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

Shiyuan Xu[1], Yibo Cao[2], Xue Chen[1,3], Yuer Yang[1], and Siu-Ming Yiu[1⋆]

[1] Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong
{syxu666, yueryang}@connect.hku.hk, smyiu@cs.hku.hk
[2] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China
yibocaobupt@gmail.com
[3] Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
xue-serena.chen@connect.polyu.hk

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, rigorous security reduction, and theoretical complexity comparison of our scheme and perform comprehensive evaluations at 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. Finally, we provide two potential applications to show its versatility.

Keywords: PEKS · Authorization · Lattice-based cryptography · LWE and SIS hardness.

1 Introduction

With the advent of security and privacy in the big data era, efficient data retrieval and sharing have become more challenging. Boneh et al. proposed the concept of public key encryption with keyword search (PEKS) primitive, which allows for searching encrypted data [1]. PEKS involves

⋆ Corresponding author

three entities: data sender, data user, and cloud server [2]. 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.

However, a searchable ciphertext is generated through a user's public key and only one corresponding secret key can search it. This 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. Moreover, 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 limits access control privileges. For instance, each user may be at a different access level and can only search for limited keywords.

To address this issue, Jiang et al. proposed a solution where an administrator holds the enterprise's secret key and grants search privileges to users [3]. In their scheme, the administrator sets up an authorized keyword set for each user, who can only search for authorized keyword sets. Then, the administrator 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 a user and thereby keep the timeliness of signatures.

The aforementioned scheme presents a promising variant of PEKS. Nevertheless, its security is based on the discrete logarithmic (DL) assumptions, which have been proven vulnerable to attacks by quantum computers [4,5]. In response to this, lattice-based cryptography has emerged as a rapidly developing alternative [6, 7]. In addition, traditional PEKS schemes typically require computing a public key for each user, resulting in significant storage overhead. To get around this, identity-based encryption (IBE) seems to be a proper approach [8], where the public key can be an arbitrary string (e.g. a user's identity number or email address). It reduces the storage overhead of the public key certificate effectively.

A straightforward approach is to adopt the conventional PEKS scheme, that is, treating identities as keywords and utilizing IBE for equality testing [1, 9]. However, directly implementing this methodology can lead to significant complexity in key management. 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 [10]. In the context of a lattice-based instantiation, using IBE primitives (e.g. [11, 12]) will result in a considerable computational overhead for generating public and secret keys, as they consist of numerous large matrices and lattice basis. Consequently, this methodology is not suitable for real-world scenarios.

Based on the above-mentioned discussions, the following question arises:

Can we construct an efficient PEKS primitive that can effectively authorize the keyword search while maintaining quantum-safe security?

1.1 Our Contribution

We resolve the above question affirmatively and summarize our fourfold contributions as follows.

- We propose a novel primitive, namely lattice-based public key encryption with authorized keyword search (L-PEAKS), which enables an authority to authorize users to search for specific sets of keywords while also providing resistance against quantum computing. In L-PEAKS, keywords are encrypted by a public key, and users who do not possess the corresponding secret keys must be authorized by the authority. To reduce the storage overhead of public key certificates, we employ an IBE structure and leverage the users' identities as their public keys.

- We incorporate a lattice signature into the PEKS to realize the authority, for the first time. We accomplish this by defining two different keyword sets and utilizing lattice basis extension algorithms. Specifically, the keyword is encrypted with a public key, and 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.
- 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 L-PEAKS scheme to eleven other PEKS primitives, evaluating their relative strengths across five security properties, and showcasing our superiority in security and privacy.
- Ultimately, we implement our scheme in two moderate security parameter settings ($n = 256$ and $n = 320$). 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. Subsequently, we present two potential applications to demonstrate its practicality in real-world scenarios.

1.2 Technique Overview

In this part, we provide a concise technical overview. Firstly, we offer a high-level idea of our scheme. Then, we present a thorough and step-by-step description of the techniques we utilized.

Technical Roadmap. To commence, we present an overall structure, with a detailed technical roadmap in Fig. 1. We construct a novel primitive L-PEAKS through three steps, that is, L-PEKS, L-ID-PEKS, and L-PEAKS. We first utilize the lattice sampling algorithms to transform the PEKS into a lattice version. In addition, we introduce the notation of IBE into L-PEKS to obtain L-ID-PEKS for the sake of efficiency. Furthermore, we incorporate the lattice basis extension algorithm into the lattice signature for authorization. Eventually, we present L-PEAKS, which is expressible as the concatenation of L-ID-PEKS and lattice signature.

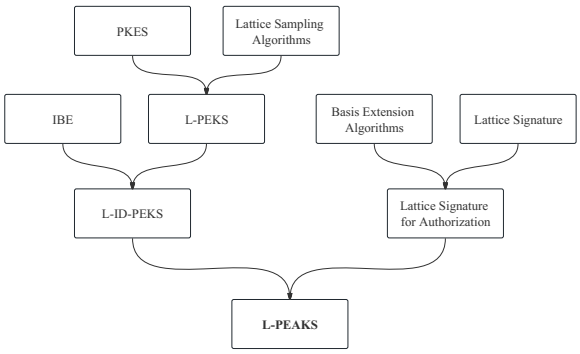


Fig. 1. Technical Roadmap.

Step 1. How to construct appropriate lattice-based PEKS. Traditional PEKS schemes use the $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ algorithms to calculate public-secret keys and a trapdoor, respectively [13, 14]. We adopt the $\mathsf{SamplePre}$ algorithm to generate the secret key of users. We do not consider $\mathsf{TrapGen}$ algorithm since we have utilized it to obtain the master secret key MSK. Additionally, our trapdoor is obtained by invoking $\mathbf{trap_2} \leftarrow \mathsf{SampleLeft}(\mathbf{A_{ID}}, \mathbf{A_{kw}}, \mathbf{SK_{ID}}, \mathbf{u}, s)$.

Step 2. Why and how to construct appropriate lattice-based ID-PEKS. In the conventional PEKS scheme, the storage overhead of public key certificates is significant, which is unsuitable for practical applications. Therefore, we leverage the concept of IBE and utilize the user's identity **ID** as the public key. To begin with, we utilize $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$ algorithm to calculate

a master secret key $\mathsf{MSK} = \mathbf{T_A}$. Next, we employ a hash function $H_1 : \{-1, 1\}^l \to \mathbb{Z}_q^n$ to convert the **ID** into a vector format, and then calculate the secret key of a user with identity **ID** as $\mathbf{sk_{ID}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, H_1(\mathbf{ID}), s)$.

Step 3. How to construct L-PEAKS. Informally speaking, constructing an L-PEASK primitive is a combination of our designed ID-PEKS and lattice signature for authorization. Technically, we begin by defining two keyword sets, that is, a universal keyword set $\mathcal{KS}_{\mathcal{U}}$ and an authorized keyword set $\mathcal{KS}_{\mathcal{W}}$, where $\mathcal{KS}_{\mathcal{W}} \subseteq \mathcal{KS}_{\mathcal{U}}$. Moreover, we devise a mapping relationship $f(\mathbf{sk_{ID}}, \mathbf{kw}) = \mathbf{sk_{ID}} + \sum_{i=1}^{k} kw_i \mathbf{m}_i \in \mathbb{Z}_q^m$ to realize a combination between one keyword **kw** and a secret key $\mathbf{sk_{ID}}$. Subsequently, when it comes to all the keywords in $\mathcal{KS}_{\mathcal{U}}$ and $\mathcal{KS}_{\mathcal{W}}$, we need to construct a new approach to realize a combination between multi-keywords and a secret key $\mathbf{sk_{ID}}$. Our intuition is to sum the above function f to get $f_1(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \mathcal{KS}_U} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m$ and $f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \mathcal{KS}_W} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m$, corresponding to the $\mathcal{KS}_{\mathcal{U}}$ and $\mathcal{KS}_{\mathcal{W}}$, respectively. After that, we use the unauthorized keywords set $(\mathcal{KS}_{\mathcal{U}} - \mathcal{KS}_{\mathcal{S}})$ to compute a polynomial $F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \{\mathcal{KS}_U - \mathcal{KS}_W\}} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m$ for latter use. Now, we show how to compute the authorize key $\mathbf{SK_{ID}}$. Since we need a matrix to be an authorized key, and the polynomial $F(\mathbf{sk_{ID}})$ is a vector, where its dimensions do not match. Therefore, we need to invoke the $\mathsf{RandBasis}$ algorithm to calculate $\mathbf{SK_{ID}} \leftarrow \mathsf{RandBasis}(F(\mathbf{sk_{ID}}), s_1)$, which serves as a stepping stone toward our goal. Finally, we can obtain the signature by processing several steps as described in the remainder (Section 4).

