Multi-Writer Searchable Encryption: An LWE-based Realization and Implementation

Lei Xu[†§], Xingliang Yuan[‡], Ron Steinfeld[‡], Cong Wang[§], and Chungen Xu[†]

[†] School of Science, Nanjing University of Science and Technology, China
[‡] Faculty of Information Technology, Monash University, Australia
[§] Department of Computer Science, City University of Hong Kong, China
xuleicrypto@gmail.com,{xingliang.yuan,ron.steinfeld}@monash.edu,congwang@cityu.edu.hk,xuchung@njust.edu.cn

ABSTRACT

Multi-Writer Searchable Encryption, also known as public-key encryption with keyword search(PEKS), serves a wide spectrum of data sharing applications. It allows users to search over encrypted data encrypted via different keys. However, most of the existing PEKS schemes are built on classic security assumptions, which are proven to be untenable to overcome the threats of quantum computers. To address the above problem, in this paper, we propose a lattice-based searchable encryption scheme from the learning with errors (LWE) hardness assumption. Specifically, we observe that the keys of each user in a basic scheme are composed of large-sized matrices and basis of the lattice. To reduce the complexity of key management, our scheme is designed to enable users to directly use their identity for data encryption. We present several optimization techniques for implementation to make our design nearly practical. For completeness, we conduct rigorous security, complexity, and parameter analysis on our scheme, and perform comprehensive evaluations at a commodity machine. With a scenario of 100 users, the cost of key generation for each user is 125s, and the cost of searching a document with 1000 keywords is 13.4ms.

CCS CONCEPTS

• **Security and privacy → Management and querying of encrypted data**; **Privacy-preserving protocols**;

KEYWORDS

Multi-writer searchable encryption; cloud computing security; practical post-quantum security

ACM Reference Format:
Lei Xu[†§], Xingliang Yuan[‡], Ron Steinfeld[‡], Cong Wang[§], and Chungen Xu[†]. 2019. Multi-Writer Searchable Encryption: An LWE-based Realization and Implementation. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3321705.3329814

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329814

1 INTRODUCTION

Searchable encryption (SE) [7, 12] is a promising cryptographic scheme to mitigate massive data breaches. Using SE, a server is enabled to search directly on encrypted data without decryption. The work on SE can be classified into two categories, i.e., searchable symmetric encryption (SSE) [12, 19, 35] and public key encryption with keyword search (PEKS) [7]. Compared to SSE, PEKS is more suitable for data sharing and exchanging applications like cloud drive, messaging, and email systems, where user data needs to be encrypted via their own different keys. In particular, PEKS allows a user (writer) to use the public key of another user (reader) to encrypt the data, which later can be searched by the reader.

Existing research on PEKS primarily focuses on how to enrich query functionality and realize data sharing, e.g., supporting boolean, conjunctive, subset, rank [8, 14, 32] and multi-user queries [30]. However, the security of these schemes relies on classic assumptions, i.e., discrete logarithms or large integer factoring, which have been proven to be unable to withstand the attack of quantum computers [26, 33]. To address this potential threat, lattice-based cryptography [2, 23, 25, 28, 29] has advanced rapidly in recent years.

In this paper, we aim to design a lattice-based PEKS scheme to resist the threat of quantum computers. One possible approach is to follow the methodology of traditional PEKS [1, 7], that is, treating identities as keywords and leveraging anonymous identity-based encryption (IBE) for equality testing whilst protecting keyword privacy. However, directly adapting the above approach will introduce high complexity in key management. By treating keywords as identities, the system parameters and the master private key of the IBE scheme are considered as the user's public key and private key in the PEKS scheme, respectively. In the context of the lattice-based construction, if a lattice-based anonymous IBE scheme like [2, 11] is used, the public and private keys of each user will be composed of a number of large-sized matrices and the basis of the lattice. With such an approach, it takes over 20 minutes to generate a user's key pair, as shown in our experiments later.

To address the above issue, we propose to further leverage the philosophy of IBE, and devise a new lattice-based PEKS scheme based on LWE. It allows the writer to use the reader's identity tag to encrypt keywords of documents directly. To realize this goal, we modify the anonymous IBE schemes [2, 11] to allow the users' private keys in PEKS to be derived from the master key of IBE. Specifically, we adapt coding techniques to map the user's identity into a matrix. After that, we use a lattice basis expansion

technique [11] to compute a basis from the master basis and the identity matrix as the user's private key.

How to implement an efficient LWE-based PEKS scheme is also not clear. The first challenge when implementing our scheme is to find a full-rank basis of the lattice in an efficient manner, which is the core operation in the key derivation algorithm. In the literature, Alwen et al.'s work [4] introduces an approach to generate a short basis associated with a random matrix. We improve their work and introduce a customized gaussian elimination algorithm to fast solve linear equations over finite fields. Second, we show how to extend the basis of a lattice to any arbitrary dimension as the user's private key, and propose an optimization to generate multiple user keys in a batch manner. Our contributions are summarized as follows:

- We propose an LWE-based PEKS scheme, which allows users to encrypt documents via their identity tag. This feature highly reduces the complexity of key management in applications of multi-writer searchable encryption.
- We perform detailed parameter analysis to ensure the correctness of the proposed scheme. Also, we present formal security definitions of the proposed scheme and conduct a rigorous security analysis.
- We implement our scheme and propose several optimization techniques to improve the performance. In a moderate security parameter setting ($n = 256$), our scheme takes 0.46 s to encrypt one keyword, 6.12 s to generate a search token, and 13.4 ms to search a document with 1000 keywords. For key derivation, the cost for a user is 226 s without optimization. Through batch processing, the cost can be reduced to 125 s per user for the scenario of 100 users.

Organization: The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 overviews the definition of searchable encryption and our proposed scheme. Section 4 lists related background knowledge on lattice and its security assumption. Section 5 presents our scheme, and Section 6 gives the security theorem of our proposed scheme under the LWE assumption and performs a formal security analysis. Implementation and optimization is introduced in Section 7, and evaluations are reported in Section 8. We conclude the paper in Section 9.

