Computers, Materials & Continua
DOI:10.32604/cmc.2021.017216
Article

Tech Science Press

Verifiable Identity-Based Encryption with Keyword Search for IoT from Lattice

Lin Mei[1], Chungen Xu[1,*], Lei Xu[1], Xiaoling Yu[2] and Cong Zuo[3]

[1]School of Science, Nanjing University of Science and Technology, Nanjing, 210094, China
[2]College of Data Science, Taiyuan University of Technology, Taiyuan, 030000, China
[3]Faculty of Information Technology, Monash University, Clayton, 3800, Australia
[*]Corresponding Author: Chungen Xu. Email: xuchung@njust.edu.cn
Received: 19 January 2021; Accepted: 24 February 2021

Abstract: Internet of Things (IoT), which provides the solution of connecting things and devices, has increasingly developed as vital tools to realize intelligent life. Generally, source-limited IoT sensors outsource their data to the cloud, which arises the concerns that the transmission of IoT data is happening without appropriate consideration of the profound security challenges involved. Though encryption technology can guarantee the confidentiality of private data, it hinders the usability of data. Searchable encryption (SE) has been proposed to achieve secure data sharing and searching. However, most of existing SE schemes are designed under conventional hardness assumptions and may be vulnerable to the adversary with quantum computers. Moreover, the untrusted cloud server may perform an unfaithful search execution. To address these problems, in this paper, we propose the first verifiable identity-based keyword search (VIBKS) scheme from lattice. In particular, a lattice-based delegation algorithm is adopted to help the data user to verify both the correctness and the integrity of the search results. Besides, in order to reduce the communication overhead, we refer to the identity-based mechanism. We conduct rigorous proof to demonstrate that the proposed VIBKS scheme is ciphertext indistinguishable secure against the semi-honest-but-curious adversary. In addition, we give the detailed computation and communication complexity of our VIBKS and conduct a series of experiments to validate its efficiency performance.

Keywords: Internet of Things; verifiable; lattice; searchable encryption

1 Introduction

Internet of Things (IoT) has developed as one of the most popular and exciting technologies in information and communications [1,2]. Recent years' studies have witnessed a broader range of applications in IoT networks, e.g., intelligent traffic, household appliances and medical devices. By 2025, forecasts suggest that there will be more than 75 billion Internet of Things (IoT) connected devices in use. A large number of devices produce tens of zettabytes of data in IoT makes the data storage and processing become a major challenger. Fortunately, cloud storage provides a solution

that helps the IoT devices store and process data. However, due to the personal nature of the information collected in IoT networks, consumers are concerned about the sharing of information that reveals their personal habits. Therefore, sensitive private data needs to be encrypted before outsourced to the IoT cloud [3].

Along with data privacy, data utility is indispensable as well. Take the intelligent traffic as an example, the sensor (i.e., the data owner) transmits its data to the cloud server every day for subsequent maintenance and dispatch tasks. However, as mentioned, the above data is encrypted which may incur many practical issues. The primary problem is that how can the data owner get the desired data from such a large amount of encrypted dataset for data analytics. The naïve decrypt to search approach involves very expensive computation and communication overhead, and thus promotes the study of efficient and practical retrieval over the encrypted data.

Searchable symmetric encryption (SSE) [4] is such a promising paradigm proposed to address aforementioned encrypted retrieval problem. Despite SSE translates encrypted retrieval into reality, this mechanism is with limits in achieving practical data-sharing principle guarantee. As known, in SSE, the data owner and the data user are required to be the same one, but the above data analyst (user) are always experts from many different areas. To fill the gap, public key encryption with keyword search (PEKS) scheme is proposed [5]. In PEKS, a data owner first encrypts keywords with the public key of a designated data user and then uploads them to the cloud together with encrypted data. Afterwards, when the data user wants to retrieve data containing a specific keyword, he generates a search token for the keyword with his secret key and sends it to the cloud server. Once the cloud server receives the above search token, it performs well-designed search algorithm to check if the data containing the expected keywords and returns the matched data to the user.

To date, PEKS make considerable progress in strengthening security, enriching functionality, and facilitating efficiency [6–8]. Nevertheless, most of these PEKS constructions are built under the traditional hardness assumptions like discrete logarithm problem. As pointed out in [9], the quantum computers of the near future hold a high potential to invalidate these schemes. More urgently, researchers have made breakthrough progress in quantum computers [10]. Most recently, some initial efforts have been made from lattice [11–14], opening a direction to designing quantum secure encrypted search schemes. Following this way, in this paper, we will focus on the investigation of quantum secure encrypted search schemes and develop a new PEKS scheme with better performance in security and efficiency.

1.1 Our Motivation

The first problem we aim to address is the integrity of the search result. Note that, the real-world cloud server is always semi-honest-but-curious [15], it may intentionally return partial search or even unmatched results to the user for saving its cost. This is not a new problem in traditional PEKS, many researchers propose to use verifiable PEKS to solve this problem. Specifically, a verification block is adopted and will be sent to the user, together with the search result. The user can use this block to verify if the search result is integral. However, to the best of our knowledge, few lattice-based verifiable searchable encryption schemes have been known before. For this concern, our first effort is to design a verifiable PEKS from lattice. Our key idea is to design a verification algorithm, which can be imposed on existing lattice-based PEKS schemes directly. To achieve this goal, we introduce a lattice-based delegation algorithm, which can flexibly generate the verification block under the circumstance of semi-honest-but-curious server.

Based on the above result, our second problem is how to deploy the above verifiable PEKS scheme to an IoT system. Observe that, in a public-key cryptosystem, before encrypting the data, a user always needs to verify the authenticity of a public-key. However, the public key in a lattice-based encryption scheme is usually made up of some high-dimension matrices, transmitting these matrices through source-limited sensors is infeasible. Inspired by Zhang et al.'s ideal [13], we deal with this problem by introducing the identity-based encryption mechanism, where a small size identity-tag is used to denote the public key of the user. In particular, when a user enrolls, he registers the system with his individual identity and gets a credential from the central authority as his secret key. After that, he uses this identity and corresponding secret key to perform encryption and verification operations.

1.2 Our Contributions

In light of the above observations, our contributions can be summarized as follows:

(1) We propose the first verifiable PEKS scheme from lattice to check the result integrity of a quantum secure search scheme.
(2) We introduce the philosophy of identity-based encryption mechanism and use it to mitigate the heavy communication cost in public key authentication.
(3) We compare our scheme with some closely-related works, and the results demonstrate that our scheme performs better than prior arts in efficiency.

Organizations. The rest of this paper is organized as follows. In Section 2, we introduce some related work. The proposed system overview, corresponding threat model and design goals are presented in Section 3. In Section 4, we first introduce preliminaries, including the lattice-based knowledge, the hardness problem and some algorithms. Then we define the VIBKS scheme with the security model. In Section 5, we show the construction of our lattice-based VIBKS scheme, as well as its correctness and verifiability proof. In Section 6, we provide the security proof of our scheme. The theoretical comparison and performance analysis are presented in Section 7. Finally, we give a short conclusion in Section 8.

