Dual Fine-Grained Public-Key Searchable Encryption from Lattices

Zike Jiang[1], Shixin Chen[1]

[1] College of Cyber Security, Jinan University, China

Correspondence: Zike Jiang, College of Cyber Security, Jinan University, China.

Received: December 27, 2021 Accepted: January 18, 2022 Online Published: January 21, 2022

doi:10.5539/cis.v15n1p66 URL: https://doi.org/10.5539/cis.v15n1p66

Abstract

Fine-grained public key encryption with keyword search (PEKS), allowing users to search on encrypted data with flexible access control policy, has been widely studied recently due to its promising application to real-world scenarios such as cloud computing. However, most of the existing fine-grained PEKS schemes are either only able to support single access control (e.g., attribute-based access control) or susceptible to being attacked or compromised by quantum computers in or after a short time. In this paper, we propose a fine-grained PEKS scheme that offers dual access control based on lattice. In particular, we first define a dual fine-grained PEKS primitive against chosen keyword attacks under selective security. Subsequently, we adapt the key homomorphic technique and noise rerandomization technique to design a concrete scheme. Particularly, the keyword space in our construction is unlimited. Then, we present a formal security proof against chosen keyword attacks on the learning with errors (LWE) problem in the standard model. Moreover, we demonstrate the theoretical performance and experimental result of our proposed scheme. Finally, we discuss that our scheme can be easily extended to support conjunctive keywords and delegation without incurring complex operations.

Keywords: access control, public-key encryption with keyword search (PEKS), lattice-based cryptography, learning with errors

1. Introduction

Public-key encryption with keyword search (PEKS), first proposed by Boneh et al. (2004), allows users to search encrypted files by specific keywords with data privacy. The general PEKS is sufficient in some multi-user scenarios. However, in more real-world scenarios, we need some cryptographic primitives with access control to realize fine-grained schemes. Specifically, data senders need to flexibly control the search permission of data receivers, such that only by matching the attribute or the identity can data receivers search successfully, which is non-negligible in practical cloud computing. One of the solutions is applying identity-based encryption and attribute-based encryption to PEKS. Along with that, the dual access control variants go a step further in flexibility. For instance, in cloud computing, dual access control can comprise attribute-based access control for the user authentication and inner product access control for verification (Sheng, Wen, Guo, & Yin, 2013). Additionally, as a consideration, it is unacceptable to require users (data senders or data receivers) to be online permanently, which could be solved by delegation. In other words, a receiver is able to delegate his/her search permission to other users by adopting an access policy.

Although there have been a number of fine-grained PEKS schemes in the literature (Wang, Li, Li, & Xu, 2013, Shi, Lai, Li, Deng, & Weng, 2014, He et al., 2018, Zheng, Xu, & Ateniese, 2014), the security of them relies on the hardness of conventional assumptions (e.g., classical number-theoretic assumptions), which may be insecure against quantum adversaries. Subsequently, many scholars work on the schemes with quantum-resistant, especially lattice-based constructions. However, the existing lattice-based fine-grained PEKS scheme only offers single access control (Li, Ma, Zhang, Fan, & Li, 2019). Therefore, we are motivated to design the lattice-based PEKS scheme with fine-grained access control that supports dual access control.

1.1 Contribution

To address the above issues, in this work, we propose a lattice-based fine-grained PEKS scheme, which enjoys dual access control and unlimited keyword space. Our construction is provable in the standard model. Also, we present our implementation and experimental result. Table 1 gives the comparison between our and recent PEKS schemes in terms of access control, security, quantum-resistant and support for unlimited keyword space. Specifically, we list the main contribution of this paper as follows:

1. We define a dual access control PEKS primitive, which enables both data senders and data receivers to specify the search permission to the encrypted data. Dual access control in our scheme consists of attribute-based access control and inner product access control (two pairs of access control) simultaneously. Our concrete construction

adopts the key homomorphic technique and the noise rerandomization to realize it. Concretely, the data sender generates the ciphertext corresponding to the keyword w by his/her attribute $\mathbf{x}$ and hidden access policy vector $\mathbf{z}$. The data receiver generates the trapdoor corresponding to the keyword w' by access policy f and public attribute vector $\mathbf{y}$. Besides $w = w'$, the successful searching needs to satisfy $f(\mathbf{x}) = 0$ and $\langle \mathbf{y}, \mathbf{z} \rangle = 0$.

2. We present a security model for our proposed scheme: ciphertext indistinguishability which can resist the chosen keyword attacks with selective attribute and access policy (**IND-sAF-CKA** for short). Then we give the proof based on the hardness assumption from lattices in the standard model, which is quantum-safe. In addition, our construction supports unlimited keyword space by collision-resistant hash function (CRHF).

3. Furthermore, we give extensions and discussions of our proposed scheme to offer conjunctive keywords and delegation. In our design, data receivers could delegate their search permission to others. There are two approaches of delegation. For one thing, the data receiver can search successfully with access policy f and attribute vector $\mathbf{y}$, he/she can generate a delegated key or delegated trapdoor for another data receiver. For another thing, the data receiver is able to delegate his/her trapdoor to others without revealing the search keyword.

Table 1. Comparison between ours and recent PEKS schemes

Schemes	Access control	Security[*]	Quantum-resistant	Unlimited keyword space
Behnia, Ozmen, & Yavuz, 2018	✗	IND-CKA, standard model	✓	✗
Zhang et al., 2019	identity-based	IND-CKA, random oracle	✓	✓
Xu, Yuan, Steinfeld, Wang, & Xu, 2019	identity-based	IND-CKA, standard model	✓	✗
Li et al., 2019	attribute-based	IND-CKA, random oracle	✓	✓
He et al., 2018	dual access control	IND-CKA, random oracle	✗	✗
Ours	dual access control	IND-sAF-CKA, standard model	✓	✓

Note. [*]IND-CKA: ciphertext indistinguishability against chosen keyword attacks.

1.2 Related Works

Boneh et al. (2004) first introduced the concept of PEKS and gave a concrete construction. Following that, Abdalla et al. (2005) revisited the PEKS and built the connection between PEKS and anonymous IBE, which has a certain impact on later PEKS schemes (Park, Kim, & Lee, 2004, Abdalla et al., 2005, Boneh, Kushilevitz, Ostrovsky, & Skeith, 2007, Bao, Deng, Ding, & Yang, 2008). Nevertheless, these PEKS schemes in single-user settings are unable to achieve more flexible and sophisticated search in multi-user settings for practical applications. Bao et al. (2008) offered security notions for multi-user settings as well as a provable construction. Besides, fine-grained access control is more desired in practical scenarios. A new primitive called ciphertext-policy attribute-based encryption with keyword search, proposed by Wang et al. (2013), solved the issue with secure construction from bilinear pairings. Aiming at the search permission about keywords, not just users' attributes, Shi et al. (2014) proposed an authorized keyword search scheme inspired by the dual-policy attribute-based encryption scheme. Similarly, He et al. (2018) proposed a searchable encryption primitive with attribute-based access control for hybrid boolean keyword search. Moreover, Zheng et al. (2014) proposed a variant keyword search scheme with verification. However, the above schemes are vulnerable to be against quantum attacks. For designing quantum-safe schemes, lattice-based constructions, especially based on LWE problem, are effective alternatives (Regev, 2006). Behnia et al. (2018) presented an LWE-based PEKS scheme by combining IBE (Agrawal, Boneh, & Boyen, 2010) and general transformation (Abdalla et al., 2005). However, the keyword space in this scheme is limited. Another LWE-based PEKS scheme with limited keyword space was presented by Xu et al. (2019), which realized the identity-based feature. Both of (Behnia et al., 2018) and (Xu et al., 2019) offer provable security in the standard model. Also, there are some fine-grained schemes in the random oracle model. Zhang et al. (2019) provided an identity-based PEKS scheme with a delegation that needs a proxy in the system model. The proxy is authorized by an original data sender and delegates encrypted data. However, this work has a dispute over security recently (Liu, Tseng, Tso, & Lee, 2021). Li et al. (2019) proposed a key policy attribute-based keywords search scheme based on LWE problem and the scheme is proven against chosen keyword attacks and keyword guessing attacks in the random oracle model, which integrates the key homomorphic technique from (Boneh et al., 2014) and construction technique from (Agrawal et al., 2010).

1.3 Paper Organization

The rest of this paper is organized as follows. Preliminaries are shown in Section 2 for listing the background of lattices, algorithms for sampling and key homomorphic features. In Section 3, we give the formal definition and security model for our proposed scheme. The construction and correctness analysis are described in Section 4. In Section 5, we prove our proposed scheme in the standard model by games. In Section 6, we illustrate the performance of our scheme in terms of computation and storage. Section 7 presents the extensions and discussions of our proposed scheme. We conclude in Section 8.

2. Preliminaries

2.1 Notations.

Let λ be the security parameter. For integer n, let $[n] = 1, \ldots, n$. We use bold capital letters to denote matrices, such as $\mathbf{A}, \mathbf{B}$. Correspondingly, we denote vectors as bold lowercase letters, such as $\mathbf{x}, \mathbf{y}$. We denote the matrix of horizontally (resp. veritically) as $(\mathbf{A}|\mathbf{B})$ (resp. $(\mathbf{A}||\mathbf{B})$).

2.2 Lattices Preliminaries

Lattices. For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and any vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define $\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{0} \bmod q\}$, $\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{u} \bmod q\}$.

Norms. For any vector $\mathbf{x}$, we use $||\mathbf{x}||$ to denote its l_2 norm. For any matrix $\mathbf{A}$, $||\mathbf{A}||$ denotes the maximum of the l_2 norm over the columns of $\mathbf{A}$. $||\mathbf{A}||_2$ denotes the operator norm of $\mathbf{A}$ defined as $||\mathbf{A}||_2 = sup_{||\mathbf{x}||=1} ||\mathbf{Ax}||$. $\widetilde{\mathbf{A}}$ denotes its Gram-Schmidt orthogonalization.

In addition, $s_1(\mathbf{A})$ denotes the largest singular value of $\mathbf{A}$. There is a lemma corresponding to computation of the largest singular value:

Lemma 1 (Ducas & Micciancio, 2014) *Let $\mathbf{X} \in \mathbb{R}^{n\times m}$ be a subgaussian random matrix with parameter s. There exists a universal constant $C \approx \frac{1}{\sqrt{2\pi}}$ such that for any $t \geq 0$, we have $s_1(\mathbf{X}) \leq Cs(\sqrt{m} + \sqrt{n} + t)$ except with probability at most $\frac{2}{e^{\pi t^2}}$.*

Gaussian Distribution. For any positive parameter $\sigma \in \mathbb{R}$, let $\rho_\sigma(\mathbf{x}) := \exp(-\pi\, ||\mathbf{x}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^n$ of center 0 and parameter σ. For any $n \in \mathbb{N}$ and any subset D of $\mathbb{Z}^n$, we define $\rho_\sigma(D) := \sum_{\mathbf{x}\in D} \rho_\sigma(\mathbf{x})$ the discrete integral of ρ_σ over D, and $\mathcal{D}_{D,\sigma}$ the discrete Gaussian distribution over D of parameter σ. That is, for any $\mathbf{y} \in D$, we have $\mathcal{D}_{D,\sigma}(\mathbf{y}) = \rho_\sigma(\mathbf{y})/\rho_\sigma(D)$. For succinct description, $\mathcal{D}_{n,\sigma}$ denotes the n-dimensional discrete Gaussian distribution with parameter σ.