2 RELATED WORK

The first study on PEKS is proposed by Boneh et al. [7]. Afterwards, many researchers are committed to this topic. For example, some work is designed to enrich the functionality of the existing PEKS schemes, e.g., supporting conjunctive keyword search [14, 37] and range search [8, 32]. However, existing PEKS schemes are built under traditional Turing computing assumptions and do not consider the potential threats of quantum computing. We know that in the existence of quantum computers, most of the cryptographic schemes and protocols based on number theory and bilinear maps are no longer secure. It has been proven that the difficult problems such as big integer decomposition and discrete logarithm are efficiently solvable in quantum computing.

To deal with the attacks from quantum computers, the lattice becomes a prevailing consensus. Many problems on lattices have been proven to be NP hard, such as the shortest vector problem (SVP) and the closest vector problem (CVP) [27]. Schemes constructed from the lattice usually bring the following advantages: (1) Quantum polynomial algorithm has not been found due to difficult problems on the lattice. (2) Operations on the lattice are normally linear operations, which are more efficient than RSA, number theory and coding techniques [9]. In the past decade, lattice-based constructions for signature, encryption, key agreement protocols, and private stream aggregation (to just list a few) [5, 15, 17, 22] have been proposed. However, little work focuses on lattice-based searchable encryption.

Recently, Behnia et al. [6] propose two searchable encryption schemes based on NTRU and LWE respectively. For the first scheme, its security is derived from NTRU, which is controversial due to lack of formal security reduction proof [27]. For the LWE-based scheme, the construction is built on a known technique [1] which provides a transformation to convert an anonymous identity-based encryption scheme to a basic PEKS scheme. This scheme has the following drawbacks. First, the public key of users in their scheme is a basis with size $O(nm \log q)$, where q is a prime integer, n is the security parameter and m is the dimension of the lattice. It is not efficient for the storage and management in applications with a large number of users. This situation is alleviated in our scheme, where users can use a short identity string rather than a large matrix as their public parameter to encrypt data. A detailed comparison can be found later in Section 5 and Section 8. Second, they do not implement their LWE-based scheme and only report simulated results. In this paper, we propose several optimization techniques and are the first to implement an LWE-based searchable encryption scheme.

There are also some SSE schemes [16, 34, 38] which focus on the multi-user setting. Their techniques are designed for the scenario of one writer and multiple readers.

3 PROBLEM STATEMENT

3.1 Overview

Our scheme is applicable to a scenario with three parties, i.e., users, a cloud server, and a private key generator (PKG). Each user plays a role of data writer or data reader. Data writers encrypt the data and store it to the cloud server, while data readers expect to query the data in the encrypted form. The PKG generates the system parameters and private key for each user according to her unique identity tag, e.g., email or address.

The overview of our scheme is illustrated as follows. When a user enrolls, she is required to register at the PKG with her identity tag. We assume that the PKG can be an authority able to verify the identity of that user. After registration, it will generate and issue her a private key based on her tag. When a user (writer) would like to share some sensitive documents, she encrypts the keywords orderly parsed from each document and sends them to the cloud server. Here the ciphertext is generated by the public system parameters and the reader's tag. When the reader wants to search for documents containing a certain keyword, she generates a search token from the keyword and sends it to the cloud server. Then the server performs the search algorithm that processes the token against the keyword ciphertexts of each document. If a match is found, all the keyword ciphertexts in the matched document are returned to the reader, and are decrypted to recover this document.

In our targeted application scenario, we assume that the PKG is a trusted entity and it sends the valid private key to the users in an encrypted and authenticated channel. The cloud server follows the protocols but is interested in the underlying contents of search tokens and ciphertext.

3.2 Functions and Security Definitions

We give the definition of our proposed searchable encryption scheme as follows:

Definition 3.1. Our scheme consists of five polynomial-time algorithms:

- **Setup**: takes the security parameter λ as input, and outputs the system parameters sp with the master secret key msk.
- **Derive**: takes sp, master secret key msk and an identity tag τ as input, and outputs the secret key sk_τ.
- **TokGen**: takes the user's private key sk_τ and a keyword w as input, and generates a token st_w.
- **Encrypt**: takes sp, τ and w as input, and generates the ciphertext C of w.
- **Search**: takes sp, C, and st_w as input, outputs 1 if C matches w and 0 otherwise.

Definition 3.2. Let $\mathcal{W}$ be a set of keywords and C be the corresponding ciphertext under the given identity tag τ, we say that our searchable encryption scheme from lattice, Π = (Setup, Derive, TokGen, Encrypt, Search), is correct if

$$\Pr[Search(C, TokGen(sk_\tau, w)) = w] = 1$$

where $w \in \mathcal{W}$.

Definition 3.3. Let λ be the security parameter and $\mathcal{A}$ be the adversary. The security game between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ is simulated as follows:

-**Setup**: $\mathcal{B}$ runs the Setup(λ) algorithm to generate (sp, msk). $\mathcal{A}$ is given with the parameter sp and outputs a target identity tag τ^*.

- **Phase 1**: $\mathcal{A}$ asks $\mathcal{B}$ for the private key of the identity tag and the token of keyword w for an identity of her choice.

(1) **Derive Query**. When $\mathcal{A}$ asks for the private key of a certain tag $\tau \neq \tau^*$, the challenger returns sk_τ as response.
(2) **Token Query**. When $\mathcal{A}$ asks for the token of keyword w for any tag τ, $\mathcal{B}$ responds by first running the Derive algorithm to obtain the private key sk_τ, and then running the TokGen algorithm to obtain a token st_w.