2 Related Work

With the rapid increase of data volume from quantities of IoT devices, cloud storage technology plays an increasingly crucial role in IoT. In terms of data confidentiality, it becomes prevalent to use cloud storage data encryption services. To achieve the goal of searching over encrypted data outsourced to the cloud server without leaking information about the messages shared by the data sender, searchable encryption (SE) is one of the essential approaches. SE is divided into symmetry searchable encryption and asymmetric searchable encryption. The SSE scheme was firstly introduced in [4]. Although the SSE scheme is with relatively high computational efficiency, it is only suitable for a single user model, but could not be well applied in multi-user model.

PEKS based on conventional hardness assumption. The first PEKS scheme was proposed to enable one to search over encrypted data generated by public key system [5]. Following with Boneh et al.'s work [5], a multitude of PEKS schemes that achieve various kinds of functionality are proposed [6–8]. Park et al. [6] considered the need for conjunctive keyword query and proposed the first public key encryption with conjunctive field keyword search scheme. To fill the gap in the area of fuzzy search on encrypted data, a PEKS scheme supporting fuzzy keyword query was proposed by Xu et al. [7]. Besides, schemes support proxy re-encryption with keyword search

under the assistance of the proxy were proposed in [8]. However, the mentioned schemes above are all fragile to the attack from quantum computers.

PEKS based on lattice assumption. Lattice has been widely used to construct cryptosystems that can resist the near future quantum computer. The first PEKS built for quantum security is proposed by Gu et al. [11], whose security can be reduced to LWE assumption in the standard model. To further pursue the practicability, several lattice-based works have been proposed to support fine-grained access control [12], proxy-oriented search [13], conjunctive keyword search [14], verifiable search [15]. In addition to extending the functionality, researches on resisting varieties of attack methods have been sufficiently developed. In order to ensure the security when the scheme suffers from key-exposure problems, Zhang et al. [16] constructed a scheme satisfying the forward security in the quantum security level. Besides, keyword guessing attack is practical in PEKS as the keywords in real-life are with inherently low entropy. To prevent inside KGA, Yu et al. [17] employed the preimage sample algorithm on a random bit and a part of ciphertext, which prevents the malicious server from forging a valid ciphertext.

Verifiable searchable encryption. The verifiable search has been substantially developed in the plaintext retrieval using special data structures (e.g., signature and merkle tree) in case a malicious cloud server returns false results or conceals data loss incidents. For example, in 2010, Benabbas et al. [18] presented the first practical verifiable computation scheme for high degree polynomial functions. In the encrypted data search scenario, Zheng et al. [19] utilized bloom filter and signature algorithm to verify whether the cloud has faithfully executed the search operations, and proposed two different schemes, i.e., KP-ABKS and CP-ABKS. Sun et al. [15] proposed a scheme to enable authenticity check over the returned search result in the multi-user multi-data-contributor search scenario. Miao et al. [20] constructed the Verifiable SE Framework (VSEF), which can withstand the inside KGA and achieve verifiable searchability. Wu et al. [21] presented a new authenticated data structure based on homomorphic encryption, and shows how to apply it to verify the correctness and completeness of search results. However, the verification proof in their scheme is generated by the cloud server, who can forge the proof to pass the verification when it is viewed as an adversary. In summary, the above verifiable searchable encryption schemes seldom consider how to ensure the integrity of returned data when the cloud server is semi-honest-but-curious and equipmented with quantum computers.

3 Problem Statement

In this section, we will first give a high-level description of our proposed scheme and introduce the role of each participant involved in. Then, to explicitly show the capability of each party, we define the threat model.

3.1 System Overview

As shown in Fig. 1, we illustrate the architecture of the proposed VIBKS in an IoT system. The designed scheme is consisted of four entities: central authority (CA), cloud server (CS), IoT sensor (IoTs) and data receiver (DR).

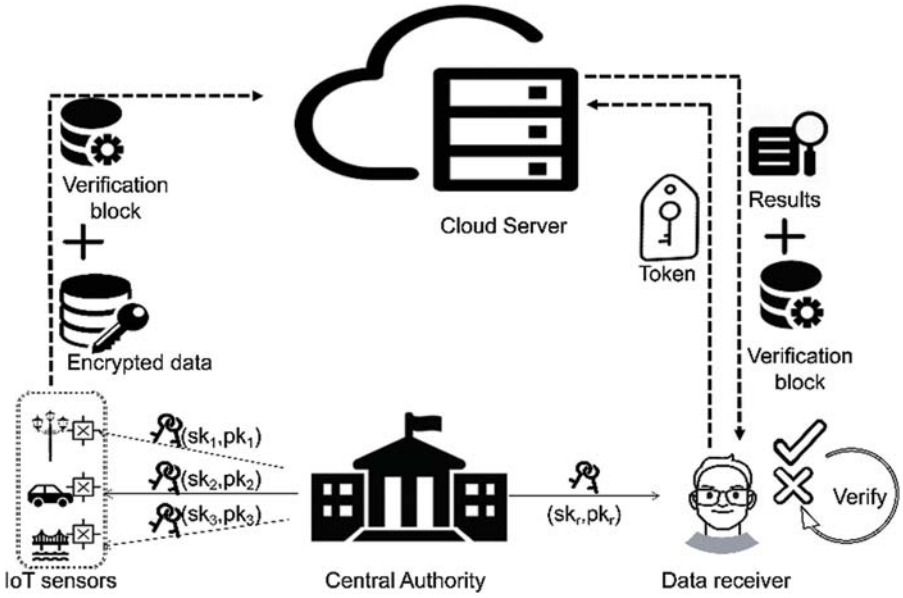

Figure 1: System overview

- CA: CA is assumed full responsibility for key generation and public parameters generation.
- IoTs: IoTs is the sensor deployed in an IoT system, who holds the duty that transmits data to CS over the Internet. Specifically, the transmitted data requires to be encrypted before sent to CS. To further pursue the goal of data verification, a verification block is attached.
- DR: DR is authorized by IoTs, it holds the ability of keyword retrieval and result verification. The former is realized by submitting search token, while the latter is due to the verification block generated by IoTs.
- CS: CS is mainly responsible for two aspects of work in the system, the first is to store the encrypted data with keyword ciphertexts and verification block, the second is to perform test operation.

Overview. Here we elaborate on the implementation process of the proposed scheme with Fig. 1. First of all, the CA generates the system public parameters and the master secret key, with which it generates the corresponding identity secret key according to the tag submitted by other entities later. After obtaining tags of IoTs and DR, CA sends the public and secret key pairs to each entity. IoTs generates the keyword ciphertexts to allow the DR visit its data and compute the verification block for integrity check, which are sent with the encrypted data to CS. When DR wants to obtain ciphertexts corresponding to a keyword, the secret key will be used to create a keyword-oriented token, which is transmitted to the CS subsequently. Then, by testing whether the token and keyword ciphertexts match, the corresponding results (including encrypted data, data identity sets and verification blocks) are returned. Finally, DR decrypts the encrypted data and detects the correctness and integrity of results.

