Lattice-Based Searchable Encryption Scheme against Inside Keywords Guessing Attack

Xiaoling Yu[1], Chungen Xu[1, *], Lei Xu[1] and Yuntao Wang[2]

Abstract: To save the local storage, users store the data on the cloud server who offers convenient internet services. To guarantee the data privacy, users encrypt the data before uploading them into the cloud server. Since encryption can reduce the data availability, public-key encryption with keyword search (PEKS) is developed to achieve the retrieval of the encrypted data without decrypting them. However, most PEKS schemes cannot resist quantum computing attack, because the corresponding hardness assumptions are some number theory problems that can be solved efficiently under quantum computers. Besides, the traditional PEKS schemes have an inherent security issue that they cannot resist inside keywords guessing attack (KGA). In this attack, a malicious server can guess the keywords encapsulated in the search token by computing the ciphertext of keywords exhaustively and performing the test between the token and the ciphertext of keywords. In the paper, we propose a lattice-based PEKS scheme that can resist quantum computing attacks. To resist inside KGA, this scheme adopts a lattice-based signature technique into the encryption of keywords to prevent the malicious server from forging a valid ciphertext. Finally, some simulation experiments are conducted to demonstrate the performance of the proposed scheme and some comparison results are further shown with respect to other searchable schemes.

Keywords: Searchable encryption, lattice assumption, inside keywords guessing attack, post-quantum secure.

1 Introduction

With the arrival of the big data era, most users use cloud services to store their data to save the local storage costs. However, this has caused a great concern about the security of outsourced data. Naturally, users apply the encryption technique to these data before uploading them to the cloud [Das, Baykara and Tuna (2019), Hayouni and Hamdi (2018)]. However, encryption reduces the readability and usability of data [Boneh, Crescenzo, Ostrovsky et al. (2004)]. Searchable encryption (SE) is a technique that can achieve

[1] School of Science, Nanjing University of Science and Technology, Nanjing, 210094, China.

[2] School of Information Science, Security and Networks, Japan Advanced Institute of Science and Technology, Ishikawa, 9231292, Japan.

[*] Corresponding Author: Chungen Xu. Email: xuchung@njust.edu.cn.

Received: 21 January 2020; Accepted: 18 April 2020.

retrieval of the encrypted data without decrypting them [Song, Wagner and Perrig (2000)]. Researches on SE can be classified into two categories, including symmetric SE and public-key encryption with keywords search (PEKS) [Xu, Yuan, Steinfeld et al. (2019)].

Compared with symmetric SE, the application of PEKS is more flexible, where more than one data owners can share their data with the data user. PEKS schemes can be used in many systems, e.g., email systems, the Internet of Thing (IoT) systems [Xu, Xu, Liu et al. (2019), Zhang, Xu, Wang et al. (2019)], and electronic health records (EHR) systems [Yang and Ma (2016)], etc. Boneh et al. [Boneh, Crescenzo, Ostrovsky et al. (2004)] first introduced PEKS model, which refers to three parties, the server, the sender (data owner) and the receiver (data user). In this model, the sender encrypts documents along with the keywords using the receiver's public key and stores the ciphertext on the server. When the receiver intends to search some documents associated with a specific keyword, he computes a token of this keyword using his secret key and sends it to the server as a search request. The server will conduct a test algorithm between the token and the ciphertext, then return the query results.

So far, most PEKS schemes were constructed under some hardness assumptions of number theory problems, such as bilinear pairing [Boneh, Crescenzo, Ostrovsky et al. (2004); Rhee, Park, Susilo et al. (2010); Huang and Li (2017); Yang and Ma (2016); Xu, Xu, Liu et al. (2019)] and others [Wu, Gan and Wang (2018); Crescenzo and Saraswat (2007); Anada, Kanaoka, Matsuzaki et al. (2018)]. However, with the proposing of a polynomial-time quantum algorithm on solving integer factorization problem, these PEKS schemes mentioned above are not secure under quantum computing attack. Thus, post-quantum cryptography (PQC) which applies mathematical primitive to construct cryptosystems against quantum computing, becomes a vital cryptography direction. Lattice-based cryptography is one of the most promising candidates of PQC. Therefore, how to construct lattice-based PEKS schemes that are secure in the future quantum era, is an urgent issue.

For the typical PEKS schemes [Boneh, Crescenzo, Ostrovsky et al. (2004)], Byun et al. [Byun, Rhee, Park et al. (2006)] proposed an attack, keywords guessing attack (KGA). When the adversary is a malicious server, this attack is called inside KGA. In a PEKS model, given a search token, anyone can compute the ciphertexts of arbitrary keywords from the keyword space using the receiver's public key, then execute a test between the ciphertext with this token, and confirm the keyword encapsulated in this token until the test succeeds. To alleviate the security concern aroused by such attack, most PEKS schemes were constructed under the assumption that the keywords space is so large that adversaries cannot generate exhaustively the ciphertexts for all keywords. However, the number of keywords in the real scenario is smaller than our imagination [Huang and Li (2017)]. Thus, this attack makes it feasible for adversaries to threaten the data privacy, such as the keywords of other returned documents, and which keyword the receiver wants to search, etc.

To address the above issues, the paper aims to construct a lattice-based PEKS scheme that can resist inside KGA. The contributions are as follows:

●We construct a lattice-based PEKS scheme to resist the quantum computing attack, where its security can be reduced to the hardness assumptions of LWE problem and ISIS problem. A special hash function is used for transforming the keywords to a non-singular matrix. This scheme adopts the trapdoor generation algorithm from lattice to obtain the sender's and receiver's key pairs. To prevent inside KGA, our scheme employs the preimage sample algorithm on a random bit and a part of ciphertext, which prevents the malicious server from forging a valid ciphertext.

●We conduct simulation experiments to evaluate the performance of the proposed scheme and show a comparison with other searchable schemes on the computational cost and the related parameter size.

1.1 Related works

Since Ajtai [Ajtai (1996)] showed a reduction from some classical worst-case problems in lattice to the average-case smallest integer solution (SIS) problem and constructed a related one-way function, then lattice is used to construct the cryptosystems which are widely believed post-quantum secure. Then, Regev [Regev (2009)] introduced another one average-case problem, learning with errors (LWE) problem, which has been proved it's as hard as approximating the worst-case Gap-SVP and SIVP with certain factors. Some lattice-based cryptography schemes were constructed, such as digital signature schemes [Gentry, Peikert and Vaikuntanathan (2008); Gordon, Katz and Vaikuntanathan (2010)], identity-based encryption schemes [Gentry, Peikert and Vaikuntanathan (2008); Agrawal, Boneh and Boyen (2010)], key exchange schemes [Zhang, Zhang, Ding et al. (2015); Peikert (2014); Ding, Gao, Takagi et al. (2019)], etc.