- **Challenge**: At some point, $\mathcal{A}$ sends $\mathcal{B}$ two keywords w_0, w_1 on which it wishes to be challenged. The only restriction is that neither w_0 nor w_1 has been queried for the token in Phase 1. $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and generates the searchable ciphertext C^* for w_b with the tag τ^*, then gives C^* to $\mathcal{A}$ as the challenge ciphertext.
- **Phase 2**: $\mathcal{A}$ continues to perform **Token Query** over keyword w of her choice as long as $w \neq w_0, w_1$.
- **Guess**: Eventually, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$. Such an adversary $\mathcal{A}$ is called an IND-sID-CKA (selective tag and chosen keyword) adversary. $\mathcal{A}$'s advantage in attacking the scheme is defined as the following function of the security parameter λ:

$$Adv_{\epsilon,A}(\lambda) = |\Pr[b = b'] - 1/2|.$$

Definition 3.4. We say that our searchable encryption scheme is IND-sID-CKA secure if for any polynomial time adversary $\mathcal{A}$ we have that $Adv_{\mathcal{A}}(s)$ is a negligible function.

Table 1: Notations

Notation	Description
λ	security parameter
τ	identity tag string
$\mathcal{T}$	tag space consists of all users' unique tags
w	keyword selected from the document
$\mathcal{W}$	keyword space with all keywords in DB
ST_w	search token for keyword w
$\|\|S\|\|$	Euclidean norm of the matrix S
$\mathbb{Z}_q$	prime finite fields with order q
$\lfloor x \rfloor$	the maximum integer not greater than x
$\lfloor x \rceil$	integer closest to the value x
$\lceil x \rceil$	minimum integer greater than x
x^{-1}	inverse element of x in $\mathbb{Z}_q$
$\mathbf{A}^\top$	transpose of matrix $\mathbf{A}$
$\mathbf{A}(i, :)$	the i-th row of matrix $\mathbf{A}$
$\mathbf{A}(:, j)$	the j-th column of matrix $\mathbf{A}$
$\mathbf{A}(i, j : n)$	the j-th to n-th elements from the i-th row of $\mathbf{A}$

4 PRELIMINARIES

4.1 Integer Lattice and Sampling

The construction for our searchable encryption scheme is based on lattice theory. We first review some basic knowledge of lattices. An m-dimensional full-rank integer lattice $\Lambda \subseteq Z_m$ can be defined as the set of all integer linear combinations of m linearly independent column vectors. Here, we define the following three sets. Note that the notations in this work can be referred in Table 1.

Definition 4.1. [2, 13] For a prime integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, define:

$$\Lambda_q(\mathbf{A}) := \{\boldsymbol{e} \in Z^m \ s.t. \ \exists \ \boldsymbol{s} \in \mathbb{Z}_q^n \text{ where } \mathbf{A}^\top \boldsymbol{s} = \boldsymbol{e} \bmod q\}$$
$$\Lambda_q^\perp(\mathbf{A}) := \{\boldsymbol{e} \in Z^m \ s.t. \ \mathbf{A}\boldsymbol{e} = \mathbf{0} \mod q\}$$
$$\Lambda_q^u(\mathbf{A}) := \{\boldsymbol{e} \in Z^m \ s.t. \ \mathbf{A}\boldsymbol{e} = \boldsymbol{u} \mod q\}$$

Note that if $t \in \Lambda_q^u(\mathbf{A})$, we will have $\Lambda_q^u(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + t$.

For any set $\mathbf{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k\} \subset \mathbb{R}^m$, we let $\|\mathbf{S}\|$ denote its maximum Euclidean norm of vectors in $\mathbf{S}$, and the vector $\widetilde{\mathbf{S}} = \{\widetilde{\boldsymbol{s}_1}, \ldots, \widetilde{\boldsymbol{s}_k}\} \subset \mathbb{R}^m$ denote the Gram-Schmidt orthogonalization taken in order. An arbitrary full-rank set $\mathbf{S}$ in a lattice can be converted to a basis with an equally low Gram-Schmidt norm for the same lattice.

The following Lemma 4.2 illustrates that there exists an effective algorithm that can construct a pair of matrices $(\mathbf{A}, \mathbf{S})$ in polynomial time, where $\mathbf{S}$ is a short basis for the lattice $\Lambda_q^\perp(\mathbf{A})$ and will be used to generate system parameters for our scheme.

Lemma 4.2. *[2] Let $q \geq 3$ be odd and $m := \lceil 6n \log q \rceil$. There is a probabilistic polynomial-time (PPT) algorithm* TrapGen(q, n, m) *that*

outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{S} \in \mathbb{Z}^{m\times m})$ *such that* $\mathbf{S}$ *is statistically close to a uniform matrix in* $\mathbb{Z}^{n\times m}$ *and* $\mathbf{S}$ *is a basis for* $\Lambda_q^{\perp}(\mathbf{A})$ *satisfying*

$$\left\|\widetilde{\mathbf{S}}\right\| \leq O(\sqrt{n\log q}) \text{ and } \|\mathbf{S}\| \leq O(n\log q)$$

except with a negligible probability $negl(n)$.

The algorithms and lemmas given below show how to extend a random basis to an arbitrary higher dimensional one through invoking a deterministic polynomial-time (DPT) algorithm and a probabilistic polynomial-time (PPT) algorithm respectively as proposed in [11]. It is guaranteed that the original basis cannot be derived from the extended one. In this paper, we exploit ExtBasis and RandBasis to derive the private key based on a user's tag.

Lemma 4.3 (**ExtBasis**). *[11] There is a DPT algorithm* ExtBasis *with the following properties: given an arbitrary rank n matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, *an arbitrary basis* $\mathbf{S} \in \mathbb{Z}_q^{m\times m}$ *of* $\Lambda^{\perp}(\mathbf{A})$, *and an arbitrary* $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n\times\overline{m}}$, ExtBasis$(\mathbf{S}, \mathbf{A}')$ *outputs a basis* $\mathbf{S}'$ *of* $\Lambda^{\perp}(\mathbf{A}') \subseteq \mathbb{Z}_q^{m+\overline{m}}$ *such that* $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$, *where* $\mathbf{A}' = \mathbf{A}\|\overline{\mathbf{A}}$.