1.3 Related Works

Jiang et al. initialized a PEAKS primitive, in which the authority can assign tokens with valid time limits to each authorized keyword [3]. Following this work, Liu et al. proposed a public key encryption with a hierarchical authorized keyword search (PEHAKS) scheme to render authorization more flexible in real scenarios [15]. However, they cannot resist quantum computing attacks.

The concept of authorized keyword search has been widely used. In particular, 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 [16]. In 2019, Xu et al. presented a scheme that enables keyword search by the authority and authorized users [17]. After that, Wang et al. proposed a novel authorized keyword search protocol over encrypted data with metadata (MD-AKS) in 2022, achieving a secure and flexible mechanism to simultaneously process different fields of metadata [18].

1.4 Outline

The remainder of this paper is organized as follows. Section 2 covers the preliminary knowledge. In section 3, we present the syntax and security models of lattice-based PEAKS primitive. The construction will be elaborated on Section 4, while its correctness and parameter setting will be specified in Section 5. In Section 6, we give the security reduction proofs. Section 7 specifies the performance evaluation and comparison. Finally, we conclude this paper.

2 Preliminaries

2.1 Public Key Encryption with Keyword Search Scheme

A standard PEKS primitive consists of four polynomial algorithms:

- $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$: Given a security parameter λ, this probabilistic-polynomial time (PPT) algorithm outputs pk and sk as a public key and secret key, respectively.
- $ct \leftarrow \mathsf{PEKS}(pk, kw)$: Given a public key pk and a keyword kw, this PPT algorithm will output a ciphertext ct.
- $\mathsf{Trap} \leftarrow \mathsf{Trapdoor}(sk, kw')$: Given a secret key sk and a keyword kw', this PPT algorithm outputs a trapdoor Trap.
- $(1 \text{ or } 0) \leftarrow \mathsf{Test}(ct, \mathsf{Trap})$: Given a ciphertext ct and a trapdoor Trap, this deterministic algorithm outputs 1 if $kw = kw'$; Otherwise, it outputs 0.

2.2 Integer Lattice and Sample Algorithms

Definition 1 (**Lattice**). *[19] 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 $\Lambda = \Lambda(\mathbf{B}) = \{x_1 \cdot \mathbf{b_1} + x_2 \cdot \mathbf{b_2} + \cdots + x_n \cdot \mathbf{b}_n | x_i \in \mathbb{Z}\}$, 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 (**q-ary Lattices**). *[20] Given $n, m, q \in \mathbb{Z}$, and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define the following q-ary Lattices and a coset: $\Lambda_q(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m | \exists \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{A}^\top \mathbf{s} = \mathbf{e} \bmod q\}$, $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m | \mathbf{Ae} = \mathbf{0} \bmod q\}$, and $\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m | \mathbf{Ae} = \mathbf{u} \bmod q\}$.*

Definition 3 (**Gaussian Distribution**). *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 (**TrapGen**(n, m, q)). *[21] 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 $\{\mathbf{A} : \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}\}$. In this way, we say $\mathbf{T_A}$ is a trapdoor of $\mathbf{A}$.*

Lemma 2 (**SamplePre**$(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, s)$). *[22] 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 $s \leq \|\tilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m)})$, where $m \geq 2n\lceil \log q \rceil$, this* PPT *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 (**SampleLeft**$(\mathbf{A}, \mathbf{M}, \mathbf{T_A}, \mathbf{u}, s)$). *[23] 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 $s \leq \|\tilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m + m_1)})$, this* PPT *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 (**SampleRight**$(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \mathbf{u}, s)$). *[23] 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 $s_R = \mathsf{sup}_{\|\mathbf{x}\|=1} \|\mathbf{Rx}\|$, this* PPT *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 (**RandBasis**$(\mathbf{T}, s_1)$). *[12] Given a trapdoor $\mathbf{T}$ of an m-dimensional integer lattice Λ, a parameter $s_1 \geq \|\tilde{\mathbf{T}_0}\| \cdot \omega(\sqrt{\log n})$, this* PPT *algorithm will output a trapdoor $\mathbf{T}'$ of Λ, where $\|\mathbf{T}'\| \leq s_1\sqrt{m}$ with overwhelming probability. Moreover, given two lattice trapdoors $\mathbf{T}_0$, $\mathbf{T}_1$ and a parameter $s_1 \geq \mathsf{max}\{\|\tilde{\mathbf{T}_0}\|, \|\tilde{\mathbf{T}_1}\|\} \cdot \omega(\sqrt{\log n})$, the statistical distance between two outputs of $\mathsf{RandBasis}(\mathbf{T}_0, s_1)$ and $\mathsf{RandBasis}(\mathbf{T}_1, s_1)$ algorithms is within $\mathsf{negl}(n)$.*

Lemma 6. *[23] Suppose there is a random matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$, a short trapdoor $\mathbf{T_A}$ of lattice $\Lambda_q^{\perp}(\mathbf{A})$ (where $m > n$ and $q > 2$), and a parameter $s > \|\tilde{\mathbf{T}_\mathbf{A}}\| \cdot \omega(\sqrt{\log m})$. After that, for any $c \in \mathbb{R}_m$ and $\mathbf{u} \in \mathbb{Z}_q^n$, we obtain $\Pr[x \leftarrow \mathcal{D}_{\Lambda_q^{\perp}(\mathbf{A}),s} : \|x\| > s\sqrt{m}] \leq \mathsf{negl}(n)$.*

2.3 The LWE and SIS Hardness Assumptions

Definition 4 (**LWE Assumption**). *[6] Suppose there exists an integer $q = q(n)$ and an error distribution $\bar{\Psi}_\alpha$ in $\mathbb{Z}_q$, $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-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 $\lfloor qX \rceil \bmod q$ over $\mathbb{Z}_q$, where X is normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$.*

Definition 5. *[6] 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 $\lfloor qX \rceil \bmod q$, where X is a normal random variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$.*

Lemma 7. *[6] Given $q > 2\sqrt{n}/\alpha$, if there exists a quantum algorithm for $\mathbb{Z}_q$, $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-LWE hardness, then it has a quantum algorithm for approximating the* SIVP *and* GapSVP *hardness to within $\tilde{O}(n/\alpha)$ in the Euclidean norm for the worst case.*

Lemma 8. *[11] 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 6 (**SIS Assumption**). *[20] 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 $\beta \geq \sqrt{m}q^{n/m}$.*

Based on the above definition, we define the search (q, n, m, β)-SIS hardness as follows. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a vector $\mathbf{t} \in \mathbb{Z}^n$ and a parameter $d << q^{n/m}$, the search (q, n, m, β)-SIS hardness contributes to finding a vector $\mathbf{v} \in \{-d, \cdots, 0, \cdots, d\}^m$ such that $\mathbf{Av} = \mathbf{t}$.

Definition 7 ((q_t, p_{min}, p_{max}) **abort-resistant function**). *[20] Suppose $H := \{h : X \to Y\}$ be a hash functions family from variable 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 $h \in H$.*

2.4 Rejection Sampling Technique

Lemma 9. *[24] 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 10. *[24] 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 more specifically, 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. Normally, this technique works in the following way. To sign a message μ, a signer with her secret key sk initially selects a random parameter y according to some distribution. Then, she computes the signature σ which is a combination of y adding to a function of sk. Let the final signature distribution be f, which is independent of sk, and also 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].

We show an example of how to utilize the rejection sampling technique in signatures [24]. To sign a message μ, a signer initially selects a random vector $\mathbf{y} \in \mathbb{Z}_q^m$ according to some distribution $\mathcal{D}$. After that, she computes a signature $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$, where matrix $\mathbf{S}$ is a secret key and vector $\mathbf{c}$ is a hash value on the input of $\mathbf{y}$ and μ. The final signature is $(\mathbf{z}, \mathbf{c})$, which its distribution is independent of $\mathbf{S}$. By Lemma 10, and the distribution of $\mathbf{z}$ is $\mathcal{D}^m_{s,\mathbf{Sc}}$, we obtain that $\mathcal{D}^m_s(\mathbf{y})/\mathcal{D}^m_{s,\mathbf{Sc}}(\mathbf{y}) \approx e$. As a result, we know that there exists a real $M(\approx e)$ such that if we output the signature $(\mathbf{z}, \mathbf{c})$ with possibility $\mathsf{min}(1, \frac{\mathcal{D}^m_s(\mathbf{z})}{M\mathcal{D}^m_{\mathbf{sk_{ID}}h,s}(\mathbf{z})})$. Then, according to Lemma 9, we have $\|\mathbf{z}\| \leq 2s\sqrt{m}$ with high probability. Therefore, the expected amount of running this process does not exceed M. Selecting the output time of $(\mathbf{z}, \mathbf{c})$ can be seen as a kind of *rejection sampling*.