The following definitions and corollaries are taken from (Brakerski & Vaikuntanathan, 2016).

Definition 1 *A distribution χ supported over $\mathbb{Z}$ is (B, ϵ)-bounded if* $\Pr_{x\leftarrow\chi}[|x| < B] < \epsilon$.

Definition 2 *A distribution $\widetilde{\chi}$ supported over $\mathbb{Z}$ is (B, ϵ)-swallowing if for all $y \in [-B, B] \cap \mathbb{Z}$ it holds that $\widetilde{\chi}$ and $y + \widetilde{\chi}$ are within ϵ statistical distance.*

Corollary 1 *For every B, ϵ, δ there exists an efficiently sampleable distribution that is both (B, ϵ)-swallowing and $(B\sqrt{log(1/\delta)}/\epsilon, O(\delta))$-bounded.* In this paper, the security of our construction relies on two assumptions: learning with errors (LWE) and inhomogeneous short integer solutions (ISIS).

Learning with errors (LWE). (Regev, 2009) Let q be a prime, χ be a distribution over $\mathbb{Z}_q$ and $\mathbf{s}$ be uniformly random over $\mathbb{Z}_q^n$. For an integer m, The $\mathsf{LWE}_{q,\chi,n}$ problem is to distinguish between the distributions $(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e})$ and $(\mathbf{A}, \mathbf{u})$ where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{u} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m$.

Proposition 1 (Regev, 2009) *Let $\alpha = \alpha(n) \in (0, 1)$ and let $q = q(n)$ be a prime such that $\alpha q > 2\sqrt{n}$.If there exist an efficient algorithm that solves $\mathsf{LWE}_{q,\alpha}$, then there exists an efficient quantum algorithm for approximating* SIVP *and* GapSVP *in the l_2 norm with $\widetilde{O}(n/\alpha)$ factors in the worst case.*

Inhomogeneous short integer solutions (ISIS). (Gentry, Peikert, & Vaikuntanathan, 2008) Given $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{y} \in \mathbb{Z}_q^n$, find $\mathbf{e} \in \mathbb{Z}^m$ satisfying $\mathbf{Ae} = \mathbf{y} \bmod q$ and $||\mathbf{e}|| \leq \beta$ where β is a small value normally.

Theorem 1 (Gentry et al., 2008) *There is a probabilistic polynomial time algorithm* SampleD *that, given a basis $\mathbf{B}$ of an n-dimensional lattice Λ, a parameter $s \geq ||\widetilde{\mathbf{B}}|| \cdot \omega(\sqrt{logm})$, and a center $\mathbf{c} \in \mathbb{R}^n$, outputs a sample from a distribution that is statistically close to $\mathcal{D}_{\Lambda,s,\mathbf{c}}$.*

Proposition 2 (Ajtai, 1999, Micciancio & Goldwasser, 2002) *For any prime $q = poly(n)$ and any $m \geq 5nlogq$, there is a probabilistic polynomial time algorithm that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a full-rank set $\mathbf{S} \subset \Lambda_q^{\perp}(\mathbf{A})$, where the distribution of $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$ and the length $||\mathbf{S}|| \leq L = m^{2.5}$. Also, $\mathbf{S}$ can be converted efficiently to a basis $\mathbf{T}$ of $\Lambda_q^{\perp}(\mathbf{A})$ such that $\widetilde{||\mathbf{T}||} \leq \widetilde{||\mathbf{S}||} \leq L$.*

Lemma 2 (Cai, 1998) *For any arbitrary integer lattice Λ, it holds that:*

$$1 \leq \min_{\mathbf{B}} ||\widetilde{\mathbf{B}}|| \leq \lambda_1(\Lambda^*) \cdot O(n), \tag{1}$$

with the minimum is over all (ordered) bases $\mathbf{B}$ of lattice Λ where Λ^* is dual lattice of Λ.

Lemma 3 (Agrawal et al., 2010) *Let q be a prime and $m > (n+1)logq + \omega(logn)$. Let $\mathbf{S}$ be chosen uniformly from $\{-1,1\}^{m\times k}$ mod q, with k polynomial in n. Let $\mathbf{A}$ and $\mathbf{B}$ be chosen uniformly from $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$ respectively. Then, for every $\mathbf{e} \in \mathbb{Z}_q^m$, it holds that the distributions $(\mathbf{A}, \mathbf{AS}, \mathbf{S}^T\mathbf{e})$ and $(\mathbf{A}, \mathbf{B}, \mathbf{S}^T\mathbf{e})$ are statistically close.*

Trapdoor generators. The following lemmas state properties of algorithms for generating short basis of lattices.

Lemma 4 (Gentry et al., 2008) *Let $n, m, q > 0$ be integers with q prime and $m = \Theta(nlogq)$. There is a probabilistic polynomial time (PPT) algorithm* TrapGen *defined as follows:*

- TrapGen$(1^n, 1^m, q)$:

Inputs: *a security parameter n, an integer m such that $m = \Theta(nlogq)$, and a prime modulus q.*

Outputs: *a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_A} \in \mathbb{Z}_q^{n\times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$ such that the distribution of $\mathbf{A}$ is close to uniform with negligible probability and $||\widetilde{\mathbf{T_A}}|| = O(\sqrt{nlogq})$.*

Lemma 5 (Boneh et al., 2014) *Let $n, m, q > 0$ be integers with q prime. There are two deterministic polynomial-time(DPT) algorithms* ExtendRight *and* ExtendLeft *defined as follows:*

- ExtendRight$(\mathbf{A}, \mathbf{T_A}, \mathbf{B})$:

Inputs: *full-rank matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$.*

Outputs: *a basis $\mathbf{T_{(A|B)}}$ of $\Lambda_q^{\perp}(\mathbf{A}|\mathbf{B})$ such that $||\widetilde{\mathbf{T_A}}|| = ||\widetilde{\mathbf{T_{(A|B)}}}||$.*

- ExtendLeft$(\mathbf{A}, \mathbf{G}, \mathbf{T_G}, \mathbf{S})$:

Inputs: *full-rank matrices $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_G}$ of $\Lambda_q^{\perp}(\mathbf{G})$.*

Outputs: *a basis $\mathbf{T_{(A|G+AS)}}$ of $\Lambda_q^{\perp}(\mathbf{A}|\mathbf{G}+\mathbf{AS})$ such that $\widetilde{\mathbf{T_{(A|G+AS)}}} \leq \widetilde{\mathbf{T_G}} \cdot (1 + ||\mathbf{S}||_2)$.*

In addition, for $m = n\lceil logq \rceil$ there is a fixed full-rank matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ s.t. the lattice $\Lambda_q^{\perp}(\mathbf{G})$ has a publicly known basis $\mathbf{T_G} \in \mathbb{Z}^{m\times m}$ with $||\mathbf{T_G}|| \leq \sqrt{5}$. The matrix $\mathbf{G}$ is such that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{G} \cdot$ BD$(\mathbf{A}) = \mathbf{A}$, where BD *is a deterministic algorithm defined in (Boneh et al., 2014).*

2.3 Sampling Algorithm

Lemma 6 (Gentry et al., 2008, Boneh et al., 2014) *Let $n, m, q > 0$ be integers with q prime. Let $\sigma = ||\widetilde{\mathbf{T_A}}|| \cdot \omega(\sqrt{logm})$. There are two PPT algorithms* SamplePre *and* RandBasis *defined as follows:*

- SamplePre$(\mathbf{A}, \mathbf{T_A}, \mathbf{U}, \sigma)$:

Inputs: *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ be a basis for $\Lambda_q^{\perp}(\mathbf{A})$ and $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$.*

Outputs: *a random sample $\mathbf{X} \in \mathbb{Z}_q^{m\times k}$ from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$.*

- RandBasis$(\mathbf{A}, \mathbf{T_A}, \sigma)$:

Inputs: *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ be a basis for $\Lambda_q^{\perp}(\mathbf{A})$.*

Outputs: *a random basis $\mathbf{T_{A'}}$ of $\Lambda_q^{\perp}(\mathbf{A})$ sampled from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^{\perp}(\mathbf{A}))$.*

Note that $||\widetilde{\mathbf{T_{A'}}}|| < \sigma\sqrt{m}$ with negligible probability.

Lemma 7 (Gentry et al., 2008) *(Preimage Samplable Functions) For any prime $q = poly(n)$, any $m \geq 5nlogq$, and any $s \geq L \cdot \omega(\sqrt{logm})$ (L be as in* ***Proposition 2****), it holds that the following distributions D_1, D_2 are statistically close:*

- $D_1 = (\mathbf{A}, \mathbf{Z}, \mathbf{U})$, *s.t.* $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q), \mathbf{U} \leftarrow \mathbb{Z}^{k\times m}, \mathbf{Z} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{s}, \mathbf{U})$.
- $D_2 = (\mathbf{A}, \mathbf{Z}, \mathbf{ZA})$, *s.t.* $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{Z} \leftarrow \mathcal{D}_{k\times n,s} : \|\mathbf{z}_i\| \leq s\sqrt{n}, i \in \{1, \ldots, n\}$ *where* $\mathbf{z}_i$ *denotes the i-row of* $\mathbf{Z}$.

Lemma 8 (Katsumata & Yamada, 2016) *(Noise Rerandomization) Given a matrix* $\mathbf{R} \in \mathbb{Z}^{m\times t}$ *and* $s \in \mathbb{R}^+$ *such that* $s^2 \geq s_1(\mathbf{RR}^T)$*, there is an algorithm* $\mathsf{NoiseGen}(\mathbf{R}, \mathbf{s})$ *as follows: It first samples* $\mathbf{e}_1 = \mathbf{Re} + \sqrt{(s^2\mathbf{I}_m - \mathbf{RR}^T)}\mathbf{e}'$ *where* $\mathbf{I}_m \in \mathbb{Z}^{m\times m}$ *denotes the identity matrix,* $\mathbf{e} \leftarrow \mathcal{D}_{t,\sigma}$ *and* $\mathbf{e}' \leftarrow \mathcal{D}_{m,\sqrt{2}\sigma}$ *are independent spherical continuous Gaussian noises. Then it samples* $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^m - \mathbf{e}_1, s\sqrt{2}\sigma}$ *and returns* $\mathbf{e}_1 + \mathbf{e}_2 \in \mathbb{Z}_q^m$*. If* $s > s_1(\mathbf{R})$*, the distribution* $(\mathbf{Re} + \mathbf{e}')$ *and the distribution* $\mathbf{e}'' \leftarrow \mathcal{D}_{m,2s\sigma}$ *are statistically close, where* $\mathbf{e} \leftarrow \mathcal{D}_{t,\sigma}$ *and* $\mathbf{e}' \leftarrow \mathsf{NoiseGen}(\mathbf{R}, s)$.