Lemma 4.4 (**RandBasis**). *[11] There is a PPT algorithm RandBasis with the following properties: given a basis* $\mathbf{S}$ *of an m-dimensional integer lattice* Λ, *a parameter* $s \geq \|\widetilde{\mathbf{S}_0}\| \cdot w(\sqrt{\log n})$, RandBasis$(\mathbf{S}, s)$ *outputs a basis* $\mathbf{S}'$ *of* Λ *such that* $\|\mathbf{S}'\| \leq s \cdot \sqrt{m}$ *with overwhelming probability. Moreover, for any two bases* $\mathbf{S}_0, \mathbf{S}_1$ *of the same lattice and any* $s \geq max\{\|\widetilde{\mathbf{S}_0}\|, \|\widetilde{\mathbf{S}_1}\|\} \cdot w(\sqrt{\log n})$, *the outputs of* RandBasis$(\mathbf{S}_0, s)$ *and* RandBasis$(\mathbf{S}_1, s)$ *are within* $negl(n)$ *statistical distance.*

Algorithm: **SampleLeft**$(\mathbf{A}, \mathbf{M}_1, \mathbf{S}_A, \boldsymbol{u}, \sigma)$[10]:

Input: a rank n matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with a short basis $\mathbf{S}_A$ of $\Lambda_q^{\perp}(\mathbf{S})$, a matrix $\mathbf{M}_1$ in $\mathbb{Z}_q^{n\times m_1}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a gaussian parameter $\sigma \geq \|\widetilde{\mathbf{S}_A}\| w(\sqrt{(\log(m+m_1))})$.

Output: a random vector $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ sampled from a distribution statistically close to the distribution $D_{\Lambda_q^u(F_1),\sigma}$, where $F_1 = (\mathbf{A}|\mathbf{M}_1)$.

Algorithm: **SampleRight**$(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{S}_B, \boldsymbol{u}, \sigma)$[10]:

Input: a rank n matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times k}$, a rank n matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ with a short basis $\mathbf{S}_B$ of $\Lambda_q^{\perp}(\mathbf{A}_1)$, a matrix $\mathbf{R} \in Z_q^{k\times m}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a gaussian parameter $\sigma \geq \|\widetilde{\mathbf{S}_B}\| \cdot s_R w(\sqrt{(\log(m+m_1))})$, where $s_R = \sup_{\|x\|=1} \|\mathbf{R}x\|$.

Output: a random vector $\boldsymbol{e} \in \mathbb{Z}^{m+k}$ sampled from a distribution statistically close to distribution $D_{\Lambda_q^u(F_2),\sigma}$, where $F_2 = (\mathbf{A}|\mathbf{AR}+\mathbf{B})$.

The following lemma helps us bound the size of samples from some certain distribution.

Lemma 4.5. *[10] Let* $\mathbf{A}$ *be a random matrix in* $\mathbb{Z}^{n\times m}$ *where* $m > n$ *and* $q > 2$. $S_{\mathbf{A}}$ *is a short basis of the lattice* $\Lambda_q^{\perp}(\mathbf{A})$ *and parameter* σ *satisfies* $\sigma > \|\widetilde{S_{\mathbf{A}}}\|\omega(\sqrt{\log m})$. *Then for any* $c \in \mathbb{R}_m$ *and* $u \in \mathbb{Z}_q^n$, *we have* $Pr[x \sim \mathcal{D}_{\Lambda_q^u(\mathbf{A}),\sigma} : \|x\| > \sigma\sqrt{m}] \leq negl(n)$.

4.2 The LWE Hardness Assumption

Our security analysis of the proposed scheme is based on the LWE assumption, which is described as follows:

*Definition 4.6 (**LWE**).* [11] For an integer $q = q(n)$ and a distribution χ on $\mathbb{Z}_q$, the goal of the learning with errors problem $\mathrm{LWE}_{q,\chi}$ is to distinguish the distribution between distribution $\mathbf{A}_{s,\chi}$ (for some uniform random and a secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$) and uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Definition 4.7. [29] For an $\alpha \in (0,1)$ and a prime q, let $\overline{\Psi}_\alpha$ denote the distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil \mod q$ where X is a normal random variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$.

Lemma 4.8. *[29] Let* $q > 2\sqrt{n}/\alpha$ *be some integer and* $\alpha = \alpha(n) \in (0,1)$, *if there exists a possibly and efficient quantum algorithm for solving the* $(\mathbb{Z}_q^n, \overline{\Psi}_\alpha)$*-LWE problem, then there is an efficient quantum algorithm for approximating the hard problems GapSVP and SIVP to within* $\widetilde{O}(n/\alpha)$ *in the Euclidean norm in the worst case.*

Lemma 4.9. *[2] Consider a vector* $\boldsymbol{e} \in \mathbb{Z}^m$ *and let* $\boldsymbol{y} \xleftarrow{R} \overline{\Psi}_\alpha^m$, *the quantity* $|\boldsymbol{e}^{\mathsf{T}}\boldsymbol{y}|$ *is treated as an integer in* $\{0, q-1\}$ *which satisfies the following inequality*

$$|\boldsymbol{e}^{\mathsf{T}}\boldsymbol{y}| \leq \|\boldsymbol{e}\| q\alpha\omega(\sqrt{\log m}) + \|\boldsymbol{e}\|\sqrt{m}/2$$

except with negligible probability $negl(m)$.

5 OUR PROPOSED SCHEME

In this section, we first present our searchable encryption scheme from lattice, and then show how to derive system parameters to guarantee the correctness of the proposed scheme. After that, we perform the space and computation complexity analysis.

5.1 Our scheme

We briefly overview the methodologies used to realize our multi-writer searchable encryption scheme. To meet the security demand of resisting quantum attacks, our scheme is built under the framework of the LWE assumption. Meanwhile, considering the troublesome certificate management in a public key encryption scheme, the idea of identity-based encryption is leveraged to simplify the key management; data and messages of users are encrypted directly via their identity tags.