Gu et al. [Gu, Zheng, Kang et al. (2015)] proposed the first lattice-based PEKS scheme as far as we know, and its security can be reduced to LWE assumption in the standard model. Zhang et al. [Zhang and Xu (2018)] presented a PEKS scheme with a designated server from lattice. Zhang et al. [Zhang, Xu, Mu et al. (2018)] applied the identity-based encryption (IBE) into the keywords search and gave a construction of identity-based searchable encryption scheme with the designated tester from lattice. Xu et al. [Xu, Yuan, Steinfeld et al. (2019)] gave an identity-based searchable encryption scheme that supports multiple senders. To enrich the query, Mao et al. [Mao, Fu, Guo et al. (2018)] proposed a lattice-based encryption scheme with conjunctive keyword search. Kuchta et al. [Kuchta and Markowitch (2016)] constructed the first attribute-based searchable encryption scheme from lattice. Yang et al. [Yang, Zheng, Chang et al. (2018)] showed a lattice-based searchable encryption scheme with fuzzy keyword search and multiple users in multimedia clouds. Behnia et al. [Behnia, Ozmen and Yavuz (2018)] proposed two PEKS schemes based on NTRU and LWE problem, respectively, and showed a specific experimental analysis. Zhang et al. [Zhang, Xu, Wang et al. (2019)] introduced the forward secure PEKS model and showed a construction on lattice, where the scheme is secure even the key is compromised.

Considering the KGA, a secure channel between the server and the receiver is required to transmit search tokens, which can prevent outside adversary from obtaining the token. PEKS with the designated tester (dPEKS) model [Rhee, Park, Susilo et al. (2009); Fang, Susilo, Ge et al. (2013)] was also proposed, where only the designated server can execute the test. However, the above methods are invalid for the malicious server (inside KGA), where the server is easily controlled by adversaries through some computer viruses, etc.

To prevent the inside KGA, Huang et al. [Huang and Li (2017)] proposed a public-key authenticated encryption with keyword search scheme based on bilinear maps, which guarantees only the legal sender can compute the ciphertext of keywords to prevent the server from doing test casually. Chen et al. [Chen, Mu, Yang et al. (2015)] employed multiple servers, which prohibits any single server from conducting the test independently. This approach requires that these servers do not collude.

1.2 Roadmap

Section 2 presents some preliminaries of lattice, hardness assumption and the related algorithms. The syntax of PEKS with security model is introduced in Section 3. In Section 4, we show a construction of lattice-based searchable encryption scheme, as well as its security and performance analysis. Finally, we give a conclusion in Section 5.

2 Preliminaries

Notation: R and Z denote the set of rational and integer, respectively. The bold font denotes the vector. $\|\tilde{A}\|$ means the Gram-Schmidt norm of matrix A, where $\tilde{A}$ is the Gram-Schmidt orthogonalization of A.

Definition 2.1 Given two random variables x and y on a countable set S, there is a defined function Eq. (1) as statistical distance.

$$\Delta(x,y)=\frac{1}{2}\sum_{s\in S}|Pr[x=s]-Pr[y=s]| \quad (1)$$

If the distance $d(\lambda)=\Delta(x(\lambda),y(\lambda))$ is negligible in λ, we say x and y are statistically close.

2.1 Lattice

Lattice: Given some linearly independent vectors $\mathbf{b}_i \in \mathrm{R}^m$ for $i\in\{1,...,n\}$, the set generated by the above vectors is a lattice, denoted by

$$\Lambda(\mathbf{b}_1,\cdots,\mathbf{b}_n)=\{\Sigma_{i=1}^n x_i\mathbf{b}_i \mid x_i\in \mathrm{Z}\}. \quad (2)$$

where $\{\mathbf{b}_1,\cdots,\mathbf{b}_n\}$ is a basis of the lattice. Here m is the dimension and n is the rank. The lattice is full-rank if $m=n$.

Definition 2.2 Given a prime q, a matrix $A\in \mathrm{Z}_q^{n\times m}$ and a vector $\mathbf{u}\in \mathrm{Z}_q^n$, we introduce three sets as Eq. (3):

$$\begin{cases}\Lambda_q^{\perp}(A):=\{\mathbf{e}\in \mathrm{Z}^m \mid A\mathbf{e}=\mathbf{0} \mod q\},\\ \Lambda_q^{\mathbf{u}}(A):=\{\mathbf{e}\in \mathrm{Z}^m \mid A\mathbf{e}=\mathbf{u} \mod q\},\\ \Lambda_q(A):=\{\mathbf{u}\in \mathrm{Z}_q^m \mid \exists \mathbf{s}\in \mathrm{Z}_q^n, \mathbf{u}=A^T\mathbf{s} \mod q\}.\end{cases} \quad (3)$$

2.2 Hardness assumptions

This section introduces two hard problems, learning with errors (LWE) problem and inhomogeneous small integer solution (ISIS) problem which are used for the security proof. It has been proved that LWE problem [Regev (2009)] and ISIS problem [Gentry, Peikert and Vaikuntanathan (2008)] are as hard as approximating the worst-case Gap-SVP and SIVP with certain factors.

Definition 2.3 [Agrawal, Boneh and Boyen (2010)] Given $\alpha \in (0,1)$ and a prime q, Ψ_α is a probability distribution over Z_q which returns $\lfloor qx \rfloor$ by choosing $x \in Z$ from the normal distribution with mean 0 and standard deviation $\frac{\alpha}{\sqrt{2\pi}}$.

Definition 2.4 (LWE problem). [Regev (2009)] For a positive integer n, set $m = m(n)$ and $\alpha \in (0,1)$ such that a prime $q = q(n) > 2$ and $\alpha q > 2\sqrt{n}$, a secret $\mathbf{s} \leftarrow_\$ Z_q^n$, the (Z_q, n, Ψ_α)-LWE problem is to distinguish between the following distributions over $Z_q^{n\times m} \times Z_q^m$:

●LWE distribution: choose uniformly a matrix $A \leftarrow_\$ Z_q^{n\times m}$, and sample $\mathbf{e} \leftarrow \Psi_\alpha^m$, output $(A, A^T\mathbf{s} + \mathbf{e}) \in Z_q^{n\times m} \times Z_q^m$;

●Uniform distribution: choose uniformly a matrix $A \leftarrow_\$ Z_q^{n\times m}$ and a vector $\mathbf{x} \leftarrow_\$ Z_q^m$, output $(A, \mathbf{x}) \in Z_q^{n\times m} \times Z_q^m$.

Definition 2.5 (ISIS problem). [Gentry, Peikert and Vaikuntanathan (2008)] Given a matrix $A \in Z_q^{n\times m}$ and a uniform vector $\mathbf{u} \in Z_q^n$, a real number $\eta > 0$, ISIS problem is to find a non-zero integer vector $\mathbf{e} \in Z^m$ such that Eq. (4) holds and $\|\mathbf{e}\| \leqslant \eta$.

$$A\mathbf{e} = \mathbf{u} \mod q. \tag{4}$$

2.3 Lattice algorithms

Let $\mathbf{D}_{\Lambda,\sigma,\mathbf{c}}$ denote a discrete Gaussian distribution over Λ with a center $\mathbf{c} \in R^m$ and parameter $\sigma \in R^+$, denoted by

$$\mathbf{D}_{\Lambda,\sigma,\mathbf{c}} = \frac{\rho_{\sigma,\mathbf{c}(x)}}{\rho_{\sigma,\mathbf{c}(\Lambda)}}, \tag{5}$$

where:

$$\begin{cases} \rho_{\sigma,\mathbf{c}(\Lambda)} = \sum_{x\in\Lambda} \rho_{\sigma,\mathbf{c}(x)}, \\ \rho_{\sigma,\mathbf{c}(x)} = \exp(-\pi \frac{\| x - \mathbf{c}\|^2}{\sigma^2}). \end{cases} \tag{6}$$

We denote it by $\mathbf{D}_{\Lambda,\sigma}$, when $\mathbf{c} = 0$.

