Lattice-based Public Key Encryption with Authorized Keyword Search: Construction, Implementation, and Applications

Anonymous Author(s)
Anonymous Institution(s)

ABSTRACT

Public key encryption with keyword search (PEKS), formalized by Boneh et al. [EUROCRYPT' 04], enables secure searching for specific keywords in the ciphertext. Nevertheless, in certain scenarios, varying user tiers are granted disparate data searching privileges, and administrators need to restrict the searchability of ciphertexts to select users exclusively. To address this concern, Jiang et al. [ACISP' 16] devised a variant of PEKS, namely public key encryption with authorized keyword search (PEAKS), wherein solely authorized users possess the ability to conduct targeted keyword searches. Nonetheless, it is vulnerable to resist quantum computing attacks. As a result, research focusing on authorizing users to search for keywords while achieving quantum security is far-reaching.

In this work, we present a novel construction, namely lattice-based PEAKS (L-PEAKS), which is the first mechanism to permit the authority to authorize users to search different keyword sets while ensuring quantum-safe properties. Specifically, the keyword is encrypted with a public key, and each authorized user needs to obtain a search privilege from an authority. The authority distributes an authorized token to a user within a time period and the user will generate a trapdoor for any authorized keywords. Technically, we utilize several lattice sampling and basis extension algorithms to fight against attacks from quantum adversaries. Moreover, we leverage identity-based encryption (IBE) to alleviate the bottleneck of public key management. Furthermore, we conduct parameter analysis, security reduction, and theoretical complexity comparison of our scheme and perform comprehensive evaluations of a commodity machine for completeness. Our L-PEAKS satisfies IND-sID-CKA and T-EUF security and is efficient in terms of space and computation complexity compared to other existing primitives.

CCS CONCEPTS

• **Security and Privacy → Management and querying of encrypted data; Data security**.

KEYWORDS

Searchable encryption, Authorization, Data management, Post-quantum security

XXX, XXX, XXX

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Anonymous Author(s). 2018. Lattice-based Public Key Encryption with Authorized Keyword Search: Construction, Implementation, and Applications. In *Proceedings of XXX*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the big data era, outsourced data retrieval and sharing have been widely used due to its well-understood benefits, i.e. convenient storage, and on-demand access [38]. Meanwhile, it also brings serious data security issues, such as private information disclosure [7], [20], [35]. In order to ensure the security for outsourced data management [37], the concept of public key encryption with keyword search (PEKS) scheme was proposed, which allows for searching encrypted data [6]. Specifically, a sender uploads encrypted data with searchable ciphertext. Then, a user sends a trapdoor associated with a specific keyword to the server for searching. If the keyword in the trapdoor matches it in the ciphertext, the server will return the corresponding encrypted data to the receiver [34].

However, in traditional PEKS schemes, a ciphertext is generated through a user's public key and only one corresponding secret key can search it, which will cause inconvenience in real-world scenarios, particularly in enterprises where multiple employees are required to hold the search right for encrypted keywords based on the enterprises' public keys. Additionally, each user must obtain the enterprise's secret key to generate the trapdoor. This trivial protocol not only suffers from the abuse of secret keys but also results in casual access for users. For instance, each user may be at a different access level and can only search for limited keywords.

To get around this issue, Jiang et al. proposed a PEAKS scheme where an administrator holds the enterprise's secret key and grants search privileges to users [18]. In their scheme, the administrator sets up an authorized dataset for each user, who can only search for authorized datasets. Then, it distributes authorization tokens for each authorized keyword, which are only valid for a short time interval to ensure security. Once the time expires, the administrator must re-authorize a token for the user and thereby keep the timeliness of signatures. Nevertheless, all existing PEAKS schemes [18], [29], [13], [16], [19], [21], [14] are based on the discrete logarithmic (DL) assumptions, which have been proven vulnerable to attacks by quantum computers [28], [31].

In this paper, we aim to design a post-quantum PEAKS scheme that supports user-authorization for outsourced data management. A straightforward approach is to adopt the conventional PEKS scheme, that is, treating identities as keywords and utilizing IBE for equality testing [6], [1]. However, directly implementing lattice-based PEKS can lead to significant complexity in key management [27], [11]. Specifically, when considering identities as keywords, the system parameter and master secret key need to be treated as the user's public and secret keys, respectively [33]. In the context of

lattice-based instantiations, simply using identity-based encryption (IBE) schemes (e.g. [2], [9]) will result in a considerable computational overhead for generating public and secret keys, as they consist of numerous large matrices and lattice basis.

To address the above-mentioned issues, we propose a lattice-based PEAKS scheme for outsourced data management, realizing a searchable encryption scheme supporting user-authorization in a quantum setting. Specifically, we incorporate a lattice-based signature into the PEKS to realize the authorization, achieved by defining two different datasets and utilizing lattice basis extension algorithms. Each authorized user needs to apply for a search privilege from an authority. The authorization process is carried out by the authority, which issues a token to the user and updates it in a timely manner. We highlight our threefold contributions as follows:

- We design a lattice-based public key encryption with authorized keyword search (L-PEAKS) scheme, enabling an authority to authorize users to search for specific sets of keywords while resisting to quantum attack. To reduce the storage overhead of public key certificates, we employ an IBE structure and leverage the users' identities as their public keys.
- Moreover, we formalize and prove the security of L-PEAKS, including indistinguishability against selective identity and chosen keywords attack (IND-sID-CKA), and trapdoor existential unforgeability (T-EUF). We further compare our scheme with prior art to showcase our superiority.
- Ultimately, we implement our scheme in two moderate security parameter settings ($n = 256$ and $n = 320$) and open-source the code. Besides, we conduct a theoretical analysis of the space and computational complexities in comparison to five other mechanisms. The results indicate that our scheme's secret key and ciphertext sizes outperform those of the other mechanisms, while the size of the trapdoor is also relatively prominent.

2 RELATED WORKS

Jiang et al. initialized a PEAKS scheme, in which the authority can assign tokens with valid time limits to each authorized keyword [18]. Following this work, Cui et al. formalized attribute-based encryption with an expressive and authorized keyword search scheme, namely ABE-EAKS, which allows the expressive keyword search and fine-grained access control in the cloud computing [13]. In 2019, Xu et al. presented a scheme that enables keyword search by the authority and authorized users [32]. Recently, Liu et al. proposed a public key encryption with a hierarchical authorized keyword search (PEHAKS) scheme to render authorization more flexible in real scenarios [21]. After that, Wang et al. proposed a novel authorized keyword search protocol over encrypted data with metadata, achieving a secure and flexible mechanism to simultaneously process different fields of metadata [29]. To our best knowledge, none of the existing searchable encryption with user-authorization schemes can resist quantum computing attacks.

3 PRELIMINARIES

3.1 Lattice and Sampling Algorithms

Definition 1. *[3] Suppose that $\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b}_n \in \mathbb{R}^m$ are n linearly independent vectors. The m-dimensional lattice Λ is generated by a set of linear combinations, denoted as $\Lambda = \{x_1 \cdot \mathbf{b_1} + x_2 \cdot \mathbf{b_2} + \cdots + x_n \cdot \mathbf{b}_n | x_i \in \mathbb{Z}\}$, where $\mathbf{B} = \{\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b}_n\} \in \mathbb{R}^{m \times n}$ is the basis of Λ.*

Definition 2. *Given a center $\mathbf{c} \in \mathbb{Z}^m$, a positive parameter $s \in \mathbb{R}^+$, and any $\mathbf{x} \in \mathbb{Z}^m$, we define $\mathcal{D}_{\mathbf{c},s} = \rho_{\mathbf{c},s}(x)/\rho_{\mathbf{c},s}(\Lambda)$ for $\forall x \in \Lambda$ as the Discrete Gaussian Distribution over Λ with a center $\mathbf{c}$, where $\rho_{\mathbf{c},s}(x) = \exp(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{s^2})$ and $\rho_{\mathbf{c},s}(\Lambda) = \sum_{x \in \Lambda} \rho_{\mathbf{c},s}(x)$. Specially, we say $\mathcal{D}_{\mathbf{0},s}$ abbreviated as $\mathcal{D}_s$ when $\mathbf{c} = 0$.*

Lemma 1. *[24] Given parameters $n, m, q \in \mathbb{Z}$, this* PPT *algorithm returns $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T_A} \in \mathbb{Z}_q^{m \times m}$, where $\mathbf{T_A}$ is a basis of $\Lambda_q^{\perp}(\mathbf{A})$* s.t. *$\{\mathbf{A} : (\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)\}$ is statistically close to $\{\mathbf{A} : \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}\}$. We say $\mathbf{T_A}$ is a trapdoor of $\mathbf{A}$.*

Lemma 2. *[15] Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its trapdoor $\mathbf{T_A} \in \mathbb{Z}_q^{m \times m}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and the parameter $s > \|\tilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m)})$, where $m \geq 2n\lceil \log q \rceil$, the SamplePre($\mathbf{A}, \mathbf{T_A}, \mathbf{u}, s$) algorithm publishes a sample $\mathbf{e} \in \mathbb{Z}_q^m$ statistically distributed in $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),s}$* s.t. *$\mathbf{Ae} = \mathbf{u} \mod q$.*

Lemma 3. *[8] Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its corresponding trapdoor $\mathbf{T_A} \in \mathbb{Z}_q^{m \times m}$, a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m_1}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a parameter $s \leq \|\tilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m + m_1)})$, the SampleLeft($\mathbf{A}, \mathbf{M}, \mathbf{T_A}, \mathbf{u}, s$) algorithm will output a sample $\mathbf{t} \in \mathbb{Z}^{m+m_1}$ from the distribution statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{u}}([\mathbf{A}|\mathbf{M}]),s}$* s.t. *$[\mathbf{A}|\mathbf{M}] \cdot \mathbf{t} = \mathbf{u} \mod q$.*

Lemma 4. *[8] Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times k}$, a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and its corresponding trapdoor $\mathbf{T_B} \in \mathbb{Z}_q^{m \times m}$, a matrix $\mathbf{R} \in \mathbb{Z}_q^{k \times m}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a parameter $s \leq \|\tilde{\mathbf{T_B}}\| \cdot s_R \cdot \omega(\sqrt{\log(m + m_1)})$, where $s_R = \sup_{\|\mathbf{x}\|=1} \|\mathbf{Rx}\|$, the SampleRight($\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \mathbf{u}, s$) algorithm will output a sample $\mathbf{t} \in \mathbb{Z}^{m+k}$ from the distribution statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{u}}([\mathbf{A}|\mathbf{AR}+\mathbf{B}]),s}$* s.t. *$[\mathbf{A}|\mathbf{AR} + \mathbf{B}] \cdot \mathbf{t} = \mathbf{u} \mod q$.*