In our scheme, the PKG generates private keys for legal users. In essence, the key generation algorithm is a process of lattice basis expansion, which takes a short basis and a tag-matrix as input, and outputs a basis in higher dimension via the ExtBasis and Randbasis algorithms. Here, the tag-matrix is an encoded user identity matrix. In the encryption algorithm, users (writers) first convert the keyword string into a keyword matrix with reader's tag and system parameters, and then randomize it and reduce its dimension by multiplying a random vector. The final ciphertext is a vector with noise. The realization of search requires the assistance of a search token. In our scheme, the search token is a vector which is sampled through the SampleLeft algorithm with the reader's private key and keyword. Now we present the following construction in details.

Let $\mathcal{T} \in \{0,1\}^l$ and $\mathcal{W} \in \{-1,1\}^k$ be the user's tag space and keyword space, respectively, where l and k are fixed constants. Our scheme is given as follows:

5.1.1 Setup. The PKG first takes a security parameter n as input to determine the system parameters $q, m, m'', l, k, \sigma, \alpha, s$, where s is the gaussian parameter that satisfies $s = \widetilde{L}\omega(\sqrt{\log n})$ and $\widetilde{L} = O(\sqrt{n\log q})$. Then it generates and outputs the system parameters as shown in Algorithm 1. As given in Lemma 4.2, TrapGen(q, n, m) is run to generate a matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times m}$ with a short basis $\mathbf{S}_0 \in \mathbb{Z}^{m\times m}$

Algorithm 1 Setup Algorithm

Input: System parameters $q, n, m, m'', l, k, \sigma, \alpha, s$;
Output: System parameters sp and master key msk

$(\mathbf{A}_0, \mathbf{S}_0) \leftarrow \text{TrapGen}(q, n, m)$;
for $i = 1$ to l **do**
  for $b = 0$ to 1 **do**
    $\mathbf{T}_i^b \xleftarrow{\$} \mathbb{Z}_q^{n \times m''}$;
  end for
end for
for $i = 1$ to k **do**
  $\mathbf{M}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times (m+m')}$;// where $m' = l \cdot m''$
end for
$\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times (m+m')}$, $\boldsymbol{u}_0 \xleftarrow{\$} \mathbb{Z}_q^n$;
$sp \leftarrow \{\mathbf{A}_0, \{\mathbf{T}_i^0, \mathbf{T}_i^1\}_{i=1}^l, \{\mathbf{M}_i\}_{i=1}^k, \mathbf{B}, \boldsymbol{u}_0, s\}$;
$msk \leftarrow \mathbf{S}_0$;
return sp, msk

Algorithm 2 Key Derive Algorithm

Input: System parameters sp, master key msk and a tag $\tau \in \mathcal{T}$;
Output: Private key sk_τ

$F(\tau) = (\mathbf{T}_1^{\tau_1} | \ldots | \mathbf{T}_l^{\tau_l})$;
$A_\tau \leftarrow (\mathbf{A}_0 | F(\tau)) \in \mathbb{Z}_q^{n \times (m+m')}$;
$sk'_\tau \leftarrow \text{ExtBasis}(\mathbf{S}_0, \mathbf{A}_\tau)$;
$sk_\tau \leftarrow \text{RandBasis}(sk'_\tau, s) \in \mathbb{Z}_q^{(m+m') \times (m+m')}$;
return sk_τ

for $\Lambda_q^\perp(\mathbf{A}_0)$ s.t. $||\widetilde{\mathbf{S}_0}|| \leq O(\sqrt{n \log q})$. $2l$ uniformly random matrices $\mathbf{T}_1^0, \mathbf{T}_1^1, \ldots, \mathbf{T}_l^0, \mathbf{T}_l^1 \in \mathbb{Z}_q^{n \times m''}$, $k+1$ uniformly random matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$ and $\mathbf{B} \in \mathbb{Z}_q^{n \times (m+m')}$, and a uniformly random n-vector $\boldsymbol{u}_0 \in \mathbb{Z}_q^n$ are selected. After that, the system parameters $sp = \{\mathbf{A}_0, \{\mathbf{T}_i^0, \mathbf{T}_i^1\}_{i=1}^l, \ldots, \{\mathbf{M}_i\}_{i=1}^k, \mathbf{B}, \boldsymbol{u}_0, s\}$ are published and the master secret key $msk = \mathbf{S}_0$ is kept at the PKG. Note that users in our scheme share the same system parameters and thus the complexity of key generation for each user is reduced.

5.1.2 Key Derivation. In this function, the PKG computes private keys for registered users. Once the user submits her registration with her tag $\tau \in \mathcal{T}$, PKG takes its master secret key and the user's tag as input, and returns the corresponding private key sk_τ to the user. As shown in Algorithm 2, for a user with tag τ, the PKG first encodes τ by a map F from an l-bit string to a matrix in $\mathbb{Z}_q^{n \times m'}$ and constructs a corresponding tag-matrix $\mathbf{A}_\tau = (\mathbf{A}_0 | F(\tau))$, where $F(\tau) = (\mathbf{T}_1^{\tau_1} | \ldots | \mathbf{T}_l^{\tau_l})$. Then it invokes the ExtBasis algorithm in Lemma 4.3 to compute the basis sk'_τ of tag-matrix $\mathbf{A}_\tau$. Finally, it invokes a randomization algorithm RandBasis in Lemma 4.4 to randomize this basis and returns sk_τ to the user as her private key. Note that, one can also use a full-rank difference function [2] instead of F, but doing so will reduce the strength of security, i.e., from adaptive security to non-adaptive security.