2.4 Key Homomorphic Features (Boneh et al., 2014)

For integers n and $q = q(n)$, let $m = \Theta(nlogq), \mathbf{G} \in \mathbb{Z}_q^{n\times m}$ be the gadget matrix from **Lemma 5**. For $x \in \mathbb{Z}_q, \mathbf{B} \in \mathbb{Z}_q^{n\times m}, \mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$ define the set

$$E_{\mathbf{s},\delta}(x, \mathbf{B}) = \{(x\mathbf{G} + \mathbf{B})^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m, \|\mathbf{e}\| < \delta\}. \quad (2)$$

There are three efficient deterministic algorithms $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}, \mathsf{Eval}_{\mathsf{sim}}$ that implement the key homomorphic features and satisfy the following properties with respect to some family of functions $\mathcal{F} = \{f : \mathbb{Z}_q^l \to \mathbb{Z}_q\}$ and a function $\alpha_{\mathcal{F}} : \mathbb{Z} \to \mathbb{Z}$.

- $\mathsf{Eval}_{\mathsf{pk}}(f \in \mathcal{F}, \mathbf{B}_{i\,i\in l} \in (\mathbb{Z}_q^{n\times m})^l) \to \mathbf{B}_f \in \mathbb{Z}_q^{n\times m}$.
- $\mathsf{Eval}_{\mathsf{ct}}(f \in \mathcal{F}, (x_i, \mathbf{B}_i, \mathbf{c}_i)_{i=1}^l) \to \mathbf{c}_f \in \mathbb{Z}_q^m$. Note that $x_i \in \mathbb{Z}_q, \mathbf{B}_i \in \mathbb{Z}_q^{n\times m}, \mathbf{c}_i \in E_{\mathbf{s},\delta}(x_i, \mathbf{B}_i)$. The output $\mathbf{c}_f$ satisfies $\mathbf{c}_f \in E_{\mathbf{s},\Delta}(f(\mathbf{x}), \mathbf{B}_f)$ where $\mathbf{x} = (x_1, \ldots, x_l)$ and $B_f = \mathsf{Eval}_{\mathsf{pk}}(f, (\mathbf{B}_1, \ldots, \mathbf{B}_l))$ with $\Delta < \delta \cdot \alpha_{\mathcal{F}}(n)$.
- $\mathsf{Eval}_{\mathsf{sim}}(f \in \mathcal{F}, (x_i^*, \mathbf{S}_i)_{i=1}^l, \mathbf{A}) \to \mathbf{S}_f \in \mathbb{Z}_q^{m\times m}$. Note that $x_i^* \in \mathbb{Z}_q$ and $\mathbf{S}_i \in \{-1, 1\}^{m\times m}$. The output $\mathbf{S}_f$ satisfies $\mathbf{AS}_f - f(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_f$ where $\|\mathbf{S}_f\|_2 < \alpha_{\mathcal{F}}(n), \mathbf{B}_f = \mathsf{Eval}_{\mathsf{pk}}(f, (\mathbf{AS}_1 - x_1^*\mathbf{G}, \ldots, \mathbf{AS}_l - x_l^*\mathbf{G}))$ and $\mathbf{x}^* = (x_1^*, \ldots, x_l^*)$ with all but negligible probability.

3. Definition and Security Model

In this section, we introduce the system model of our proposed primitive for cloud computing, which is based on the classical model of PEKS and consists of four entities: a data sender, a data receiver, a trusted authority and a server:

- Data sender: A data sender who sends his/her encrypted data (ciphertext) to the server by adopting access control. A data sender has a vector as a public attribute, and a secret vector as an access policy vector.
- Data receiver: A data receiver who retrieves outsourced data after sending the trapdoor corresponding to keywords to the server. A data receiver has a public access policy and a public vector as an attribute vector.
- Trust Authority: An authority who generates a master public key, master secret key, and secret key for data receivers with access policy.
- Server: An entity that offers storage and searches for encrypted data.

We give the definition of our proposed primitive as follows:

Definition 3 *Our fine-grained PEKS scheme consists of five polynomial time algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. *The setup algorithm takes the security parameter* λ *as input, and outputs the master public key* mpk *with the master secret key* msk.
- $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, f) \to \mathsf{sk}_f$. *The key generation algorithm takes the master public key* mpk*, master secret key* msk *and access policy* f *of data receiver as input. It outputs the secret key* sk_f *corresponding to* f.
- $\mathsf{Trapdoor}(\mathsf{sk}_f, f, w, \mathbf{y}) \to \mathbf{R}$. *The trapdoor generation algorithm takes the secret key* sk_f *corresponding to access policy* f*, keyword* w *and attribute vector* $\mathbf{y}$ *corresponding to data receiver. It outputs a trapdoor* $\mathbf{R}$.
- $\mathsf{Enc}(\mathsf{mpk}, w', \mathbf{x}, \mathbf{z}) \to \mathsf{ct}$. *The encrypt algorithm takes the master public key* mpk*, keyword* w'*, attribute* $\mathbf{x}$ *of data sender and access policy vector* $\mathbf{z}$ *corresponding to data sender. It outputs the ciphertext* ct.
- $\mathsf{Test}(\mathsf{mpk}, \mathsf{ct}, \mathbf{R}) \to 0/1$. *The test algorithm takes the master public key* mpk*, ciphertext* ct *and trapdoor* $\mathbf{R}$*. It outputs the 1 for successful search or 0 for failed search.*

Correctness. For any keyword w, attribute $\mathbf{x}$, attribute $\mathbf{y}$, access policy $\mathbf{z}$ and any honestly generated master public key mpk and secret key sk_f corresponding to access policy f, we require that $\mathsf{Test}(\mathsf{mpk}, \mathsf{ct}, \mathbf{R})$ outputs 1, where $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, w', \mathbf{x}, \mathbf{z})$, $\mathbf{R} \leftarrow \mathsf{Trapdoor}(\mathsf{sk}_f, f, w, \mathbf{y})$,$\langle \mathbf{y}, \mathbf{z} \rangle = 0, w = w'$, and $f(\mathbf{x}) = 0$.

We give a security model definition of our scheme: ciphertext indistinguishability which can resist the chosen keyword attacks with selective attribute and access policy (**IND-sAF-CKA**). We first give the definition of **IND-sAF-CKA** as follows:

Definition 4 ***Game IND-sAF-CKA:***

- ***Setup:*** *The adversary $\mathcal{A}$ sends a challenge attribute $\mathbf{x}^*$ and access policy f^* such that $f^*(\mathbf{x}^*) = 0$ to challenger C. The challenger C computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends mpk to the adversary $\mathcal{A}$.*
- ***Query Phase 1.*** *The adversary $\mathcal{A}$ can make trapdoor queries $(f, \mathbf{y}, w)$ for trapdoor oracle $OTrapdoor()$ and key queries f for key oracle $OKeyGen()$, that is $\mathcal{A}^{OTrapdoor(),OKeyGen()}$ with the restriction that $f(\mathbf{x}^*) \neq 0$ or $f = f^*$.*
- ***Challenge Phase.*** *The adversary $\mathcal{A}$ sends challenge $(w_0, \vec{z_0}), (w_1, \vec{z_1})$. The challenger C chooses $\beta \leftarrow \{0, 1\}$ and $\mathsf{Enc}(\mathsf{mpk}, w_\beta, \mathbf{x}^*, \vec{z_\beta})$. Note that w_0, w_1 have not been queried before and $\langle \vec{z_0}, \mathbf{y} \rangle = \langle \vec{z_1}, \mathbf{y} \rangle$ for previous queries $\mathbf{y}$.*
- ***Query Phase 2.*** *The same as **Query Phase 1**.*
- ***Guess Phase.*** *The adversary $\mathcal{A}$ outputs the guess bit β'.*

The adversary $\mathcal{A}$ wins the above game if $\beta = \beta'$. We can define the advantage of $\mathcal{A}$ as $Adv_{\mathcal{A}} = |Pr[\beta = \beta'] - 1/2|$.

4. Our Construction

We define a CRHF $H : \{0, 1\}^* \to \mathbb{Z}_q^{n\times m}, \mathbf{y} \in \{0, \ldots, Y\}^k, \mathbf{z} \in \{0, \ldots, Z\}^k$. $|\langle \mathbf{y}, \mathbf{z} \rangle| < kYZ$. A keyword is w. The length of the attribute $\mathbf{x}$ is l. we denote $\mathbf{B}$ is a public matrix corresponding to keywords and choose l matrixs $\{\mathbf{A}_i\}_{i\in[l]}$ corresponding to the attribute. Before presenting details of our construction, we give a technique overview for better understanding.

Technical Overview. Our proposed scheme is inspired by the key homomorphic technique in Boneh et al. (2014) and the noise rerandomization used in inner product encryption (Agrawal, Libert, & Stehle, 2016). Note that, we call the inner product encryption scheme in (Agrawal et al., 2016) as ALS scheme. First, we adopt $\mathsf{ExtendRight}$ and $\mathsf{RandBasis}$ to generate the secret key for the user with access policy. In addition, for generating the trapdoor corresponding to the keyword, we continue to use $\mathsf{ExtendRight}$ and $\mathsf{RandBasis}$ algorithms. Besides that, the algorithm $\mathsf{SamplePre}$ connects the trapdoor generation and the encryption by a random matrix. In the encryption phase, the ciphertext consists of three parts corresponding to LWE instances, the one is corresponding to the attribute, another is corresponding to the (private) attribute vector and ciphertext in ALS scheme, and the rest is corresponding to the keyword.