Lemma 5. *[9] Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a basis $\mathbf{S} \in \mathbb{Z}_q^{m \times m}$ of $\Lambda^{\perp}(\mathbf{A})$, and a matrix $\mathbf{A'} \in \mathbb{Z}_q^{n \times m'}$, the ExtBasis($\mathbf{A''}$,$\mathbf{S}$) algorithm outputs a basis $\mathbf{S''}$ of $\Lambda^{\perp}(\mathbf{A''}) \subseteq \mathbb{Z}_q^{m \times m''}$* s.t. *$\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S''}}\|$, and $\mathbf{A''} = \mathbf{A}||\mathbf{A'}$, $m'' = m + m'$.*

3.2 The LWE and SIS Hardness Assumptions

Definition 3. *[27] Suppose there exists an integer $q = q(n)$ and an error distribution $\bar{\Psi}_\alpha$ in $\mathbb{Z}_q$, the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-*LWE *contributes to distinguishing the distribution $(\mathbf{u}_i, v_i) = (\mathbf{u}_i, \mathbf{u}_i^\top \mathbf{s} + x_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ randomly and uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Among that, $\bar{\Psi}_\alpha$ is the distribution of $\lfloor qX \rceil \mod q$ over $\mathbb{Z}_q$, where X is normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$.*

Definition 4. *[27] Given a prime q, a parameter $\alpha \in (0, 1)$, we denote $\bar{\Psi}_\alpha$ as the distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil \mod q$, where X is a normal random variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$.*

Lemma 6. *[2] Given a vector $\mathbf{e} \in \mathbb{Z}^m$ and a uniformly random vector $\mathbf{y} \xleftarrow{\$} \bar{\Psi}_\alpha^m$, the parameter $|\mathbf{e}^\top \mathbf{y}|$ is deemed as an integer in $\{0, q-1\}$, satisfying $|\mathbf{e}^\top \mathbf{y}| \leq \|e\| \frac{\sqrt{m}}{2} + \|e\| q\alpha \cdot \omega(\sqrt{\log m})$, except with negligible probability* $\mathsf{negl}(m)$.

Definition 5. *[26] Suppose a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a bound parameter $\beta > 0$, (q, n, m, β)-*SIS *contributes to finding a vector $\mathbf{v} \in$*

$\mathbb{Z}^m \backslash \{0\}$ *such that* $\mathbf{Av} = 0$ *and* $\|\mathbf{v}\| \leq \beta$. *Among that, in order to ensure the existence of vector* $\mathbf{v}$, *we set* $\beta \geq \sqrt{m} q^{n/m}$.

Definition 6. *[26] Given* $H := \{h : X \to Y\}$ *be a hash function family from* X *to* Y, *we define that* H *is a* (q_t, p_{min}, p_{max}) *abort-resistant function, if the non-abort probability of* x *satisfies* $p(x) = \Pr[h(x_0) = 0 \wedge h(x_1) \neq 0 \wedge h(x_2) \neq 0 \wedge \cdots \wedge h(x_{q_t}) \neq 0] \in [p_{min}, p_{max}]$, *where the input is* $x = (x_0, x_1, \cdots, x_{q_t})$ *with* $x_0 \notin \{x_1, x_2, \cdots, x_{q_t}\}$ *and the probability is over the selection of* $h \in H$.

3.3 Rejection Sampling Technique

Lemma 7. *[23] Given a positive integer* m, *for any* $s > 0$, *we have* $\Pr[x \leftarrow \mathcal{D}_s^1 : |x| > \omega(s\sqrt{\log m})] = 2^{-\omega(\log m)}$, $\Pr[x \leftarrow \mathcal{D}_s^1 : |x| > 12s] < 2^{-100}$, *and* $\Pr[\mathbf{x} \leftarrow \mathcal{D}_s^m : \|\mathbf{x}\| > 2s\sqrt{m}] < 2^{-m}$.

Lemma 8. *[23] Given a positive real number* α, *for any vector* $c \in \mathbb{Z}^m$, *if* $s = \omega(\|\mathbf{v}\|\sqrt{\log m})$, *we have* $\Pr[\mathbf{x} \leftarrow \mathcal{D}_s^m : \mathcal{D}_s^m(\mathbf{x})/\mathcal{D}_{s,\mathbf{c}}^m(\mathbf{x}) = O(1)] = 1 - 2^{\omega(\log m)}$, *and if* $s = \alpha\|\mathbf{c}\|$, *we have* $\Pr[\mathbf{x} \leftarrow \mathcal{D}_s^m : \mathcal{D}_s^m(\mathbf{x})/\mathcal{D}_{s,\mathbf{c}}^m(\mathbf{x}) < e^{12/\alpha + 1/(2\alpha^2)}] > 1 - 2^{-100}$.

The core idea of the rejection sampling technique for signature protocols is to make the distribution of output signatures independent of the signing key. Concretely, to sign a message μ, a signer with its secret key sk initially selects a random parameter y. Then, it computes the signature σ which is a combination of y and sk. Let the final signature distribution be f, which is independent of sk, and let the whole candidate signatures distribution be g, which may be related to sk. If f and g are both probability distributions and it satisfies $f(x) \leq Mg(x)$ for all x and positive number M, then the candidate signature parameter σ can be published with $f(\sigma)/(Mg(\sigma))$. In this way, the resulting distribution is f, and the expected number of this process outputs a sample is M [25].

4 FRAMEWORK DESCRIPTION

4.1 System Architecture

The basic system model of our proposed scheme is illustrated in Fig. 1, which involves four entities, that is, sender, server, authority, and users. We introduce their specific workflows below.

- Sender: It uploads the ciphertext and encrypted keyword to the server for storage, allowing the data to be accessed by all users.
- Authority: It serves as a system manager and authorizes users in a timely manner by issuing a unique authorization token. Note that the master secret key is kept confidential.
- User: It calculates the trapdoor for the authorized dataset and submits it to the server for searching operation. Note that only authorized users can obtain a valid trapdoor.
- Server: It performs authorization verification and keyword matching operations to determine whether to provide the corresponding data to the user. The server is honest but curious, meaning it performs search operations honestly but may try to infer keywords from encrypted keywords.

The proposed scheme offers data indexing services to authorized users and ensures that there is no collaboration among users. When a sender uploads the data, it includes encrypted data CT for sharing and encrypted keywords for indexing. The authority grants different users access to specific datasets of keywords, and each user is

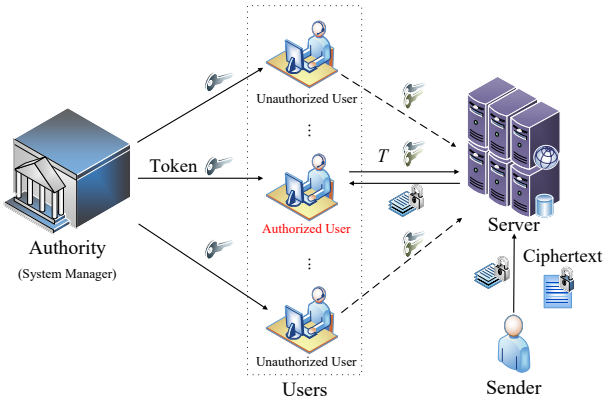


Figure 1: Architecture overview of the proposed scheme

authorized to use its respective dataset. During the authorization process at time period t, the authority generates an authorization token for the user. The authorized user then generates a trapdoor by the token and a keyword $\mathbf{kw}$ from the dataset $\mathcal{KS}_W$, and submits the trapdoor Trap to the server. Upon the server received the trapdoor, if the authorization is fresh and CT is valid, then it returns the corresponding encrypted data to the user.

4.2 Algorithm Descriptions

Our L-PEAKS scheme consists of six algorithms, Π = (Setup, KeyGen, Authorize, Encrypt, Trapdoor, Test).

- Setup(λ): Given a security parameter λ, this PPT algorithm outputs a public parameter pp and a master secret key MSK.
- KeyGen(pp, MSK, $\mathbf{ID}$): Given a public parameter pp, a master secret key MSK, and a user $\mathbf{ID}$, this PPT algorithm outputs a secret key $\mathbf{sk_{ID}}$ for the user $\mathbf{ID}$.
- Authorize(pp, $\mathbf{sk_{ID}}$, $\mathcal{KS}_W$, t): Given a public parameter pp, a secret key $\mathbf{sk_{ID}}$ of user $\mathbf{ID}$, an authorized dataset $\mathcal{KS}_W$, and the authorized time t, this PPT algorithm outputs an authorized token $token$.
- Encrypt(pp, $\mathbf{ID}$, $\mathbf{kw}$): Given a public parameter pp, a user $\mathbf{ID}$, and a keyword $\mathbf{kw}$, this PPT algorithm outputs a ciphertext CT.
- Trapdoor(pp, $\mathbf{ID}$, $token$, $\mathbf{kw}'$): Given a public parameter pp, a user $\mathbf{ID}$, an authorized token $token$, and a keyword $\mathbf{kw}' \in \mathcal{KS}_W$, this PPT algorithm outputs a trapdoor Trap.
- Test(pp, $\mathbf{ID}$, CT, Trap, t'): Given a public parameter pp, a user $\mathbf{ID}$, a searchable ciphertext CT, a trapdoor Trap, and the trapdoor-received time t', this deterministic algorithm outputs 1 or 0 based on judgment conditions.

Correctness. The correctness of our proposed scheme must satisfy that for system initialization (pp, MSK) $\leftarrow$ Setup, user registration $\mathbf{sk_{ID}} \leftarrow$ KeyGen, dataset authorization $token \leftarrow$ Authorize, ciphertext generation CT $\leftarrow$ Encrypt, and trapdoor generation Trap $\leftarrow$ Trapdoor, we have Test = 1.

4.3 Security Models

We now define two games from terms of indistinguishability against selective identity and chosen keywords attack (IND-sID-CKA) as well as trapdoor existential unforgeability (T-EUF) below.

4.3.1 IND-sID-CKA Game. In this game, an adversary $\mathcal{A}$ has the permission to launch chosen keyword attacks and it plays this game with a challenger C as below.