3.2 Threat Model

First of all, CA is supposed to be completely trusted in our model, who knows the secret key of the whole system but never launch an attack on the system. Then, we primarily stress that the server in our model behaves entirely in a "semi-honest-but-curious" fashion, as described in

some previous literature. We assume that the server will not honestly follow the algorithm we set up and possibly returns a fraction of results or unrelated files. Lastly, in our model, IoTs with plaintexts and DR with search capability are both considered reliable, which means they are not considered the outside adversary.

To prove the security of VIBEKS under the above threat model, we follow the security definition of ciphertext indistinguishability. Details will be provided in Section 4.2.

3.3 Design Goals

In order to achieve secure and efficient searchable encryption over IoT data, we design our scheme to accomplish the following objects:

(a) **Efficiency**. It may involve some operations with high computational cost when designing a lattice-based SE scheme, e.g., matrix inverse operation. In order to gain higher efficiency, considering constructing the scheme in which the matrix inverse operation need to be performed only once.
(b) **Security**. Our scheme should achieve ciphertext indistinguishability under selective-ID attacks. The quantum computer cannot violate the privacy of outsourced data as it does for some SE schemes based on traditional hardness problems. Lattice-based tools are deployed in our scheme and the security of our scheme is based on the LWE problems, which has been proved to be secure under the attack from quantum computer.
(c) **Functionality**. Our scheme also aims to enable verifiable search on encrypted outsourced data. Verifiable search is more practical than the common encrypted search, since quantities of deceptions occur every day. The lattice-based delegation algorithm applied in the proposed scheme makes sure that any cheated behavior of server can be recognized by the user.

4 Preliminaries

4.1 Lattice and Hardness Problem

Now, we first give some notations applied in our scheme and definitions about lattice-based cryptography as follows.

Notation $\mathbb{Z}$ denotes the set of integers. The bold font denotes vector. $\|\widetilde{A}\|$ denotes the Gram-Schmidt norm of matrix A, where $\widetilde{A}$ is the Gram-Schmidt orthogonalization of A.

Definition 4.1 Given two random variables x and y on a countable set S, there is a defined function $\Delta(x,y) = \frac{1}{2}\sum_{s\in S}|Pr[x=s] - Pr[y=s]|$ as statistical distance. If the distance $d(\lambda) = \Delta(x(\lambda), y(\lambda))$ is negligible in λ, we say x and y are statistically close.

Definition 4.2 Let $\boldsymbol{b}_i \in R^m$ for $i \in \{1,\ldots,n\}$ be some linearly independent vectors and $B = \{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n\}$. The m-dimensional lattice Λ generated by the above vectors is $\Lambda(B) = \{\Sigma_{i=1}^{n} x_i\boldsymbol{b}_i \mid x_i \in \mathbb{Z}\}$. Here m is the dimension and n is the rank. When $n=m$, the lattice is full-rank. The basis of the lattice Λ is $\{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n\}$.

Definition 4.3 [8] Given a matrix $A \in \mathbb{Z}_q^{n\times m}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a prime q, define:

(1) $\Lambda_q^{\perp}(A) := \{\boldsymbol{e} \in \mathbb{Z}^m \mid A\boldsymbol{e} = \boldsymbol{0} \ mod\ q\}$,
(2) $\Lambda_q^{\boldsymbol{u}}(A) := \{\boldsymbol{e} \in \mathbb{Z}^m \mid A\boldsymbol{e} = \boldsymbol{u} \ mod\ q\}$,
(3) $\Lambda_q(A) := \{\boldsymbol{\mu} \in \mathbb{Z}_q^m \mid \exists\ \boldsymbol{s} \in \mathbb{Z}_q^n, \boldsymbol{\mu} = A^T\boldsymbol{s} \ mod\ q\}$.

The ciphertext indistinguishability of VIBKS can be reduced to the learning with errors (LWE) problem. We introduce the LWE problem as follows.

Definition 4.4 [22] Given a prime q, a positive integer n, and a Guassian noise distribution χ over $\mathbb{Z}_q$, for a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, the LWE problem is to distinguish the distribution between the following two distributions:

(1) LWE distribution: samples are in the form of $(\boldsymbol{v}, u) = (\boldsymbol{v}, \boldsymbol{v}^{\mathrm{T}}\boldsymbol{s} + e)$, where $\boldsymbol{v}$ is a random vector chosen from $\mathbb{Z}_q^n$, e is a sample drawn from distribution χ.
(2) Uniform distribution: samples are uniformly chosen from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Then, we introduce the TrapGen algorithm and preimage sampleable algorithm SamplePre, which are utilized in **Setup**, **Encrypt** and **TokGen** of the proposed scheme.

Definition 4.5 [23] The probabilistic polynomial-time (PPT) algorithm TrapGen(q,n) generates a uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$ together with a short basis $T_A \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}(A)$, satisfying $\|\widetilde{T}_A\| \le O\left(\sqrt{n\log q}\right)$, and $\|T_A\| \le O\left(\sqrt{n\log q}\right)$ with all but negligible probability in n.

Definition 4.6 [24] Given a matrix $A \in \mathbb{Z}_q^{n\times m}$ with basis $T_A \in \mathbb{Z}_q^{m\times m}$ of $\Lambda_q^{\perp}(A)$, a vector $\boldsymbol{\mu} \in \mathbb{Z}_q^n$, $m \ge 2n\lceil\log q\rceil$, and a parameter $\sigma \ge \|\widetilde{T}_A\| \cdot \omega\left(\sqrt{\log m}\right)$ as inputs, the PPT algorithm SamplePre$(A, T_A, \boldsymbol{\mu}, \sigma)$ generates a vector $\boldsymbol{t} \in \mathbb{Z}_q^m$ sampled from a distribution statistically close to $D_{\Lambda_{q(\mathrm{A})}^{\boldsymbol{\mu}},\sigma}$, such that $A\boldsymbol{t} = \boldsymbol{\mu} \bmod q$.

Then we give the definition of the lattice basis delegation BasisDel [25], which is used in the private key derivation in our VIBKS scheme.

Definition 4.7 [25] Given a matrix $A \in \mathbb{Z}_q^{n\times m}$ with a short basis $T_A \in \mathbb{Z}_q^{m\times m}$ of $\Lambda_q^{\perp}(A)$, a $\mathbb{Z}_q$-invertible matrix R sampled from $D_{m\times m}$, and a parameter $\delta \ge \|\widetilde{T}_A\| \cdot \delta_R\sqrt{m}\omega\left(\log^{3/2} m\right)$ as inputs, where $\delta_R = \sqrt{n\log q} \cdot \omega\left(\sqrt{\log m}\right)$, the PPT algorithm BasisDel(A, T_A, R, δ) generates a random short lattice basis T_B of $\Lambda_q^{\perp}(B)$, where $B = AR^{-1}$.