3 Syntax and Security Models

3.1 Syntax

A lattice-based PEAKS (L-PEAKS) scheme consists of five PPT algorithms and one deterministic algorithm, $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Authorize}, \mathsf{Encrypt}, \mathsf{Trapdoor}, \mathsf{Test})$.

- $\mathsf{Setup}(\lambda)$: Given a security parameter λ, this PPT algorithm outputs a public parameter pp and a master secret key MSK.
- $\mathsf{KeyGen}(pp, \mathsf{MSK}, \mathbf{ID})$: Given a public parameter pp, a master secret key MSK, and a user with identity $\mathbf{ID}$, this PPT algorithm outputs a secret key $\mathbf{sk_{ID}}$ for the user $\mathbf{ID}$.
- $\mathsf{Authorize}(pp, \mathbf{sk_{ID}}, \mathcal{KS}_W, t)$: Given a public parameter pp, a secret key $\mathbf{sk_{ID}}$ of user with identity $\mathbf{ID}$, an authorized keyword set $\mathcal{KS}_W$, and the authorized time t, this PPT algorithm outputs an authorized token $token$.
- $\mathsf{Encrypt}(pp, \mathbf{ID}, \mathbf{kw})$: Given a public parameter pp, a user with identity $\mathbf{ID}$, and a keyword $\mathbf{kw}$, this PPT algorithm outputs a searchable ciphertext CT.
- $\mathsf{Trapdoor}(pp, \mathbf{ID}, token, \mathbf{kw}')$: Given a public parameter pp, a user with identity $\mathbf{ID}$, an authorized token $token$, and a keyword $\mathbf{kw}' \in \mathcal{KS}_W$, this PPT algorithm outputs a trapdoor Trap.
- $\mathsf{Test}(pp, \mathbf{ID}, \mathrm{CT}, \mathrm{Trap}, t')$: Given a public parameter pp, a user with identity $\mathbf{ID}$, 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. We say a lattice-based PEAKS primitive is *correct* if the following conditions hold:

$$\Pr\left[\mathsf{Test}(pp, \mathbf{ID}, \mathrm{CT}, \mathrm{Trap}, t') = 1 \;\middle|\; \begin{array}{c} (pp, \mathsf{MSK}) \leftarrow \mathsf{Setup}(\lambda); \\ \mathbf{sk_{ID}} \leftarrow \mathsf{KeyGen}(pp, \mathsf{MSK}, \mathbf{ID}); \\ token \leftarrow \mathsf{Authorize}(pp, \mathbf{sk_{ID}}, \mathcal{KS}_W, t); \\ \mathrm{CT} \leftarrow \mathsf{Encrypt}(pp, \mathbf{ID}, \mathbf{kw}); \\ \mathrm{Trap} \leftarrow \mathsf{Trapdoor}(pp, \mathbf{ID}, token, \mathbf{kw}') \end{array}\right] = 1 - \mathsf{negl}(\lambda). \quad (1)$$

3.2 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.

IND-sID-CKA Game of Lattice-based PEAKS The IND-sID-CKA game follows the original IND-CKA model proposed by Boneh et al. [1]. In this game, an adversary $\mathcal{A}$ has the permission to launch chosen keyword attacks and she plays this game with a challenger $\mathcal{C}$ as below.

- **Setup**. The adversary $\mathcal{A}$ announces a challenge keyword set $\mathcal{KS}_W^* \subseteq \mathcal{KS}_U$. Then, the challenger $\mathcal{C}$ executes Setup algorithm to generate (pp, MSK) and sends pp to $\mathcal{A}$. After that, $\mathcal{A}$ chooses a challenge user with identity $\mathbf{ID}^*$.
- **Phase 1**. $\mathcal{A}$ performs a polynomially bounded number of queries below.
  - **KeyGen Query**. $\mathcal{A}$ enquires the secret key of a user with identity $\mathbf{ID} \neq \mathbf{ID}^*$. $\mathcal{C}$ executes KeyGen algorithm to generate $\mathbf{sk_{ID}}$ and responds it to $\mathcal{A}$.
  - **Authorization Query**. $\mathcal{A}$ sends the keyword set $\mathcal{KS}_W = \mathcal{KS}_U - \mathcal{KS}_W^*$ and an authorized time t to $\mathcal{C}$. Then, $\mathcal{C}$ executes $\mathsf{Authorize}$ 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 $\mathcal{C}$. $\mathcal{C}$ executes $\mathsf{Trapdoor}$ 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 $\mathbf{kw}_0$, $\mathbf{kw}_1$. $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$ and calculates a searchable ciphertext CT^* for $\mathbf{kw}_b$ with user's identity $\mathbf{ID}^*$. After that, $\mathcal{C}$ 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 she 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:

$$Adv_{\mathcal{A}}^{\text{IND-sID-CKA}}(\lambda) := |\Pr[b = b'] - \frac{1}{2}|. \tag{2}$$

Definition 8 (**IND-sID-CKA secure of Lattice-based PEAKS**)**.** *We say that our lattice-based PEAKS scheme is* IND-sID-CKA secure, if for any PPT adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{\text{IND-sID-CKA}}(\lambda)$ is negligible.

T-EUF Game of Lattice-based PEAKS 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 $\mathcal{C}$ as below.

- **Setup**. The adversary $\mathcal{A}$ announces a challenge keyword set $\mathcal{KS}_W^* \subseteq \mathcal{KS}_U$. Then, the challenger $\mathcal{C}$ executes Setup algorithm to generate (pp, MSK) and sends pp to $\mathcal{A}$. After that, $\mathcal{A}$ chooses a challenge user with identity $\mathbf{ID}^*$.
- **Query**. $\mathcal{A}$ performs a polynomially bounded number of queries below.

- **Hash Query**. $\mathcal{C}$ maintains a list $L(\mathbf{SK_{ID}}, t, h)$, which is initially empty. When $\mathcal{A}$ queries about $(\mathbf{SK_{ID}}, t)$, $\mathcal{C}$ searches the list L and then responds the answer to $\mathcal{A}$.
- **KeyGen Query**. $\mathcal{A}$ enquires the secret key of a user with identity $\mathbf{ID} \neq \mathbf{ID}^*$. $\mathcal{C}$ executes KeyGen algorithm to generate $\mathbf{sk_{ID}}$ and responds it to $\mathcal{A}$.
- **Authorization Query**. $\mathcal{A}$ issues a keyword set and an authorized time t to $\mathcal{C}$. Then, $\mathcal{C}$ 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 $\mathcal{C}$. $\mathcal{C}$ executes Trapdoor algorithm to generate Trap and responds it to $\mathcal{A}$.

– **Forgery**. $\mathcal{A}$ outputs a trapdoor tuple for keyword set $\mathcal{KS}_{W^*}$, which has not been queried before.

Definition 9 (**T-EUF secure of Lattice-based PEAKS**). *We say that 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.

4 Lattice-based Public Key Encryption with Authorized Keyword Search

In this sector, we illustrate the concrete PEAKS scheme from lattice hardness. We start with a brief overview of the techniques. We construct a lattice-based PEAKS primitive that allows an administrative authority to authorize users to search different sets of keywords. In this scheme, the keywords are encrypted with a public key, and users without the corresponding secret key must be authorized by the authority to search the keywords. In addition, considering the troublesome certificate management of traditional public key encryption, we combine with identity-based encryption to simplify the key management and encrypt the user's data directly through her identity, which is more suitable for practical scenarios. Meanwhile, our scheme is designed to resist quantum computing attacks.