- **Setup**. The adversary $\mathcal{A}$ announces a challenge dataset $\mathcal{KS}^*_W \subseteq \mathcal{KS}_U$. Then, the challenger C executes Setup algorithm to generate (pp, MSK) and sends pp to $\mathcal{A}$. After that, $\mathcal{A}$ chooses a challenge user $\mathbf{ID}^*$.
- **Phase 1**. $\mathcal{A}$ performs a polynomially bounded number of queries.
  - **KeyGen Query**. $\mathcal{A}$ enquires the secret key of a user $\mathbf{ID} \neq \mathbf{ID}^*$. C executes KeyGen algorithm to generate $\mathbf{sk_{ID}}$ and responds it to $\mathcal{A}$.
  - **Authorization Query**. $\mathcal{A}$ sends the dataset $\mathcal{KS}_W = \mathcal{KS}_U - \mathcal{KS}^*_W$ and an authorized time t to C. Then, C executes Authorize algorithm to generate the authorized token $token$ and responds it to $\mathcal{A}$.
  - **Trapdoor Query**. $\mathcal{A}$ sends a keyword $\mathbf{kw} \in \mathcal{KS}_W$ and an authorized time t to C. C executes Trapdoor algorithm to generate Trap and responds it to $\mathcal{A}$.
- **Challenge**. $\mathcal{A}$ generates and sends two equal length keywords $\mathbf{kw}_0, \mathbf{kw}_1$ on which it wishes to be challenged. The restrictions are that $\mathcal{A}$ did not query the authorized token for $\mathcal{KS}^*_W$, or the trapdoor for $\mathbf{kw}_0, \mathbf{kw}_1$. C selects a random bit $b \in \{0, 1\}$ and calculates a searchable ciphertext CT^* for $\mathbf{kw}_b$ with user $\mathbf{ID}^*$. After that, C responds CT^* to $\mathcal{A}$ as the challenge ciphertext.
- **Phase 2**. $\mathcal{A}$ continues to perform the **Authorization Query** and **Trapdoor Query** for any keyword $\mathbf{kw}$ excepts for $\mathbf{kw_0}, \mathbf{kw_1}$.
- **Guess**. Ultimately, $\mathcal{A}$ outputs a guess bit $b' = \{0, 1\}$ and it wins the game if $b' = b$. We call this kind of adversary $\mathcal{A}$ as IND-sID-CKA adversary. The advantage for $\mathcal{A}$ in attacking this scheme is defined as a function related to the security parameter λ: $Adv^{\text{IND-sID-CKA}}_{\mathcal{A}}(\lambda) := |\Pr[b = b'] - \frac{1}{2}|$.

Definition 7 (**IND-sID-CKA security**). *Our lattice-based PEAKS scheme is* IND-sID-CKA secure, if for any PPT adversary $\mathcal{A}$, the advantage $Adv^{\text{IND-sID-CKA}}_{\mathcal{A}}(\lambda)$ is negligible.

4.3.2 T-EUF Game. In the T-EUF game, an adversary $\mathcal{A}$ has the permission to launch impersonation attack. Markedly, if $\mathcal{A}$ has been authorized, it is difficult to prevent the impersonation attack. $\mathcal{A}$ plays this game with a challenger C as below.

- **Setup**. The adversary $\mathcal{A}$ announces a challenge dataset $\mathcal{KS}^*_W \subseteq \mathcal{KS}_U$. Then, the challenger C executes Setup algorithm to generate (pp, MSK) and sends pp to $\mathcal{A}$. After that, $\mathcal{A}$ chooses a challenge user $\mathbf{ID}^*$.
- **Query**. $\mathcal{A}$ performs a polynomially bounded number of queries.
  - **Hash Query**. C maintains a list $L(\mathbf{SK_{ID}}, t, h)$, which is initially empty. When $\mathcal{A}$ queries about $(\mathbf{SK_{ID}}, t)$, C searches the list L and then responds the answer to $\mathcal{A}$.
  - **KeyGen Query**. $\mathcal{A}$ enquires the secret key of a user $\mathbf{ID} \neq \mathbf{ID}^*$. C executes KeyGen algorithm to generate $\mathbf{sk_{ID}}$ and responds it to $\mathcal{A}$.
  - **Authorization Query**. $\mathcal{A}$ issues a dataset and an authorized time t to C. Then, C executes Authorize algorithm to generate the authorized token $token$ and responds it to $\mathcal{A}$.
  - **Trapdoor Query**. $\mathcal{A}$ sends a keyword $\mathbf{kw} \in \mathcal{KS}_W$ and an authorized time t to C. C executes Trapdoor algorithm to generate Trap and responds it to $\mathcal{A}$.
- **Forgery**. $\mathcal{A}$ outputs a trapdoor tuple for dataset $\mathcal{KS}_{W^*}$, which has not been queried before.

Definition 8 (**T-EUF security**). *Our lattice-based PEAKS scheme is* T-EUF secure, if there is no PPT adversary $\mathcal{A}$ who has the ability to forge a valid trapdoor with a correct signature and authorized time with a non-negligible advantage.

5 THE PROPOSED SCHEME

In this sector, we illustrate the concrete PEAKS scheme from the lattice hardness. It allows an administrative authority to authorize users to search different datasets of keywords. Users without the corresponding secret keys must be authorized by the authority to search the keywords. Besides, considering the troublesome certificate management of public key encryption, we leverage IBE to simplify the key management and encrypt the data directly through its identity, which is more suitable for practical scenarios.

5.1 System Initialization

The initialization stage of our scheme is conducted by the authority, which takes a security parameter λ as input and then sets several parameters $\alpha = [m^2k^2\omega(\log n)]^{-1}$, $q > 2\sqrt{n}/\alpha$, $m = \lceil 6n \log q \rceil$, $s = km\omega(\sqrt{\log n})$, $q \geq m^{2.5}\omega(\sqrt{\log n})$, $M \approx \exp\left(\frac{12\kappa\sqrt{m}}{s} + \left(\frac{\kappa\sqrt{m}}{2s}\right)^2\right)$. Each user has its own identity vector $\mathbf{ID}$. Then it generates and outputs the public parameters pp to other entities and keeps MSK in secret, as described in Algorithm 1. As mentioned in Lemma 1, $\text{TrapGen}(n, m, q)$ is run to generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with a short basis $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$.

Algorithm 1 Setup(λ): Initialize the system parameters

Input: Security parameter λ, system parameters n, m, s, prime number q, and Gaussian distribution parameter s.
Output: Public parameter pp and master secret key MSK.

1: Define l as the bit length of $\mathbf{ID}$ and a a universal dataset $\mathcal{KS}_U$;
2: Invoke $(\mathbf{A}, \mathbf{T_A}) \leftarrow \text{TrapGen}(n, m, q)$ algorithm to obtain a matrix $\mathbf{A}$ and its basis $\mathbf{T_A}$ for $\Lambda_q^{\perp}(\mathbf{A})$.
3: Select two uniformly random matrices $\mathbf{B} \in \mathbb{Z}_q^{n\times 2m}$ and $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$, where k is the bit length of keyword $\mathbf{kw}$;
4: Select k uniformly random matrices $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k \in \mathbb{Z}_q^{n\times 2m}$, and k uniformly random vectors $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k \in \mathbb{Z}_q^m$;
5: Select a uniformly random vector $\mathbf{u} \in \mathbb{Z}_q^n$;
6: Select four collision-resistant hash functions: $H_1 : \{-1, 1\}^l \rightarrow \mathbb{Z}_q^n, H_2 : \mathbb{Z}_q^n \times \{0, 1\}^* \rightarrow \{h : h \in \mathbb{Z}_q, |h| \leq \kappa\}, H_3 : \mathbb{Z}_q^m \rightarrow \mathbb{Z}_q^{m\times m}, H_4 : \{-1, 1\}^l \rightarrow \mathbb{Z}_q^{n\times m}$;
7: **Return** $pp = (\mathbf{A}, \mathbf{B}, \mathbf{U}, \mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k, \mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k, \mathbf{u}, H_1, H_2, H_3, H_4, s)$, $\text{MSK} = \mathbf{T_A}$.

5.2 User Registration

In this function, the authority computes secret keys for users registration. Note that a user's identity **ID** is unique in the system and also serves as its public key. Once a user **ID** submits its registration, the authority takes the master secret key and the user's identity as input, and returns the corresponding secret key $\mathbf{sk_{ID}}$ to the user. As described in Algorithm 2, the authority runs the $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, H_1(\mathbf{ID}), s)$ algorithm to sample $\mathbf{sk_{ID}}$.

Algorithm 2 KeyGen(pp, MSK, **ID**): Register the user's key

Input: Public parameter pp, master secret key MSK, and user **ID**.
Output: Secret key $\mathbf{sk_{ID}}$ of user **ID**.

1: Parse $\mathbf{ID} = (\mathrm{id}_1, \mathrm{id}_2, \cdots, \mathrm{id}_l) \in \{-1, 1\}^l$ as the identity (public key) of a user;
2: Sample a vector $\mathbf{sk_{ID}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, H_1(\mathbf{ID}), s)$, where $\mathbf{sk_{ID}} \in \mathbb{Z}_q^m$, and $\mathbf{sk_{ID}}$ s.t. $\mathbf{A} \cdot \mathbf{sk_{ID}} = H_1(\mathbf{ID}) \bmod q$;
3: **Return** $\mathbf{sk_{ID}}$ as a secret key of the user **ID**.

5.3 User Authorization

For the valid user, the authority would authorize the user to access the dataset $\mathcal{KS}_W$ ($\mathcal{KS}_W \subseteq \mathcal{KS}_U$) and the concrete steps are specified in Algorithm 3. It firstly set a polynomial $f(\mathbf{sk_{ID}}, \mathbf{kw}) = \mathbf{sk_{ID}} + \sum_{i=1}^{k} kw_i \mathbf{m}_i \in \mathbb{Z}_q^m$, where $\mathbf{kw} = (kw_1, kw_2, \cdots, kw_k)$, and $kw_i \in \{0, 1\}$ is each bit of **kw**. After that, the authority defines two polynomials $f_1(\mathbf{sk_{ID}})$ and $f_2(\mathbf{sk_{ID}})$ corresponding to the different keyword datasets, and then subtract them to set another polynomial $F(\mathbf{sk_{ID}})$. Accordingly, the authority calculates the signature σ, which involves a time period t. With time updates, the authority will re-authorize the user with a novel authorization token to ensure the signature timeliness. Finally, the authority sends an authorized token *token* to the user through a secure communication channel.