5.1.3 Encryption. In this part, we show how to construct searchable ciphertext for the keyword w extracted from a certain document. The detailed steps are given in Algorithm 3. For a given keyword w and the data reader with identity tag τ, the data writer first computes the tag-matrix $\mathbf{A}_\tau = (\mathbf{A}_0 | F(\tau)$ and keyword-matrix $G(w)$, where $F_\tau = (\mathbf{T}_1^{\tau_1} | \ldots | \mathbf{T}_l^{\tau_l})$ and $G(w) = \mathbf{B} + \sum_{i=1}^k w_i \mathbf{M}_i$. Then she computes a matrix $\mathbf{R}_w = \sum_{i=k}^l w_i \mathbf{R}_i$, where $\mathbf{R}_i$ for $i = 1, \ldots, k$ are k matrices selected randomly from $\{-1, 1\}^{(m+m') \times (m+m')}$. Noise value $x \in \mathbb{Z}_q, \boldsymbol{y} \in \mathbb{Z}_q^{m+m'}$ are picked randomly from $\overline{\Psi}_\alpha$ and ${\overline{\Psi}_\alpha}^{m+m'}$. Finally, the data writer computes $z = \mathbf{R}_w^\top y \in \mathbb{Z}_q^{m+m'}$ and uploads the ciphertext tuple $C = (C_1, C_2)$ to the server, where $C_1 = \boldsymbol{u}_0^\top \boldsymbol{r} + x \in \mathbb{Z}_q$ and $C_2 = \mathbf{A}_{\tau,w}^\top \boldsymbol{r} + (\boldsymbol{y}, \boldsymbol{z}) \in \mathbb{Z}_q^{2(m+m')}$.

Algorithm 3 Encrypt Algorithm

Input: System parameters sp, tag $\tau \in \mathcal{T}$ and keyword $w \in \mathcal{W}$;
Output: Keyword Ciphertext C

$\mathbf{A}_\tau \leftarrow (\mathbf{A}_0 | \mathbf{T}_1^{\tau_1} | \ldots | \mathbf{T}_l^{\tau_l})$;
$G(w) \leftarrow \mathbf{B} + \sum_{i=1}^k w_i \mathbf{M}_i$;
$\mathbf{A}_{\tau,w} \leftarrow (\mathbf{A}_\tau | G(w)) \in \mathbb{Z}_q^{n \times 2(m+m')}$;
for $i = 1 : k$ **do**
  $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{(m+m') \times (m+m')}$;
end for
$\mathbf{R}_w \leftarrow \sum_{i=1}^k w_i \mathbf{R}_i$;
$x \xleftarrow{\overline{\Psi}_\alpha} \mathbb{Z}_q, \boldsymbol{y} \xleftarrow{\overline{\Psi}_\alpha^{m+m'}} \mathbb{Z}_q^{m+m'}, \boldsymbol{r} \leftarrow \mathbb{Z}_q^n$;
$\boldsymbol{z} \leftarrow \mathbf{R}_w^\top \boldsymbol{y} \in \mathbb{Z}_q^{m+m'}$;
$C_0 \leftarrow \boldsymbol{u}_0^\top \boldsymbol{r} + x, C_1 \leftarrow \mathbf{A}_{\tau,w}^\top \boldsymbol{r} + (\boldsymbol{y}, \boldsymbol{z})$;
return $C \leftarrow (C_0, C_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2(m+m')}$

5.1.4 Search. In this function, a user searches for the documents with a keyword that others users share to her. When a user (reader) wants to search the documents with w, she first computes a token and sends it to the server, and the server then assists her in retrieving the matched ciphertexts. The steps are shown as Algorithm 4. First, the reader encodes w to a matrix $G(w)$, where $G(w) = \sum_{i=1}^k w_i \mathbf{M}_i + \mathbf{B}$, then she takes her private key, tag-matrix $\mathbf{A}_\tau$, a vector $\boldsymbol{u}_0$, and $G(w)$ as inputs to invoke SampleLeft algorithm in Section 4. The sample result $\boldsymbol{st}_w$ satisfying $(\mathbf{A}_\tau | G(w)) \cdot \boldsymbol{st}_w = \boldsymbol{u}_0$ is sent to the server as the search token. Once the server receives the search request with token $\boldsymbol{st}_w$, she calculates the error term $\gamma \leftarrow |C_0 - \boldsymbol{st}_w^\top C_1|$ with all ciphertext $C = (C_0, C_1)$ and returns the ciphertexts that satisfy $\gamma \leq \lfloor \frac{q}{4} \rfloor$.

Remark. Our scheme leverages the identity tag to simplify the key management for the lattice-based searchable encryption scheme. Compared to a basic LWE-based searchable encryption scheme [6], our scheme has two advantages. First, the user storage overhead is reduced. The reason is that the public key in the basic scheme always consists of several matrices with size of $O(n^2)$, while the identity tag in our scheme is a fixed length string. Second, the computation cost of deriving a private key is much smaller than the cost of generating a public/private key pair for each user, which will be illustrated later in Section 8.

Note that, our design currently does not support the multi-reader setting, where a writer can encrypt a document once and it can later be searched by different readers. To enable the above setting, one possible approach is to leverage the proxy re-encryption (PRE)

Algorithm 4 Search Algorithm

User:

1: $G(w) \leftarrow \sum_{i=1}^{k} w_i \mathbf{M}_i + \mathbf{B}$
2: $\boldsymbol{st}_w \leftarrow \text{SampleLeft}(\mathbf{A}_\tau, G(w), sk_\tau, \boldsymbol{u}_0, \sigma) \in \mathbb{Z}_q^{2(m+m')}$
3: Send $\boldsymbol{st}_w$ to the server.

Server:

4: $\gamma \leftarrow |C_0 - \boldsymbol{st}_w^\top C_1|$
5: **if** $\gamma \leq \lfloor \frac{q}{4} \rfloor$ **then**
6: return 1 // the keyword ciphertext matches the token
7: **else if**
8: **then**
9: return 0
10: **end if**

technique to re-encrypt the ciphertext [30] for other readers. Lattice based PRE constructions like [18] could be adapted. We leave this as our future work.

5.2 Parameters Analysis

In this section, we prove that our scheme can work correctly under the given parameters setting, i.e. the error term calculated by matched token and keyword ciphertext is smaller than $\lfloor \frac{q}{4} \rfloor$ with overwhelming probability in the search process.