– Setup(λ): After inputs a security parameter λ and sets several system parameters q, n, m, s, s_1, where q is a prime, s is the Gaussian distribution parameter, s_1 is related to RandBasis algorithm, and also defines a universal keyword space $\mathcal{KS}_U$, this algorithm processes the following steps.
  - Invoke $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$ algorithm to obtain a uniformly random $n \times m$-matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its basis $\mathbf{T_A} \in \mathbb{Z}_q^{m \times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$.
  - Select l uniformly random $n \times m$-matrices $\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_l \in \mathbb{Z}_q^{n \times m}$, where l is the bit length of identity vector $\mathbf{ID}$.
  - Select two uniformly random matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$, where k is the bit length of keyword $\mathbf{kw}$.
  - Select k uniformly random matrices $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k \in \mathbb{Z}_q^{n \times m}$, and k uniformly random vectors $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k \in \mathbb{Z}_q^m$.
  - Select a uniformly random vector $\mathbf{u} \in \mathbb{Z}_q^n$.
  - Select two collision-resistant hash functions:

$$H_1 : \{-1, 1\}^l \rightarrow \mathbb{Z}_q^n; H_2 : \mathbb{Z}_q^n \times \{0, 1\}^* \rightarrow \{h : h \in \mathbb{Z}_q, |h| \leq \kappa\}; \tag{3}$$

  - Set and output a public parameter and master secret key as

$$pp = (\mathbf{A}, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_l, \mathbf{B}, \mathbf{U}, \mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_k, \mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k, \mathbf{u}, H_1, H_2, s, s_1), \mathsf{MSK} = \mathbf{T_A}.$$

– $\mathsf{KeyGen}(pp, \mathsf{MSK}, \mathbf{ID})$: Taking a public parameter pp, a master secret key MSK, and a user with identity $\mathbf{ID}$ as input, this algorithm executes the following procedures.
  - Set $\mathbf{ID} = (\mathrm{id}_1, \mathrm{id}_2, \cdots, \mathrm{id}_l) \in \{-1, 1\}^l$ as the identity (public key) of a user.
  - Sample a vector $\mathbf{sk_{ID}} \in \mathbb{Z}_q^m$ as

$$\mathbf{sk_{ID}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, H_1(\mathbf{ID}), s), \tag{4}$$

    where $\mathbf{sk_{ID}}$ s.t. $\mathbf{A} \cdot \mathbf{sk_{ID}} = H_1(\mathbf{ID})$ and $\mathbf{sk_{ID}}$ is statistically distributed in $\mathcal{D}_{\Lambda_q^{H_1(\mathbf{ID})}(\mathbf{A}),s}$.
  - Output $\mathbf{sk_{ID}}$ as a secret key of the user with identity $\mathbf{ID}$.

– $\mathsf{Authorize}(pp, \mathbf{sk_{ID}}, \mathcal{KS}_W, t)$: After inputs a public parameter pp, a secret key $\mathbf{sk_{ID}}$ of user with identity $\mathbf{ID}$ and defines the authorized keyword set $\mathcal{KS}_W$ ($\mathcal{KS}_W \subseteq \mathcal{KS}_U$) together with the authorized time t, this algorithm processes the following steps.
  - Set a polynomial as

$$f(\mathbf{sk_{ID}}, \mathbf{kw}) = \mathbf{sk_{ID}} + \sum_{i=1}^{k} kw_i \mathbf{m}_i \in \mathbb{Z}_q^m, \tag{5}$$

    where $\mathbf{kw} = (kw_1, kw_2, \cdots, kw_k)$, and $kw_i \in \{0, 1\}$ is each bit of $\mathbf{kw}$.
  - Set three polynomials as

$$f_1(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \mathcal{KS}_U} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m, \tag{6}$$

$$f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \mathcal{KS}_W} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m, \tag{7}$$

$$F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \{\mathcal{KS}_U - \mathcal{KS}_W\}} f(\mathbf{sk_{ID}}, \mathbf{kw}) \in \mathbb{Z}_q^m. \tag{8}$$

    We then involve a time period t in the signature σ. With time updates, the authority will re-authorize the user with a novel authorization token to ensure the signature timeliness. The authority generates the authorized key with its signature as below.
  - Calculate an authorized key $\mathbf{SK_{ID}} \in \mathbb{Z}_q^{m \times m}$ as

$$\mathbf{SK_{ID}} \leftarrow \mathsf{RandBasis}(F(\mathbf{sk_{ID}}), s_1). \tag{9}$$

  - Select a random vector $\mathbf{y} \in \mathbb{Z}_q^m$.
  - Calculate a parameter $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t) \in \mathbb{Z}_q$, and a vector $\mathbf{z} = \mathbf{sk_{ID}} \cdot h + \mathbf{y} \in \mathbb{Z}_q^m$.
  - Calculate a signature σ of an authorized key $\mathbf{SK_{ID}}$ as

$$\sigma = (h, \mathbf{z}) = (H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t), \mathbf{sk_{ID}}h + \mathbf{y}) \tag{10}$$

    with probability $\mathsf{min}(1, \frac{\mathcal{D}_s^m(\mathbf{z})}{M\mathcal{D}_{\mathbf{sk_{ID}}h,s}^m(\mathbf{z})})$, where t is the authorized time.
  - The authority distributes an authorized token $token = (\mathbf{SK_{ID}}, \sigma, \mathbf{y}, t)$ to the user through a secure communication channel.

– $\mathsf{Encrypt}(pp, \mathbf{ID}, \mathbf{kw})$: Taking a public parameter pp, a user with identity $\mathbf{ID}$, and a keyword $\mathbf{kw} \in \mathcal{KS}_U$ as input, this algorithm performs the following operations.
  - Calculate a matrix $\mathbf{A_{kw}} = \sum_{i=1}^{k} kw_i \mathbf{M}_i + \mathbf{B} \in \mathbb{Z}_q^{n \times m}$, where $kw_i \in \{0, 1\}$ is each bit of $\mathbf{kw}$.
  - Calculate a matrix $\mathbf{A_{ID}} = \sum_{i=1}^{l} (\mathrm{id}_i + \mathbf{A}_i) + \mathbf{B} \in \mathbb{Z}_q^{n \times m}$.

- Set a matrix $\mathbf{A_{ID,kw}} = (\mathbf{A_{ID}}|\mathbf{A_{kw}}) \in \mathbb{Z}_q^{n\times 2m}$.
- For $i = 1, 2, \cdots, k$, uniformly random select k matrices $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m\times m}$.
- Calculate a matrix $\mathbf{R_{kw}} = \sum_{i=1}^{k} kw_i \cdot \mathbf{R}_i$.
- Uniformly random select a noise value $noi \xleftarrow{\$} \bar{\Psi}_\alpha \in \mathbb{Z}_q$ and a noise vector $\mathbf{noi} \xleftarrow{\$} \bar{\Psi}_\alpha^m \in \mathbb{Z}_q^m$.
- Uniformly random select a vector $\mathbf{r} \in \mathbb{Z}_q^n$.
- Calculate two ciphertexts as

$$\mathrm{c}_0 = \mathbf{u}^\top \mathbf{r} + noi \in \mathbb{Z}_q, \text{and } \mathbf{c}_1 = \mathbf{A}_{\mathbf{ID,kw}}^\top \cdot \mathbf{r} + (\mathbf{noi}, \mathbf{R}_{\mathbf{kw}}^\top \cdot \mathbf{noi}) \in \mathbb{Z}_q^{2m}. \tag{11}$$

- Output a ciphertext tuple $\mathrm{CT} = (\mathrm{c}_0, \mathbf{c}_1)$.

– $\mathsf{Trapdoor}(pp, \mathbf{ID}, token, \mathbf{kw})$: Given a public parameter pp, a user with identity $\mathbf{ID}$, an authorized token $token$, and a keyword $\mathbf{kw} \in \mathcal{KS}_W$, this algorithm executes these operations below.

- Calculate a matrix $\mathbf{A_{kw}} = \sum_{i=1}^{k} kw_i \mathbf{M}_i + \mathbf{B} \in \mathbb{Z}_q^{n\times m}$, where $kw_i \in \{0, 1\}$.
- Calculate a matrix $\mathbf{A_{ID}} = \sum_{i=1}^{l} (\mathrm{id}_i + \mathbf{A}_i) + \mathbf{B} \in \mathbb{Z}_q^{n\times m}$.
- Calculate a vector $\mathbf{trap_1} = \mathbf{SK_{ID}y} \in \mathbb{Z}_q^m$.
- Sample a vector $\mathbf{trap_2} \in \mathbb{Z}_q^{2m}$ as

$$\mathbf{trap_2} \leftarrow \mathsf{SampleLeft}(\mathbf{A_{ID}}, \mathbf{A_{kw}}, \mathbf{SK_{ID}}, \mathbf{u}, s). \tag{12}$$

- Output a trapdoor tuple $\mathrm{Trap} = (\mathbf{trap_1}, \mathbf{trap_2}, \sigma, t)$.

– $\mathsf{Test}(pp, \mathbf{ID}, \mathrm{CT}, \mathrm{Trap}, t')$: After input a public parameter pp, a user with identity $\mathbf{ID}$, a ciphertext CT, a trapdoor Trap, and t' refers to the time when the server received the trapdoor Trap, this algorithm will judge the validity of the above parameters.