Algorithm 3 Authorize(pp, $\mathbf{sk_{ID}}$, $\mathcal{KS}_W$, t): Authorize the user

Input: Public parameter pp, secret key $\mathbf{sk_{ID}}$ of user **ID**, and an authorized dataset $\mathcal{KS}_W$ with the authorized time t.
Output: Authorized token *token*.

1: Set a polynomial as $f(\mathbf{sk_{ID}}, \mathbf{kw}) = \mathbf{sk_{ID}} + \sum_{i=1}^{k} kw_i \mathbf{m}_i \in \mathbb{Z}_q^m$;
2: Set three polynomials as: $f_1(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \mathcal{KS}_U} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m$, $f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \mathcal{KS}_W} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m$, and $F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}}) \in \mathbb{Z}_q^m$;
3: Calculate an authorized key $\mathbf{SK_{ID}} = H_3(F(\mathbf{sk_{ID}})) \in \mathbb{Z}_q^{m \times m}$;
4: Select a random vector $\mathbf{y} \in \mathbb{Z}_q^m$;
5: Calculate a parameter $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t) \in \mathbb{Z}_q$, and a vector $\mathbf{z} = \mathbf{sk_{ID}} \cdot h + \mathbf{y} \in \mathbb{Z}_q^m$;
6: Calculate a signature σ of an authorized key $\mathbf{SK_{ID}}$ as $\sigma = (h, \mathbf{z}) = (H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t), \mathbf{sk_{ID}}h + \mathbf{y})$ with probability $\min(1, \frac{\mathcal{D}_s^m(\mathbf{z})}{M\mathcal{D}_{\mathbf{sk_{ID}}h,s}^m(\mathbf{z})})$;
7: **Return** an authorized token *token* = $(\mathbf{SK_{ID}}, \sigma, \mathbf{y}, t)$.

5.4 Ciphertext Generation

In this part, we show how to construct searchable ciphertext for the keyword **kw** in the dataset $\mathcal{KS}_U$. The concrete steps are illustrated in Algorithm 4. For a given keyword **kw** with user **ID**, the data sender firstly calculates two matrix $\mathbf{A_{kw}} = \sum_{i=1}^{k} kw_i \mathbf{M}_i + \mathbf{B} \in \mathbb{Z}_q^{n \times 2m}$, and $\mathbf{A_{ID}} = (\mathbf{A} \mid H_4(\mathbf{ID})) \in \mathbb{Z}_q^{n \times 2m}$, where $kw_i \in \{0, 1\}$ is each bit of **kw**. Then, it sets a new matrix $\mathbf{A_{ID,kw}}$ as a combination of $\mathbf{A_{kw}}$ and $\mathbf{A_{ID}}$. After that, it generates a matrix $\mathbf{R_{kw}}$ and selects several noise parameters. Ultimately, the authority computes two ciphertexts $\mathrm{c}_0 = \mathbf{u}^\top \mathbf{r} + noi$, and $\mathbf{c}_1 = \mathbf{A}_{\mathbf{ID,kw}}^\top \cdot \mathbf{r} + (\mathbf{noi}, \mathbf{R}_{\mathbf{kw}}^\top \cdot \mathbf{noi})$ and uploads the ciphertext tuple $\mathrm{CT} = (\mathrm{c}_0, \mathbf{c}_1)$ to the server.

Algorithm 4 Encrypt(pp, **ID**, **kw**): Generate the ciphertext

Input: Public parameter pp, user **ID**, and keyword **kw** in the dataset $\mathcal{KS}_U$.
Output: Ciphertext tuple CT.

1: Calculate a matrix $\mathbf{A_{kw}} = \sum_{i=1}^{k} kw_i \mathbf{M}_i + \mathbf{B} \in \mathbb{Z}_q^{n \times 2m}$;
2: Calculate a matrix $\mathbf{A_{ID}} = (\mathbf{A} \mid H_4(\mathbf{ID})) \in \mathbb{Z}_q^{n \times 2m}$;
3: Set a matrix $\mathbf{A_{ID,kw}} = (\mathbf{A_{ID}} | \mathbf{A_{kw}}) \in \mathbb{Z}_q^{n \times 4m}$;
4: **for all** $i = 1, 2, \cdots, k$ **do**
5: Uniformly random select k matrices $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{2m \times 2m}$;
6: **end for**
7: Calculate a matrix $\mathbf{R_{kw}} = \sum_{i=1}^{k} kw_i \cdot \mathbf{R}_i$;
8: Uniformly random select a noise value $noi \xleftarrow{\$} \bar{\Psi}_\alpha \in \mathbb{Z}_q$ and a noise vector $\mathbf{noi} \xleftarrow{\$} \bar{\Psi}_\alpha^{2m} \in \mathbb{Z}_q^{2m}$;
9: Uniformly random select a vector $\mathbf{r} \in \mathbb{Z}_q^n$;
10: Calculate two ciphertexts $\mathrm{c}_0 = \mathbf{u}^\top \mathbf{r} + noi \in \mathbb{Z}_q$, and $\mathbf{c}_1 = \mathbf{A}_{\mathbf{ID,kw}}^\top \cdot \mathbf{r} + (\mathbf{noi}, \mathbf{R}_{\mathbf{kw}}^\top \cdot \mathbf{noi}) \in \mathbb{Z}_q^{4m}$;
11: **Return** a ciphertext tuple $\mathrm{CT} = (\mathrm{c}_0, \mathbf{c}_1)$.

5.5 Trapdoor Generation

In this function, a user **ID** generates a trapdoor tuple Trap for some authorized keyword **kw** in the dataset $\mathcal{KS}_W$ and then submits it to the server for searching, where only authorized user can generate a valid trapdoor. The detailed procedures are depicted in Algorithm 5. Initially, The user calculates two matrices $\mathbf{A_{kw}}$ and $\mathbf{A_{ID}}$. Then, it invokes the $\mathsf{ExtBasis}(\mathbf{A_{ID}}, \mathbf{T_A})$ algorithm to calculate $\mathbf{T}_{\mathbf{A_{ID}}} \in \mathbb{Z}_q^{2m \times 2m}$. Subsequently, it calculates two vectors $\mathbf{trap_1}$ and $\mathbf{trap_2}$ by multiplication operation and $\mathsf{SampleLeft}(\mathbf{A_{ID}}, \mathbf{A_{kw}}, \mathbf{T}_{\mathbf{A_{ID}}}, \mathbf{u}, s)$ algorithm mentioned in Lemma 3, respectively. Eventually, the user sends a trapdoor $\mathrm{Trap} = (\mathbf{trap_1}, \mathbf{trap_2}, \sigma, t)$ to the server.

Algorithm 5 Trapdoor(pp, **ID**, *token*, **kw**): Generate the trapdoor

Input: Public parameter pp, user **ID**, authorized token *token*, and keyword **kw** in the dataset $\mathcal{KS}_W$.
Output: Trapdoor tuple Trap.

1: Calculate a matrix $\mathbf{A_{kw}} = \sum_{i=1}^{k} kw_i \mathbf{M}_i + \mathbf{B} \in \mathbb{Z}_q^{n \times 2m}$;
2: Calculate a matrix $\mathbf{A_{ID}} = (\mathbf{A} \mid H_4(\mathbf{ID})) \in \mathbb{Z}_q^{n \times 2m}$;
3: Sample a basis $\mathbf{T}_{\mathbf{A_{ID}}} \leftarrow \mathsf{ExtBasis}(\mathbf{A_{ID}}, \mathbf{T_A})$ for $\mathbf{A_{ID}}$;
4: Calculate a vector $\mathbf{trap_1} = \mathbf{SK_{ID}y} \in \mathbb{Z}_q^m$;
5: Sample a vector $\mathbf{trap_2} \leftarrow \mathsf{SampleLeft}(\mathbf{A_{ID}}, \mathbf{A_{kw}}, \mathbf{T}_{\mathbf{A_{ID}}}, \mathbf{u}, s)$, where $\mathbf{trap_2} \in \mathbb{Z}_q^{4m}$;
6: **Return** a trapdoor tuple $\mathrm{Trap} = (\mathbf{trap_1}, \mathbf{trap_2}, \sigma, t)$.

5.6 Matching Phase

The server checks whether the parameters are matched properly and then returns the corresponding results, in which the specific steps are as follows. After taking public parameter pp, user **ID**, ciphertext CT, trapdoor Trap, and t' as input, the Test(pp, **ID**, CT, Trap, t') algorithm judges the validity of the above parameters. Note that t' refers to the time when the server received the trapdoor Trap.

- The server initially checks whether $t' \leq t$. If it satisfies this condition, the trapdoor is in the authorized time.
- After that, the server checks if $h = H_2(\mathbf{Az} - H_1(\mathbf{ID})h, \mathbf{trap_1}, t)$ and $\|\mathbf{z}\| \leq 2s\sqrt{m}$. If this equation holds, the trapdoor is actually from an authorized user and processes the following step.
- Finally, the server checks the error term $|c_0 - \mathbf{trap}_2^\top \mathbf{c_1}| \leq \lfloor q/4 \rfloor$. If this equation holds, it outputs 1 and the server publishes the corresponding encrypted data to the authorized user; Otherwise, it outputs 0 and returns an error.

6 SECURITY ANALYSIS

In this section, we illustrate that the lattice-based PEAKS construction satisfies the correctness, IND-sID-CKA and T-EUF security.

Correctness analysis. We start by proving that the signature in Trapdoor algorithm is issued by an authority correctly: $\mathbf{Az}-H_1(\mathbf{ID})h = H_1(\mathbf{ID})h + \mathbf{Ay} - H_1(\mathbf{ID})h = \mathbf{Ay}$. Thus, we obtain that $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t) = H_2(\mathbf{Az} - H_1(\mathbf{ID})h, \mathbf{trap_1}, t)$. We also know the distribution $\mathbf{z}$ is close to $\mathcal{D}_s^m$ according to the rejection sampling technique and Lemma 3 and then we say $\|\mathbf{z}\| \leq 2s\sqrt{m}$ with probability of over $1 - 2^{-m}$ due to Lemma 2 .

Furthermore, we illustrate that the error term computed by a matched trapdoor and keyword ciphertext in Trapdoor and Test algorithms is less than $\lfloor q/4 \rfloor$ with overwhelming probability.