Lemma 2.1 (*TrapGen* algorithm). [Ajtai (1999); Gentry, Peikert and Vaikuntanathan (2008); Alwen and Peikert (2009)] Given integers n, m, q as the input where $q > 2$ and $m \geqslant 6n \log q$, there is a probability polynomial time (PPT) algorithm *TrapGen*, outputs a matrix $A \in Z_q^{n\times m}$ along with a basis T_A of the lattice $\Lambda^{\perp}(A)$, where

$$A \cdot T_A = 0 \quad \bmod q . \tag{7}$$

Here the distribution of A is statistically close to uniform on $Z_q^{n\times m}$, and $\|\tilde{T}_A\| \leqslant O(\sqrt{n \log q})$.

Lemma 2.2 (Sample preimage algorithm). [Gentry, Peikert and Vaikuntanathan (2008)] Given a matrix $A \in Z_q^{n\times m}$ with a basis $T_A \in Z_q^{m\times m}$, a vector $\mathbf{u} \in Z_q^n$ and a parameter $\sigma \geqslant \|\tilde{T}_A\| \cdot \omega(\sqrt{\log m})$ as the input, where $m \geqslant 2n\lceil \log q \rceil$, there is a PPT algorithm *SamplePre*, outputs a sample $\mathbf{e} \in Z_q^m$ distributed in $\mathbf{D}_{\Lambda_q^{\mathbf{u}}(A),\sigma}$, such that Eq. (8) holds.

$$A\mathbf{e} = \mathbf{u} \quad \bmod q \tag{8}$$

Lemma 2.3 (*SampleLeft* algorithm). [Agrawal, Boneh and Boyen (2010)] Given a matrix $A \in Z_q^{n\times m}$ with basis T_A of lattice $\Lambda^{\perp}(A)$ and a matrix $A' \in Z_q^{n\times m_1}$, a vector $\mathbf{u} \in Z_q^n$, a gaussian parameter σ as the input, there is an algorithm *SampleLeft* to output a short vector $\mathbf{e}$ sampled from a distribution statistically close to $\mathbf{D}_{\Lambda^{\mathbf{u}}_{q(A|A')},\sigma}$ such that Eq. (9) holds.

$$(A \mid A')\mathbf{e} = \mathbf{u} \quad \bmod q \tag{9}$$

Lemma 2.4 (*SampleRight* algorithm). [Agrawal, Boneh and Boyen (2010)] Given a matrix $A \in Z_q^{n\times k}$ and a matrix $B \in Z_q^{n\times m}$, a random matrix $R \in Z_q^{k\times m}$, a basis T_B of $\Lambda^{\perp}(B)$, a vector $\mathbf{u} \in Z_q^n$, and a parameter σ as the input, there is an algorithm *SampleRight* to output a vector $\mathbf{e}$ sampled from a distribution statistically close to $\mathbf{D}_{\Lambda^{\mathbf{u}}_{q(A|AR+B)},\sigma}$ such that Eq. (10) holds

$$(A \mid AR + B)\mathbf{e} = \mathbf{u} \quad \bmod q \tag{10}$$

Our scheme employs the following injective encoding function, full-rank difference map (FRD), to transform from a keyword to a matrix. The keywords set can be expanded to $\{0,1\}^*$ using a collision-resistant hash function over Z_q^n.

Definition 2.6 [Agrawal, Boneh and Boyen (2010)] Given a prime q and a positive integer n, a function $H : Z_q^n \to Z_q^{n\times n}$ is a full-rank difference function, if it satisfies the following conditions:

- For all distinct vectors $\mathbf{u}, \mathbf{v} \in Z_q^n$, the matrix $H(\mathbf{u}) - H(\mathbf{v}) \in Z_q^{n\times n}$ is full rank;
- H is computable in polynomial time (in $n \log q$).

3 Syntax of PEKS against inside KGA

This section introduces the system model of PEKS against inside KGA as Fig. 1, which consists of three parties, the sender, the receiver and the server. Firstly, the central authority will compute the public and secret key pairs for the sender and the receiver. Compared with the traditional PEKS scheme [Boneh, Crescenzo, Ostrovsky et al. (2004)], the sender's secret key is used for the generation of the ciphertext, which offers an authentication and guarantees only the legal sender can generate the valid ciphertext. Thus, a malicious server cannot compute the ciphertext to execute the inside KGA.

● **Sender:** The sender encrypts the data document along with keywords and then uploads the encrypted data as well as the ciphertext of keywords to the server for sharing their data with the receiver (data user). Here we focus on the encryption of keywords where the computing of its ciphertext requires the sender's secret key and the receiver's public key, while the classical symmetric encryption technique is adopted to the encryption of data documents.

● **Receiver:** The receiver computes a token of the queried keyword using his secret key, then sends it to the server as a search request.

● **Server:** The server offers storage for the encrypted data and performs the keywords search. Once receiving a token from the receiver, the server will execute a test algorithm on keywords and returns search results to the receiver.

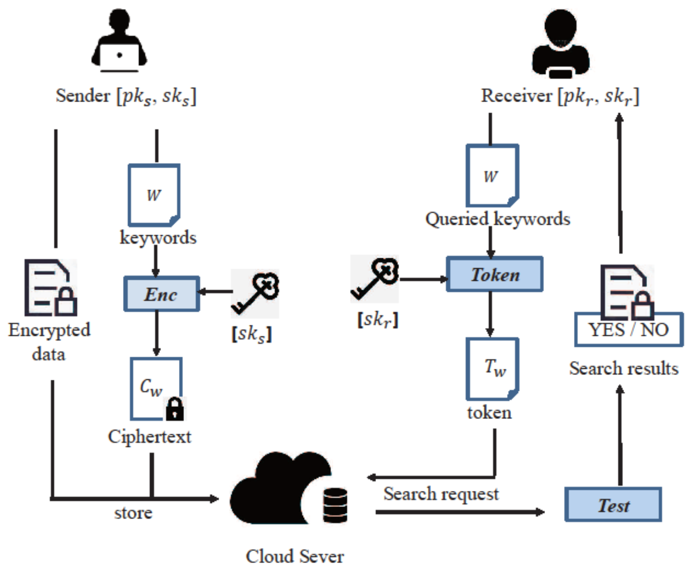


Figure 1: The framework of PEKS

3.1 System model

Our PEKS model consists of six algorithms, denoted by

$\Pi = (\mathbf{SetUp}, \mathbf{SKeyGen}, \mathbf{RKeyGen}, \mathbf{Enc}, \mathbf{Token}, \mathbf{Test})$.

● $pp \leftarrow \mathbf{SetUp}(\lambda)$: SetUp algorithm inputs the security parameter λ and returns the public parameter pp.