- 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 $h \stackrel{?}{=} H_2(\mathbf{Az} - H_1(\mathbf{ID})h, \mathbf{trap_1}, t)$ and $\|\mathbf{z}\| \stackrel{?}{\leq} 2s\sqrt{m}$.
  If this equation holds, the trapdoor is actually from an authorized user and processes the following step.
- Finally, the server checks the error term $|\mathrm{c}_0 - \mathbf{trap_2}^\top \mathbf{c_1}| \leq \lfloor \frac{q}{4} \rfloor$.
  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.

5 Correctness and Parameters

5.1 Parameters Setting

We need to make sure that these parameters are well set for our scheme to work properly: $\alpha = [m^2k^2\omega(\log n)]^{-1}$, $q > \frac{2\sqrt{n}}{\alpha}$, $m = \lceil 6n \log q \rceil$, $s = km\omega(\sqrt{\log n})$, $q \geq m^{2.5}\omega(\sqrt{\log n})$, κ s.t. $2^\kappa \cdot \binom{512}{\kappa} \geq 2^{100}$, $M \approx \exp\left(\frac{12\kappa\sqrt{m}}{s} + \left(\frac{\kappa\sqrt{m}}{2s}\right)^2\right)$.

5.2 Correctness

We demonstrate that our scheme satisfies the correctness requirements we described previously.

Theorem 1. *The proposed lattice-based PEAKS primitive is correct with overwhelming probability if the above parameters are set properly.*

Proof. We start by proving that the signature in Trapdoor algorithm is issued by an authority correctly:

$$\begin{aligned}\mathbf{Az} - H_1(\mathbf{ID})h &= \mathbf{A}(\mathbf{sk_{ID}} \cdot h + \mathbf{y}) - H_1(\mathbf{ID})h \\ &= \mathbf{A} \cdot \mathbf{sk_{ID}} \cdot h + \mathbf{Ay} - H_1(\mathbf{ID})h \\ &= H_1(\mathbf{ID})h + \mathbf{Ay} - H_1(\mathbf{ID})h \\ &= \mathbf{Ay}\end{aligned} \tag{13}$$

Thus, we obtain that $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t) \stackrel{?}{=} H_2(\mathbf{Az} - H_1(\mathbf{ID})h, \mathbf{trap_1}, t)$. 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 $\lfloor \frac{q}{4} \rfloor$ with overwhelming probability.

To begin with, we parse the trapdoor as $\mathbf{trap_2} = (\mathbf{trap_2^1}, \mathbf{trap_2^2}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^m$. After that, through the triangle inequality, the error term is bounded by:

$$\begin{aligned}|\mathrm{c}_0 - \mathbf{trap_2^\top c_1}| &= |\mathbf{u}^\top \mathbf{r} + noi - \mathbf{trap_2^\top}(\mathbf{A_{ID,kw}^\top} \cdot \mathbf{r} + (\mathbf{noi}, \mathbf{R_{kw}^\top} \cdot \mathbf{noi}))| \\ &= |\mathbf{u}^\top \mathbf{r} + noi - (\mathbf{A_{ID,kw}} \cdot \mathbf{trap}_2)^\top \mathbf{r} - \mathbf{trap}_2^\top(\mathbf{noi}, \mathbf{R_{kw}^\top} \cdot \mathbf{noi})| \\ &= |\mathbf{u}^\top \mathbf{r} + noi - \mathbf{u}^\top \mathbf{r} - \mathbf{trap_2^\top}(\mathbf{noi}, \mathbf{R_{kw}^\top} \cdot \mathbf{noi})| \\ &= |noi - \begin{pmatrix} \mathbf{trap_2^1}^\top \\ \mathbf{trap_2^2}^\top \end{pmatrix} (\mathbf{noi}, \mathbf{R_{kw}^\top} \cdot \mathbf{noi})| \\ &= |noi - (\mathbf{trap_2^1}^\top + \mathbf{trap_2^2}^\top \cdot \mathbf{R_{kw}^\top})\mathbf{noi}| \\ &= |noi - (\mathbf{trap_2^1} + \mathbf{R_{kw}} \cdot \mathbf{trap_2^2})^\top \mathbf{noi}| \\ &\leq |noi| + |(\mathbf{trap_2^1} + \mathbf{R_{kw}} \cdot \mathbf{trap_2^2})^\top \mathbf{noi}|\end{aligned} \tag{14}$$

According to the standard Gaussian tail bound, we obtain $|noi| < q\alpha\omega(\sqrt{\log m}) + \frac{1}{2}$. Then, according to Lemma 8, we have

$$|(\mathbf{trap_2^1} + \mathbf{R_{kw}} \cdot \mathbf{trap_2^2})^\top \mathbf{noi}| < s(km + \sqrt{m})(q\alpha \cdot \omega(\log m) + \frac{\sqrt{m}}{2}), \tag{15}$$

with negligible probability $\mathsf{negl}(n)$. Hence, we now conclude that the error term is bounded by

$$\begin{aligned}|\mathrm{c}_0 - \mathbf{trap_2^\top c_1}| &\leq O(sm^{1.5}) + s(\sqrt{m} + km)(q\alpha \cdot \omega(\sqrt{\log m}) + \frac{\sqrt{m}}{2}) \\ &\leq O(sm^{1.5}) + qskm\alpha \cdot \omega(\sqrt{\log m}) \\ &\leq O(m^{2.5} \cdot \omega(\sqrt{\log m})) := \frac{q}{5},\end{aligned} \tag{16}$$

with overwhelming probability $1 - \mathsf{negl}(n)$. This is the end of correctness proof. In this way, it also realizes that the inequality $q > 2\sqrt{n}/\alpha$, ensuring the LWE assumption is as hard as the worst-case SIVP and GapSVP hardness [6].

6 Security Analysis

This section illustrates that the lattice-based PEAKS construction satisfies IND-sID-CKA and T-EUF security. We analyze two security theorems and reduced them to the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-LWE and (q, n, m, β)-SIS hardness, respectively.

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 11. *[11] Given a prime number q and $0 \leq q_t \leq q$, then we say the hash family H defined above satisfies $(q_t, \frac{1}{q}(1 - \frac{q_t}{q}), \frac{1}{q})$ abort-resistant.*

Theorem 2. *The proposed lattice-based PEAKS primitive 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 $\mathcal{C}$ can solve the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-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 the real game, which is the same as described in Definition 8.

Game 1: This game is identical to **Game 0**, except that calculate $\mathbf{M}_i = \mathbf{A}_{\mathbf{ID}^*} \cdot \mathbf{R}_i^* + \theta_i \cdot \mathbf{B}$ instead of choosing is uniformly random from $\mathbb{Z}_q^{n \times m}$, where $\mathbf{R}_i^* \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and $\theta_i \xleftarrow{\$} \mathbb{Z}_q$. As for the challenge keyword $\mathbf{kw}$, let $\mathbf{R}_{\mathbf{kw}} = \sum_{i=1}^{k} kw_i \mathbf{R}_i^*$ and $(\mathbf{R}_{\mathbf{kw}}^*)^\top \cdot \mathbf{noi} \in \mathbb{Z}_q^m$ 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 $\mathbb{Z}_q^{n \times m}$. 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, $\mathcal{C}$ selects a hash function $h \in H$ at random and keeps it secret. When $\mathcal{C}$ receives the secret key of user with identity $\mathbf{ID} \neq \mathbf{ID}^*$ 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 $\mathcal{A}$, she will respond the Trapdoor queries and the challenge ciphertext same as in **Game 1**. On the other hand, given a user with identity $\mathbf{ID}^*$ and a keyword $\mathbf{kw}^*$, $\mathcal{A}$ sends the final guess to $\mathcal{C}$. After that, $\mathcal{C}$ 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, $\mathcal{C}$ re-selects a random bit $b' \in \{0, 1\}$ and then aborts the game. Furthermore, $\mathcal{C}$ 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 presented in [26]. 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 $\mathbf{A}$ and $\mathbf{B}$. In this game, $\mathcal{C}$ uniformly selects $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ at random. Then, $\mathcal{C}$ calls the TrapGen algorithm to generate the matrix $\mathbf{B}$ and its basis $\mathbf{T}_\mathbf{B}$. In this way, the response for KeyGen

queries of $\mathcal{C}$ are the same as those in **Game 2**. As for the Trapdoor query, a user with identity **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 $\mathbf{trap_2} \leftarrow \mathsf{SampleRight}(\mathbf{A}, h(\mathbf{kw})\mathbf{B}, \mathbf{R_{kw}}, \mathbf{T_B}, \mathbf{u}, s) \in \mathbb{Z}_q^{2m}$ 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 $\mathcal{C}$ chooses the challenge ciphertext $\mathrm{CT} = (\mathrm{c}_0, \mathbf{c_1}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2m}$ 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 $\mathcal{C}$ has the ability to solve the LWE hardness over ϵ' probability for a target user with identity $\mathbf{ID}^*$.