Finally, to prove the ciphertext indistinguishability of our scheme, we introduce a PPT algorithm SampleR, referring to [24], to sample matrices from a distribution statistically close to $D_{m\times m}$ over $\mathbb{Z}_q^{m\times m}$. Moreover, we employ a PPT algorithm SampleRwithBasis to simulate a random short lattice basis, which is defined as follows.

Definition 4.8 [25] Given a prime $q > 2$, $m \ge 2n\lceil\log q\rceil$, a matrix $A \in \mathbb{Z}_q^{n\times m}$ as inputs, the PPT algorithm SampleRwithBasis(A) generates a low-form matrix R which is statistically close to $D_{m\times m}$ over $\mathbb{Z}_q^{m\times m}$, and a random short lattice basis T_B of $\Lambda_q^{\perp}(B)$, where $B = AR^{-1}$, such that $\|\widetilde{T}_B\| \le \delta_R/\omega\left(\sqrt{\log m}\right)$ with an overwhelming probability.

4.2 VIBKS and Security Definition

A formal definition of VIBKS is given in the following part.

Definition 4.9 VIBKS consists of the following six algorithms.

(1) **Setup** (1^n): This probabilistic algorithm is run by the CA to setup the system. It inputs a security parameter n and returns the public parameter PP, the master secret key MSK.

(2) **KeyDerive** (PP, MSK, τ): This probabilistic algorithm is run by the CA to derive keys for users. It inputs the public parameter SP, the master secret key MSK and the user's tag τ, and returns the private key sk_τ and public key pk_τ for the user.
(3) **Encrypt** (PP, τ, w, F, KS): This probabilistic algorithm is run by the IoTs to generate the keyword ciphertexts tuple set and the verification tuple set. It inputs the public parameter PP, the tag τ of data receiver, the keyword w, the file set F and the keyword set KS, and returns the keyword ciphertext tuple set CT and verification tuple set VT.
(4) **TokGen** (PP, τ, sk_τ, w*): This probabilistic algorithm is run by the receiver whenever he wants to search for some files containing the keyword w*. It inputs the keyword w*, the user's private key sk_τ and the public parameter PP, and returns the search token ST_{w*}.
(5) **Test** (CT, ST_{w^*}): This algorithm is run by the cloud server to search for the matched files containing the keyword w^*. It inputs the search token ST_{w*} and the keyword ciphertext tuple set CT and verification tuple set VT, and returns the corresponding file set R with its verification block V, otherwise returns nothing.
(6) **Verify** (R, V): This algorithm is run by the data receiver to verify if the cloud server returns the correct and complete results.

The formal definition given above shows that our proposed VIBKS scheme provides the function of result verification, i.e., data receiver can check if the cloud server honestly performs the test operation and returns complete results. For security, we introduce a security game to define the security model of our proposed scheme.

Definition 4.10 Given the security parameter n, the game is carried out between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup: Take the security parameter n as input, $\mathcal{C}$ outputs the system public parameters PP and sends it to $\mathcal{A}$. $\mathcal{A}$ submits a target identity id_r^* to $\mathcal{C}$, and then it adaptively queries the following oracles.

KeyDerive oracle: Upon receiving the **KeyDerive** query on id from $\mathcal{A}$, where the only restriction is $id \neq id_r^*$, $\mathcal{C}$ generates the private key sk_{id} and returns it to $\mathcal{A}$.

TokGen oracle: Upon receiving the **TokGen** query on w from $\mathcal{A}$, $\mathcal{C}$ generates the search token st_w and returns it to $\mathcal{A}$.

Challenge phase: After finishing the above queries, $\mathcal{A}$ sends (id_r^*, w_0^*, w_1^*) to $\mathcal{C}$, where w_0^*, w_1^* are two different randomly chosen keywords that have never been queried in the above oracles, id_r^* is the target identity. $\mathcal{C}$ randomly chooses a bit $b \in \{0,1\}$, and returns a random searchable ciphertext tuple $C^* = (c_0^*, c_1^*)$ to $\mathcal{A}$.

$\mathcal{A}$ continues to query the **TokGen oracle** as above, the only restriction is that neither w_0^* nor w_1^* can be queried to obtain its search token.

Guess: Finally, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. It wins the game if and only if $b' = b$.

The probability of $\mathcal{A}$ winning the above game is defined as $\mathrm{Adv}_{\mathcal{A}}(n) = \left|\Pr\left[b' = b\right] - 1/2\right|$.

Definition 4.11 We say that our VIBKS scheme achieves ciphertext indistinguishability under the selective-ID attack if $\mathrm{Adv}_{\mathcal{A}}(n)$ is negligible for any PPT adversary $\mathcal{A}$.

5 Our Construction

As introduced in Section 3, our scheme is composed of six algorithms, which will be described one by one in this section. Note that, we adopt a symmetric encryption scheme $SE = (KeyGen, Enc, Dec)$ to encrypt data files.

5.1 Intuition Behind Our Construction

After we noticed that the existing lattice-based SE schemes only consider the server is honest-but-curious, we began to think about efficient lattice-based SE scheme that can thwart the server perform lazy search (i.e., search only a fraction of data or fooling users with previous search results) or return unfaithful result. Bloom filter can achieve this, but we want to avoid its false positive rate. We notice that we can solve this by adopting signature technique. The idea is to combine an efficient hash function with a lattice-based delegation algorithm to make sure that the verification result is deterministic, which implies that the user can check the completeness and correctness with 100% accuracy.

5.2 Construction

Now we describe the construction details of our lattice-based VIBKS scheme. It is composed of the following six polynomial-time algorithms.

Setup: Taking as input a security parameter n, CA generates the system public parameters $\mathrm{q}, \mathrm{m}, \sigma, \alpha, \mathrm{s}$, where q is prime, s is the gaussian parameter. Then CA performs the following steps:

(1) Set two secure hash function $H_1\colon \{0,1\}^{l_1} \to \mathbb{Z}_q^{m\times m}$, $H_2\colon \{0,1\}^{l_2} \to \mathbb{Z}_q^n$, $H_3\colon \{0,1\}^* \to \mathbb{Z}_q^n$, where the outputs of H_1 are distributed in $D_{m\times m}$.
(2) Invoke $TrapGen(q,n)$ to generate a uniformly random matrix $A_0 \in \mathbb{Z}_q^{n\times m}$ together with a short basis $T_{A_0} \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}(A_0)$.
(3) Select a uniformly random vector $\boldsymbol{t} \in \mathbb{Z}_q^n$.

Finally, CA outputs the system public parameters PP and master secret key MSK given by $\mathrm{MSK} = \mathbf{T}_{A_0}$, $\mathrm{PP} = (q, m, n, l_1, l_2, \sigma, \alpha, s, \boldsymbol{t}, A_0, H_1, H_2, H_3)$.