- $(pk_s, sk_s) \leftarrow \textbf{SKeyGen}(pp)$: The sender's key generation algorithm inputs public parameter pp and generates the sender's public key pk_s and secret key sk_s.
- $(pk_r, sk_r) \leftarrow \textbf{RKeyGen}(pp)$: The receiver's key generation algorithm inputs public parameter pp and generates the receiver's public key pk_r and secret key sk_r.
- $C_w \leftarrow \textbf{Enc}(w, sk_s, pk_s, pk_r)$: The encryption algorithm inputs a keyword w, the sender's public key pk_s, the secret key sk_s, and the receiver's public key pk_r, outputs the ciphertext C_w.
- $T_w \leftarrow \textbf{Token}(w, sk_r, pk_r)$: The token generation algorithm inputs the keyword w, the receiver's secret key sk_r and the public key pk_r, outputs the search token T_w.
- $\{0,1\} \leftarrow \textbf{Test}(T_w, C_w, pk_s)$: The test algorithm inputs the search token T_w, the ciphertext C_w as well as the sender's public key pk_s, outputs 1 if T_w and C_w own the same keyword, otherwise outputs 0.

Correctness: The correctness guarantees that for any honestly generated key pairs (pk_s, sk_s) and (pk_r, sk_r), and for any keyword w, $Test(pk_s, pk_r, T_w, C) = 1$ holds with probability 1, where $C_w \leftarrow Enc(w, sk_s, pk_s, pk_r)$ and $T_w \leftarrow Token(w, pk_r, sk_r)$.

3.2 Security model

This section introduces a game to define the security model on the proposed scheme, ciphertext indistinguishability under selectively chosen keywords attack. The security model requires that there is no PPT adversary who can distinguish the ciphertext and a randomly selected element from the ciphertext space. Given the security parameter λ, there are two parties, a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ carry out the following games.

Setup (λ): The adversary chooses a target keyword w^* and sends it to the challenger $\mathcal{C}$. Given security parameter λ as input, $\mathcal{C}$ runs the algorithm to output the public parameter pp and sends it to $\mathcal{A}$.

Query phase 1: $\mathcal{A}$ is allowed to query adaptively the tokens for any keywords except w^*, that is to say, $\mathcal{A}$ can issue the q_t th query with the knowledge of the $q_1, \ldots, q_{t-1}$ queries. Then the challenger $\mathcal{C}$ returns tokens to $\mathcal{A}$.

Challenge phase: After $\mathcal{A}$ finishes the above queries, $\mathcal{C}$ randomly chooses a bit $r \in \{0,1\}$, if $r = 0$, returns $C_{w^*} \leftarrow Enc(w^*, pk_r, sk_s, pk_s)$ to $\mathcal{A}$, otherwise selects randomly an element C from the ciphertext space and returns it to $\mathcal{A}$.

Query phase 2: $\mathcal{A}$ can continue to query adaptively the tokens for any keywords except w^*.

Guess: $\mathcal{A}$ guesses a bit $r' \in \{0,1\}$. When $r = r'$, $\mathcal{A}$ wins the game.

The proposed model satisfies ciphertext indistinguishability under selectively chosen keywords attack if the advantage as Eq. (11) is negligible for any PPT adversary $\mathcal{A}$

$$Adv_{\mathcal{A}}(\lambda) = | Pr[r = r'] - \frac{1}{2} | . \qquad (11)$$

Then we introduce a game on the inside KGA, where there are two parties, an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The adversary, including a malicious server, intends to perform this attack by forging a valid ciphertext of keyword which can pass during the test phase. The challenger will do the simulation and return results to answer queries from the adversary. The game is as follows:

Setup (λ)**:** Given security parameter λ as input, $\mathcal{C}$ runs the algorithm to output public parameter pp as well as the public keys of the sender and the receiver, then sends them to $\mathcal{A}$.

Token query: The adversary $\mathcal{A}$ can query the token of any keyword, then $\mathcal{C}$ returns the results to the adversary.

Ciphertext query: The adversary $\mathcal{A}$ can query the ciphertext of keyword, then the challenger $\mathcal{C}$ returns the results to the adversary.

Forge phase: From the above queries, $\mathcal{A}$ outputs a forged ciphertext related to a keyword w^* which can pass during the test phase.

4 Lattice-based construction

4.1 Our construction

Here we give our construction in lattice as Fig. 2. In the phase of encryption, our scheme uses the sender's secret key to generate a signature for a random bit and a part of ciphertext, which guarantees anyone including a malicious server cannot forge the ciphertext and then he cannot perform inside KGA. During the test phase, the server has to recover the random bit and then verifies this signature.

4.1.1 Setup

Given some system parameters $n, m, q, \delta, \sigma, \alpha$, where q is prime, the central authority runs the **Setup** algorithm in Fig. 2 to return the public parameter which is published for this whole system. During the process, a vector $\mathbf{u}$ is selected randomly from the uniform distribution over Z_q^n and two matrices A, B are chosen randomly from the uniform distribution over $Z_q^{n\times m}$. According to the Definition 2.6, a function $H:\{0,1\}^* \to Z_q^{n\times n}$ is chosen from the FRD hash function family $\mathcal{H}$ and a hash function $H_0 : Z_q^{2m} \times \{0,1\} \to Z_q^n$ is chosen from a hash function family $\mathcal{H}_0$.

4.1.2 Key generation

Given public parameters, the central authority runs **KeyGen** algorithm in Fig. 2 to generate the user's public and secret key pair, where the *TrapGen* algorithm is invoked. As the Lemma 2.1, *TrapGen* algorithm outputs a uniform matrix $A \in Z_q^{n\times m}$ along with a short basis T of lattice $\Lambda^{\perp}(A)$ which satisfies Eq. (12) hold and $\|\tilde{T}\| \le O(\sqrt{n\log q})$.

$$AT = 0 \mod q. \tag{12}$$

The matrix A is public key pk and the basis T is secret key sk. In our scheme, we employ this algorithm to generate the sender's public and secret key pair $(pk_s, sk_s) = (A_s, T_s)$ and the receiver's public and secret key pair $(pk_r, sk_r) = (A_r, T_r)$. Then the central authority returns these key pairs to the sender and the receiver via a secure channel, respectively.

4.1.3 Encryption

In this part, the sender runs **Enc** algorithm in Fig. 2 to encrypt keywords using her secret key and the receiver's public key, and stores the ciphertext to the server. During this phase, a random vector $\mathbf{s}$ is selected from the uniform distribution over Z_q^n. Then the sender chooses a noise vector $\mathbf{y} \leftarrow \Psi_\alpha^m$ and $x \leftarrow \Psi_\alpha$, where Ψ_α is a distribution over Z_q as Definition 2.3, and selects a single bit $b \in \{0,1\}$, a random matrix $R \in \{-1,1\}^{m\times m}$.

Next, the sender computes

$$\begin{cases} C_1 = \mathbf{u}^T\mathbf{s} + x + b\left\lfloor \dfrac{q}{2} \right\rfloor \in Z_q, \\ C_2 = (A_r \mid A + H(w)B)^T\mathbf{s} + \mathbf{z} \in Z_q^{2m}. \end{cases} \tag{13}$$

where $\mathbf{z} \leftarrow (\mathbf{y}, R^T\mathbf{y})^T$, and a hash function H_0 in (C_2, b) to obtain a hash value $\mathbf{h} = H_0(C_2 \mid b)$. We can note the preimage sample algorithm as Definition 2.2 is invoked to obtain a sample $\mathbf{e} \in Z_q^m$ which satisfies as follows:

$$A_s\mathbf{e} = \mathbf{h} \mod q. \tag{14}$$

Finally, the sender returns $(C_1, C_2, \mathbf{e})$ as the ciphertext.

