmod q - \lfloor \frac{q}{2} \rfloor.$$

(3) Furthermore, it treats b as an integer in $\mathbb{Z}$, and sets $m = 1$ if $\mathsf{Abs}(b) < \lfloor \frac{q}{4} \rfloor$; otherwise, $m = 0$. (4) Finally, it returns the plaintext m. (5) In summary, $\mathsf{Decrypt}(id, \mathsf{mpk}, sk_{id}, ct) \to m$.

Theorem A1. *Let integer parameters* $n = \mathcal{O}(\lambda), \widetilde{m} = \mathcal{O}(n), \sigma = \mathcal{O}(n^{0.5}), q = \mathcal{O}(\widetilde{m}^{3.5})$. *Consider a cipertext*

$$(c_0, \mathbf{c}_1) = \left(\mathbf{u}^\top \mathbf{s} + e_0 + \lfloor \frac{q}{2} \rfloor \cdot m, \mathbf{A}^\top \mathbf{s} + \mathbf{e}\right) \textit{ mod } q$$

of one bit message m. Then, the decryption algorithm $\mathsf{Decrypt}$ *with the identity secret key* $sk_{id} = \mathbf{r}$ *can decrypt the ciphertext ct correctly with a probability* $1 - \mathsf{negl}(\lambda)$.

IBE encrypting n-bit message. Then, we show the scheme that can encrypt more than one bit at a time.

- KeyGen: (1) It runs $\mathsf{TrapGen}(\widetilde{m}, n, q)$ to obtain a uniformly random $n \times \widetilde{m}-$matric $\mathbf{A} \in \mathbb{Z}_q^{n \times \widetilde{m}}$ and $\mathbf{T_A} \in \mathbb{Z}_q^{\widetilde{m} \times \widetilde{m}}$ which is a good basis for $\Lambda_q^{\perp}(\mathbf{A})$. (2) Then, it selects a hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{Z}_q^{n \times n}$ which maps an identity to a matrix. (3) Finally, it outputs $\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H})$ and $\mathsf{msk} = (\mathbf{T_A})$. (4) In summary, $\mathsf{KeyGen}(\lambda, q, \widetilde{m}, n) \to (\mathsf{mpk} = (\mathbf{A}, q, \widetilde{m}, n, \mathsf{H}), \mathsf{msk} = (\mathbf{T_A}))$.
- $\mathsf{Extract}$: (1) On input mpk, msk, and an identity $id \in \{0,1\}^n$, it computes $\mathbf{U} = \mathsf{H}(id)$ which is an $n \times n$-matrix. (2) It takes advantage of the algorithm $\mathsf{SampleD}$ to generate the identity secret key $sk_{id} = \mathbf{R}$ such that $\mathbf{R}$ is composed of $\mathbf{r}_i = \mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}_i, \sigma)$ where u_i is the i-th column of $\mathbf{U}$. It is easy to see that $\mathbf{U} = \mathbf{AR} \bmod q$. (3) In a word, $\mathsf{Extract}(\mathsf{msk}, \mathsf{mpk}, id) \to sk_{id} = \mathbf{R}$.

- Encrypt: (1) On input of an identity id, mpk, and one n-bit message $\mathbf{m} \in \{0,1\}^n$, it first computes $\mathbf{U} = \mathsf{H}(id)$ and chooses a uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0 \in \mathcal{D}_{\mathbb{Z}^n,\sigma}$ and $\mathbf{e} \in \mathcal{D}_{\mathbb{Z}^{\tilde{m}},\sigma}$. (2) Then, it computes $\mathbf{x} = \mathbf{U}^\top \mathbf{s} + \mathbf{e}_0$ and $\mathbf{c}_1 = (\mathbf{A}^\top \mathbf{s} + \mathbf{e}) \bmod q$, and sets $ct = (\mathbf{c}_0, \mathbf{c}_1)$, where $\mathbf{c}_0 = \mathbf{x} + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m} \bmod q$. (3) In summary, $\mathsf{Encrypt}(id, \mathsf{mpk}, \mathbf{m}) \rightarrow ct = (\mathbf{c}_0, \mathbf{c}_1)$.
- Decrypt: (1) To decrypt a ciphertext $ct = (\mathbf{c}_0, \mathbf{c}_1)$ using an identity secret key $sk_{id} = \mathbf{R}$, it computes $\mathbf{y} = \mathbf{R}^\top \mathbf{c}_1 \bmod q$. (2) Then, it computes

$$\mathbf{b} = (\mathbf{c}_0 - \mathbf{y}) \bmod q - \lfloor \frac{q}{2} \rfloor \cdot (1, 1, \cdots, 1)^\top.$$

(3) Furthermore, it treats each coordinate of $\mathbf{b} = (b_1, \cdots, b_n)^\top$ as an integer in $\mathbb{Z}$, and sets $m_i = 1$ if $\mathsf{Abs}(b_i) < \lfloor \frac{q}{4} \rfloor$, else $m_i = 0$. (4) Finally, it returns the plaintext $\mathbf{m}$. (5) In a word, $\mathsf{Decrypt}(id, \mathsf{mpk}, sk_{id}, ct) \rightarrow \mathbf{m}$.