KeyDerive: Taking as inputs the system public parameters PP, the master secret key MSK and a tag $\tau \in \{0,1\}^{l_1}$, CA generates the public and secret key pair for the tag-belonged IoTs as follows:

(1) Compute the tag-matrix $R_\tau = H_1(\tau) \in \mathbb{Z}_q^{m\times m}$.
(2) Run $sk_\tau \leftarrow BasisDel\left(A_0, R_\tau, T_{A_0}, s\right)$ to obtain a short random basis for $\Lambda_q^{\perp}(pk_\tau)$, where $pk_\tau = A_0 {R_\tau}^{-1}$.

Finally, CA sends the private key sk_τ and the public key pk_τ to the IoTs whose tag is τ.

Encrypt: Taking as inputs the system public parameters PP, the file set $F = \{F_1, \ldots, F_h\}$, the keyword set $KS = \{w_1, \ldots, w_u\}$, the IoTs's key pair (sk_o, pk_o), the receiver's tag $\tau \in \{0,1\}^{l_1}$, the IoTs generates the keyword ciphertext tuple and the verification tuple as follows:

(1) For each file F_i in F, generate the symmetric key $sk = SE \cdot KeyGen(1^n)$ and encrypt the file with it to get the ciphertext $C_i = SE \cdot Enc(sk, F_i)$.

(2) For each keyword $w_{ij} \in \{0,1\}^{l_2}$ in file F_i, first compute the tag-matrix $R_\tau = H_1(\tau) \in \mathbb{Z}_q^{m \times m}$ and set $F_\tau = A_0 {R_\tau}^{-1} \in \mathbb{Z}_q^{n \times m}$, then compute the keyword-vector $G_{w_{ij}} = \mathbf{t} + H_2\left(w_{ij}\right) \in \mathbb{Z}_q^n$, finally compute the keyword ciphertexts $c_{ij} = \left(c_{ij}^0 = {G_{w_{ij}}}^{\mathrm{T}}\mathbf{r} + x, c_{ij}^1 = {F_\tau}^{\mathrm{T}}\mathbf{r} + \mathbf{y}\right)$, where $\mathbf{r}$ is a uniformly random vector from $\mathbb{Z}_q^n$, $x \in \mathbb{Z}_q$ and $\mathbf{y} \in \mathbb{Z}_q^m$ are noise vectors chosen from $\overline{\Psi}_\alpha$ and $\overline{\Psi}_\alpha^m$ respectively.

(3) Let $CT_i = \left(C_i, \left\{c_{ij} \middle| w_{ij} \in F_i\right\}\right)$ be the keyword ciphertext tuple of F_i.

(4) For each keyword $w_k \in KS$, first generate the keyword ciphertext c_k as in step 2, then compute $h_{w_k} = H_3\left(id_1 \| \ldots \| id_z \| w_k\right) \in \mathbb{Z}_q^n$, where $\{id_1, \ldots, id_z\}$ is the identity set of files containing the keyword w_k, finally evaluate the verification block $V_k \leftarrow SamplePre\left(pk_s, sk_s, \mathbf{h}_{w_k}, \sigma\right)$. Note that $V_k \cdot pk_s = \mathbf{h}_{w_k} \in \mathbb{Z}_q^n$, and the verification block is distributed in $D_{\Lambda_{q(pk_s)}^{\mathbf{hw}_k}, \sigma}$.

(5) Let $VT_k = (c_k, V_k)$ be the verification tuple of keyword w_k.

Finally, the IoTs outsources the keyword ciphertext tuple set $CT = \{CT_i\}\,(i \in \{1, \ldots, h\})$ and the verification tuple set $VT = \{VT_k\}\,(k \in \{1, \ldots, u\})$ to the cloud server.

TokGen: Taking as inputs the system public parameters PP, the receiver's tag $\tau \in \{0,1\}^{l_1}$, the keyword $w^* \in \{0,1\}^{l_2}$ and the secret key $sk_\tau \in \mathbb{Z}_q^{m \times m}$, the data receiver computes the search token for searching as follows:

(1) Compute the tag-matrix $R_\tau = H_1(\tau) \in \mathbb{Z}_q^{m \times m}$ and set $F_\tau = A_0 {R_\tau}^{-1} \in \mathbb{Z}_q^{n \times m}$.

(2) Compute the keyword-vector $G_{w^*} = \mathbf{t} + H_2(w^*) \in \mathbb{Z}_q^n$.

(3) Set $st_{w^*} \leftarrow SamplePre\left(F_\tau, sk_\tau, G_{w^*}, \sigma\right)$. Note that $F_\tau st_{w^*} = G_{w^*} \in \mathbb{Z}_q^n$, and the search token is distributed in $D_{\Lambda_{q(F(\tau))}^{G(w^*)}, \sigma}$.

The data receiver sends the search token st_{w^*} to the cloud server.

Test: Taking as inputs the keyword ciphertext tuple set CT, the verification tuple set VT and the search token st_{w^*}, the cloud server searches for the matched files and the verification block as follows:

(1) For each keyword ciphertext tuple $CT_i \in CT$, check if the file F_i contains the same keyword as st_{w^*}. To achieve this goal, for all $c_{ij} \in CT_i$, compute $\gamma_{ij} = c_{ij}^0 - st_{w^*}^{\mathrm{T}} c_{ij}^1$, if there exist γ_{ij}, such that $|\gamma_{ij}| \leq \lfloor q/4 \rfloor$, add the corresponding encrypted file C_i to the result set R. Otherwise, the cloud server aborts.

(2) For each verification tuple $VT_k \in VT$, compute $\gamma_k = c_k^0 - st_{w^*}^{\mathrm{T}} c_k^1$, if $|\gamma_k| \leq \lfloor q/4 \rfloor$, return the verification block V_k. Otherwise, the cloud server aborts.

The cloud server returns the result set R and the verification block V_k to the data receiver.

Verify: Taking as inputs the result set R and the verification block V_k, the data receiver verifies if the cloud server returns the integral results as follows:

(1) Decrypt the returning ciphertexts $F_i = SE.Dec\left(sk, C_i\right)$.

(2) Compute $\mathbf{h}_w = H_3\left(id_1 \| \ldots \| id_h \| w^*\right) \in \mathbb{Z}_q^n$, where w^* is the queried keyword, $\{id_1, \ldots, id_h\}$ is the identity set of the returned files.

(3) Check whether the equation $pk_o V_k = \mathbf{h}_{w^*}$ holds, and whether V_k is distributed in $D_{\Lambda^{\mathbf{h}_w}_{q(pk_o)},\sigma}$. If they hold, it implies that the cloud server has faithfully returned all results. Otherwise, the cloud server behaviors dishonestly.