4.1.4 Token generation

When the receiver intends to query some documents associated with keyword w, he runs **Token** algorithm in Fig. 2 to obtain a token T_w related to w using his secret key. This algorithm firstly computes a matrix $M_w = A + H(w)B$, then invokes the *SampleLeft* algorithm to sample $\mathbf{t} \in Z_q^{2m}$ which satisfies

$$(A_r \mid A + H(w)B)\mathbf{t} = \mathbf{u} \mod q. \tag{15}$$

Finally, the receiver returns $\mathbf{t}$ as the token T_w to the server.

4.1.5 Test

The server runs **Test** algorithm to decide if the search token matches the ciphertext or not.

Firstly, this algorithm computes Eq. (16) to recover the random bit b,

$$r = C_1 - \mathbf{t}^T \cdot C_2 \tag{16}$$

then computes the hash function on the part of ciphertext C_2 along with b to verify if Eq. (17) holds.

$$A_s \mathbf{e} = \mathbf{h} \mod q \tag{17}$$

```
Setup(λ)
Central authority:
1:  set parameters $n, m, q, \delta, \sigma, \alpha$
2:  $\mathbf{u} \leftarrow_\$ Z_q^n$
3:  $H \leftarrow \mathcal{H}$
4:  $H_0 \leftarrow \mathcal{H}_0$
5:  $A, B \leftarrow_\$ Z_q^{n\times m}$
6:  $pp \leftarrow (q, n, m, \delta, \sigma, \alpha, A, B, \mathbf{u}, H, H_0)$
7:  return $pp$

KeyGen(pp)
Central authority:
1:  $(A, T) \leftarrow TrapGen(n, m, q)$
2:  $(pk, sk) \leftarrow (A, T)$
3:  return $pk, sk$

Enc(w, sk_s, pk_s, pk_r)
Sender:
1:  $\mathbf{s} \leftarrow_\$ Z_q^n$
2:  $\mathbf{y} \leftarrow \Psi_\alpha^m$
3:  $x \leftarrow \Psi_\alpha$
4:  $R \leftarrow_\$ \{-1,1\}^{m\times m}$
5:  $\mathbf{z} \leftarrow \begin{pmatrix} \mathbf{y} \\ R^T \mathbf{y} \end{pmatrix}$
6:  $b \leftarrow_\$ \{0,1\}$
7:  $F_w \leftarrow (A_r \mid A + H(w)B)$
8:  $C_1 \leftarrow \mathbf{u}^T \mathbf{s} + x + b\left\lfloor \frac{q}{2} \right\rfloor$
9:  $C_2 \leftarrow F_w^T \mathbf{s} + \mathbf{z}$
10: $\mathbf{h} \leftarrow H_0(C_2 \mid b)$
11: $\mathbf{e} \leftarrow SamplePre(A_s, T_s, \mathbf{h}, \delta)$
12: $C_w \leftarrow (C_1, C_2, \mathbf{e})$
13: return $C_w$

Token(w, sk_r, pk_r)
Receiver:
1:  $M_w \leftarrow A + H(w)B$
2:  $\mathbf{t} \leftarrow SampleLeft(A_r, M_w, T_r, \mathbf{u}, \sigma)$
3:  $T_w \leftarrow \mathbf{t}$
4:  return $T_w$

Test(T_w, C_w, pk_s)
Server:
1:  $r \leftarrow C_1 - \mathbf{t}^T \cdot C_2$
2:  if $|r - \left\lfloor \frac{q}{2} \right\rfloor| < \left\lfloor \frac{q}{4} \right\rfloor$ then
3:      $b \leftarrow 1$
4:      $\mathbf{h} \leftarrow H_0(C_2 \mid 1)$
5:  else
6:      $b \leftarrow 0$
7:      $\mathbf{h} \leftarrow H_0(C_2 \mid 0)$
8:  end if
9:  if $A_s \mathbf{e} = \mathbf{h} \mod q$ then
10:     return 1
11: else
12:     return 0
13: end if
```

Figure 2: Our lattice-based search scheme construction

Eq. (17) holds which indicates the token matches ciphertext, namely, the ciphertext includes the keyword encapsulated in token.

Correctness: From the **Test** algorithm, we have the results as Eq. (18).

$$\begin{aligned} r = C_1 - \mathbf{t}^T \cdot C_2 &= \mathbf{u}^T\mathbf{s} + x + b\left\lfloor \frac{q}{2} \right\rfloor - \mathbf{t}^T \cdot (F_w^T\mathbf{s} + \begin{pmatrix} \mathbf{y} \\ R^T\mathbf{y} \end{pmatrix}) \\ &= x + b\left\lfloor \frac{q}{2} \right\rfloor - \mathbf{t}^T \cdot \begin{pmatrix} \mathbf{y} \\ R^T\mathbf{y} \end{pmatrix} = b\left\lfloor \frac{q}{2} \right\rfloor + x - \mathbf{t}^T \begin{pmatrix} \mathbf{y} \\ R^T\mathbf{y} \end{pmatrix} \end{aligned} \tag{18}$$

where $x - \mathbf{t}^T(\mathbf{y}, R^T\mathbf{y})^T$ is error term. To decrypt correctly, we set the parameters (q, m, σ, α) as the discussion in Agrawal et al. [Agrawal, Boneh and Boyen (2010)] to make sure:

- The error term is less than $\frac{q}{5}$;
- *TrapGen* algorithm can operate, i.e., $m > 6n\log q$;
- δ is large enough for *SamplePre* algorithm;
- σ satisfies *SampleLeft* algorithm and *SampleRight* algorithm;
- Regev's reduction applies [Regev (2009)], i.e., $q > \frac{2\sqrt{n}}{\alpha}$.

4.2 Security analysis

In the section, we analyze the security of the proposed scheme, including ciphertext indistinguishability under selectively chosen keywords attack, and inside KGA resistance. Intuitively, a part of the ciphertext forms an LWE instance. Our scheme encapsulates a GPV signature [Gentry, Peikert and Vaikuntanathan (2008)] (which can be generated by invoking preimage sample algorithm and be reduced to ISIS problem) into the ciphertext, which prevents any adversary including the malicious server from computing (or forging) a valid ciphertext of keywords freely. In the **Test** algorithm, the server firstly is required to recover a random bit, then he needs to verify the signature. Thus, this construction can prevent the inside KGA.

Lemma 4.1 For any PPT adversary $\mathcal{A}$, the proposed scheme is ciphertext indistinguishability under selectively chosen keywords attack, if the (Z_q, n, Ψ_α)-LWE problem is intractable.

Proof: Let $\mathcal{A}$ be a PPT adversary to break the ciphertext indistinguishability under selectively chosen keywords attack of the proposed scheme. Then we can construct a challenger $\mathcal{C}$ who can solve the LWE problem.

Setup: The adversary $\mathcal{A}$ chooses w^* as the target keyword. $\mathcal{C}$ queries the LWE oracle $\mathcal{O}$ for $m+1$ times and obtains fresh pairs $(\mathbf{u}_i, v_i) \in Z_q^n \times Z_q$, where $i = 0, 1, \ldots, m$. Then $\mathcal{C}$ prepares the system parameter pp and public key as follows:

● Construct a random matrix $A_r \in Z_q^{n\times m}$ from m of the given LWE samples, where the i-th column of A_r is $\mathbf{u}_i$ for all $i=1,2,...,m$. Let $\mathbf{u}$ be $\mathbf{u}_0$.