We parse the trapdoor as $\mathbf{trap_2} = (\mathbf{trap_2^1}, \mathbf{trap_2^2}) \in \mathbb{Z}_q^{2m} \times \mathbb{Z}_q^{2m}$. Then, by the triangle inequality, the error term is bounded as:

$$
\begin{aligned}
&|\mathbf{u}^\top \mathbf{r} + noi - (\mathbf{A_{ID,kw}} \cdot \mathbf{trap}_2)^\top \mathbf{r} - \mathbf{trap}_2^\top (\mathbf{noi}, \mathbf{R}_{\mathbf{kw}}^\top \cdot \mathbf{noi})| \\
&= |noi - \begin{pmatrix} \mathbf{trap}_2^{1\top} \\ \mathbf{trap}_2^{2\top} \end{pmatrix} (\mathbf{noi}, \mathbf{R}_{\mathbf{kw}}^\top \cdot \mathbf{noi})| \\
&= |noi - (\mathbf{trap_2^1} + \mathbf{R_{kw}} \cdot \mathbf{trap_2^2})^\top \mathbf{noi}| \\
&\leq |noi| + |(\mathbf{trap_2^1} + \mathbf{R_{kw}} \cdot \mathbf{trap_2^2})^\top \mathbf{noi}|
\end{aligned}
$$

According to the Gaussian tail bound, we get $|noi| < q\alpha\omega(\sqrt{\log m})$ +1/2. Then, from Lemma 6, we have $|(\mathbf{trap_2^1} + \mathbf{R_{kw}} \cdot \mathbf{trap_2^2})^\top \mathbf{noi}| < s(km + \sqrt{m})(q\alpha \cdot \omega(\log m) + \sqrt{m}/2)$, with negligible probability. Hence, we conclude that the error term is bounded by

$$
\begin{aligned}
|c_0 - \mathbf{trap}_2^\top \mathbf{c_1}| &\leq O(sm^{1.5}) + s(\sqrt{m} + km)(q\alpha \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2) \\
&\leq O(sm^{1.5}) + qskm\alpha \cdot \omega(\sqrt{\log m}) \\
&\leq O(m^{2.5} \cdot \omega(\sqrt{\log m})) := q/5,
\end{aligned}
$$

with overwhelming probability. It also shows that $q > 2\sqrt{n}/\alpha$, ensuring the LWE assumption is as hard as the worst-case SIVP and GapSVP hardness [27].

We introduce a hash family and a lemma for later proof H: $\{h_\theta : \mathbb{Z}_q^k/\{0^k\} \to \mathbb{Z}_q\}$ as $h(\mathbf{kw}) = 1 + \sum_{i=1}^k \theta \cdot kw_i \in \mathbb{Z}_q$, where $\mathbf{kw} = (kw_1, kw_2, \cdots, kw_k) \in \{0,1\}^k$ and $\theta = (\theta_1, \theta_2, \cdots, \theta_k) \in \mathbb{Z}_q^k$.

Lemma 9. *[2] Given a prime number q and $0 \leq q_t \leq q$, then we say the hash family H defined above satisfies $(q_t, 1/q(1-q_t/q), 1/q)$ abort-resistant.*

Theorem 1. *The proposed scheme is* IND-sID-CKA *secure assuming that the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-*LWE *hardness holds. For a polynomial $(q_k, q_a, q_t, \epsilon)$ adversary $\mathcal{A}$, if $\mathcal{A}$ has the ability to win the game with advantage ϵ by performing at most q_k* KeyGen *queries, q_a* Authorization *queries, and q_t* Trapdoor *queries, then a challenger C can solve the* LWE *hardness.*

Proof. We adopt a sequence of games to prove this theorem and also illustrate that it does not exist a polynomial $(q_k, q_a, q_t, \epsilon)$ adversary $\mathcal{A}$ can distinguish these games.

Game 0: This is a real game, the same as shown in Definition 7.

Game 1: This game is identical to **Game 0**, except that calculate $\mathbf{M}_i = \mathbf{A_{ID^*}} \cdot \mathbf{R}_i^* + \theta_i \cdot \mathbf{B}$ instead of choosing is uniformly random from $\mathbb{Z}_q^{n \times 2m}$, where $\mathbf{R}_i^* \xleftarrow{\$} \{-1,1\}^{2m \times 2m}$ and $\theta_i \xleftarrow{\$} \mathbb{Z}_q$. As for the challenge keyword **kw**, let $\mathbf{R_{kw}} = \sum_{i=1}^k kw_i \mathbf{R}_i^*$ and $(\mathbf{R}_{\mathbf{kw}}^*)^\top \cdot \mathbf{noi} \in \mathbb{Z}_q^{2m}$ to construct a challenge ciphertext CT*. In this way, if $\mathcal{A}$ can distinguish **Game 0** and **Game 1**, then there exists a challenger C who can distinguish between $\mathbf{M}_i$ and a matrix chosen uniformly random in $\mathbb{Z}_q^{n \times 2m}$. In the view of $\mathcal{A}$, **Game 0** and **Game 1** are statistically indistinguishable due to the fact that $\mathbf{M}_i$ for $i = 1, 2, \cdots, k$ are statistically close to uniform distribution. Hence, we acquire: $\Pr[\mathbf{Game}_0(1^\lambda) = 1] = \Pr[\mathbf{Game}_1(1^\lambda) = 1]$.

Game 2: This game is identical to **Game 1**, except except for the addition of an artificial abort. The difference between these two games is reflected in the initial and final parts. On the one hand, C selects a hash function $h \in H$ at random and keeps it secret. When C receives the secret key of user $\mathbf{ID} \neq \mathbf{ID}^*$ and Trapdoor queries of keywords set $\mathbf{kw}_1, \mathbf{kw}_2, \cdots, \mathbf{kw}_{q_t}$ (the selected keyword is not in it) from adversary $\mathcal{A}$, it will respond the Trapdoor queries and the challenge ciphertext same as in **Game 1**. On the other hand, given a user $\mathbf{ID}^*$ and a keyword $\mathbf{kw}^*$, $\mathcal{A}$ sends the final guess to C. After that, C checks the guess if $h(\mathbf{kw}_i \neq 0)$ for $i = 1, 2, \cdots, q_t$ and $h(\mathbf{kw}^* = 0)$. If the conditions are not fulfilled, C re-selects a random bit $b' \in \{0,1\}$ and then aborts the game. Furthermore, C randomly chooses a bit $\beta \in \{0,1\}$ s.t. $\Pr[\beta = 1] = \tau(\mathbf{kw}^*, \mathbf{kw}_1, \mathbf{kw}_2, \cdots, \mathbf{kw}_{q_t})$, where $\tau(\cdot)$ is a function in [30]. As a result, the above-mentioned changes are independent for $\mathcal{A}$, we conclude that: $\Pr[\mathbf{Game}_1(1^\lambda) = 1] \leq \frac{1}{4q} \Pr[\mathbf{Game}_2(1^\lambda) = 1]$.

Game 3: This game is identical to **Game 2**, except that change the calculation methods of **A** and **B**. In this game, C uniformly selects $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ at random. Then, C calls the TrapGen algorithm to generate the matrix **B** and its basis $\mathbf{T_B} \in \mathbb{Z}_q^{2m \times 2m}$. In this way, the response for KeyGen queries of C are the same as those in **Game 2**. As for the Trapdoor query, a user **ID** sets $\mathbf{A_{ID,kw}} = (\mathbf{A_{ID}}|\mathbf{A_{kw}}) = (\mathbf{A_{ID}}|\sum_{i=1}^k kw_i \mathbf{M}_i + \mathbf{B}) = (\mathbf{A_{ID}}|\mathbf{A_{ID}} \cdot \mathbf{R_{kw}} + h(\mathbf{kw}) \cdot \mathbf{B})$, where $\mathbf{R_{kw}} = \sum_{i=1}^k kw_i \mathbf{R}_i^*$ and $h(\mathbf{kw}) = 1 + \sum_{i=1}^k kw_i \cdot \theta_i$. After that, sample $\mathbf{trap}_2 \leftarrow \mathsf{SampleRight}(\mathbf{A_{ID}}, h(\mathbf{kw})\mathbf{B}, \mathbf{R_{kw}}, \mathbf{T_B}, \mathbf{u}, s) \in \mathbb{Z}_q^{4m}$ as the output of the Trapdoor query. In this way, $\mathbf{trap}_2$ is close to the distribution $\mathcal{D}_{\lambda_q^{\mathbf{u}}(\mathbf{A_{ID,kw}})}$ in **Game 2**. Accordingly, for the view of $\mathcal{A}$, **Game 2** and **Game 3** are statistically indistinguishable. Therefore, we have: $\Pr[\mathbf{Game}_2(1^\lambda) = 1] = \Pr[\mathbf{Game}_3(1^\lambda) = 1]$.

Game 4: This game is identical to **Game 3**, except that C chooses the challenge ciphertext $\text{CT} = (c_0, \mathbf{c_1}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{4m}$ at random. In this way, $\mathcal{A}$ cannot win the game since CT is calculated from a random ciphertext space. We define a simulator to illustrate the computational indistinguishability between **Game 3** and **Game 4** from the perspective of $\mathcal{A}$.

Reduction from LWE: Suppose that there exists an adversary $\mathcal{A}$ and a PPT challenger C has the ability to solve the LWE hardness over ϵ' probability for a target user $\mathbf{ID}^*$.

Setup: For $i = 0, 1, \cdots, 2m$, C randomly samples the entries of a LWE hardness as $(\mathbf{u}_i, v_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Afterward, C performs these operations. (1) Assume $\mathbf{A}_{\mathbf{ID}^*} = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_m)$. (2) Calculate the matrix $\mathbf{A}$ same as **Game 0** and matrices $(\mathbf{B}, \mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k)$ same as **Game 2**. (3) Respond the new public parameter pp=($\mathbf{A}$, $\mathbf{B}$, $\mathbf{M}_1$, $\mathbf{M}_2$, $\cdots$, $\mathbf{M}_k$, $\mathbf{u}_0$) to $\mathcal{A}$.

Phase 1: In this phase, $\mathcal{A}$ enquires the secret key of a user $\mathbf{ID}$. C responds the queries as below.