We show our concrete construction as follows (the relevant parameters are described after correctness):

$\mathsf{Setup}(1^\lambda)$: Take as input a security parameter λ. The trust authority computes the master public key and master secret key as follows:

1. Generate $(\mathbf{A}, \mathbf{T_A})$ from $\mathsf{TrapGen}(n, m, q)$.
2. Sample two uniformly random matrixs $\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{D} \leftarrow \mathbb{Z}_q^{n\times k}$.
3. Sample l uniformly random matrixs $\mathbf{A}_1, \ldots, \mathbf{A}_l \leftarrow \mathbb{Z}_q^{n\times m}$.
4. Output the master public key and master secret key

$$\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{A}_1, \ldots, \mathbf{A}_l), \mathsf{msk} = (\mathbf{T_A}).$$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, f)$: Take as input a master public key mpk, a master secret key msk and an access policy f. The trust authority computes the secret key corresponding to the access policy f as follows:

1. Compute $\mathbf{A}_f = \mathsf{Eval}_{\mathsf{pk}}(f, (\mathbf{A}_1, \ldots, \mathbf{A}_l))$.
2. Sample a random basis $\mathbf{T}_f \leftarrow \mathsf{RandBasis}(\mathbf{F}_f, \mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{A}_f), \sigma), \mathbf{F}_f = (\mathbf{A}|\mathbf{A}_f)$.
3. Output the secret key $\mathsf{sk}_f = \mathbf{T}_f$.

$\mathsf{Trapdoor}(\mathsf{sk}_f, f, w, \mathbf{y})$: Take as input a secret key corresponding to access policy f, an access policy f, a keyword w and attribute vector $\mathbf{y}$. The data receiver computes the trapdoor corresponding to the access policy f, the keyword w and the attribute vector $\mathbf{y}$ as follows:

1. Compute $\mathbf{A}_f = \mathsf{Eval}_{\mathsf{pk}}(f, (\mathbf{A}_1, \ldots, \mathbf{A}_l))$.
2. Sample a random basis
$$\mathbf{T}_{f,w} \leftarrow \mathsf{RandBasis}(\mathbf{F}_{f,w}, \mathsf{ExtendRight}(\mathbf{F}_f, \mathbf{T}_f, \mathbf{B} + H(w)), \sigma)$$
where $\mathbf{F}_f = (\mathbf{A}|\mathbf{A}_f)$, $\mathbf{F}_{f,w} = (\mathbf{A}|\mathbf{A}_f|\mathbf{B} + H(w))$.
3. Sample a trapdoor matrix $\mathbf{R}_{f,w} \leftarrow \mathsf{SamplePre}(\mathbf{F}_{f,w}, \mathbf{T}_{f,w}, \mathbf{D}, \sigma)$, s.t. $\mathbf{F}_{f,w} \cdot \mathbf{R}_{f,w} = \mathbf{D}$.
4. Output the trapdoor $\mathbf{R} = (\mathbf{y}, \mathbf{r}_{f,\mathbf{y},w})$ where $\mathbf{r}_{f,\mathbf{y},w} = \mathbf{R}_{f,w}\mathbf{y}$.

$\mathsf{Enc}(\mathsf{mpk}, w, \mathbf{x}, \mathbf{z})$: Take as input a master public key mpk, a keyword w, an attribute $\mathbf{x}$ and an access policy vector $\mathbf{z}$. The data sender computes the ciphertext corresponding to the keyword w, the attribute $\mathbf{x}$ and the access policy vector $\mathbf{z}$ as follows:

1. Compute $\mathbf{H}_\mathbf{x} = (\mathbf{A}|\mathbf{A}_1 + x_1\mathbf{G}| \cdots |\mathbf{A}_l + x_l\mathbf{G}) \in \mathbb{Z}_q^{n\times m(l+1)}$ where $\mathbf{x} = (x_1, \ldots, x_l)$.
2. Sample a secret vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and sample $l+1$ matrixs $\mathbf{S}_1, \ldots, \mathbf{S}_l, \mathbf{S}_w \leftarrow \{-1, 1\}^{m\times m}$.
3. Sample four noise vector $\mathbf{e}_1 \leftarrow \mathcal{D}_{m,\sigma_1}, \mathbf{e}_2 \leftarrow \mathcal{D}_{k,\sigma_1}, \mathbf{e}_3 \leftarrow \mathcal{D}_{k,\sigma_2}, \mathbf{e}_4 \leftarrow \mathcal{D}_{k,\sigma_3}$.
4. Compute
$$\mathbf{c_H} = \mathbf{H}_\mathbf{x}^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^{m(l+1)}, \mathbf{c}_w = (\mathbf{B} + H(w))^T\mathbf{s} + \mathbf{S}_w^T\mathbf{e}_1,$$
$$\mathbf{c_D} = \mathbf{D}^T\mathbf{s} + \mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_4 + \lfloor q/K \rfloor \cdot \mathbf{z} \in \mathbb{Z}_q^k$$
where $\mathbf{e} = (\mathbf{I}_m|\mathbf{S}_1| \cdots |\mathbf{S}_l)^T\mathbf{e}_1 \in \mathbb{Z}_q^{m(l+1)}$.
5. Output the ciphertext $\mathsf{ct} = (\mathbf{c_H}, \mathbf{c_D}, \mathbf{c}_w)$.

$\mathsf{Test}(\mathsf{mpk}, \mathsf{ct}, \mathbf{R})$: Take as input a master public key mpk, a ciphertext ct and a trapdoor $\mathbf{R}$. The server tests the searching as follows:

1. If $f(\mathbf{x}) \neq 0$, the searching is abort, otherwise do as follows:
2. Parse $\mathsf{ct} = (\mathbf{c_H}, \mathbf{c_D}, \mathbf{c}_w), \mathbf{R} = (\mathbf{y}, \mathbf{r}_{f,\mathbf{y},w} = \mathbf{R}_{f,w}\mathbf{y})$ and $\mathbf{c_H} = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_l)$.
3. Compute $\mathbf{c}_f = \mathsf{Eval}_{\mathsf{ct}}(f, (x_i, \mathbf{A}_i, \mathbf{c}_i)_{i\in[l]})$.
4. Compute $\mu' = \mathbf{y}^T \cdot \mathbf{c_D} - \mathbf{r}_{f,\mathbf{y},w}^T \cdot (\mathbf{c}_0|\mathbf{c}_f|\mathbf{c}_w)$.
5. Then, the server computes $\mu = argmin_{\mu\in[0,K+1]}|\lfloor q/K \rfloor \cdot \mu - \mu'|$.
6. If $\mu = 0$ the server returns 1, otherwise it returns 0.

Correctness. We consider correctness in our scheme as follows:

$$\begin{aligned}\mu' &= \mathbf{y}^T \cdot \mathbf{c_D} - \mathbf{r}_{f,\mathbf{y},w}^T \cdot (\mathbf{c}_0|\mathbf{c}_f|\mathbf{c}_w)\\ &= \mathbf{y}^T\mathbf{D}^T\mathbf{s} + \mathbf{y}^T\mathbf{e}_2 + \mathbf{y}^T\mathbf{e}_3 + \mathbf{y}^T\mathbf{e}_4 + \lfloor q/K \rfloor \cdot \langle \mathbf{y}, \mathbf{z} \rangle\\ &- \mathbf{y}^T\mathbf{R}_{f,w}^T \cdot ((\mathbf{A}|\mathbf{A}_f|\mathbf{B} + H(w))^T\mathbf{s} + \mathbf{e}')\\ &= \lfloor q/K \rfloor \cdot \langle \mathbf{y}, \mathbf{z} \rangle + \mathbf{y}^T(\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{y}^T\mathbf{e}_4 - \mathbf{R}_{f,w}^T\mathbf{e}')\end{aligned}$$

where $\mathbf{e}' = (\mathbf{e}_1|\mathbf{e}_f|\mathbf{S}_w^T\mathbf{e}_1)$.

If $f(\mathbf{x}) = 0$ and $\langle \mathbf{y}, \mathbf{z} \rangle = 0$, the test process returns 1, otherwise returns 0. We have that $||e_2|| < \sigma_1\sqrt{k}, ||e_3|| < \sigma_2\sqrt{k}, ||e_4|| < \sigma_3\sqrt{k}, ||\mathbf{R}_{f,w}|| < \sigma\sqrt{3mk}$, so $||\mathbf{y}^T(\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_4 - \mathbf{R}_{f,w}^T\mathbf{e}')|| < Yk(\sigma_1 + \sigma_2 + \sigma_3) + 3\sigma\sigma_1 Ykm(1 + \alpha_{\mathcal{F}}(n))$.

Parameters. For the correctness and security of our proposed scheme, we give the setting of parameters as follows:

- $m \geq 6nlogq$.
- $\sigma > n \cdot \omega(\sqrt{n}) \cdot \alpha_{\mathcal{F}}(n)$.
- Since $s > s_1(\mathbf{Z}_2^T)$ and $s > s_1(\mathbf{Z}_3^T)$ we can set $s > C\sigma(2\sqrt{m} + \sqrt{k})$.
- $\sigma_2 > 2s\tau, \sigma_3 > 2s\sigma_1, \tau > \sigma_1 \sqrt{m} \cdot \alpha_{\mathcal{F}}(n) \sqrt{log(1/\delta)}/\epsilon$.
- $\alpha q > 2\sqrt{n}$.
- $q > 4ZY^2k^2(\sigma_1 + \sigma_2 + \sigma_3) + 12Z\sigma\sigma_1 Y^2k^2m(1 + \alpha_{\mathcal{F}}(n))$ where q is prime.

5. Security Analysis

In this section, we consider the security of our proposed scheme. The proof idea of **IND-sAF-CKA** is inspired by inner product functional encryption (IPFE) schemes (Agrawal et al., 2016, Abdalla, Catalano, Gay, & Ursu, 2020, Pal & Dutta, 2021), but we makes some modifications and improvements to them, which is suitable for our security requirements. Recall that, besides naming the inner product encryption scheme in (Agrawal et al., 2016) as ALS scheme, the challenger in the security proof of ALS scheme is referred to as ALS challenger.

Theorem 2 *Our proposed scheme with parameters as above is* ***IND-sAF-CKA*** *secure provided that the* $\mathsf{LWE}_{q,\alpha,n}$ *assumption holds.*

Proof. Although we can give the reduction between our scheme and LWE assumption, for succinct proof, we use oracles from the scheme in Agrawal et al. (2016) (Since the scheme in Agrawal et al. (2016) is under LWE problem and the proof idea of our scheme follows it).

Game 0: This game is identical to the real game described above.

Game 1: This game is identical to the **Game 0** except that the challenger C select a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, instead of sampling $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$. This game is inefficient since we must enumerate all short possible basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$ and use one of short basis to answer key queries and trapdoor queries. For realizing that, we use **Lemma 2** and **Theorem 1** with appropriate parameters set in **Parameters** to ensure that the lattice spanned by $\mathbf{A}$ has a short basis. By **Lemma 4**, this game is indistinguishable from the previous one.