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

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

- KeyGen Query. With regard to the q_t queries for the secret key of a user with identity $\mathbf{ID} \neq \mathbf{ID}^*$, $\mathcal{C}$ responds to the following answer. (1) Calculate $\mathbf{A_{ID}} = \sum_{i=1}^{l}(\mathrm{id_i} + \mathbf{A}_i) + \mathbf{B}$. (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 the keyword set $\mathcal{KS}_U - \mathcal{KS}_{W^*}$ on time t. For each keyword $\mathbf{kw} \in \mathcal{KS}_U - \mathcal{KS}_{W^*}$, and each user with identity $\mathbf{ID} \neq \mathbf{ID}^*$, perform the $\mathsf{RandBasis}(\mathsf{F}(\mathbf{sk_{ID}}), \mathsf{s}_1)$ algorithm to calculate $\mathbf{SK_{ID}}$, where $F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \{\mathcal{KS}_U - \{\mathcal{KS}_U - \mathcal{KS}_{W^*}\}\}} f(\mathbf{sk_{ID,kw}}) = \sum_{\mathbf{kw} \in \{\mathcal{KS}_U - \{\mathcal{KS}_U - \mathcal{KS}_{W^*}\}\}} f(\mathbf{sk_{ID}} + \sum_{i=1}^{k} kw_i \cdot \mathbf{m}_i)$. After that, $\mathcal{C}$ 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 $\mathsf{min}(1, \frac{\mathcal{D}_s^m(\mathbf{z})}{M\mathcal{D}^m_{\mathbf{sk_{ID}}^h, s}(\mathbf{z})})$, where t is the authorized time. Finally, $\mathcal{C}$ responds $token = (\mathbf{SK_{ID}}, \sigma, \mathbf{y}, t)$ to $\mathcal{A}$.
- Trapdoor Query. $\mathcal{A}$ queries a trapdoor of a keyword $\mathbf{kw}$ for a user with identity $\mathbf{ID}^*$. 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 herself.
  - If $\mathbf{kw} \in \mathcal{KS}_{W^*}$, $\mathcal{C}$ responds the query as follows. (1) Perform the TrapGen algorithm to generate $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ together with its trapdoor $\mathbf{T_B}$ of $\Lambda_q^\top(\mathbf{B})$. (2) For $i = 1, 2, \cdots, k$, randomly select $\mathbf{R}_i^* \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and k random parameters $\theta_i \in \mathbb{Z}_q$. (3) Then, compute $\mathbf{M}_i = \mathbf{A_{ID^*}} \cdot \mathbf{R}_i^* + \theta_i \cdot \mathbf{B} \in \mathbb{Z}_q^{n \times m}$. In this case, $\mathbf{A_{ID,kw}} = (\mathbf{A_{ID}}|\mathbf{A_{kw}}) = (\mathbf{A_{ID}}|\sum_{i=1}^{k} kw_i\mathbf{M}_i + \mathbf{B}) = (\mathbf{A_{ID}}|\mathbf{A_{ID}} \cdot \mathbf{R_{kw}} + h(\mathbf{kw})\mathbf{B})$, where $\mathbf{R_{kw}} = \sum_{i=1}^{k} kw_i\mathbf{R}_i^* \in \mathbb{Z}_q^{m \times m}$. (4) After that, compute $\mathbf{trap_1} = \mathbf{SK_{ID}} \cdot \mathbf{y} \in \mathbb{Z}_q^m$. (5) Further, invoke the $\mathsf{SampleRight}(\mathbf{A_{ID^*}}, h(\mathbf{kw})\mathbf{B})$ algorithm to generate $\mathbf{trap_2}$ of user with identity $\mathbf{ID}^*$ on the keyword $\mathbf{kw}$. (6) Ultimately, respond a trapdoor tuple $\mathrm{Trap} = (\mathbf{trap_1}, \mathbf{trap_2}, \sigma, t)$ to $\mathcal{A}$.

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

- If the above equation holds, $\mathcal{C}$ will abort the game and respond a bit $b' \in \{0,1\}$ at random.
- Otherwise, $\mathcal{C}$ calculates the challenge ciphertext for a target user with identity $\mathbf{ID}^*$ through the following procedures. (1) Calculate a vector $\mathbf{v}^* = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \in \mathbb{Z}_q^m$ and a scalar $\mathrm{c}_0^* = \mathrm{v}_0 \in \mathbb{Z}_q$. (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 $\mathbf{c}_1^* = \begin{bmatrix} \mathbf{v}^* \\ (\mathbf{R}_{\mathbf{kw}^*})^\top \cdot \mathbf{v}^* \end{bmatrix} \in \mathbb{Z}_q^{2m}$ and select a bit $b \in \{0,1\}$. (4) If $b = 0$, respond $\mathrm{CT}^* = (\mathrm{c}_0^*, \mathbf{c}_1^*)$ to $\mathcal{A}$. If $b = 1$, randomly select $\mathrm{CT} = (\mathrm{c}_0, \mathbf{c}_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2m}$ 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 $\mathcal{C}$ processes the artificial abort operation at the outset. For $i = 1, 2, \cdots, q_t$, $\mathcal{C}$ 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, $\mathcal{C}$ publishes the guess as the response for solving the LWE hardness problem. Otherwise, $\mathcal{C}$ 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 $\mathbf{noi} \xleftarrow{\$} \bar{\Psi}_\alpha^m \in \mathbb{Z}_q^m$, we have $\mathbf{v}^* = \mathbf{A}_{\mathbf{ID}^*}^\top \cdot \mathbf{r} + \mathbf{noi}$. Consequently, we obtain:

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

Under this circumstances, $\mathrm{CT}^* = (\mathrm{c}_0^*, \mathbf{c}_1^*)$ is a valid challenge ciphertext since both $\mathrm{c}_0^* = \mathbf{u}^\top \mathbf{r} + noi$ 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 $\mathbf{v}^* \xleftarrow{\$} \mathbb{Z}_q^m$. In this context, the challenge ciphertext CT^* is uniform in $\mathbb{Z}_q \times \mathbb{Z}_q^{2m}$.

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 11, 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 $\mathsf{negl}(n)$ to be a negligible function.

Thus, the advantage for $\mathcal{C}$ to solve the LWE hardness is summarized as:

$$Adv_{\mathcal{C}}^{\mathrm{LWE}} \geq \frac{1}{2}((p_{\max} - p_{\min}) + p_{\min} | \Pr[b' = b] - \frac{1}{2}|) \geq \frac{p}{4q} + \mathsf{negl}(n).$$

In a nutshell, we conclude: $|\Pr[\mathbf{Game}_3(1^\lambda) = 1] - \Pr[\mathbf{Game}_4(1^\lambda) = 1]| \leq Adv_{\mathcal{C}}^{\mathrm{LWE}}$.

Up to this point, we have verified that the above theorem holds.