- KeyGen Query. With regard to the q_t queries for the secret key of a user $\mathbf{ID} \neq \mathbf{ID}^*$, C responds to the following answer. (1) Calculate $\mathbf{A}_{\mathbf{ID}} = (\mathbf{A} \mid H_4(\mathbf{ID}))$. (2) Perform the $\text{SamplePre}(\mathbf{A}, \mathbf{T_A}, H_1(\mathbf{ID}), s)$ algorithm to generate $\mathbf{sk_{ID}}$ and then respond it to $\mathcal{A}$.
- Authorization Query. $\mathcal{A}$ queries an authorized token for the dataset $\mathcal{KS}_U - \mathcal{KS}_{W^*}$ on time t. For each keyword $\mathbf{kw} \in \mathcal{KS}_U - \mathcal{KS}_{W^*}$, and each user $\mathbf{ID} \neq \mathbf{ID}^*$, calculate $\mathbf{SK_{ID}}$, where $F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}})$ After that, C chooses a random vector $\mathbf{y} \in \mathbb{Z}^m$ and calculates $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}} \cdot \mathbf{y}, t)$ and $\mathbf{z} = \mathbf{sk_{ID}} \cdot h + \mathbf{y}$ to obtain the signature $\sigma = (h, \mathbf{z})$ with probability $\min(1, \frac{\mathcal{D}_s^m(\mathbf{z})}{M\mathcal{D}^m_{\mathbf{sk_{ID}}h,s}(\mathbf{z})})$, where t is the authorized time. Finally, C responds $token = (\mathbf{SK_{ID}}, \sigma, \mathbf{y}, t)$ to $\mathcal{A}$.
- Trapdoor Query. $\mathcal{A}$ queries a trapdoor of keyword $\mathbf{kw}$ for user $\mathbf{ID}^*$. If $h(\mathbf{kw}) = 0$, abort this game and return a bit $b' \in \{0, 1\}$ at random. Otherwise, operate the steps below.
  - If $\mathbf{kw} \in \mathcal{KS}_U - \mathcal{KS}_{W^*}$, $\mathcal{A}$ can calculate the authorized token according to access the Authorization Query and then generate the trapdoor by itself.
  - If $\mathbf{kw} \in \mathcal{KS}_{W^*}$, C responds the query as follows. (1) Perform the TrapGen algorithm to generate $\mathbf{B} \in \mathbb{Z}_q^{n \times 2m}$ together with its trapdoor $\mathbf{T_B}$. (2) For $i = 1, 2, \cdots, k$, randomly select $\mathbf{R}_i^* \xleftarrow{\$} \{-1, 1\}^{2m \times 2m}$ and k random parameters $\theta_i \in \mathbb{Z}_q$. (3) Then, compute $\mathbf{M}_i = \mathbf{A}_{\mathbf{ID}^*} \cdot \mathbf{R}_i^* + \theta_i \cdot \mathbf{B} \in \mathbb{Z}_q^{n \times 2m}$. In this case, $\mathbf{A}_{\mathbf{ID},\mathbf{kw}} = (\mathbf{A}_{\mathbf{ID}} | \mathbf{A}_{\mathbf{kw}}) = (\mathbf{A}_{\mathbf{ID}} | \sum_{i=1}^k kw_i \mathbf{M}_i + \mathbf{B}) = (\mathbf{A}_{\mathbf{ID}} | \mathbf{A}_{\mathbf{ID}} \cdot \mathbf{R}_{\mathbf{kw}} + h(\mathbf{kw})\mathbf{B})$, where $\mathbf{R}_{\mathbf{kw}} = \sum_{i=1}^k kw_i \mathbf{R}_i^* \in \mathbb{Z}_q^{2m \times 2m}$. (4) After that, compute $\mathbf{trap_1} = \mathbf{SK_{ID}} \cdot \mathbf{y} \in \mathbb{Z}_q^m$. (5) Further, invoke the $\text{SampleRight}(\mathbf{A}_{\mathbf{ID}^*}, h(\mathbf{kw})\mathbf{B}, \mathbf{R}_{\mathbf{kw}}, \mathbf{T_B}, \mathbf{u}, s)$ algorithm to generate $\mathbf{trap_2} \in \mathbb{Z}_q^{4m}$ of user $\mathbf{ID}^*$ on the keyword $\mathbf{kw}$. (6) Ultimately, respond a trapdoor tuple $\text{Trap} = (\mathbf{trap_1}, \mathbf{trap_2}, \sigma, t)$ to $\mathcal{A}$.

Challenge: When the queries end, $\mathcal{A}$ publishes $\mathbf{kw}_0$ and $\mathbf{kw}_1$ to C, C checks if $h(\mathbf{kw}^*) = 0$.

- If the above equation holds, C will abort the game and respond a bit $b' \in \{0, 1\}$ at random.
- Otherwise, C calculates the challenge ciphertext for a target user $\mathbf{ID}^*$ through the following procedures. (1) Calculate a vector $\mathbf{v}^* = [v_1 \cdots v_{2m}]^\top \in \mathbb{Z}_q^{2m}$ and a scalar $c_0^* = v_0 \in \mathbb{Z}_q$. (2) Select a challenge keyword $\mathbf{kw}^* \in \{\mathbf{kw}_1, \mathbf{kw}_2\}$ at random, and for $i = 1, 2, \cdots, k$, compute $\mathbf{R}_{\mathbf{kw}^*} = \sum_{i=1}^k kw_i^* \mathbf{R}_i^*$. (3) Calculate $\mathbf{c}_1^* = \begin{bmatrix} \mathbf{v}^* \\ (\mathbf{R}_{\mathbf{kw}^*})^\top \cdot \mathbf{v}^* \end{bmatrix} \in \mathbb{Z}_q^{4m}$ and select a bit $b \in \{0, 1\}$. (4) If $b = 0$, respond $\text{CT}^* = (c_0^*, \mathbf{c}_1^*)$ to $\mathcal{A}$. If $b = 1$, randomly select $\text{CT} = (c_0, \mathbf{c_1}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{4m}$ and send it to $\mathcal{A}$.

Phase 2: In this phase, the simulator repeats the same procedures as in **Phase 1** with forbidden to query the challenge keywords $\mathbf{kw_0}$ and $\mathbf{kw_1}$.

Guess: The challenger C processes the artificial abort operation at the outset. For $i = 1, 2, \cdots, q_t$, C determines that if both $h(\mathbf{kw}^*) = 0$ and $h(\mathbf{kw}_i) \neq 0$ are satisfied, where $\mathbf{kw}^*$ is the target keyword and $\mathbf{kw}^* \notin \{\mathbf{kw}_1, \mathbf{kw}_2, \cdots, \mathbf{kw}_{q_t}\}$. If the conditions are met, C publishes the guess as the response for solving the LWE hardness problem. Otherwise, C outputs a random bit b' from {0,1} and then aborts this game. At the end of all queries, $\mathcal{A}$ outputs a guess b'.

On the one hand, if the LWE hardness is a pseudo-random oracle, we have $\mathbf{A}_{\mathbf{kw}^*} = (\mathbf{A}_{\mathbf{ID}^*} | \mathbf{R}_{\mathbf{kw}^*})$ due to $h(\mathbf{kw}^*) = 0$. As for the random noise $\mathbf{noi} \xleftarrow{\$} \bar{\Psi}_\alpha^{2m} \in \mathbb{Z}_q^{2m}$, we have $\mathbf{v}^* = \mathbf{A}_{\mathbf{ID}^*}^\top \cdot \mathbf{r} + \mathbf{noi}$. We obtain:

$$\mathbf{c}_1^* = \begin{bmatrix} \mathbf{A}_{\mathbf{ID}^*}^\top \mathbf{r} + \mathbf{noi} \\ (\mathbf{A}_{\mathbf{ID}^*} \mathbf{R}_{\mathbf{kw}^*})^\top \mathbf{r} + (\mathbf{R}_{\mathbf{kw}^*})^\top \mathbf{noi} \end{bmatrix} = \mathbf{A}_{\mathbf{kw}^*}^\top \mathbf{r} + \begin{bmatrix} \mathbf{noi} \\ (\mathbf{R}_{\mathbf{kw}}^*)^\top \mathbf{noi} \end{bmatrix}.$$

Under this circumstances, $\text{CT}^* = (c_0^*, \mathbf{c}_1^*)$ is a valid challenge ciphertext since both $c_0^* = \mathbf{u}^\top \mathbf{r} + noi$ and $\mathbf{c}_1^*$ are valid.

On the other hand, if the LWE hardness is a purely random oracle, we have $v_0 \xleftarrow{\$} \mathbb{Z}_q$, and $\mathbf{v}^* \xleftarrow{\$} \mathbb{Z}_q^{2m}$. In this context, the challenge ciphertext CT^* is uniform in $\mathbb{Z}_q \times \mathbb{Z}_q^{4m}$.

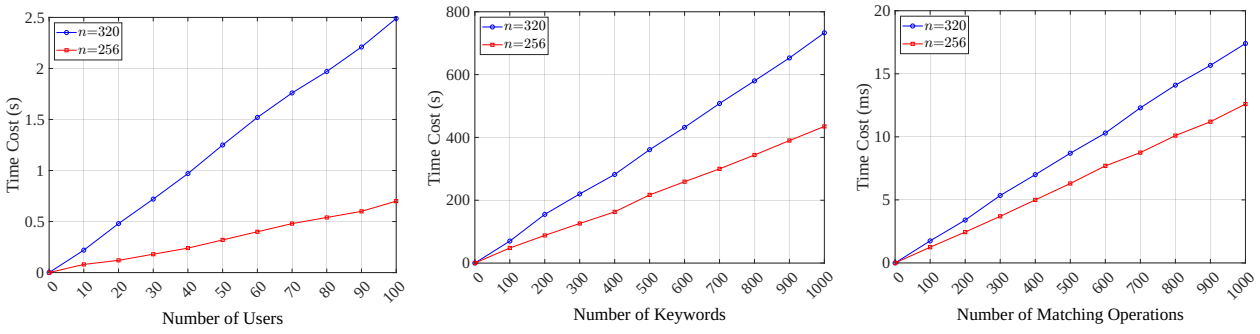
Now, we assume $[p_{\min}, p_{\max}]$ be the probability interval of artificial abort which does not address in the Trapdoor query. Based on the above discussion and Lemma 9, we obtain that $|p_{\max} - p_{\min}| \geq \frac{q_t}{q^2}$, where $q_t \leq \frac{q}{2}$ and $p_{\min} \geq \frac{1}{2q}$. As a result, we need to ensure that the parameter q is large enough to construct $\text{negl}(n)$ to be a negligible function.