5.3 Correctness of VIBKS

Let (sk_τ, pk_τ) be the data receiver's key pair, w be the keyword contained in CT_w and w^* be that in st_{w^*}. In *Test*, take the token st_{w^*} as the input, the cloud server can compute γ as follows:

$$\gamma = c_0 - st_{w^*}^{\mathrm{T}} c_1 = {G_w}^{\mathrm{T}}\mathbf{r} + x - st_{w^*}^{\mathrm{T}}\left({F_\tau}^{\mathrm{T}}\mathbf{r} + \mathbf{y}\right) = {G_w}^{\mathrm{T}}\mathbf{r} + x - {G_{w^*}}^{\mathrm{T}}\mathbf{r} - st_{w^*}^{\mathrm{T}}\mathbf{y}.$$

When $w = w^*$, $\gamma = x - st_w^{\mathrm{T}}\mathbf{y}$, as discussed in [23], $|\gamma| \le \lfloor q/4 \rfloor$. When $w \neq w^*$, as $\gamma \neq x - st_w^{\mathrm{T}}\mathbf{y}$, $|\gamma| > \lfloor q/4 \rfloor$, thus we conclude that the ciphertext CT_w and the token contain different keyword.

5.4 Verifiability of VIBKS

Let $\{id_1, \ldots, id_h\}$ be the identity set of decrypted files, V be the verification block, pk_o be the public key of data owner. In *Verify*, the data receiver computes $\mathbf{h}_w = H_3(id_1 \| \ldots \| id_h \| w^*)$ and $pk_o V_k$. When the identity set of decrypted files is the same as the file identity set ID_w that the data owner embeds in the verification block, the equation $pk_o V_k = \mathbf{h}_{w^*}$ holds from the definition of algorithm SamplePre. Thus, we come to the conclusion that the cloud server has returned the correct and complete results. Otherwise, we consider that the cloud server performs unfaithfully.

6 Security

In this section, we will give the security proof of our proposed scheme.

Theorem 6.1 VIBKS from lattice achieves ciphertext indistinguishability under the selective-ID attack in the random oracle model, provided that the hardness assumption of decisional-LWE problem holds.

Proof. Let $\mathcal{A}$ be an adversary, which breaks the ciphertext indistinguishability under the selective-ID attack with a non-negligible probability ε in the random oracle model. We construct a challenger $\mathcal{C}$ with a non-negligible probability solving the hardness assumption of LWE problem.

Setup: $\mathcal{C}$ requests from LWE oracle $\mathcal{O}$ for $m+1$ times to obtain fresh pair $(\mathbf{u}_k, v_k) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $k = 0, 1, \ldots, m$. Then sets two lists L_1, L_2, L_3, and let q_{H_i} be the maximum number of $\mathcal{A}$'s queries to H_i, where $i = 1, 2, 3$. Finally, $\mathcal{C}$ prepares the system public parameters PP for $\mathcal{A}$ as follows.

(1) $\mathcal{C}$ samples a random matrix $R^* \leftarrow \mathcal{D}_{m\times m}$ by running SampleR.
(2) $\mathcal{C}$ constructs a random matrix $F^* \leftarrow \mathbb{Z}_q^{n\times m}$ from m of the given LWE samples, where the k-th column of F^* be the vector $\boldsymbol{u}_k \in \mathbb{Z}_q^n$ for all $k = 1, \ldots, m$.
(3) $\mathcal{C}$ sets $A_0 = F^* R^*$, where A_0 is uniform in $\mathbb{Z}_q^{n\times m}$ since $R^* \in \mathbb{Z}_q^{m\times m}$ are invertible modulo q and F^* is uniform in $\mathbb{Z}_q^{n\times m}$. $\mathcal{C}$ also let $\boldsymbol{t} = \boldsymbol{u}_0$, and finally sets $\mathrm{PP} = (A_0, \boldsymbol{t}, H_1, H_2, H_3)$ and sends it to $\mathcal{A}$. Ahead of time, the target identity is id_r^* submitted by $\mathcal{A}$.

Note that, regardless when $\mathcal{A}$ performs the **KeyDrive oracle** query, it has made all relevant H_1 queries before. Now, $\mathcal{A}$ performs the following queries.

H_1 **query**: For the l-th query, where $l = 1, 2, \ldots, q_{H_1}$, $\mathcal{A}$ queries to H_1 on any identity id adaptively. If $l = I_1^*$, such that $id = id_r^*$, $\mathcal{C}$ sets $H_1(id) = R^*$ and returns it to $\mathcal{A}$. Otherwise, $\mathcal{C}$ runs

SampleRwithBasis(A_0) to obtain a random matrix $R_{id} \in \mathbb{Z}_q^{m\times m}$ and a short random lattice basis $T_{id} \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}\left(A_0 R_{id}^{-1}\right)$. Then $\mathcal{C}$ adds $(id, A_0 R_{id}^{-1}, R_{id}, T_{id})$ to list L_1, and returns R_{id} to $\mathcal{A}$.

H_2 **query**: For the l-th query, where $l = 1, 2, \ldots, q_{H_2}$, $\mathcal{A}$ queries to H_2 on any keyword w adaptively. If $l = q_{H_2}$, such that $w = w^*$, $\mathcal{C}$ sets $H_2(w) = \boldsymbol{t}_0 - \boldsymbol{t} = \boldsymbol{t}_0 - \boldsymbol{u}_0$ and adds it to list L_2, then returns it to $\mathcal{A}$. Otherwise, $\mathcal{C}$ sets $H_2(w) = \boldsymbol{t}_0$, where $\boldsymbol{t}_0 \leftarrow \mathbb{Z}_q^n$ and adds it to list L_2, then returns it to $\mathcal{A}$.

KeyDerive oracle: Upon receiving the **KeyDerive** query on id from $\mathcal{A}$, where the only restriction is $id \neq id_r^*$, $\mathcal{C}$ checks the list L_1 to find $(id, A_0 R_{id}^{-1}, R_{id}, T_{id})$, if it is found, $\mathcal{C}$ returns T_{id} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ performs the same operation as H_1 query on id, obtains $(id, A_0 R_{id}^{-1}, R_{id}, T_{id})$ and returns T_{id} to $\mathcal{A}$.

TokenGen oracle: Upon receiving the **TokenGen** query on w from $\mathcal{A}$, where the only restriction is $id \neq id_r^*$, $\mathcal{C}$ finds $(id, A_0 R_{id}^{-1}, R_{id}, T_{id})$ in list L_1 and $\boldsymbol{t}_0$ in list L_2, it they are found, $\mathcal{C}$ computes $G_w = \mathbf{t} + \mathbf{t}_0$ and runs SamplePre$(A_0 R_{id}^{-1}, T_{id}, G_w, \sigma)$ to obtain st_w. Finally, $\mathcal{C}$ returns st_w to $\mathcal{A}$.