Theorem 3. *The proposed lattice-based PEAKS primitive 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* $\mathcal{C}$ *can solve the* (q, n, m, β)-SIS *hardness.*

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

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

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

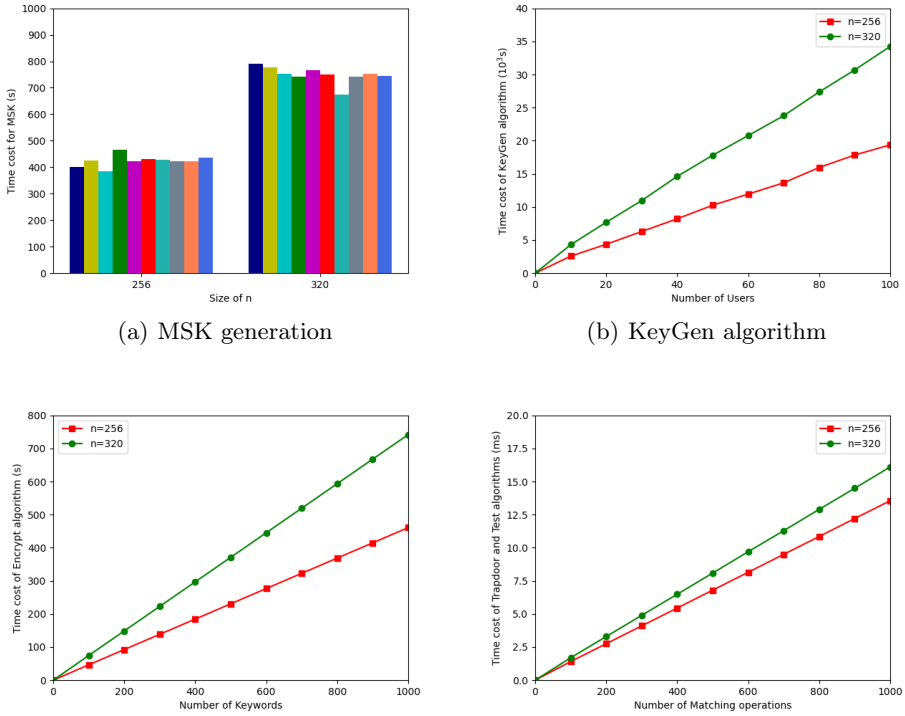
- Hash Query. $\mathcal{C}$ maintains a list $L(\mathbf{SK_{ID}}, t, h)$, which is initially empty. When $\mathcal{A}$ queries about $(\mathbf{SK_{ID}}, t)$, $\mathcal{C}$ searches the list L. If the tuple $(\mathbf{SK_{ID}}, t, h)$ exists, $\mathcal{C}$ will return h to $\mathcal{A}$. Otherwise, $\mathcal{C}$ randomly selects a vector $\mathbf{y}$ and calculates $h = H_2(\mathbf{Ay}, \mathbf{SK_{ID}y}, t)$. After that, $\mathcal{C}$ 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 with identity $\mathbf{ID} \neq \mathbf{ID}^*$, $\mathcal{C}$ responds to the following answer. (1) Calculate $\mathbf{A_{ID}} = \sum_{i=1}^{l}(\mathrm{id_i} + \mathbf{A}_i) + \mathbf{B}$. Assume that the i-th bit of the user with identity $\mathbf{ID}^*$ 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 keyword set on time t. For each keyword $\mathbf{kw}$ and each user with identity $\mathbf{ID} \neq \mathbf{ID}^*$, perform the $\mathsf{RandBasis}(\mathsf{F}(\mathbf{sk_{ID}}), \mathsf{s}_1)$ algorithm to compute $\mathbf{SK_{ID}}$, where $F(\mathbf{sk_{ID}}) = f_1(\mathbf{sk_{ID}}) - f_2(\mathbf{sk_{ID}}) = \sum_{\mathbf{kw} \in \{\mathcal{KS}_U - \mathcal{KS}_W\}} f(\mathbf{sk_{ID,kw}}) = \sum_{\mathbf{kw} \in \{\mathcal{KS}_U - \mathcal{KS}_W\}} f(\mathbf{sk_{ID}} + \sum_{i=1}^{k} kw_i \cdot \mathbf{m}_i)$. After that, $\mathcal{C}$ 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 $\mathsf{min}(1, \frac{\mathcal{D}_s^m(\mathbf{z})}{M\mathcal{D}^m_{\mathbf{sk_{ID}}h,s}(\mathbf{z})})$, where t is the authorized time. Finally, $\mathcal{C}$ responds $token = (\mathbf{SK_{ID}}, \sigma, \mathbf{y}, t)$ to $\mathcal{A}$.
- Trapdoor Query. $\mathcal{A}$ queries a trapdoor of a keyword $\mathbf{kw}$. Then, $\mathcal{A}$ calculates the authorized token by asking for the Authorization Query and generates the trapdoor herself.

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 valid forged signature and the tuple $(\mathbf{SK_{ID}^*}, t^*, h^*)$ is in the list L. In this way, we have $H_2(\mathbf{Az}^* - H_1(\mathbf{ID})h, \mathbf{SK_{ID}^*y}, t) = H_2(\mathbf{Az} - H_1(\mathbf{ID})h, \mathbf{SK_{ID}y}, t)$. If $\mathbf{SK_{ID}} \neq \mathbf{SK_{ID}^*}$ 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 $\mathbf{SK_{ID}} = \mathbf{SK_{ID}^*}$ 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 $\mathbf{z} - \mathbf{z}^* \neq 0$, and due to $\|\mathbf{z}\|, \|\mathbf{z}^*\| \leq 2s\sqrt{m}$, we have that $\|\mathbf{z} - \mathbf{z}^*\| \leq 4s\sqrt{m}$. Consequently, $\|\mathbf{z} - \mathbf{z}^*\|$ is a solution of the (q, n, m, β)-SIS assumption.

According to Lemma 5.3 and 5.4 [24], the probability for finding a solution of the (q, n, m, β)-SIS assumption is $\epsilon = (\frac{1}{2} - 2^{-100})(\delta - 2^{-100})(\frac{\delta - 2^{-100}}{q_h + q_a} - 2^{-100}) \approx \frac{\delta^2}{2(q_h + q_a)}$. Therefore, $\mathcal{C}$ can solve the (q, n, m, β)-SIS assumption with probability ϵ.

7 Performance Evaluation and Comparison

All the experiments[4] for our primitive is accomplished on a Windows 10 Laptop with Core i7-11800H CPU 2.30 GHz and 24 GB RAM. We use Matlab to implement our scheme and set the security parameter as $n = 256, m = 9753, q = 4093$ and $n = 320, m = 13133, q = 8191$ following the existing works [10, 27, 28]. Our comprehensive performance evaluations are illustrated in Fig. 2. As for 2(a), we obtain the time cost for the MSK generation algorithm ten times under two security parameter settings ($n = 256$ and $n = 320$). In addition, we indicate the performance of KeyGen, Encrypt, and Trapdoor & Test algorithms in Fig. 2(b), 2(c), and 2(d), respectively.



(a) MSK generation

(b) KeyGen algorithm

(c) Encrypt algorithm

(d) Trapdoor and Test algorithms

Fig. 2. Performance Evaluation of L-PEAKS Scheme.

Subsequently, we compare our L-PEAKS primitive with existing schemes [1–3,10,14,15,18,29–33] in three aspects: security properties, space complexity, and computation complexity.

[4] 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 for each run should be consistent with those in this paper.

A comparative security analysis of our L-PEAKS scheme and existing PEKS and PEAKS mechanisms is elaborated in Table 1. From the table, it is evident that schemes [1, 10, 14, 18] as well as our scheme incorporates the concept of identity-based encryption. Moreover, schemes [2, 10, 14, 32] along with our scheme demonstrate resistance against quantum computing attacks.

For the authorized property, scheme [3, 15, 18, 29] and our scheme meets the requirement. The security property IND-CKA is essential and is not satisfied by only one scheme [30] among the existing schemes. As for the T-EUF security, it is achieved by schemes [3, 29] 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.

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

Schemes	**ID**	**QR**	**Authorize**	**IND-CKA**	**T-EUF**
Boneh et al. [1]	✓	×	×	✓	×
Xu et al. [2]	×	✓	×	✓	×
Jiang et al. [3]	×	×	✓	✓	✓
Jiang et al. [29]	×	×	✓	✓	✓
Huang et al. [30]	×	×	×	×	×
Xu et al. [10]	✓	✓	×	✓	×
Wang et al. [18]	✓	×	✓	✓	×
Chen et al. [31]	×	×	×	✓	×
Liu et al. [32]	×	✓	×	✓	×
Zhang et al. [14]	✓	✓	×	✓	×
Liu et al. [15]	×	×	✓	✓	×
Our scheme	✓	✓	✓	✓	✓

Notes. **ID**: Identity-based scheme. **QR**: Quantum-resistance.