Game 2: This game is identical to the **Game 1** except that the answer of the trapdoor queries. For the trapdoor queries, the challenger C programs the $\mathbf{D}$. First, C samples $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3 \in \mathcal{D}_{m\times k,\sigma}$ and set $\mathbf{Z} = (\mathbf{Z}_1 \| \mathbf{Z}_2 \| \mathbf{Z}_3) \in \mathbb{Z}^{3m\times k}$, $\mathbf{D}_1 = \mathbf{A}\mathbf{Z}_1$. Then, if $\mathcal{A}$ make queries for $(f^*, w, \mathbf{y})$, C computes $\mathbf{A}_{f^*} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f^*, (\mathbf{A}_1, \ldots, \mathbf{A}_l))$ and set $\mathbf{D} = \mathbf{D}_1 + (\mathbf{A}_{f^*})\mathbf{Z}_2 + (\mathbf{A})\mathbf{Z}_3 = (\mathbf{A}|\mathbf{A}_{f^*}|\mathbf{A})\mathbf{Z}$. We can apply **Lemma 7** and argue that $\mathbf{D}_1$ is close to uniformly random over $\mathbb{Z}_q^{n\times k}$ statistically, so $\mathbf{D}$ is also close to uniformly random over $\mathbb{Z}_q^{n\times k}$ statistically. C can answer the trapdoor queries with $\mathbf{r}_{f,\mathbf{y},w} = \mathbf{Z}\cdot\mathbf{y}$ where the distributions of $\mathbf{Z}$ and real $\mathbf{R}_{f,w}$ are the same $(\mathcal{D}_{3m\times k,\sigma})$ by **Lemma 6**; otherwise, if $\mathcal{A}$ make queries for $(f, w, \mathbf{y})$ and $f(\mathbf{x}^*) \neq 0$, C computes $\mathbf{A}_f, \mathbf{T}_{f,w}, \mathbf{R}_{f,w}$ and $\mathbf{R}$ the same as the real algorithm $\mathsf{Trapdoor}$.

Game 3: This game is identical to the **Game 2** except that the $\mathbf{A}_i$ for all $i \in [l]$. C selects $\mathbf{S}_i^* \leftarrow \{-1, 1\}^{m\times m}, \mathbf{S}_w^* \leftarrow \{-1, 1\}^{m\times m}$ and computes $\mathbf{A}_i = \mathbf{A}\mathbf{S}_i^* - x_i^*\mathbf{G}, \mathbf{B} = \mathbf{A}\mathbf{S}_w^*$ instead of selecting $\mathbf{A}_i, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$. For trapdoor queries and key queries, the adversary $\mathcal{A}$ first make a polynomial number of key queries on f such that $f(\mathbf{x}^*) \neq 0$. C computes $\mathbf{S}_f \leftarrow \mathsf{Eval}_{\mathsf{sim}}(f, x_i^*, \mathbf{S}_i)$, $\mathbf{A}_f = \mathsf{Eval}_{\mathsf{pk}}(f, (\mathbf{A}_1, \ldots, \mathbf{A}_l))$. After that, C can answer the key queries which is identical to the previous game. By **Lemma 3**, this game is statistically indistinguishable from the **Game 2**.

Game 4: This game is identical to the **Game 3** except that the challenger C is efficient. In this game, the short basis $\mathbf{T_A}$ is not required in key queries and trapdoor queries. Both the trapdoor queries $(f, \mathbf{y}, w)$ and key queries f should satisfy $f(\mathbf{x}^*) \neq 0$ or $f = f^*$. If $f = f^*$, the challenger uses matrix $\mathbf{Z} = (\mathbf{Z}_1 \| \mathbf{Z}_2 \| \mathbf{Z}_3)$ to answer the trapdoor as $\mathbf{Z}\cdot\mathbf{y}$ that is identical to the previous game. If $f(\mathbf{x}^*) = y \neq 0$, the challenger computes $\mathbf{S}_f = \mathsf{Eval}_{\mathsf{sim}}(f, (x_i^*, \mathbf{S}_i)_{i\in[l]}, \mathbf{A})$ such that $\mathbf{A}\mathbf{S}_f - y\mathbf{G} = \mathbf{A}_f$. Then it computes $\mathbf{T}_f \leftarrow \mathsf{RandBasis}((\mathbf{A}|\mathbf{A}_f), \mathsf{ExtendLeft}(\mathbf{A}, -y\mathbf{G}, \mathbf{T_G}, \mathbf{S}_f), \sigma)$. The challenger can return $\mathbf{T}_f$ for key queries with $f(\mathbf{x}^*) = y \neq 0$. Furthermore, the challenger computes $\mathbf{T}'_f \leftarrow \mathsf{RandBasis}\ (\mathbf{F}_{f,w}, \mathsf{ExtendRight}(\mathbf{A}|\mathbf{A}_f, \mathbf{T}_f, \mathbf{B} + H(w)), \sigma)$, where $\mathbf{F}_{f,w} = (\mathbf{A}|\mathbf{A}_f|\mathbf{B} + H(w))$. The challenger samples $\mathbf{R}_f \leftarrow \mathsf{SamplePre}(\mathbf{F}_{f,w}, \mathbf{T}'_f, \mathbf{D}, \sigma)$ and outputs trapdoor $\mathbf{R}_f \cdot \mathbf{y}$. By **Lemma 5** and **Lemma 6**, the public parameters and queries are statistically close to the previous game.

Game 5: This game is identical to the **Game 4** except that the ciphertext corresponding to keyword is replaced with the uniform one from $\mathbb{Z}_q^m$. Follow the LWE definition in (Regev, 2009), this game is indistinguishable to **Game 4** computationally.

Game 6: In this game, we use the security proof of ALS scheme described in (Agrawal et al., 2016) to get the indistinguishability of challenge ciphertext. we recall that we consider the interaction between the challenger C and the ALS challenger for succinct proof. The challenger C receives public key $(\mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$ from the ALS challenger and target $f^*, \mathbf{x}^*$ from the adversary $\mathcal{A}$. The challenger C can simulate the view of the adversary $\mathcal{A}$ as follows:

- **Setup:** $\mathbf{A} = \mathbf{A}_{\mathsf{ALS}}^T, \mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{Z}_2, \mathbf{Z}_3 \leftarrow \mathcal{D}_{m\times k,\sigma}$, $\mathbf{S}_i^* \leftarrow \{-1,1\}^{m\times m}$, $\mathbf{A}_i = \mathbf{A}\mathbf{S}_i^* - x_i^*\mathbf{G}$ for all $i \in [l]$. $\mathbf{D} = \mathbf{D}_{\mathsf{ALS}}^T + \mathbf{A}_f^*\mathbf{Z}_2 + \mathbf{A}\mathbf{Z}_3$ where $\mathbf{A}_f^* = \mathsf{Eval}_{\mathsf{pk}}(f^*, (\mathbf{A}_1, \dots, \mathbf{A}_l))$. The challenger C sends the master public key $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{A}_1, \dots, \mathbf{A}_l, \mathbf{D})$.
- **Key queries:** These key queries are the same as ones in the previous game.
- **Trapdoor queries:** The adversary $\mathcal{A}$ queries for $f, w, \mathbf{y}$. If $f = f^*$, the challenger asks decryption key $\mathsf{sk}_\mathbf{y}$ for $\mathbf{y}$ from the ALS challenger. When the challenger obtains $\mathsf{sk}_\mathbf{y}$, it returns $\mathbf{r}_{f,w,\mathbf{y}} = (\mathsf{sk}_\mathbf{y}^T \| \mathbf{Z}_2\mathbf{y} \| \mathbf{Z}_3\mathbf{y})$. Otherwise, the output is identical to the previous game.
- **Challenge:** The challenge ciphertext $\mathsf{ct}^* = (\mathbf{c}_\mathbf{H}^*, \mathbf{c}_\mathbf{D}^*, \mathbf{c}_w^*)$ as:

$$\mathbf{c}_\mathbf{H}^* = (\mathsf{ct}_1^{\mathsf{ALS}} | \mathbf{C}') = (\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_l),$$
$$\mathbf{c}_\mathbf{D}^* = \mathsf{ct}_2^{\mathsf{ALS}} + \mathbf{Z}_2^T\mathbf{c}'_{f^*} + \mathbf{Z}_3^T\mathbf{c}_0 + \mathsf{NoiseGen}(\mathbf{Z}_2^T, s) + \mathsf{NoiseGen}(\mathbf{Z}_3^T, s)$$

  where

$$\mathbf{C}' = (\mathbf{S}_1^* | \cdots | \mathbf{S}_l^*)^T \mathsf{ct}_1^{\mathsf{ALS}}, \mathsf{ct}_1^{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{s} + \mathbf{e}_1,$$
$$\mathsf{ct}_2^{\mathsf{ALS}} = \mathbf{D}_{\mathsf{ALS}}\mathbf{s} + \mathbf{e}_2 + \lfloor q/K \rfloor \cdot \mathbf{z}_\beta, \mathbf{e}_1 \leftarrow \mathcal{D}_{m,\sigma_1}, \mathbf{e}_2 \leftarrow \mathcal{D}_{k,\sigma_1}.$$

  Then the challenger computes $\mathbf{c}'_{f^*} = \mathbf{c}_{f^*} + \mathbf{e}'_3$, where $\mathbf{e}'_3 \leftarrow \mathcal{D}_{m,\tau}, \mathbf{c}_{f^*} = \mathsf{Eval}_{\mathsf{ct}}(f^*, (x_i^*, \mathbf{A}_i, \mathbf{c}_i)_{i\in[l]}) = \mathbf{A}_{f^*}^T\mathbf{s} + \mathbf{e}_{f^*}$ by features in Section *2.4*. Note that $\mathbf{c}_w^*$ corresponding to the keyword is identical to the one in **Game 5**.

It remains to show that **Game 5** is indistinguishable from **Game 6**. First, we consider the master public key and secret key for queries. In the ALS scheme, $\mathbf{D}_{\mathsf{ALS}} = \mathbf{Z}_{\mathsf{ALS}}\mathbf{A}_{\mathsf{ALS}}$. The matrix $\mathbf{D} = \mathbf{D}_{\mathsf{ALS}}^T + \mathbf{A}_{f^*}\mathbf{Z}_2 + (\mathbf{A})\mathbf{Z}_3 = (\mathbf{Z}_{\mathsf{ALS}}\mathbf{A}_{\mathsf{ALS}})^T + \mathbf{A}_{f^*}\mathbf{Z}_2 + (\mathbf{A})\mathbf{Z}_3 = (\mathbf{A}|\mathbf{A}_{f^*}|\mathbf{A})\mathbf{Z}$ where $\mathbf{Z} = (\mathbf{Z}_{\mathsf{ALS}} \| \mathbf{Z}_2 \| \mathbf{Z}_3)$. So we have $\mathbf{r}_{f,w,\mathbf{y}} = (\mathsf{sk}_\mathbf{y}^T \| \mathbf{Z}_2\mathbf{y} \| \mathbf{Z}_3\mathbf{y}) = (\mathbf{Z}_{\mathsf{ALS}}^T\mathbf{y} \, \| \mathbf{Z}_2\mathbf{y} \| \mathbf{Z}_3\mathbf{y}) = \mathbf{Z}\mathbf{y}$.