Thus, the advantage for C to solve the LWE hardness is summarized as: $Adv_C^{\text{LWE}} \geq \frac{1}{2}((p_{\max} - p_{\min}) + p_{\min} | \Pr[b' = b] - \frac{1}{2}|) \geq \frac{p}{4q} + \text{negl}(n)$. In a nutshell, we conclude: $|\Pr[\mathbf{Game}_3(1^\lambda) = 1] - \Pr[\mathbf{Game}_4(1^\lambda) = 1]| \leq Adv_C^{\text{LWE}}$.

□

Theorem 2. *The proposed scheme is* T-EUF *secure assuming that the* (q, n, m, β)-SIS *hardness holds, where* $\beta \approx \tilde{O}(\|\mathbf{z}\|)$. *For a polynomial* $(q_h, q_k, q_a, q_t, \epsilon)$ *adversary* $\mathcal{A}$, *if* $\mathcal{A}$ *can win the game with advantage* ϵ *by performing at most* q_h Hash *queries,* q_k KeyGen *queries,* q_a Authorization *queries, and* q_t Trapdoor *queries, and* $\mathcal{A}$ *can break the signature generation process of* Authorization *algorithm with advantage* δ, *then a challenger* C *can solve the* SIS *hardness.*

Proof. If there is an adversary $\mathcal{A}$ who can attack the trapdoor existential unforgeability with a non-negligible advantage, then it can also solve the (q, n, m, β)-SIS assumption. $\mathcal{A}$ initially sets the challenge user $\mathbf{ID}^*$. We simulate the interaction between $\mathcal{A}$ and a challenger C as follows.



(a) User Registration (b) Ciphertext Generation (c) Trapdoor Generation and Matching

Figure 2: Performance Evaluation of L-PEAKS Scheme.

Setup: The adversary $\mathcal{A}$ announces a challenge dataset $\mathcal{KS}^*_W \subseteq \mathcal{KS}_U$. Then, C performs these procedures. (1) Calculate several matrices $(\mathbf{A}, \mathbf{B}, \mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k)$ same as the original **T-EUF Game** in Section 3.2. (2) Select a challenge user $\mathbf{ID}^*$ and calculate a matrix $\mathbf{A}_{\mathbf{ID}^*} = (\mathbf{A} \mid H_4(\mathbf{ID}^*)) \in \mathbb{Z}_q^{n\times 2m}$. (3) Respond a public parameter pp=$(\mathbf{A}, \mathbf{B}, \mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k)$ to $\mathcal{A}$.

Query: $\mathcal{A}$ performs the queries, and C responds them to $\mathcal{A}$.

- Hash Query. C maintains a list $L(\mathbf{SK_{ID}}, t, h)$, which is initially empty. When $\mathcal{A}$ queries about $(\mathbf{SK_{ID}}, t)$, C searches the list L. If the tuple $(\mathbf{SK_{ID}}, t, h)$ exists, C will return h to $\mathcal{A}$. Otherwise, C randomly selects a vector $\mathbf{y}$ and calculates $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t)$. After that, C returns h to $\mathcal{A}$ and adds the tuple $(\mathbf{SK_{ID}}, t, h)$ to list L.
- KeyGen Query. With regard to the q_t queries for the secret key of a user $\mathbf{ID} \neq \mathbf{ID}^*$, C responds to the following answer. (1) Calculate $\mathbf{A_{ID}} = (\mathbf{A} \mid H_4(\mathbf{ID}))$. Assume that the i-th bit of the user $\mathbf{ID}^*$ and $\mathbf{ID}$ are different due to $\mathbf{ID}^* \neq \mathbf{ID}$. (2) Perform the $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, H_1(\mathbf{ID}), s)$ algorithm to generate $\mathbf{sk_{ID}}$ and then respond it to $\mathcal{A}$.
- Authorization Query. $\mathcal{A}$ queries an authorized token for a dataset on time t. For each keyword $\mathbf{kw}$ and each user $\mathbf{ID} \neq \mathbf{ID}^*$, compute $\mathbf{SK_{ID}}$, where $F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw}\in\{\mathcal{KS}_U - \mathcal{KS}_W\}} f(\mathbf{sk_{ID,kw}})$. After that, C chooses a random vector $\mathbf{y} \in \mathbb{Z}^m$ and calculates $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}} \cdot \mathbf{y}, t)$ and $\mathbf{z} = \mathbf{sk_{ID}} \cdot h + \mathbf{y}$ to obtain the signature $\sigma = (h, \mathbf{z})$ with probability $\min(1, \frac{\mathcal{D}_s^m(\mathbf{z})}{M\mathcal{D}^m_{\mathbf{sk_{ID}}h,s}(\mathbf{z})})$, where t is the authorized time. Finally, C responds $token = (\mathbf{SK_{ID}}, \sigma, \mathbf{y}, t)$ to $\mathcal{A}$.
- Trapdoor Query. $\mathcal{A}$ queries a trapdoor of $\mathbf{kw}$. Then, $\mathcal{A}$ calculates the authorized token by asking for the Authorization Query and generates the trapdoor itself.

Forgery: Assume there exists a signature $\sigma = (h, \mathbf{z})$. For $\mathbf{kw} \in \mathcal{KS}_{W^*}$, $\mathcal{A}$ executes the Trapdoor algorithm to generate a trapdoor tuple $\mathrm{Trap}^* = (\mathbf{trap}_1^*, \mathbf{trap}_2^*, \sigma^*, t^*) = (\mathbf{trap}_1^*, \mathbf{trap}_2^*, h^*, \mathbf{z}^*, t^*)$. Since Trap^* is a valid trapdoor, we obtain that $\sigma^* = (h^*, \mathbf{z}^*)$ is a forged signature and the tuple $(\mathbf{SK}^*_{\mathbf{ID}}, t^*, h^*)$ is in the list L. Thus, we have $H_2(\mathbf{Az}^* - H_1(\mathbf{ID})h, \mathbf{SK}^*_{\mathbf{ID}}\mathbf{y}, t) = H_2(\mathbf{Az} - H_1(\mathbf{ID})h, \mathbf{SK_{ID}}\mathbf{y}, t)$. If $\mathbf{SK_{ID}} \neq \mathbf{SK}^*_{\mathbf{ID}}$ or $\mathbf{Az}^* - H_1(\mathbf{ID})h \neq \mathbf{Az} - H_1(\mathbf{ID})h$, it means $\mathcal{A}$ finds the pre-image of the hash function. Therefore, we have $\mathbf{SK_{ID}} = \mathbf{SK}^*_{\mathbf{ID}}$ and $\mathbf{Az}^* - H_1(\mathbf{ID})h = \mathbf{Az} - H_1(\mathbf{ID})h$ and so $\mathbf{A}(\mathbf{z} - \mathbf{z}^*) = 0$. In addition, we notice that $\mathbf{z} - \mathbf{z}^* \neq 0$ due to the fact that σ^* is not equal to the old signature σ. Also, because of $\|\mathbf{z}\|, \|\mathbf{z}^*\| \leq 2s\sqrt{m}$, we have $\|\mathbf{z} - \mathbf{z}^*\| \leq 4s\sqrt{m}$. Consequently, $\|\mathbf{z} - \mathbf{z}^*\|$ is a solution of the (q, n, m, β)-SIS assumption.

According to Lemma 5.3 and 5.4 in [23], the probability for finding a solution of the (q, n, m, β)-SIS assumption is $\epsilon = (\frac{1}{2} - 2^{-100})(\delta - 2^{-100})(\frac{\delta - 2^{-100}}{q_h + q_a} - 2^{-100}) \approx \frac{\delta^2}{2(q_h+q_a)}$. Thus, C can solve the (q, n, m, β)-SIS assumption with probability ϵ.

□

7 PERFORMANCE EVALUATION AND COMPARISON WITH PRIOR ARTS

In this section, we give a comprehensive performance evaluation of our scheme at the beginning. Then, we conduct a comparative analysis of the proposed L-PEAKS scheme with other state-of-the-art PEKS and PEAKS schemes with regard to three aspects, security properties, space complexity, and computation complexity.

All the experiments[1] of our scheme is accomplished on a Windows 10 Laptop with Core i7-11800H CPU 2.30 GHz and 24 GB RAM. We use Python to implement our scheme and set two different types of security parameters as $n = 256, m = 9728, q = 4093$ and $n = 320, m = 12800, q = 8191$ following the existing works [4, 5, 33]. Our comprehensive performance evaluations are illustrated in Fig. 2 under two security parameter settings ($n = 256$ and $n = 320$). To be more specific, we depict the performance of User Registration, Ciphertext Generation and Trapdoor Generation & Matching phases in Fig. 2(a), Fig. 2(b), and Fig. 2(c), respectively.

Subsequently, we compare our L-PEAKS scheme with existing PEKS and PEAKS schemes [6], [34], [18], [19], [17], [33], [29], [10], [22], [36], [21] in terms of five aspects, including identity-based, quantum-safe, authorize, IND-CKA, T-EUF. This comparison is elaborated in Table 1. From this table, it is evident that schemes [6], [33], [29], [36] as well as our scheme incorporates the concept of identity-based encryption. Moreover, schemes [34], [33], [36], [22] along with our scheme demonstrate resistance against quantum computing attacks. For the authorize property, scheme [18], [19],

[1]Note that since the running time of the program will vary depending on the state of the machine at the time of each measurement, the trend and speed ratios should be consistent with those in this paper. The source code is published at https://github.com/BatchClayderman/LB-PEAKS.

[29], [21] and our scheme meets the requirement. The security property IND-CKA is essential and is not satisfied by only one scheme [17] among the existing schemes. As for the T-EUF security, it is achieved by schemes [18],[19] and our scheme out of the eleven existing schemes we compared. In summary, our L-PEAKS scheme is the only one that effectively addresses all the security concerns.

We also demonstrate the comparison analysis of space complexity and computation complexity with other five schemes [12], [33], [34], [22], [36] in Tables 2 and 3, respectively. To ensure a fair comparison, the schemes we compare are all based on the lattice hardness. Specifically, we give a comparison of the secret key size, ciphertext size, and trapdoor size with others in Table 2. As for Table 3, we analyze the computation complexity of the encryption, trapdoor, and search algorithms, respectively. From Table 2 and Table 3, it is evident that our L-PEAKS scheme guarantees excellent security performance without introducing additional space and computation overheads in general. Especially for the secret key size and ciphertext size in Table 2 as well as trapdoor computation and search computation in Table 3, our scheme offers more security properties while still remaining the overhead minimal compared with prior arts.