Theorem 5.1. *Let $m = \lceil 6n \log q \rceil, q \geq m^{2.5} \cdot \omega(\sqrt{\log n})$, $\sigma = km \cdot \omega(\sqrt{\log n})$, $\alpha = [k^2 m^2 \cdot \omega(\log n)]^{-1}$, $m' \leq m$, then our scheme could work correctly with overwhelming probability.*

Proof. Let $C = (C_0, C_1) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{2(m+m')}$ be the ciphertext of keyword w and $\boldsymbol{st}_w = \text{SampleLeft}(\mathbf{A}_\tau, G(w), sk_\tau, \boldsymbol{u}_0, \sigma) \in \mathbb{Z}_q^{2(m+m')}$ be the corresponding search token, where $C_0 = \boldsymbol{u}_0^\top \boldsymbol{r} + x$ and $C_1 = \mathbf{A}_{\tau,w}^\top \boldsymbol{r} + (\boldsymbol{y}, \boldsymbol{z})$. According to the step in line 5 of the Search algorithm, the proof for the correctness of our scheme is to show that the inequality $\gamma = \left|C_0 - \boldsymbol{st}_w^\top C_1\right| < \lfloor \frac{q}{4} \rfloor$ holds with overwhelming probability under the given parameters.

After parsing the token $\boldsymbol{st}_w = (\boldsymbol{st}_w^1, \boldsymbol{st}_w^2) \in \mathbb{Z}_q^{m+m'} \times \mathbb{Z}_q^{m+m'}$, because $\boldsymbol{z} = \mathbf{R}_w^\top \boldsymbol{y} \in \mathbb{Z}_q^{m+m'}$, by triangle inequality, the error term is

$$\begin{aligned}
\gamma &= \left|C_0 - \boldsymbol{st}_w^\top C_1\right| = \left|\boldsymbol{u}_0^\top \boldsymbol{r} + x - \boldsymbol{st}_w^\top (\mathbf{A}_{\tau,w}^\top \boldsymbol{r} + (\boldsymbol{y}, \boldsymbol{z}))\right| \\
&= \left|\boldsymbol{u}_0^\top \boldsymbol{r} + x - (\mathbf{A}_{\tau,w} \mathbf{st}_w)^\top \boldsymbol{r} - \mathbf{st}_w^\top (\boldsymbol{y}, \boldsymbol{z})\right| \\
&= \left|\boldsymbol{u}_0^\top \boldsymbol{r} + x - \boldsymbol{u}^\top \boldsymbol{r} - (\mathbf{st}_w^1, \mathbf{st}_w^2)^\top (\boldsymbol{y}, \mathrm{R}_w \boldsymbol{y}))\right| \\
&= \left|x - (\boldsymbol{st}_w^1 + \mathbf{R}_w \boldsymbol{st}_w^2)^\top \boldsymbol{y}\right| \\
&< |x| + \left|(\boldsymbol{st}_w^1 + \mathbf{R}_w \boldsymbol{st}_w^2)^\top \boldsymbol{y}\right|
\end{aligned} \tag{1}$$

For the matrix $\mathbf{R}_w$, as $\mathbf{R}_w = \sum_{i=1}^{k} w_i \mathbf{R}_i$, so

$$\|\mathbf{R}_w\| \leq \sum_{i=1}^{k} \|\mathbf{R}_i\| \leq k\sqrt{m+m'}. \tag{2}$$

Since $\boldsymbol{st}_w$ is sampled by SampleLeft algorithm, the inequality $\|\boldsymbol{st}_w\| < \sigma\sqrt{2(m+m')}$ holds with all but negligible probability in n. According to the Lemma 6 of [2], this leads to

$$\|(\boldsymbol{st}_{w_1} + \mathbf{R}_w \boldsymbol{st}_{w_2})^\top\| \leq O(\sigma km + \sqrt{m}) \tag{3}$$

with overwhelming probability.

Note that the first part of the results $|x|$ in equation (1) is selected from the distribution $\overline{\Psi}_\alpha$, which can be bounded by the standard gaussian tail bound theorem. Combining with equations (2) and (3), the second part $\left|(\boldsymbol{st}_{w_1} + \mathbf{R}_w \boldsymbol{st}_{w_2})^\top \boldsymbol{y}\right|$ also can be bounded by Lemma 4.9. By the standard gaussian tail bound and Lemma 4.9, we have

$$|x| < q\alpha\omega(\sqrt{\log m}) + \frac{1}{2} \tag{4}$$

and

$$\left|(\boldsymbol{st}_{w_1} + \mathbf{R}_w \boldsymbol{st}_{w_2})^\top \boldsymbol{y}\right| < \sigma(km + \sqrt{m})[q\alpha\omega(\log m) + \sqrt{m}/2] \tag{5}$$

with all but negligible probability in m.

Through the results discussed above, we know that the error term can be bounded by

$$\begin{aligned}
\gamma &\leq \sigma(km + \sqrt{m}) \cdot (\sqrt{m}/2 + q\alpha\omega(\sqrt{\log m})) + O(\sigma m^{3/2}) \\
&\leq q\sigma\alpha km\omega(\sqrt{\log m}) + O(\sigma m^{3/2}) \\
&\leq O(m^{2.5} \cdot \omega(\sqrt{\log m})) := q/5
\end{aligned}$$

with overwhelming probability. □

The above proof completes the correctness analysis. In addition, this setting also makes the inequality $q > 2\sqrt{n}/\alpha$ hold, which ensures the LWE assumption is as hard as the worst-case SIVP and GapSVP under a quantum reduction. Under this setting, we continue to discuss the computation and space cost for our scheme.

5.3 Space and Computation Complexity

In this section, we analyze the efficiency of our proposed scheme in terms of space and computation cost in each algorithm. Let $|\mathbb{Z}_q|$ be the size integer element $\mathbb{Z}_q$ and ρ be the computation complexity of sampling an integer from $\mathbb{Z}_p$ with the gaussian distribution. Let n be the security parameter and m be the dimension of lattice basis. Set $\overline{m} = m + m'$, then the comparison of our scheme with Behnia et al.'s LWE-based scheme [6] is listed in Table 2 and Table 3.