● By running *TrapGen* algorithm, the challenger $\mathcal{C}$ obtains a random matrix $B \in Z_q^{n\times m}$ as well as a trapdoor T_B for $\Lambda_q^{\perp}(B)$. Then $\mathcal{C}$ selects a random matrix $R^* \in \{-1,1\}^{m\times m}$ to construct $A \leftarrow A_r R^* - H(w^*)B$.

Query phase 1: $\mathcal{C}$ answers each token query for $w \neq w^*$. Then we have

$$\begin{aligned} F_w &= (A_r \mid A + H(w)B) = (A_r \mid A_r R^* - H(w^*)B + H(w)B) \\ &= (A_r \mid A_r R^* + (H(w) - H(w^*))B) \end{aligned} . \tag{19}$$

According to the definition of FRD function, we can note that $H(w)-H(w^*)$ is a non-singular matrix and the trapdoor T_B is a trapdoor for $\Lambda_q^{\perp}(B')$, where

$$B' = (H(w) - H(w^*))B . \tag{20}$$

Then $\mathcal{C}$ can answer the token query by returning $\mathbf{t}$ to $\mathcal{A}$ which is generated from *SampleRight* $(A_r, (H(w)-H(w^*))B, R^*, T_B, \mathbf{u}, \delta)$.

H_0 **query:** For an $H_0(C_2 \mid b)$ query, $\mathcal{C}$ returns the hash value if the related input $(C_2 \mid b)$ has been queried. Otherwise, $\mathcal{C}$ selects randomly $\mathbf{h}$ from Z_q^n and returns it into the hash list L_0.

Challenge phase: $\mathcal{A}$ selected randomly a bit $b^* \in \{0,1\}$, then $\mathcal{C}$ generates the ciphertext of the target keyword w^* as follows:

● Set $\mathbf{v}^* = (v_1,...,v_m) \in Z_q^m$, compute

$$\begin{cases} C_1^* = v_0 + b^* \left\lfloor \dfrac{q}{2} \right\rfloor \in Z_q \\ C_2^* = (\mathbf{v}^* \mid (R^*)^T \mathbf{v}^*) \in Z_q^{2m} \end{cases} . \tag{21}$$

Compute Eq. (22) and invokes *SamplePre* algorithm to generate a preimage sample $\mathbf{e}^*$.

$$\mathbf{h}^* = H_0(C_2^* \mid b^*) \tag{22}$$

● Select randomly a bit $r \in \{0,1\}$, if $r=0$, return $(C_1^*, C_2^*, \mathbf{e}^*)$ to $\mathcal{A}$, otherwise return a random (C_1, C_2) from the part of ciphertext space $Z_q \times Z_q^{2m}$, and check if Eq. (23) holds for a randomly chosen $b \in \{0,1\}$ has been queried or not,

$$\mathbf{h} = H_0(C_2 \mid b) \tag{23}$$

if it hasn't been queried, compute the hash value $\mathbf{h} = H_0(C_2 \mid b)$ and add it into the hash list L_0. The challenger generates $\mathbf{e} \leftarrow \mathbf{D}_{\Lambda_q^{\mathbf{u}}(A_s),\delta}$ by *SampleDom* algorithm in [Gentry, Peikert and Vaikuntanathan (2008)]. Finally, $\mathcal{C}$ returns $(C_1, C_2, \mathbf{e})$ to the adversary.

Query phase 2: $\mathcal{A}$ continues to execute the queries for keywords $w \neq w^*$.
Guess: $\mathcal{A}$ outputs a guess for $r \in \{0,1\}$.

If the LWE oracle is pseudorandom, we can note that $F_{w^*} = (A_r \mid A_r R^*)$. For the random noise vector $\mathbf{y}$, $\mathbf{v}^* = A_r^T \mathbf{s} + \mathbf{y}$ where $\mathbf{y} \leftarrow \Psi_\alpha^m$. Thus

$$C_2^* = \begin{pmatrix} A_r^T \mathbf{s} + \mathbf{y} \\ (R^*)^T A_r^T \mathbf{s} + (R^*)^T \mathbf{y} \end{pmatrix} = \begin{pmatrix} A_r^T \mathbf{s} + \mathbf{y} \\ (A_r R^*)^T \mathbf{s} + (R^*)^T \mathbf{y} \end{pmatrix} = F_{w^*}^T \mathbf{s} + \begin{pmatrix} \mathbf{y} \\ (R^*)^T \mathbf{y} \end{pmatrix}, \tag{24}$$

which is a valid ciphertext for the challenged keyword w^*. From the LWE instance, we have:

$$v_0 = \mathbf{u}_0^T \mathbf{s} + x \tag{25}$$

then Eq. (26) shows a part of the ciphertext for w^*.

$$C_1^* = \mathbf{u}_0^T \mathbf{s} + x + b^* \left\lfloor \frac{q}{2} \right\rfloor \tag{26}$$

When the oracle is a truly random oracle, we have v_0 and $\mathbf{v}^*$ are uniform. Then C_2^* is uniform and independent. Thus the part of the ciphertext (C_1^*, C_2^*) is uniform in $Z_q \times Z_q^{2m}$. From the Lemma 2.2, the preimage sample $\mathbf{e}^*$ is distributed in $\mathbf{D}_{\Lambda_q^{\mathbf{u}}(A_s),\delta}$. Thus if the adversary $\mathcal{A}$ can distinguish the ciphertext, then $\mathcal{C}$ can solve the LWE problem.

Lemma 4.2 For any PPT adversary $\mathcal{A}$, the proposed scheme can resist inside KGA, if ISIS problem is intractable.

Proof: Let $\mathcal{A}$ be an adversary who can perform the inside KGA by forging the ciphertext of keywords. Then we can construct a challenger $\mathcal{C}$ to solve ISIS problem by invoking the adversary $\mathcal{A}$.

Setup: $\mathcal{C}$ runs the setup algorithm to generate the public parameters and sends them to $\mathcal{A}$. Then $\mathcal{C}$ queries the ISIS oracle and obtains a pair $(A^*, \mathbf{h}^*) \in Z_q^{n\times m} \times Z_q^n$. Let $A_s = A^*$ be the public key of the sender, $\mathcal{C}$ chooses randomly a matrix $A_r \in Z_q^{n\times m}$ as the public key of the receiver, $\mathcal{C}$ sends these two public keys to $\mathcal{A}$.

Hash query: H_0 query: For a distinct C_2 with a random bit $b \in \{0,1\}$, $\mathcal{C}$ returns the hash value $\mathbf{h}$ to $\mathcal{A}$ if it has existed in the list L_0. If $C_2 = C_2^*$ is the second part of the ciphertext related to the keyword w^* and $b = b^*$, $\mathcal{C}$ adds $(C_2^*, b^*, \perp, \mathbf{h}^*)$ into the list L_0 and returns $\mathbf{h}^*$ to $\mathcal{A}$. Otherwise, the related input $(C_2 \mid b)$ has been queried. Otherwise, $\mathcal{C}$ runs *SampleDom* algorithm and obtains $\mathbf{e} \leftarrow \mathbf{D}_{\mathbf{Z}_q^m,\sigma}$, then computes $\mathbf{h} = A^*\mathbf{e} \mod q$ and adds $(C_2, b, \mathbf{e}, \mathbf{h})$ to the hash list L_0.