Challenge phase: $\mathcal{A}$ sends (id_r^*, w_0^*, w_1^*) to $\mathcal{C}$, where w_0^*, w_1^* are two different keywords that have never been queried before, id_r^* is the challenge identity. $\mathcal{C}$ randomly chooses a bit $b \in \{0, 1\}$, if $b = 0$, $\mathcal{C}$ returns a random searchable ciphertext $C^* = (c_0^*, c_1^*)$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ performs as follows:

For each $k = 0, 1, \ldots, m$, retrieve $v_k \in \mathbb{Z}_q$ from LWE instance, set $v^* = (v_1, v_2, \ldots, v_m)^{\mathrm{T}} \in \mathbb{Z}_q^m$, and set $c_0^* = v_0, c_1^* = v^*$. Finally, $\mathcal{C}$ returns the searchable ciphertext $C^* = (c_0^*, c_1^*)$ to $\mathcal{A}$.

Guess: At last, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$.

The goal of $\mathcal{A}$ is to decide which keyword is contained in the challenged ciphertext C^*, since the challenger $\mathcal{C}$ returns the searchable ciphertext which is associated with each keyword with probability 1/2. Now, we evaluate the probability of $\mathcal{A}$ in breaking the ciphertext indistinguishability under the selective-ID attack in the random oracle model. With the system public parameters PP, we have $\boldsymbol{t} + H_2(w_0^*) = \boldsymbol{t} + \boldsymbol{t}_0 - \boldsymbol{t} = \boldsymbol{t}_0$, $A_0 R_{id}^{-1} = F^* R^* H_1(id)^{-1} = F^* R^* R^{*-1} = F^*$, and $c_0^* = v_0 = \boldsymbol{t}_0{}^{\mathrm{T}} \boldsymbol{r}^* + x^* c_1^* = v^* = F^{*\mathrm{T}} \boldsymbol{r}^* + \boldsymbol{y}^*$ for some random values $x^* \in \mathbb{Z}_q$ and vectors $\boldsymbol{r}^* \in \mathbb{Z}_q^n$ and $\boldsymbol{y}^* \in \mathbb{Z}_q^m$ with Gaussian distribution. Therefore, c_0^*, c_1^* have the correct ciphertext form. We assume that $\mathcal{A}$ can successfully guess the keyword w_0^* is associated with the ciphertext C^* with the probability ε, where the keyword w_0^* is the q_{H_2}-th H_2 query when $w_0^* = w^*$ with the probability $1/q_{H_2}$. When id_r^* is the I_1^*-th H_1 query, it occurs with the probability $1/q_{H_1}$. Thus, if $\mathcal{A}$ breaks the ciphertext indistinguishability under the selective-ID attack with a non-negligible probability ε in the random oracle model, $\mathcal{C}$ can solve the hardness assumption of decisional-LWE problem with the non-negligible probability $\varepsilon' = \varepsilon/(2q_{H_1}q_{H_2})$, which is contradictory. Consequently, $\mathrm{Adv}_{\mathcal{A}}(n)$ is negligible, and VIBKS achieves ciphertext indistinguishability under the selective-ID attack in the random oracle model.

7 Performance Analysis

7.1 Theoretical Analysis

In this section, we show a theoretical performance comparison of our proposed scheme and some other searchable encryption schemes [13,16,20]. First, we give the theoretical analysis of the computation overhead of keyword encryption, token generation, test operation, and verification process. Moreover, we describe the hardness problems on which the security models of each

scheme are based. Finally, with regard to the functionalities, we describe the verifiability and post-quantum security.

Now, we specify some notations used in the comparison below. In particular, m denotes the number of results, E denotes the modular exponentiation operation, h denotes the hash operation which maps the arbitrary string into G, P denotes the bilinear pairing operation, H denotes the hash operation which maps the arbitrary string into a matrix. I denotes the matrix inversion operation, M denotes the matrix multiplication operation, BD denotes the BasisDel, SP denotes the SamplePre. The performance comparison is listed in Tab. 1.

Table 1: Performance comparison

Schemes	[20]-Basic	[13]	[16]	Ours
Encrypt	$5E+2h$	$2H+I+3M+SP$	$H+3M+I$	$2H+I+5M+SP$
TokGen	$3E+h+P$	$H+I+M+SP+BD$	$H+M+I+SP+BD$	$2H+I+M+SP$
Test	2P	$l*M+H$	$l*M$	M
Verify	$3mE+2P$	–	–	$H+M$
Hardness	DL/CDH/DBDH	LWE/ISIS	LWE/ISIS	LWE
Verifiable	Yes	No	No	Yes
Post-quantum	No	Yes	Yes	Yes

7.2 Experiment Analysis

In this section, we conduct experiments of our proposed scheme in a laptop using Matlab R2017a. The environment is Windows 10 Ultimate (x64) with an Intel (R) Core (TM) i7-10750H CPU@2.60 GHz. We set the parameters $m \geq 6n\log q$. The security level of [13,16] is set to $l = 10$, as it utilized in their experiments. Under the above parameters, the time cost of SamplePre, hash operation and matrix multiplication are about 0.326574 s, 0.13221 s, and 0.028051 s respectively, which are far less than 183.067383 s required for matrix inverse operation. Therefore, we can only pay attention to the time cost of matrix inverse when considering the performance of algorithms that involves matrix inverse.

As shown in Tab. 1, although the theoretical computation cost of *Encrypt* in both [13,16] and our scheme contains matrix inverse operation, we found that the efficiency of our scheme is much higher than that of [13,16] in experiments. The advantage is due to that in our scheme, inverse matrix is only related to the user's identity. It indicates that the matrix inverse operation can be done before the *Encrypt* process and keep it locally for the next *Encrypt* process. In [13,16], the keyword-related matrix needs to be inversed, so that the matrix inverse operation must be conducted once the keyword updates, which brings huge computation overhead. For the same reason, the efficiency of *TokGen* in our scheme is superior to that of [13,16]. Indeed, our scheme enjoys a greater advantage in *TokGen* because of the additional BasisDel overhead in [13,16].

Fig. 2 provides the comparison of test time with the increasing of the number of keywords. The left graph of Fig. 3 proves that our scheme and the scheme in [13] spend less time to complete the test no matter what the number of keywords is. The scheme in [20] involves time-consuming operation (i.e., bilinear pairing) when matching the token and encrypted keywords. We can observe that *Test* in [20] consumes around 9 times more computation time than that in [16] with the same number of keywords. Then, we consider the test time comparison of our scheme and [13,16],

which are all constructed based on lattice hardness assumptions. From the left figure of Fig. 2, we can also see that the test time of [16] and our scheme is almost the same, far less than that in [13]. In order to further explore the relationship of time cost in [16] and our scheme, we list them in the right figure of Fig. 2. When the number of keywords is 100, we observe that our scheme and [16] take about 0.126 ms and 0.333 ms, respectively. The results again confirm the superiority of our scheme.