Table 2. Space complexity comparison

Schemes	**Secret key Size**	**Ciphertext Size**	**Trapdoor Size**
Cheng et al. [33]	$\lvert\mathbb{Z}^{m\times m}\rvert$	$\lvert\mathbb{Z}_q^{8m\times \ell_S}\rvert$	$\lvert\mathbb{Z}_q^{8m\times \ell_R}\rvert$
Xu et al. [10]	$\lvert\mathbb{Z}_q^{(m+m')\times(m+m')}\rvert$	$\lvert\mathbb{Z}_q^{2(m+m')}\rvert$	$\lvert\mathbb{Z}_q^{2(m+m')}\rvert$
Xu et al. [2]	$\lvert\mathbb{Z}_q^{m+\kappa}\rvert$	$\lvert\mathbb{Z}_q^{\rho((d+3)m+2)}\rvert$	$\lvert\mathbb{Z}_q^{(d+3)m}\rvert$
Liu et al. [32]	$\lvert\mathbb{Z}_q^{m+n\kappa+m\times m}\rvert$	$\lvert\mathbb{Z}_q^{\rho(2m+2)}\rvert$	$\lvert\mathbb{Z}_q^{2m}\rvert$
Zhang et al. [14]	$\lvert\mathbb{Z}^{m\times m}\rvert$	$\lvert\mathbb{Z}_q^{(m+1)\times\rho}\rvert$	$\lvert\mathbb{Z}_q^{m}\rvert$
Our scheme	$\lvert\mathbb{Z}_q^{m}\rvert$	$\lvert\mathbb{Z}_q^{2m+1}\rvert$	$\lvert\mathbb{Z}_q^{4m+2}\rvert$

We also demonstrate the comparison analysis of space complexity and computation complexity with other five primitives [2, 10, 14, 32, 33] in Tables 2 and 3, respectively. In Table 2, we compare the secret key size, ciphertext size, and trapdoor size with others. As for Table 3, we analyze the computation complexity of the encryption, trapdoor, and search algorithms. From Table 2 and

Table 3. Computation complexity comparison

Schemes	Encrypt Comp.	Trapdoor Comp.	Search Comp.
Cheng et al. [33]	$O(\rho m \ell_S)$	$O(\rho m \ell_R)$	$O(n^2)$
Xu et al. [10]	$O(n^3)$	$O(\rho(m+m'))$	$O(n^2)$
Xu et al. [2]	$O(\kappa n^3)$	$O(n^3)$	$O(\kappa n^2)$
Liu et al. [32]	$O(\kappa n^3)$	$O(n^3)$	$O(\kappa n^2)$
Zhang et al. [14]	$O(n^2)$	$O(\rho m^2)$	$O(n^2)$
Our scheme	$O(n^3)$	$O(\rho m)$	$O(n^2)$

Table 3, it is evident that our L-PEAKS scheme guarantees excellent security performance without introducing additional space and computation overheads.

8 Potential Applications

We show two applications in a novel context of utilizing the proposed scheme.

Application in the cloud-assisted IoT scenario. Cloud-assisted IoT combines the advantages of cloud computing and IoT to facilitate data collection from the real world, data sharing, and data analysis [34,35]. While traditional PEKS schemes can support ciphertext search over cloud-assisted IoT. However, sometimes it may be necessary to manage the authority of searchers hierarchically. For instance, in an enterprise, ordinary employees can only search for authorized information authorized by the managers. Our L-PEAKS scheme features quantum resistance, ciphertext search, and search privilege control, which is suitable for cloud-assisted IoT scenarios.

Application in the flexible metadata scenario. Metadata plays a crucial role in organizing, finding, and comprehending data [18,36]. However, it also holds a large amount of sensitive information. In addition to the requirement for ciphertext search, it is also important to consider the search privilege over metadata. For instance, when it contains commercial attributes, we should appropriately label them as private or public for different scenario needs. Therefore, our proposed L-PEAKS not only ensures a secure and flexible mechanism for handling metadata search privileges but also enjoys quantum security.

9 Conclusion

In this paper, we propose an L-PEAKS primitive from LWE and SIS hardness. It allows an authority to authorize users to search different keyword sets 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 notation to support users to encrypt data directly via their identities to reduce the storage overhead of public key certificates. After that, we conduct a detailed security reduction and comprehensive performance evaluation of our scheme. Finally, we provide theoretical comparisons with regard to space and computation complexity and show two application scenarios.

References

1. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 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*, pages 506–522. Springer, 2004.
2. Shiyuan Xu, Yibo Cao, Xue Chen, Yanmin Zhao, and Siu-Ming Yiu. Post-quantum public-key authenticated searchable encryption with forward security: General construction, and applications. Cryptology ePrint Archive, Paper 2023/591, 2023. `https://eprint.iacr.org/2023/591`.
3. Peng Jiang, Yi Mu, Fuchun Guo, and Qiaoyan Wen. 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*, pages 170–186. Springer, 2016.
4. Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
5. Gang Xu, Shiyuan Xu, Yibo Cao, Fan Yun, Yu Cui, Yiying Yu, Ke Xiao, et al. Ppseb: a postquantum public-key searchable encryption scheme on blockchain for e-healthcare scenarios. *Security and Communication Networks*, 2022, 2022.
6. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.
7. Xue Chen, Shiyuan Xu, Tao Qin, Yu Cui, Shang Gao, and Weimin Kong. Aq–abs: Anti-quantum attribute-based signature for emrs sharing with blockchain. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1176–1181. IEEE, 2022.
8. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*, pages 22–41. Springer, 2014.
9. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Journal of cryptology*, 21:350–391, 2008.
10. Lei Xu, Xingliang Yuan, Ron Steinfeld, Cong Wang, and Chungen Xu. Multi-writer searchable encryption: An lwe-based realization and implementation. In *Proceedings of the 2019 ACM Asia conference on computer and communications security*, pages 122–133, 2019.
11. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h) ibe in the standard model. In *Eurocrypt*, volume 6110, pages 553–572. Springer, 2010.
12. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25:601–639, 2012.
13. Chunxiang Gu, Yan Guang, Yuefei Zhu, and Yonghui Zheng. Public key encryption with keyword search from lattices. *International journal of information technology*, 19(1):1–10, 2013.
14. Xiaojun Zhang, Chunxiang Xu, Huaxiong Wang, Yuan Zhang, and Shixiong Wang. Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things. *IEEE Transactions on dependable and secure computing*, 18(3):1019–1032, 2021.
15. Zi-Yuan Liu, Chu-Chieh Chien, Yi-Fan Tseng, Raylin Tso, and Masahiro Mambo. Public key encryption with hierarchical authorized keyword search. In *International Conference on Information Security and Cryptology*, pages 147–170. Springer, 2022.
16. Hui Cui, Robert H Deng, Joseph K Liu, and Yingjiu Li. 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*, pages 106–126. Springer, 2017.
17. Lingling Xu, Wanhua Li, Fangguo Zhang, Rong Cheng, and Shaohua Tang. Authorized keyword searches on public key encrypted data with time controlled keyword privacy. *IEEE Transactions on Information Forensics and Security*, 15:2096–2109, 2019.

18. Jiabei Wang, Rui Zhang, Jianhao Li, and Yuting Xiao. Owner-enabled secure authorized keyword search over encrypted data with flexible metadata. *IEEE Transactions on Information Forensics and Security*, 17:2746–2760, 2022.
19. Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.
20. Chris Peikert. 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*, pages 80–97. Springer, 2010.
21. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.
22. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.
23. David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. *Cryptology ePrint Archive*, 2009.
24. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.
25. Von Neumann. Various techniques used in connection with random digits. *Notes by GE Forsythe*, pages 36–38, 1951.
26. Brent Waters. 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*, pages 114–127. Springer, 2005.
27. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
28. Rouzbeh Behnia, Muslum Ozgur Ozmen, and Attila Altay Yavuz. Lattice-based public key searchable encryption from experimental perspectives. *IEEE Transactions on Dependable and Secure Computing*, 17(6):1269–1282, 2018.
29. Peng Jiang, Yi Mu, Fuchun Guo, and Qiaoyan Wen. Secure-channel free keyword search with authorization in manager-centric databases. *Computers & Security*, 69:50–64, 2017.
30. Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14, 2017.
31. Biwen Chen, Libing Wu, Sherali Zeadally, and Debiao He. Dual-server public-key authenticated encryption with keyword search. *IEEE Transactions on Cloud Computing*, 10(1):322–333, 2019.
32. Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. 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*, pages 423–436, 2022.
33. Leixiao Cheng and Fei Meng. Public key authenticated encryption with keyword search from lwe. In *European Symposium on Research in Computer Security*, pages 303–324. Springer, 2022.
34. Hu Xiong, Tianang Yao, Hanxiao Wang, Jun Feng, and Shui Yu. A survey of public-key encryption with search functionality for cloud-assisted iot. *IEEE Internet of Things Journal*, 9(1):401–418, 2021.
35. Wei Wang, Peng Xu, Dongli Liu, Laurence Tianruo Yang, and Zheng Yan. Lightweighted secure searching over public-key ciphertexts for edge-cloud-assisted industrial iot devices. *IEEE Transactions on Industrial Informatics*, 16(6):4221–4230, 2019.
36. Yang Yang, Ximeng Liu, Robert H Deng, and Jian Weng. Flexible wildcard searchable encryption system. *IEEE Transactions on Services Computing*, 13(3):464–477, 2017.