Query phase: The token query is same as the Lemma 4.1. For the generation of part of ciphertext (C_1, C_2), $\mathcal{C}$ computes them according to the **Enc** algorithm. For the third part of the ciphertext, $\mathcal{C}$ runs *SampleDom* algorithm and obtains $\mathbf{e} \leftarrow \mathbf{D}_{\mathbf{Z}_q^m,\sigma}$. Then $\mathcal{C}$ returns

$(C_1, C_2, \mathbf{e})$ to $\mathcal{A}$.

Forge phase: $\mathcal{A}$ forges a ciphertext $(C_1^*, C_2^*, \mathbf{e}^*)$ associated with w^* and a random bit b^* to $\mathcal{C}$ which cannot be queried for the ciphertext oracle. $\mathcal{A}$ obtains the token $\mathbf{t}^*$ by running the *SamplePre* algorithm. Then $\mathcal{C}$ can recover the random bit b^* from $C_1^* - (\mathbf{t}^*)^T C_2^*$ and outputs $\mathbf{e}^*$ as the third part of the forged ciphertext. $\mathcal{C}$ returns $\mathbf{e}^*$ as an answer for ISIS problem $(A^*, \mathbf{h}^*)$.

If $\mathbf{e}^*$ is a valid ciphertext as the third part related to w^* and a bit b^*, then we have

$$A_s \mathbf{e}^* = H_0(C_2^* \mid b^*) \mod q, \tag{27}$$

where $\| \mathbf{e}^* \| \leqslant \sigma\sqrt{m}$. We note that $A_s = A^*$ and $\mathcal{C}$ can guess the result as Eq. (28) successfully with probability $\frac{1}{M}$,

$$H_0(C_2^* \mid b^*) = \mathbf{h}^* \tag{28}$$

where M denotes the time of hash query and is a polynomial. Hence if $\mathcal{A}$ forges successfully the ciphertext with a non-negligible probability ε, then the ISIS problem can be solved with the non-negligible probability $\frac{\varepsilon}{M}$, which breaks the hardness assumption of ISIS problem.

4.2 Performance evaluation

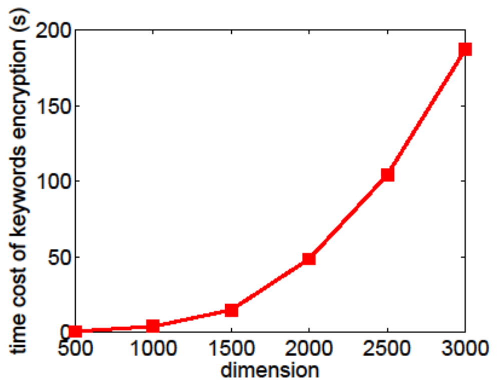
In this section, we show a performance comparison with other PEKS schemes as Tab. 1 which lists the computational cost of the related algorithms and the size of parameters, etc. There are some notations, H: hash operation, E: exponent operation in group, P: pairing operation, M_P: multiplication operation of group elements, T_{BD}: *NewBasisDel* operation, T_{SL}: *SampleLeft* operation, T_{SP}: *SamplePre* operation; M: modular multiplication, I: matrix inversion operation, m: the dimension of matrix, l_q: the size of element in Z_q, l_w: the size of keyword, $|G|$: the size of group elements, κ: the security parameter used in the scheme [Behnia, Ozmen and Yavuz (2018)].

Huang et al. [Huang and Li (2017)] constructed a searchable scheme against inside KGA, whose security is based the hardness of on the DBDH (Decisional Bilinear Diffie-Hellman) problem and mDLIN (modified Decision linear) problem which can be solved under the quantum computer, compared with the schemes [Behnia, Ozmen and Yavuz (2018); Zhang and Xu (2018)] and our scheme. For realizing the standard model, Behnia et al. [Behnia, Ozmen and Yavuz (2018)] doesn’t use hash function to deal with the keywords but bit-by-bit encrypt each keyword. Furthermore, this scheme does not consider the inside KGA. The scheme proposed by Zhang et al. [Zhang and Xu (2018)] can guarantee only the designated tester can execute **Test** algorithm, which prevents the outside adversary except malicious servers. This scheme needs some matrix inversion operation and new basis delegation operation, etc., while our scheme runs the *SamplePre* algorithm to generate a signature and prevents anyone including a malicious server from forging a ciphertext to realize inside KGA resistance. Since our scheme is constructed based on the hardness assumptions from lattice, it can resist quantum computers.

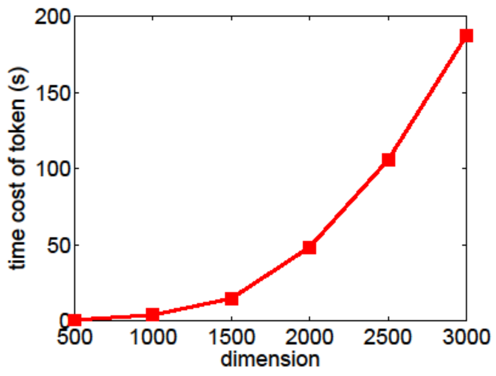
Table 1: The performance comparison with other schemes

Scheme	[Huang and Li (2017)]	[Behnia, Ozmen and Yavuz (2018)]	[Zhang and Xu (2018)]	Ours
PEKS cost	$H+3E+M_P$	$3\kappa M$	$6M+2H+2I$	$3M+2H+T_{SP}$
Token cost	$P+H+E$	T_{SL}	$T_{SP}+T_{BD}+H+2M+I$	$H+M+T_{SL}$
Test cost	$2P+M_P$	κM	$3M+T_{SP}+T_{BD}+I+H$	$M+H$
Ciphertext size	$2\|G\|$	$(2m+1)l_q+1$	$2(m+1)l_q+l_w$	$(3m+1)l_q$
Token size	$\|G\|$	$2ml_q$	ml_q	$2ml_q$
Hardness	DBDH/mDLIN	LWE	LWE	LWE/ISIS
Insider KGA	Yes	No	No	Yes
Post-quantum	No	Yes	Yes	Yes

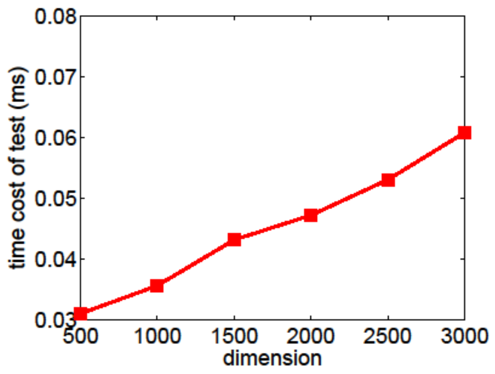
We also conduct simulation experiments on the computational cost of the related algorithms with the increase of the number of keywords using Python 3.7. The environment is Windows 10 Ultimate (×64) with an Intel (R) Core (TM) i7-8770 CPU and a 3.20-GHz processor. We choose $q=32749$ and obtain the time cost of **Enc** algorithm as Fig. 3(a), token generation algorithm as Fig. 3(b), and **Test** algorithm as Fig. 3(c). In the above simulations, each experiment is repeated 30 times under different dimensions and the average time is shown.