Then we notice that we can rewrite the ciphertext as follows:

$$\begin{aligned}\mathbf{c}_\mathbf{H}^* &= (\mathsf{ct}_1^{\mathsf{ALS}} | \mathbf{C}') = (\mathsf{ct}_1^{\mathsf{ALS}} | (\mathbf{S}_1^* | \cdots | \mathbf{S}_l^*)^T \mathsf{ct}_1^{\mathsf{ALS}}) \\ &= (\mathbf{A}_{\mathsf{ALS}}\mathbf{s} + \mathbf{e}_1 | (\mathbf{S}_1^* | \cdots | \mathbf{S}_l^*)^T (\mathbf{A}_{\mathsf{ALS}}\mathbf{s} + \mathbf{e}_1)) \\ &= (\mathbf{A}^T\mathbf{s} + \mathbf{e}_1 | (\mathbf{S}_1^* | \cdots | \mathbf{S}_l^*)^T (\mathbf{A}^T\mathbf{s} + \mathbf{e}_1)) \\ &= \mathbf{H}_x^T\mathbf{s} + \mathbf{e}\end{aligned}$$

where $\mathbf{e} = (\mathbf{I}_m | \mathbf{S}_1^* | \cdots | \mathbf{S}_l^*)\mathbf{e}_1$.

Also, we consider the $\mathbf{c}_\mathbf{D}^*$ as follows:

$$\begin{aligned}\mathbf{c}_\mathbf{D}^* &= \mathsf{ct}_2^{\mathsf{ALS}} + \mathbf{Z}_2^T\mathbf{c}'_{f^*} + \mathbf{Z}_3^T\mathbf{c}_0 + \mathsf{NoiseGen}(\mathbf{Z}_2^T, s) + \mathsf{NoiseGen}(\mathbf{Z}_3^T, s) \\ &= (\mathbf{D} - \mathbf{A}_{f^*}\mathbf{Z}_2 - \mathbf{A}\mathbf{Z}_3)^T\mathbf{s} + \mathbf{e}_2 + \lfloor q/K \rfloor \cdot \mathbf{z}_\beta + \mathbf{Z}_2^T(\mathbf{A}_{f^*}^T\mathbf{s} + \mathbf{e}_{f^*} + \mathbf{e}'_3) \\ &+ \mathbf{Z}_3^T(\mathbf{A}^T\mathbf{s} + \mathbf{e}_1) + \mathsf{NoiseGen}(\mathbf{Z}_2^T, s) + \mathsf{NoiseGen}(\mathbf{Z}_3^T, s) \\ &= \mathbf{D}^T\mathbf{s} + \mathbf{e}_2 + \mathbf{Z}_2^T(\mathbf{e}_{f^*} + \mathbf{e}'_3) + \mathsf{NoiseGen}(\mathbf{Z}_2^T, s) + \mathbf{Z}_3^T\mathbf{e}_1 + \mathsf{NoiseGen}(\mathbf{Z}_3^T, s) \\ &+ \lfloor q/K \rfloor \cdot \mathbf{z}_\beta.\end{aligned}$$

We have $\|\mathbf{e}_{f^*}\| < \sigma_1\sqrt{m} \cdot \alpha_\mathcal{F}(n)$ by the feature in Section *2.4* and $s_1(\mathbf{Z}_2^T), s_1(\mathbf{Z}_3^T) \le C\sigma(2\sqrt{m} + \sqrt{k})$ by **Lemma 1**.

Note that, by **Corollary 1** and **Lemma 8**, the noise $\mathbf{Z}_2^T(\mathbf{e}_{f^*} + \mathbf{e}'_3) + \mathsf{NoiseGen}(\mathbf{Z}_2^T, s)$ and $\mathbf{Z}_3^T\mathbf{e}_1 + \mathsf{NoiseGen}(\mathbf{Z}_3^T, s)$ in $\mathbf{c}_\mathbf{D}^*$ is statistically close to $\mathbf{e}_3$ and $\mathbf{e}_4$ respectively (in real scheme), so we have $\mathbf{c}_\mathbf{D}^* = \mathbf{D}^T\mathbf{s} + \mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_4 + \lfloor q/K \rfloor \cdot \mathbf{z}_\beta$. Finally, all ciphertexts are indistinguishable between **Game 5** and **Game 6**. From what has been discussed above, we may safely draw a conclusion that, the first game **Game 0** and the last game **Game 6** are indistinguishable computationally, so the theorem is proved.

6. Performance

In this section, we present the performance of our scheme in terms of computation and storage.

6.1 Theoretical Performance

The costs of computation and storage listed in Table 2 and Table 3 are devided into five algorithms among Setup, KeyGen, Enc, Trapdoor and Test. For precise evaluation, let $|\mathbb{Z}|$, $|\mathbb{G}|$ and $|\mathbb{G}_T|$ represent the size of one element in $\mathbb{Z}_q$, $\mathbb{G}$ and $\mathbb{G}_T$. Let T_{TrapGen}, $T_{\mathsf{SamplePre}}$, $T_{\mathsf{RandBasis}}$, $T_{\mathsf{ExtendRight}}$, $T_{\mathsf{SampleLeft}}$, T_{SampleD} denote the running time of algorithms TrapGen, SamplePre, RandBasis, ExtendRight, SampleLeft and SampleD, respectively. Let $T_{\mathsf{NewBasisDel}}$ denotes the running time of algorithm NewBasisDel from Zhang et al. (2019) Let T_H denote the running time of hash function H. Let T_{expmod} and T_{pairing} denote the running time of modular exponentiation and bilinear pairing operation in group $\mathbb{G}$ respectively. Let $T_{\mathsf{exppairing}}$ denotes the exponentiation computation in prime-order group $\mathbb{G}_T$. Let T_{mul} denote the computation cost of multiplication operation in $\mathbb{Z}_q$. Let κ denotes the length of keyword. Let λ denotes the security parameter. Let T_{pkf} and T_{ctf} denote the computational cost of $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$ respectively. Additionally, we consider the following scalar in (He et al., 2018): S is the number of user's attributes; W is the number of keywords in the ciphertext; N is the number of attributes in the access structure of the ciphertext, M is the number of keywords in the access structure of the search token.

Table 2. Computation cost among ours and recent PEKS schemes

Schemes	Setup	KeyGen	Enc	Trapdoor	Test
Behnia et al., 2018	-	T_{TrapGen}	$(mn\kappa + m^2\lambda\kappa + m^2\lambda + n\lambda + 2nm\lambda)T_{\mathsf{mul}}$	$T_{\mathsf{SampleLeft}}$	$2m\lambda T_{\mathsf{mul}}$
Zhang et al., 2019	T_{TrapGen}	$T_{\mathsf{NewBasisDel}} + T_H + T_{mul}$	$(nm^2 + nm\lambda + n\lambda)T_{\mathsf{mul}} + T_{\mathsf{SamplePre}}$	$T_{\mathsf{NewBasisDel}} + T_{\mathsf{SamplePre}}$	$(nm + m\lambda)T_{\mathsf{mul}}$
Xu et al., 2019	T_{TrapGen}	$T_{\mathsf{RandBasis}} + T_{\mathsf{ExtendRight}}$	$O(n^3)$	$T_{\mathsf{SampleLeft}}$	$2mT_{\mathsf{mul}}$
Li et al., 2019	$2T_{\mathsf{TrapGen}}$	$T_{\mathsf{pkf}} + T_{\mathsf{RandBasis}} + T_{\mathsf{ExtendRight}}$	$(nk + 3mnk)T_{\mathsf{mul}} + T_H + T_{\mathsf{SamplePre}}$	$T_{\mathsf{pkf}} + T_{\mathsf{SampleLeft}}$	$T_{\mathsf{ctf}} + 3msT_{\mathsf{mul}} + T_H$
He et al., 2018	$3T_{\mathsf{expmod}}$	$(2 + 2S)T_{\mathsf{expmod}}$	$(1 + 2N + 3W)T_{\mathsf{expmod}}$	$(1 + 2S + 3M)T_{\mathsf{expmod}}$	$(1 + 2N + 3M)T_{\mathsf{pairing}} + (N + M)T_{\mathsf{pairing}}$
Ours	T_{TrapGen}	$T_{\mathsf{pkf}} + T_{\mathsf{RandBasis}} + T_{\mathsf{ExtendRight}}$	$(nm(l + 1) + nm + mm + nk)T_{\mathsf{mul}}$	$T_{\mathsf{pkf}} + T_{\mathsf{SampleLeft}} + 3mkT_{\mathsf{mul}}$	$T_{\mathsf{ctf}} + 4mkT_{\mathsf{mul}}$

Table 3. Space overhead among ours and recent PEKS schemes

Schemes	Setup paramater	Private key	Ciphertext	Trapdoor
Behnia et al., 2018	$((\kappa + 2)nm + n)\lvert\mathbb{Z}\rvert$	$m^2\lvert\mathbb{Z}\rvert$	$(2m + 2)\lambda\lvert\mathbb{Z}\rvert$	$2m\lvert\mathbb{Z}\rvert$
Zhang et al., 2019	$(2nm + n)\lvert\mathbb{Z}\rvert$	$m^2\lvert\mathbb{Z}\rvert$	$(\lambda + m\lambda + m)\lvert\mathbb{Z}\rvert$	$m\lvert\mathbb{Z}\rvert$
Xu et al., 2019	$(3nm + n)\lvert\mathbb{Z}\rvert$	$4m^2\lvert\mathbb{Z}\rvert$	$(4m + 1)\lvert\mathbb{Z}\rvert$	$4m\lvert\mathbb{Z}\rvert$
Li et al., 2019	$((l + 3)mn + n)\lvert\mathbb{Z}\rvert$	$4m^2\lvert\mathbb{Z}\rvert$	$(k + (l + 2)mk + m)\lvert\mathbb{Z}\rvert$	$3m\lvert\mathbb{Z}\rvert$
He et al., 2018	$5\lvert\mathbb{G}\rvert$	$(2S + 1)\lvert\mathbb{G}\rvert$	$(1 + 2N + 2W)\lvert\mathbb{G}\rvert$	$(1 + 2S + 2M)\lvert\mathbb{G}\rvert$
Ours	$((l + 3)mn + n)\lvert\mathbb{Z}\rvert$	$4m^2\lvert\mathbb{Z}\rvert$	$(m(l + 1) + k + m)\lvert\mathbb{Z}\rvert$	$(3m + k)\lvert\mathbb{Z}\rvert$

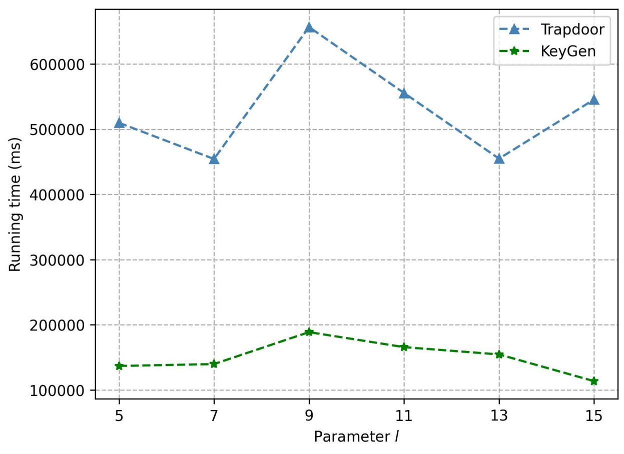
Table 2 presents the comparison of computational cost. We use similar techniques to Li et al. (2019), so the approximate computational cost can be observed obviously. However, we achieve more complex access control which is dual fine-grained including functional access control and matching on vectors. As the techniques mentioned, our scheme considers the computational costs of evaluation function $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$. If we adopt the algorithms in Boneh et al. (2014), the computational costs of $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$ are lnm^2T_{mul} and $lnmT_{mul}$ with addition function, respectively. Correspondingly, the computational costs of $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$ are nm^2T_{mul} and l^2nmT_{mul} with multiplication function, respectively.