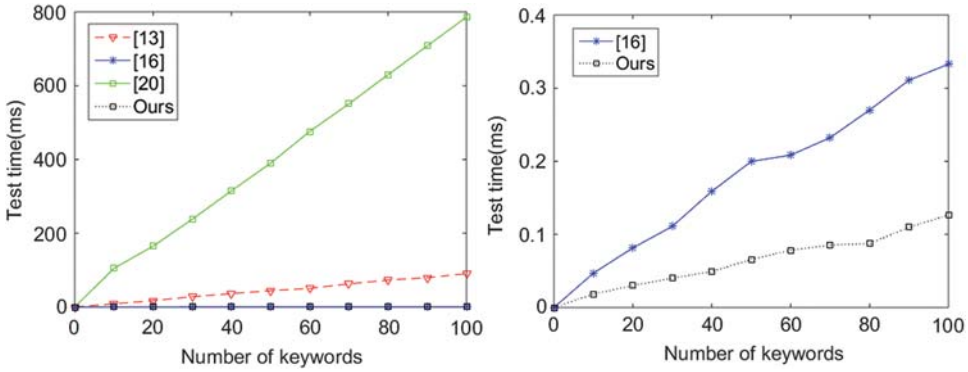

Figure 2: Comparison of *Test*

With respect to *Verify*, we implement the experiments to compare the verification time of our scheme and the basic scheme in [20]. As the basic scheme in [20] supports multi-keyword search, we set the number of keywords in one query to 1 to facilitate comparison. Furthermore, from Tab. 1, we can see that the performance of [20] in *Verify* depends on the number of results. In Fig. 3, we set the number of results to 1, which is the optimal case for [20]. Under the above setting, we notice that the performance of our scheme is significantly better than [20] as the verification times increases. It costs only 0.106 s to finish 100 times verification in our scheme, while it costs 2.899 s in [20]. The reason is that the scheme in [20] adopts time-consuming operations such as bilinear pairing and modular exponentiation, while our proposed scheme only involves in efficient matrix multiplication and hash operations.

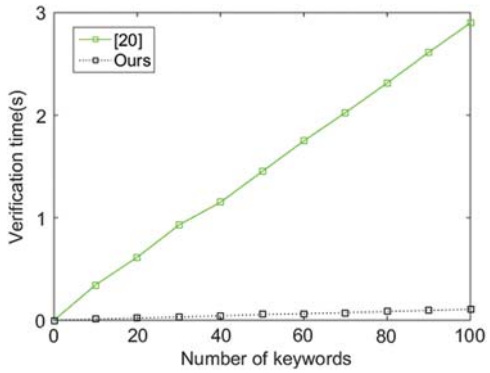

Figure 3: Comparison of *Verify*

8 Conclusion

Due to the booming of post-quantum cryptography, in this paper, we have constructed a lattice-based identity-based encryption with keyword search which can protect against quantum computer attacks. This paper proposes a lattice-based VIBKS scheme that can detect the misbehavior of cloud server and resist quantum computing attack. In the scheme, we use the preimage sample algorithm to generate a verification block for keyword ciphertext, which can be utilized by users to verify if the server returns correct and complete results. Then, we prove that our proposed scheme is secure under the selective-ID attack, and achieves both correctness and verifiability. Finally, we give theoretical and experimental comparisons with other searchable encryption schemes. The results demonstrate that our scheme achieves a better balance in efficiency and security. As future works, we consider the construction of lattice-based PEKS schemes with rich query functions.

Funding Statement: This work is supported by the National Natural Science Foundation of China (No: 62072240) and the National Key Research and Development Program of China (No. 2020YFB1804604).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Ashton, "That internet of things thing," *RFID Journal*, vol. 22, no. 7, pp. 97–114, 2009.
[2] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
[3] L. Xu, C. G. Xu, Z. Y. Liu, Y. L. Wang and J. F. Wang, "Enabling comparable search over encrypted data for IoT with privacy-preserving," *Computers, Materials & Continua*, vol. 60, no. 2, pp. 675–690, 2019.
[4] D. X. Song, D. A. Wagner and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. S&P*, Berkeley, California, USA, pp. 44–55, 2000.
[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, Interlaken, Switzerland, pp. 506–522, 2004.
[6] D. J. Park, K. Kim and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proc. WISA*, Jeju Island, Korea, pp. 73–86, 2004.
[7] P. Xu, H. Jin, Q. H. Wu and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.
[8] J. Shao, Z. F. Cao, X. H. Liang and H. Lin, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.
[9] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. FOCS*, Santa Fe, New Mexico, USA, pp. 124–134, 1994.
[10] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin *et al.,* "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
[11] C. X. Gu, Y. H. Zheng, F. Kang and D. Xin, "Keyword search over encrypted data in cloud computing from lattices in the standard model," in *Proc. Cloud Computing and Big Data*, Huangshan, China, pp. 335–343, 2015.
[12] L. Xu, X. L. Yuan, R. Steinfeld, C. Wang and C. G. Xu, "Multi-writer searchable encryption: An LWE-based realization and implementation," in *Proc. Asia CCS*, Auckland, New Zealand, pp. 122–133, 2019.

[13] X. J. Zhang, Y. Tang, H. X. Wang, C. X. Xu, Y. B. Miao *et al.,* "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Information Sciences*, vol. 494, no. 3, pp. 193–207, 2019.

[14] Y. J. Mao, X. B. Fu, C. Guo and G. H. Wu, "Public key encryption with conjunctive keyword search secure against keyword guessing attack from lattices," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 11, pp. 789, 2019.

[15] W. Sun, S. Yu, W. Lou, Y. T. Hou and H. Li, "Protecting your right: Verifiable attribute-based keyword search with fine-grained owner enforced search authorization in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1187–1198, 2016.

[16] X. J. Zhang, C. X. Xu, H. X. Wang, Y. Zhang and S. X. Wang, "FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Transactions on Dependable and Secure Computing*, 2019. https://doi.org/10.1109/TDSC.2019.2914117.

[17] X. L. Yu, C. G. Xu, L. Xu and Y. T. Wang, "Lattice-based searchable encryption scheme against inside keywords guessing attack," *Computers, Materials & Continua*, vol. 64, no. 2, pp. 1107–1125, 2020.

[18] S. Benabbas, R. Gennaro and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Proc. CRYPTO*, Santa Barbara, CA, USA, pp. 111–131, 2011.

[19] Q. J. Zheng, S. H. Xu and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. InfoCom.*, Toronto, Canada, pp. 522–530, 2014.

[20] Y. B. Miao, Q. Y. Tong, R. H. Deng, K. R. Choo, X. M. Liu *et al.,* "Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage," *IEEE Transactions on Cloud Computing*, 2020. https://doi.org/10.1109/TCC.2020.2989296.

[21] D. N. Wu, Q. Q. Gan and X. M. Wang, "Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting," *IEEE Access*, vol. 6, pp. 42445–42453, 2018.

[22] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, pp. 34:1–34:40, 2009.

[23] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," *Theory of Computing Systems*, vol. 48, no. 3, pp. 535–553, 2011.

[24] C. Gentry, C. Peikert and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. STOC*, Victoria, British Columbia, Canada, pp. 197–206, 2008.

[25] S. Agrawal, D. Boneh and X. Boyen, "Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE," in *Proc. CRYPTO*, Santa Barbara, CA, USA, pp. 98–115, 2010.