(a) The keywords encryption



(b) The token generation



(c)The **Test** algorithm

Figure 3: The computational cost of PEKS scheme

5 Conclusion

This paper proposed a lattice-based PEKS scheme which can resist inside keywords guessing attack and quantum computing attack. In the scheme, we use the preimage sample algorithm to generate a signature for a random bit and a part of ciphertext, which can prevent anyone including the malicious server from forging the ciphertext of keywords. Then, we give a comparison with other searchable encryption schemes and simulation experiments. As future works, we consider the construction of lattice-based PEKS scheme with rich query function.

Funding Statement: The authors would like to thank the support from Fundamental Research Funds for the Central Universities (No. 30918012204). The authors also gratefully acknowledge the helpful comments and suggestions of other researchers, which has improved the presentation.

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

References

Agrawal, S.; Boneh, D.; Boyen, X. (2010): Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. *Proceedings of the 30th Annual Cryptology Conference*, pp. 98-115.

Ajtai, M. (1996): Generating hard instances of lattice problems (extended abstract). *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pp. 99-108.

Ajtai, M. (1999): Generating hard instances of the short basis problem. *Proceeding of the 26th International Colloquium, Automata, Languages and Programming*, pp. 1-9.

Alwen, J.; Peikert, C. (2009): Generating shorter bases for hard random lattices. *Proceeding of the 26th International Symposium on Theoretical Aspects of Computer Science*, pp. 75-86.

Anada, H.; Kanaoka, A.; Matsuzaki, N.; Watanabe, Y. (2018): Key-updatable public-key encryption with keyword search: models and generic constructions. *Proceeding of the 23rd Australasian Conference Information Security and Privacy*, pp. 341-359.

Behnia, R.; Ozmen, M. O.; Yavuz, A. A. (2018): Lattice-based public key searchable encryption from experimental perspectives. *IEEE Transactions on Dependable and Secure Computing*. DOI: 10.1109/TDSC.2018.2867462.

Boneh, D.; Crescenzo, G. D.; Ostrovsky, R.; Persiano, G. (2004): Public key encryption with keyword search. *Proceeding of EUROCRYPT, International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 506-522.

Byun, J. W.; Rhee, H. S.; Park, H.; Lee, D. H. (2006): Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. *Secure Data Management, Third VLDB Workshop*, pp. 75-83.

Chen, R.; Mu, Y.; Yang, G.; Guo, F.; Wang, X. (2015): A new general framework for secure public key encryption with keyword search. *Proceeding of the 20th Australasian Conference Information Security and Privacy*, pp. 59-76.

Crescenzo, G. D.; Saraswat, V. (2007): Public key encryption with searchable keywords based on Jacobi symbols. *Proceeding of the 8th International Conference on Cryptology in India*, pp. 282-296.

Das, R.; Baykara, M.; Tuna, G. (2019): A novel approach to steganography: enhanced least significant bit substitution algorithm integrated with self-determining encryption feature. *Computer Systems Science and Engineering*, vol. 34, no. 1, pp. 23-32.

Ding, J.; Gao, X.; Takagi, T.; Wang, Y. (2019): One sample ring-LWE with rounding and its application to key exchange. *Applied Cryptography and Network Security*, pp. 323-343.

Fang, L.; Susilo, W.; Ge, C.; Wang, J. (2013): Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Sciences*, vol. 238, pp. 221-241.

Gentry, C.; Peikert, C.; Vaikuntanathan, V. (2008): Trapdoors for hard lattices and new cryptographic constructions. *Proceeding of ACM Symposium on Theory of Computing*. pp. 197-206.

Gordon, S. D.; Katz, J.; Vaikuntanathan, V. (2010): A group signature scheme from lattice assumptions. *Proceeding of the 16th International Conference on the Theory and Application of Cryptology and Information Security-ASIACRYPT*, pp. 395-412.

Gu, C.; Zheng, Y.; Kang, F.; Xin, D. (2015): Keyword search over encrypted data in cloud computing from lattices in the standard model. *Cloud Computing and Big Data, CloudCom-Asia*. pp. 335-343.

Hayouni, H.; Hamdi, M. (2018): A data aggregation security enhancing scheme in WSNs using homomorphic encryption. *Intelligent Automation and Soft Computing*, vol. 24, no. 4, pp. 729-737.

Huang, Q.; Li, H. (2017): An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, vol. 403, pp. 1-14.

Kuchta, V.; Markowitch, O. (2016): Multi-authority distributed attribute-based encryption with application to searchable encryption on lattices. *Mycrypt. Malicious and*

Exploratory Cryptology, pp. 409-435.

Mao, Y.; Fu, X.; Guo, C.; Wu, G. (2018): Public key encryption with conjunctive keyword search secure against keyword guessing attack from lattices. *Transactions on Emerging Telecommunications Technologies*, vol. 30, no, 11, pp. 1-14.

Peikert, C. (2014): Lattice cryptography for the internet. *Proceeding of Post-Quantum Cryptography—6th International Workshop*, pp. 197-219.

Regev, O. (2009): On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, vol. 56, no. 6, pp. 34:1-34:40.

Rhee, H. S.; Park, J. H.; Susilo, W.; Lee, D. H. (2009): Improved searchable public key encryption with designated tester. *Proceedings of Symposium on Information, Computer and Communications Security-ASIACCS*, pp. 376-379.

Rhee, H. S.; Park, J. H.; Susilo, W.; Lee, D. H. (2010): Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, vol. 83, no. 5, pp. 763-771.

Song, D. X.; Wagner, D.; Perrig, A. (2000): Practical techniques for searches on encrypted data. *Proceedings of IEEE Symposium on Security and Privacy*, pp. 44-55.

Wu, D. N.; Gan, Q.; Wang, X. (2018): Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting. *IEEE Access*, vol. 6, pp. 42445-42453.

Xu, L.; Xu, C.; Liu, Z.; Wang, Y.; Wang, J. (2019): Enabling comparable search over encrypted data for IoT with privacy-preserving. *Computers, Materials & Continua*, vol. 60, no. 2, pp. 675-690.

Xu, L.; Yuan, X.; Steinfeld, R.; Wang, C.; Xu, C. (2019): Multi-writer searchable encryption: an LWE-based realization and implementation. *Proceedings of the Asia Conference on Computer and Communications Security-AsiaCCS*, pp. 122-133.

Yang, Y.; Ma, M. (2016): Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. *IEEE Transactions Information Forensics and Security*, vol. 11, no. 4, pp. 746-759.

Yang, Y.; Zheng, X.; Chang, V.; Ye, S.; Tang, C. (2018): Lattice assumption based fuzzy information retrieval scheme support multi-user for secure multimedia cloud. *Multimedia Tools and Applications*, vol. 77, no. 8, pp. 9927-9941.

Zhang, J.; Zhang, Z.; Ding, J.; Snook, M.; Dagdelen, O. (2015): Authenticated key exchange from ideal lattices. *Proceeding of EUROCRYPT, International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 719-751.

Zhang, X.; Xu, C. (2018): Trapdoor security lattice-based public-key searchable encryption with a designated cloud server. *Wireless Personal Communications*, vol. 100, no. 3, pp. 907-921.

Zhang, X.; Xu, C.; Mu, L.; Zhao, J. (2018): Identity-based encryption with keyword search from lattice assumption. *China Communications*, vol. 15, no. 4, pp. 164-178.

Zhang, X.; Xu, C.; Wang, H.; Zhang, Y.; Wang, S. (2019): 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*.