Table 3 presents the comparison of space overhead. The analysis is analogous to the computational cost. Notably, our scheme enjoying dual access control makes the ciphertext and trapdoor, with two pairs access control, tighter respectively.

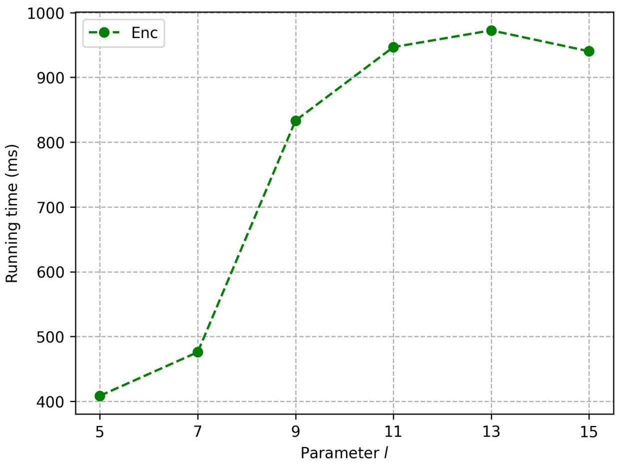
Specifically, according to Table 1, Table 2 and Table 3, the computation cost and space overhead of the ciphertext and trapdoor in our scheme is nearly half of the ones in Li et al. (2019), if we directly adopt the scheme (single access control) in Li et al. (2019) to realize dual access control.

6.2 Experimental Evaluation

Our implementations are built on a Linux system with i5-10210U CPU and 16GB DDR RAM. We exploit C++ language and NTL library version 11.5.1. For the correctness and security of our scheme, we set the parameters satisfying the parameters setting based on both dimensions $n = 10$ and $m = 600$. We fix the parameters except the attribute length l which is an independent variable. We show the comparison of space overhead and running time among our and recent PEKS schemes. We evaluate the experimental result, 10 trials on average displaying the running time of the five algorithms, as attribute length l grows.



(a) Running time of Trapdoor and KeyGen



(b) Running time of Enc

Figure 1. Running time of Enc, KeyGen and Trapdoor

Unlike Wang, Chen, Xiang, & Wang (2022), in our implementation, we leverage the second version of algorithm TrapGen instead of the first version in Alwen and Peikert (2011). The second version of algorithm TrapGen is better than the first version in terms of the running time and the space overhead.

Table 4. The running time of our scheme when l grows

Parameter l	5	7	9	11	13	15
Setup(ms)	1593.35	1816.65	1795.52	1781.57	1773.15	1010.01
KeyGen(ms)	136848.49	139735.13	188783.76	165725.72	154534.44	113543.55
Enc(ms)	408.50	475.86	833.31	946.73	972.56	940.29
Trapdoor(ms)	509938.09	454417.37	656917.52	555700.06	454954.93	545722.78
Test(ms)	152.48	159.00	333.21	263.97	287.20	286.90

Table 5. The space overheads of our scheme when l grows

Parameter l	5	7	9	11	13	15
Setup(bytes)	144	176	208	240	272	304
KeyGen(bytes)	16	16	16	16	16	16
Enc(bytes)	24	24	24	24	24	24
Trapdoor(bytes)	24	24	24	24	24	24

It is clear as Table 4 shows that the algorithm Trapdoor is time-consuming. The substantial reason is the generation of the lattice basis. Theoretically, the running time of KeyGen and Trapdoor should be close since they are almost identical except for the dimensions. Furthermore, considering the reality, we pull away from the algorithm $\mathsf{Eval}_{\mathsf{pk}}$ from KeyGen and reuse it in Trapdoor. We implement the algorithm $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$ based on Boneh et al. (2014) and Dai et al. (2017), using evaluation of public keys and ciphertexts on gates. However, the running time of Trapdoor is more than KeyGen, which is because of the dimension of matrices. Note that, not the same as the implementations based on ring-LWE in Dai et al. (2017), we have greatly improved the implementations of $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$ by the optimization on the standard lattice.

As shown in Figure 1.a, the running times of Trapdoor and KeyGen in our scheme are independent of the increase of l. This is caused mainly by two reasons: From the previous analysis, the stability of Trapdoor is mainly dependent on the generation of the lattice basis and the reuse of $\mathsf{Eval}_{\mathsf{pk}}$. Moreover and similarly, the stability of KeyGen is mainly dependent on the algorithm $\mathsf{Eval}_{\mathsf{pk}}$ which is high complexity algorithm and is independent of the increase of l. Subsequently, we give Figure 1.b to show the running time of Enc increases as the paramater l grows clearly in the result of theoretical performance.

As shown in Table 5, when l grows, the space overheads of our scheme are embodied in setup parameters and ciphertexts from algorithms Setup and Enc respectively. The algorithm Enc is related with l, but the size of ciphertexts stems from three matrices of the NTL library. As a result, the scope of the three matrices is not beyond 24 bytes in our implementations and experiments.

7. Extensions and Discussions

We consider the extensions and discussions for functionality. In this section, we give the sketches about how to build additional schemes for conjunctive keywords and delegation based on our proposed scheme in Section *4*.

Conjunctive Keywords. A critical part applied to this extension is secret sharing. We present the secret sharing $\sum_{i=1}^{l} \mathbf{D}_i = \mathbf{D}$ at the phase of $\mathsf{Trapdoor}$. In particular, instead of the variable $\mathbf{D}$, the data receiver invoke $\mathsf{SamplePre}$ with $\mathbf{D}_i$ to get $\mathbf{R}_{f,w_i}$ for all $i \in [l]$ respectively, where $\sum_{i=1}^{l} \mathbf{D}_i = \mathbf{D}$. Then, in Enc, the data sender computes ciphertexts $\mathbf{c}_{w_i}$ corresponding to keyword w_i for all $i \in [l]$ respectively. Finally, in Test, the server computes the $\mu' = \mathbf{y}^T \cdot \mathbf{c}_\mathbf{D} - \sum_{i=1}^{l}(\mathbf{r}_{f,\mathbf{y},w_i}^T \cdot (\mathbf{c}_0|\mathbf{c}_f|\mathbf{c}_{w_i}))$. We omit the formal proof since the modification in this extension does not impact the security and the proof only has a slight change without subtle techniques.

Delegation. For discussing the delegation, we add another entity: proxy, a data receiver who is delegated by an original data receiver so that the proxy cooperates with the original data receiver for searching. A proxy receives the key or trapdoor from an original data receiver and sends his/her trapdoor to the server for searching. The inputs of $\mathsf{Delegate}$ are master public key mpk, private key sk_f corresponding to access policy f and access policy g. The delegation process needs the master public key mpk and is initiated by the original data receiver with secret key sk_f. On the one hand, the original data receiver corresponding to access policy f could extend his/her key to the other data receiver who has access policy g. On the other hand, the original data receiver corresponding to access policy f could delegate his/her trapdoor to the other data receiver who has access policy g without leaking his/her keyword information.

For designing the delegation, we divide it into two solutions. If the data receiver U_f has an access policy f that match the attribute $\mathbf{x}$ and he/she intends to delegate his/her search permission to another data receiver U_g with the access policy g, the delegation adopts the key delegation from Boneh et al. (2014) for the transformation from sk_f to sk_g. On the other hand, U_f could delegate his/her trapdoor to another data receiver U_g to search keywords. It is similar to proxy re-encryption, the difference is that it is interactive and acts on the trapdoor. Besides, the security requirements are different. When U_g search successfully with $g(\mathbf{x}) = 0$, U_g returns the result (encrypted data file) to U_f. In this case, we recall that we change the trapdoor directly. We first give a naive idea: First, U_f sends the $\mathbf{I}_w = \mathbf{B} + H(w)$ to U_g. U_g computes $\mathbf{A}_g = \mathsf{Eval}_{\mathsf{pk}}(g, \mathbf{A}_1, \dots, \mathbf{A}_l)$, $\mathbf{T}_{g,w} \leftarrow \mathsf{RandBasis}(\mathbf{F}_w, \mathsf{ExtendRight}(\mathbf{A}|\mathbf{A}_g, \mathbf{T}_g, \mathbf{I}_w), \sigma), \mathbf{F}_{g,w} = (\mathbf{A}|\mathbf{A}_g|\mathbf{I}_w)$. But this method may leakage the keyword information. Luckily, we propose a solution to avoid exposing $H(w)$: in high level, U_f can generate a transformation key $\mathbf{V}_{g,w}$ for transferring the trapdoor corresponding to f to the trapdoor corresponding to g. After receiving $\mathbf{V}_{g,w}$, U_g can integrate his/her trapdoor without keyword with $\mathbf{V}_{g,w}$ in order to get the new trapdoor.

The algorithm description as follows, which could be proven based on ISIS assumption and the proof is omitted for the sake of brevity.

$\mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_f, g)$: The delegation process needs the master public key mpk and is initiated by the original data receiver with secret key sk_f.