Theorem A2. *Let integer parameters* $n = \mathcal{O}(\lambda), \tilde{m} = \mathcal{O}(n), \sigma = \mathcal{O}(n^{0.5}), q = \mathcal{O}(\tilde{m}^{3.5})$. *Consider a cipertext*

$$(\mathbf{c}_0, \mathbf{c}_1) = \left(\mathbf{U}^\top \mathbf{s} + \mathbf{e}_0 + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}\right) \ mod \ q$$

of n-bit message $\mathbf{m}$. *Then, the decryption algorithm* Decrypt *with the identity secret key* $sk_{id} = \mathbf{R}$ *can decrypt the ciphertext ct correctly with a probability* $1 - \mathsf{negl}(\lambda)$.

References

1. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology, Proceedings of CRYPTO*; Springer: Berlin/Heidelberg, Germany, 1984; Volume 196, pp. 47–53.
2. Boneh, D.; Franklin, M.K. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology—CRYPTO 2001*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2139, pp. 213–229.
3. Boneh, D.; Boyen, X. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2004*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 223–238.
4. Waters, B. Efficient Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2005*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3494, pp. 114–127.
5. Cocks, C.C. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Cryptography and Coding, 8th IMA International Conference*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2260, pp. 360–363.
6. Boneh, D.; Gentry, C.; Hamburg, M. Space-Efficient Identity Based Encryption Without Pairings. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), Providence, RI, USA, 21–23 October 2007; pp. 647–657.
7. Jhanwar, M.P.; Barua, R. A Variant of Boneh-Gentry-Hamburg's Pairing-Free Identity Based Encryption Scheme. In *Information Security and Cryptology*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5487, pp. 314–331.
8. Joye, M. Identity-Based Cryptosystems and Quadratic Residuosity. In *Public-Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9614, pp. 225–254.
9. Gentry, C.; Peikert, C.; Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the 40th Annual Symposium on Theory of Computing, Victoria, BC, Canada, 17–20 May 2008; pp. 197–206.
10. Cash, D.; Hofheinz, D.; Kiltz, E.; Peikert, C. Bonsai Trees, or How to Delegate a Lattice Basis. In *Advances in Cryptology—EUROCRYPT 2010*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6110, pp. 523–552.
11. Agrawal, S.; Boneh, D.; Boyen, X. Efficient Lattice (H)IBE in the Standard Model. In *Advances in Cryptology—EUROCRYPT 2010*; Gilbert, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6110, pp. 553–572. [CrossRef]
12. Xie, X.; Xue, R.; Zhang, R. Deterministic Public Key Encryption and Identity-Based Encryption from Lattices in the Auxiliary-Input Setting. In *Security and Cryptography for Networks*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7485, pp. 1–18.
13. Yamada, S. Adaptively Secure Identity-Based Encryption from Lattices with Asymptotically Shorter Public Parameters. In *Advances in Cryptology—EUROCRYPT 2016*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9666, pp. 32–62.
14. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
15. Chinese-State-Cryptography-Administration. Chinese SSL VPN Technology Specification. Available online: http://gmbz.org.cn/main/viewfile/20180110021416665180.html (accessed on 1 January 2022).
16. Van Oorschot, P.C.; Menezes, A.J.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1996.
17. Wootters, W.K.; Zurek, W.H. A single quantum cannot be cloned. *Nature* **1982**, *299*, 802–803. [CrossRef]
18. Nielsen, M.A.; Chuang, I. Quantum Computation and Quantum Information; American. *Assoc. Phys. Teach.* **2002**, *70*, 558–559.

19. Häner, T.; Jaques, S.; Naehrig, M.; Roetteler, M.; Soeken, M. Improved quantum circuits for elliptic curve discrete logarithms. In *International Conference on Post-Quantum Cryptography*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 425–444.
20. Cuccaro, S.A.; Draper, T.G.; Kutin, S.A.; Moulton, D.P. A new quantum ripple-carry addition circuit. *arXiv* **2004**, arXiv:0410184.
21. Liu, X.; Yang, H.; Yang, L. CNOT-count optimized quantum circuit of the extended Shor's algorithm for ECDLP. *arXiv* **2021**, arXiv:2112.11358.
22. Roetteler, M.; Naehrig, M.; Svore, K.M.; Lauter, K. Quantum resource estimates for computing elliptic curve discrete logarithms. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 241–270.
23. Markov, I.L.; Saeedi, M. Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation. *Quantum Inf. Comput.* **2012**, *12*, 361–394. [CrossRef]
24. Ajtai, M. *Generating Hard Instances of the Short Basis Problem*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 1–9.
25. Alwen, J.; Peikert, C. *Generating Shorter Bases for Hard Random Lattices*; STACS Schloss Dagstuhl–Leibniz-Zentrum für Informatik: Dagstuhl, Germany, 2009; pp. 75–86.
26. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *STOC* **2005**, *56*, 84–93.
27. Wu, C.; Yang, L. A complete classification of quantum public-key encryption protocols. In *Electro-Optical and Infrared Systems*; International Society for Optics and Photonics: Bellingham, WA, USA, 2015; Volume 9648, p. 964818.
28. Xiang, C.; Yang, L.; Peng, Y.; Chen, D. The classification of quantum symmetric-key encryption protocols. In *Quantum and Nonlinear Optics III*; International Society for Optics and Photonics: Bellingham, WA, USA, 2014; Volume 9269, p. 926909.
29. Deutsch, D. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A Math. Phys. Sci.* **1985**, *400*, 97–117.
30. Amy, M.; Maslov, D.; Mosca, M.; Roetteler, M. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2013**, *32*, 818–830. [CrossRef]