8 CONCLUSION

In this paper, we propose a L-PEAKS scheme from the LWE and SIS hardness. It allows an authority to authorize users to search different datasets and maintains quantum resistance for the first time. In particular, we leverage the philosophy of lattice sampling algorithms and basis extension algorithms to achieve these properties. We also introduce the IBE to support users encrypting data directly via their identities to reduce the storage overhead. Furthermore, we conduct a detailed security analysis and give a comprehensive performance evaluation of our proposed scheme. Finally, we provide security properties comparison and theoretical comparisons with regard to space and computation complexity. For future work, it would be interesting to design a PEAKS scheme supporting multi-user scenarios and resist secret key leakage problem.

Table 1: Security properties comparison with other existing PEKS and PEAKS schemes

Schemes	Identity-based	Quantum-safe	Authorize	IND-CKA	T-EUF
Boneh et al. [6]	✓	×	×	✓	×
Xu et al. [34]	×	✓	×	✓	×
Jiang et al. [18]	×	×	✓	✓	✓
Jiang et al. [19]	×	×	✓	✓	✓
Huang et al. [17]	×	×	×	×	×
Xu et al. [33]	✓	✓	×	✓	×
Wang et al. [29]	✓	×	✓	✓	×
Chen et al. [10]	×	×	×	✓	×
Liu et al. [22]	×	✓	×	✓	×
Zhang et al. [36]	✓	✓	×	✓	×
Liu et al. [21]	×	×	✓	✓	×
Our scheme	✓	✓	✓	✓	✓

Table 2: Space complexity comparison

Schemes	Secret key Size	Ciphertext Size	Trapdoor Size
Cheng et al. [12]	$\lvert\mathbb{Z}^{m\times m}\rvert$	$\lvert\mathbb{Z}_q^{8m\times \ell_S}\rvert$	$\lvert\mathbb{Z}_q^{8m\times \ell_R}\rvert$
Xu et al. [33]	$\lvert\mathbb{Z}_q^{(m+m')\times(m+m')}\rvert$	$\lvert\mathbb{Z}_q^{2(m+m')}\rvert$	$\lvert\mathbb{Z}_q^{2(m+m')}\rvert$
Xu et al. [34]	$\lvert\mathbb{Z}_q^{m+\kappa}\rvert$	$\lvert\mathbb{Z}_q^{\rho((d+3)m+2)}\rvert$	$\lvert\mathbb{Z}_q^{(d+3)m}\rvert$
Liu et al. [22]	$\lvert\mathbb{Z}_q^{m+n\kappa+m\times m}\rvert$	$\lvert\mathbb{Z}_q^{\rho(2m+2)}\rvert$	$\lvert\mathbb{Z}_q^{2m}\rvert$
Zhang et al. [36]	$\lvert\mathbb{Z}^{m\times m}\rvert$	$\lvert\mathbb{Z}_q^{(m+1)\times\rho}\rvert$	$\lvert\mathbb{Z}_q^{m}\rvert$
Our scheme	$\lvert\mathbb{Z}_q^{m}\rvert$	$\lvert\mathbb{Z}_q^{4m+1}\rvert$	$\lvert\mathbb{Z}_q^{5m+2}\rvert$

Table 3: Computation complexity comparison

Schemes	Encrypt Comp.	Trapdoor Comp.	Search Comp.
Cheng et al. [12]	$O(\rho m\ell_S)$	$O(\rho m\ell_R)$	$O(n^2)$
Xu et al. [33]	$O(n^3)$	$O(\rho(m+m'))$	$O(n^2)$
Xu et al. [34]	$O(\kappa n^3)$	$O(n^3)$	$O(\kappa n^2)$
Liu et al. [22]	$O(\kappa n^3)$	$O(n^3)$	$O(\kappa n^2)$
Zhang et al. [36]	$O(n^2)$	$O(\rho m^2)$	$O(n^2)$
Our scheme	$O(n^3)$	$O(\rho m)$	$O(n^2)$

REFERENCES

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. 2008. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of cryptology* 21 (2008), 350–391.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. 2010. Efficient lattice (h) ibe in the standard model.. In *Eurocrypt*, Vol. 6110. Springer, 553–572.

[3] Miklós Ajtai. 1996. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 99–108.

[4] Martin R Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology* 9, 3 (2015), 169–203.

[5] Rouzbeh Behnia, Muslum Ozgur Ozmen, and Attila Altay Yavuz. 2018. Lattice-based public key searchable encryption from experimental perspectives. *IEEE Transactions on Dependable and Secure Computing* 17, 6 (2018), 1269–1282.

[6] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004. Public key encryption with keyword search. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*. Springer, 506–522.

[7] Luca Bonomi, Sepand Gousheh, and Liyue Fan. 2023. Enabling Health Data Sharing with Fine-Grained Privacy. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 131–141.

[8] David Cash, Dennis Hofheinz, and Eike Kiltz. 2009. How to delegate a lattice basis. *Cryptology ePrint Archive* (2009).

[9] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. 2012. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology* 25 (2012), 601–639.

[10] Biwen Chen, Libing Wu, Sherali Zeadally, and Debiao He. 2019. Dual-server public-key authenticated encryption with keyword search. *IEEE Transactions on Cloud Computing* 10, 1 (2019), 322–333.

[11] Xue Chen, Shiyuan Xu, Tao Qin, Yu Cui, Shang Gao, and Weimin Kong. 2022. AQ–ABS: Anti-quantum attribute-based signature for EMRs sharing with blockchain. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1176–1181.

[12] Leixiao Cheng and Fei Meng. 2022. Public key authenticated encryption with keyword search from LWE. In *European Symposium on Research in Computer Security*. Springer, 303–324.

[13] Hui Cui, Robert H Deng, Joseph K Liu, and Yingjiu Li. 2017. Attribute-based encryption with expressive and authorized keyword search. In *Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3–5, 2017, Proceedings, Part I 22*. Springer, 106–126.

[14] Xiaolei Dong, Zhenfu Cao, Jiachen Shen, et al. 2019. Revocable Public Key Encryption with Authorized Keyword Search. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 857–860.

[15] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 197–206.

[16] Jialu Hao, Jian Liu, Huimei Wang, Lingshuang Liu, Ming Xian, and Xuemin Shen. 2019. Efficient attribute-based access control with authorized search in cloud storage. *IEEE Access* 7 (2019), 182772–182783.

[17] Qiong Huang and Hongbo Li. 2017. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences* 403 (2017), 1–14.

[18] Peng Jiang, Yi Mu, Fuchun Guo, and Qiaoyan Wen. 2016. Public key encryption with authorized keyword search. In *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II 21*. Springer, 170–186.

[19] Peng Jiang, Yi Mu, Fuchun Guo, and Qiaoyan Wen. 2017. Secure-channel free keyword search with authorization in manager-centric databases. *Computers & Security* 69 (2017), 50–64.

[20] Chunchi Liu, Hechuan Guo, Minghui Xu, Shengling Wang, Dongxiao Yu, Jiguo Yu, and Xiuzhen Cheng. 2022. Extending on-chain trust to off-chain–trustworthy blockchain data collection using trusted execution environment (tee). *IEEE Trans. Comput.* 71, 12 (2022), 3268–3280.

[21] Zi-Yuan Liu, Chu-Chieh Chien, Yi-Fan Tseng, Raylin Tso, and Masahiro Mambo. 2022. Public Key Encryption with Hierarchical Authorized Keyword Search. In *International Conference on Information Security and Cryptology*. Springer, 147–170.

[22] Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. 2022. Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation. In *Proceedings of the 2022 ACM on Asia conference on computer and communications security*. 423–436.

[23] Vadim Lyubashevsky. 2012. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 738–755.

[24] Daniele Micciancio and Chris Peikert. 2012. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 700–718.

[25] Von Neumann. 1951. Various techniques used in connection with random digits. *Notes by GE Forsythe* (1951), 36–38.

[26] Chris Peikert. 2010. An efficient and parallel Gaussian sampler for lattices. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*. Springer, 80–97.

[27] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* 56, 6 (2009), 1–40.

[28] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

[29] Jiabei Wang, Rui Zhang, Jianhao Li, and Yuting Xiao. 2022. Owner-enabled secure authorized keyword search over encrypted data with flexible metadata. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2746–2760.

[30] Brent Waters. 2005. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*. Springer, 114–127.

[31] Gang Xu, Shiyuan Xu, Yibo Cao, Fan Yun, Yu Cui, Yiying Yu, Ke Xiao, et al. 2022. PPSEB: a postquantum public-key searchable encryption scheme on blockchain for E-healthcare scenarios. *Security and Communication Networks* 2022 (2022).

[32] Lingling Xu, Wanhua Li, Fangguo Zhang, Rong Cheng, and Shaohua Tang. 2019. Authorized keyword searches on public key encrypted data with time controlled keyword privacy. *IEEE Transactions on Information Forensics and Security* 15 (2019), 2096–2109.

[33] Lei Xu, Xingliang Yuan, Ron Steinfeld, Cong Wang, and Chungen Xu. 2019. Multi-writer searchable encryption: An LWE-based realization and implementation. In *Proceedings of the 2019 ACM Asia conference on computer and communications security*. 122–133.

[34] Shiyuan Xu, Yibo Cao, Xue Chen, Yanmin Zhao, and Siu-Ming Yiu. 2023. Post-Quantum Public-Key Authenticated Searchable Encryption with Forward Security: General Construction, and Applications. In *International Conference on Information Security and Cryptology*. Springer, 274–298.

[35] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzheng Cheng. 2024. On Protecting the Data Privacy of Large Language Models (LLMs): A Survey. *arXiv preprint arXiv:2403.05156* (2024).

[36] Xiaojun Zhang, Chunxiang Xu, Huaxiong Wang, Yuan Zhang, and Shixiong Wang. 2021. 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* 18, 3 (2021), 1019–1032.

[37] Yifang Zhang, Mingyue Wang, Yu Guo, and Fangda Guo. 2023. Towards dynamic and reliable private key management for hierarchical access structure in decentralized storage. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3371–3380.

[38] Jie Zhu, Qi Li, Cong Wang, Xingliang Yuan, Qian Wang, and Kui Ren. 2018. Enabling generic, verifiable, and secure data search in cloud services. *IEEE Transactions on Parallel and Distributed Systems* 29, 8 (2018), 1721–1735.