- Key Delegation: The original data receiver U_f extend his/her key to another receiver U_g who has an access policy g.
    1. The original data receiver U_f computes $\mathbf{A}_g = \mathsf{Eval}_{\mathsf{pk}}(g, \mathbf{A}_1, \dots, \mathbf{A}_l)$, $\mathbf{T}_g \leftarrow \mathsf{RandBasis}(\mathbf{F}, \mathsf{ExtendRight}(\mathbf{A}|\mathbf{A}_f, \mathbf{T}_f, \mathbf{A}_g), \sigma), \mathbf{F} = (\mathbf{A}|\mathbf{A}_f|\mathbf{A}_g)$.
    2. Then U_f sends $\mathsf{sk}_g = \mathbf{T}_g$ to a proxy U_g.
    3. U_g can search as the normal receiver.
- Trapdoor Delegation: The original data receiver U_f delegate his/her trapdoor to another receiver U_g who has an access policy g. A proxy U_g cooperates with the U_f as follows:
    1. U_f computes $\mathbf{A}_f = \mathsf{Eval}_{\mathsf{pk}}(f, \mathbf{A}_1, \dots, \mathbf{A}_l), \mathbf{A}_g = \mathsf{Eval}_{\mathsf{pk}}(g, \mathbf{A}_1, \dots, \mathbf{A}_l)$,
    $\mathbf{T}_{f,w} \leftarrow \mathsf{RandBasis}(\mathbf{F}_{f,w}, \mathsf{ExtendRight}(\mathbf{A}|\mathbf{A}_f, \mathbf{T}_f, \mathbf{B} + H(w))$,
    $\mathbf{F}_{f,w} = (\mathbf{A}|\mathbf{A}_f|\mathbf{B} + H(w))$, and sends $\mathbf{V}_{g,w} \leftarrow \mathsf{SamplePre}(\mathbf{F}_{f,w}, \mathbf{T}_{f,w}, \mathbf{F}_g = (\mathbf{A}|\mathbf{A}_g|\mathbf{B})), \sigma)$ to U_g.
    2. U_g computes the same as algorithm $\mathsf{Trapdoor}$ and gets $\mathbf{R}_g$ without keyword where $\mathbf{F}_g = (\mathbf{A}|\mathbf{A}_g|\mathbf{B})$ such that $\mathbf{F}_g \cdot \mathbf{R}_g = \mathbf{D}$.
    3. U_g outputs trapdoor $\mathbf{R} = (\mathbf{y}, \mathbf{r}_{g,\mathbf{y}})$ where $\mathbf{r}_{g,\mathbf{y}} = (\mathbf{V}_{g,w}\mathbf{R}_g)\mathbf{y}$ and sends it to the server.

4. When U_g search successfully with $g(\mathbf{x}) = 0$, U_g returns the result (encrypted data) to U_f.

Finally, we analyses the correctness with delegation. In key delegation, we omit the verification since it is essentially relying on the delegation algorithm in Boneh et al. (2014). In trapdoor delegation, we focus on the Test algorithm: since $\mathbf{F}_{f,w} \cdot \mathbf{V}_{g,w} = \mathbf{F}_g$ and $\mathbf{F}_g \cdot \mathbf{R}_g = \mathbf{D}$, we have $\mathbf{y}^T(\mathbf{R}_g^T \mathbf{V}_{g,w}^T) \cdot ((\mathbf{A}|\mathbf{A}_f|\mathbf{B} + H(w))^T \mathbf{s} + \mathbf{e}') = \mathbf{y}^T \mathbf{D}^T \mathbf{s} + \mathbf{y}^T \mathbf{R}_g^T \mathbf{V}_{g,w}^T \mathbf{e}'$. After that, we can get the equation similar to the one in the correctness without delegation. For correctness, we need $\|\mathbf{y}^T(\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_4 - \mathbf{R}_g^T \mathbf{V}_{g,w}^T \mathbf{e}')\| < Yk(\sigma_1 + \sigma_2 + \sigma_3) + 9\sigma^2\sigma_1 Yk^2m^2(1 + \alpha_{\mathcal{F}}(n))$.

8. Conclusions

In this paper, we proposed a lattice-based public key encryption with keyword search which is dual fine-grained and provable for chosen keyword attacks. Additionally, we implement our scheme and present the computation costs and space overheads. Furthermore, we give extensions and discussions about conjunctive keywords and delegation based on our proposed scheme. In our future work, we will focus on the optimization of our scheme which can be applied to real scenarios by reducing complex algorithms like TrapGen and SamplePre.

Acknowledgements

The first author is supported by Special Funds for the Cultivation of Guangdong College StudentsScientific and Technological Innovation. (”Climbing Program” Special Funds.) (No. pdjh2021a0050)

References

Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., ... Shi, H. (2005, August). Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Annual international cryptology conference* (pp. 205-222). Springer, Berlin, Heidelberg. https://doi.org/10.1007/s00145-007-9006-6

Abdalla, M., Catalano, D., Gay, R., & Ursu, B. (2020, December). Inner-product functional encryption with fine-grained access control. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 467-497). Springer, Cham. https://doi.org/10.1007/978-3-030-64840-4_16

Agrawal, S., Boneh, D., & Boyen, X. (2010, May). Efficient lattice (H) IBE in the standard model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 553-572). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13190-5_28

Agrawal, S., Libert, B., & Stehl, D. (2016, August). Fully secure functional encryption for inner products, from standard assumptions. In *Annual International Cryptology Conference* (pp. 333-362). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53015-3_12

Ajtai, M. (1999, July). Generating hard instances of the short basis problem. In *International Colloquium on Automata, Languages, and Programming* (pp. 1-9). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48523-6_1

Alwen, J., & Peikert, C. (2011). Generating shorter bases for hard random lattices. *Theory of Computing Systems, 48*(3), 535-553. https://doi.org/10.1007/s00224-010-9278-3

Bao, F., Deng, R. H., Ding, X., & Yang, Y. (2008, April). Private query on encrypted data in multi-user settings. In *International Conference on Information Security Practice and Experience* (pp. 71-85). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-79104-1_6

Behnia, R., Ozmen, M. O., & Yavuz, A. A. (2018). Lattice-based public key searchable encryption from experimental perspectives. *IEEE Transactions on Dependable and Secure Computing, 17*(6), 1269-1282. https://doi.org/10.1109/TDSC.2018.2867462

Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004, May). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques* (pp. 506-522). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24676-3_30

Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., ... Vinayagamurthy, D. (2014, May). Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 533-556). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-55220-5_30

Boneh, D., Kushilevitz, E., Ostrovsky, R., & Skeith, W. E. (2007, August). Public key encryption that allows PIR queries. In *Annual International Cryptology Conference* (pp. 50-67). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74143-5_4

Brakerski, Z., & Vaikuntanathan, V. (2016, August). Circuit-ABE from LWE: unbounded attributes and semi-adaptive security. In *Annual International Cryptology Conference* (pp. 363-384). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53015-3_13

Cai, J. Y. (1998). A relation of primal-dual lattices and the complexity of shortest lattice vector problem. *Theoretical Computer Science, 207*(1), 105-116. https://doi.org/10.1016/S0304-3975(98)00058-9

Dai, W., Dor?z, Y., Polyakov, Y., Rohloff, K., Sajjadpour, H., Sava?, E., & Sunar, B. (2017). Implementation and evaluation of a lattice-based key-policy ABE scheme. *IEEE Transactions on Information Forensics and Security, 13*(5), 1169-1184. https://doi.org/10.1109/TIFS.2017.2779427

Ducas, L., & Micciancio, D. (2014, August). Improved short lattice signatures in the standard model. In *Annual Cryptology Conference* (pp. 335-352). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44371-2_19

Gentry, C., Peikert, C., & Vaikuntanathan, V. (2008, May). Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing* (pp. 197-206). https://doi.org/10.1145/1374376.1374407

He, K., Guo, J., Weng, J., Weng, J., Liu, J. K., & Yi, X. (2018). Attribute-based hybrid Boolean keyword search over outsourced encrypted data. *IEEE Transactions on Dependable and Secure Computing, 17*(6), 1207-1217. https://doi.org/10.1109/TDSC.2018.2864186

Katsumata, S., & Yamada, S. (2016, December). Partitioning via non-linear polynomial functions: more compact IBEs from ideal lattices and bilinear maps. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 682-712). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53890-6_23

Li, J., Ma, M., Zhang, J., Fan, S., & Li, S. (2019, December). Attribute-based keyword search from lattices. In *International Conference on Information Security and Cryptology* (pp. 66-85). Springer, Cham. https://doi.org/10.1007/978-3-030-42921-8_4

Liu, Z. Y., Tseng, Y. F., Tso, R., & Lee, C. Y. (2021). *Notes on a lattice-based proxy-oriented identity-based encryption with keyword search.* IACR Cryptol. ePrint Arch., 2021, 7. Retrieved from https://eprint.iacr.org/2021/007

Micciancio, D., & Goldwasser, S. (2002). *Complexity of lattice problems: a cryptographic perspective* (Vol. 671). Springer Science & Business Media. https://doi.org/10.1007/978-1-4615-0897-7

Pal, T., & Dutta, R. (2021). *Attribute-Based Access Control for Inner Product Functional Encryption from LWE.* IACR Cryptol. ePrint Arch., 2021, 178. Retrieved from https://eprint.iacr.org/2021/178

Park, D. J., Kim, K., & Lee, P. J. (2004, August). Public key encryption with conjunctive field keyword search. In *International Workshop on Information Security Applications* (pp. 73-86). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-31815-6_7

Regev, O. (2006, August). Lattice-based cryptography. In *Annual International Cryptology Conference* (pp. 131-141). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11818175_8

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM), 56*(6), 1-40. https://doi.org/10.1145/1568318.1568324

Sheng, G., Wen, T., Guo, Q., & Yin, Y. (2013, May). Verifying correctness of inner product of vectors in cloud computing. In *Proceedings of the 2013 international workshop on Security in cloud computing* (pp. 61-68). https://doi.org/10.1145/2484402.2484416

Shi, J., Lai, J., Li, Y., Deng, R. H., & Weng, J. (2014, September). Authorized keyword search on encrypted data. In *European symposium on research in computer security* (pp. 419-435). Springer, Cham. https://doi.org/10.1007/978-3-319-11203-9_24

Wang, C., Li, W., Li, Y., & Xu, X. (2013, November). A ciphertext-policy attribute-based encryption scheme supporting keyword search function. In *International Symposium on Cyberspace Safety and Security* (pp. 377-386). Springer, Cham. https://doi.org/10.1007/978-3-319-03584-0_28

Wang, P., Chen, B., Xiang, T., & Wang, Z. (2022). Lattice-based public key searchable encryption with fine-grained access control for edge computing. *Future Generation Computer Systems, 127*, 373-383. https://doi.org/10.1016/j.future.2021.09.012

Xu, L., Yuan, X., Steinfeld, R., Wang, C., & Xu, C. (2019, July). Multi-writer searchable encryption: An LWE-based realization and implementation. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security* (pp. 122-133). https://doi.org/10.1145/3321705.3329814

Zhang, X., Tang, Y., Wang, H., Xu, C., Miao, Y., & Cheng, H. (2019). Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage. *Information Sciences, 494*, 193-207. https://doi.org/10.1016/j.ins.2019.04.051

Zheng, Q., Xu, S., & Ateniese, G. (2014, April). VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In *IEEE INFOCOM 2014-IEEE conference on computer communications* (pp. 522-530). IEEE. https://doi.org/10.1109/INFOCOM.2014.6847976