In the aspect of space complexity, our private key size of each user is $|\mathbb{Z}_q^{\overline{m} \times \overline{m}}|$ and theirs is $|\mathbb{Z}_q^{m \times m}|$. The difference is caused by the identity. However, with the identity, our scheme no longer needs to use the matrix as their public key, and only one fixed length string is sufficient. Then all the users can share the same system parameters and use them to perform the encryption and search operations with their identities. The size of ciphertext in [6] is $|\mathbb{Z}_q^{\kappa O(\log n) \times (2m+1)}|$. This is because their scheme uses a κ-bit message for encryption, and the message is encrypted bit by bit, where the complexity of ciphertext for each bit is $O(\log n)$ as indicated in [6]. We note that the above treatment is not necessary in an PEKS scheme, where the message can be a single bit '0' or '1' to indicate matching after search. Therefore, our scheme only encrypts a one-bit message during keyword encryption. The token generation of these two schemes adapts the same SampleLeft algorithm, and the only difference is the input. Note that it makes the token size different; ours is $|\mathbb{Z}_q^{2\overline{m}}|$ and theirs is $|\mathbb{Z}_q^{2m}|$.

In the aspect of computation, the complexity of encryption in our scheme is $O(n^3)$. As mentioned before, the scheme in [6] encrypts a κ-bit message and thus their computational complexity reaches $O(\kappa O(\log n) n^3)$. This also affects the efficiency of the search algorithm. The computation cost of our Search algorithm without

Table 2: Space Complexity Analysis

Scheme	Private-key Size	Ciphertext Size	Token Size
[6]	$\lvert\mathbb{Z}_q^{m\times m}\rvert$	$\lvert\mathbb{Z}_q^{\kappa O(\log n)(2m+1)}\rvert$	$\lvert\mathbb{Z}_q^{2m}\rvert$
Ours	$\lvert\mathbb{Z}_q^{\overline{m}\times\overline{m}}\rvert$	$\lvert\mathbb{Z}_q^{2\overline{m}+1}\rvert$	$\lvert\mathbb{Z}_q^{2\overline{m}}\rvert$

Table 3: Computation Complexity Analysis

Scheme	TokGen Comp	Enc. Comp	Search Comp
[6]	$O(\rho m)$	$O(\kappa O(\log n)n^3)$	$O(\kappa O(\log n)n^2)$
Ours	$O(\rho\overline{m})$	$O(n^3)$	$O(n^2)$

token generation is $O(n^2)$ which just needs one matrix-vector multiplication and one integer subtraction to encrypt one bit, while it is $O(\kappa O(\log n)n^2)$ for [6]. The computation complexity of the token generation depends on the size of the input, and the complexity of token generation in the two schemes is $O(\rho m)$ and $O(\rho\overline{m})$, respectively.

6 SECURITY ANALYSIS

The security of the scheme relies on the assumption of LWE mentioned in Definition 4.6. Following the security game in Definition 3.3, for a given adversary $\mathcal{A}$ who can perform queries defined in the security game, we show how to reduce the security of the scheme to the LWE assumption. No adversary $\mathcal{A}$ can break our scheme with non-negligible probability. Otherwise, there exists a challenger $\mathcal{B}$ that can prove that the LWE assumption does not hold.

Before the security analysis, we introduce an artificial abort function from [2].

Definition 6.1. [2] Let $H := \{h : X \to Y\}$ be a family of hash functions form X to Y where $0 \in Y$. For a set of $q_t + 1$ inputs $x = (x_0, \ldots, x_{q_t})$, the non-abort probability of x is defined as

$$p(x) = \Pr[h(x_0) = 0 \wedge h(x_1) \neq 0 \wedge \ldots h(x_{q_t}) \neq 0]$$

where the probability is over the random choice of $h \in H$. We say that H is (q_t, p_{min}, p_{max}) abort-resistant if $p(x) \in [p_{min}, p_{max}]$ for all $x = (x_0, x_1, \ldots, x_{q_t})$ with $x_0 \notin \{x_1, \ldots, x_{q_t}\}$.

Following the work in [2], we introduce the hash family $H : \{h_\theta : \mathbb{Z}_q^k/\{0^k\} \to \mathbb{Z}_q\}$ as $h_\theta(w) = 1 + \sum_{i=1}^k \theta_i w_i \in \mathbb{Z}_q$, where $w = (w_1, \ldots, w_k) \in \{-1, 1\}^k$ and $\theta = (\theta_1, \ldots, \theta_k) \in \mathbb{Z}_q^k$. It has the following properties.

Lemma 6.2. *[2] Let q be a prime and $0 \leq q_t \leq q$, then the hash family H introduced above is $(q_t, \frac{1}{q}(1 - \frac{q_t}{q}), \frac{1}{q})$ abort-resistant.*

Theorem 6.3. *Our searchable encryption scheme $(q, n, m, m', \sigma, \alpha)$ is IND-sTag-CKA secure assuming that the $(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)$-LWE assumption holds. For a probabilistic polynomial (q_e, q_t, ϵ) adversary $\mathcal{A}$, if she can win the game with advantage ϵ by making no more than q_e derive queries and q_t token queries, then challenger $\mathcal{B}$ can solve the $(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)$-LWE problem as $\mathcal{A}$ with advantage $\epsilon' \geq \epsilon/(4q)$ in the same time.*

Proof. To prove this theorem, we introduce a sequence of games and show that no (t, q_e, q_t, ϵ)-adversary $\mathcal{A}$ can distinguish these games.

Game$_0$ is the real game described in definition 3.3.

Game$_1$ is identical to **Game**$_0$ except invoking the TrapGen algorithm to obtain $(\mathbf{A}_\tau, \mathbf{S}) \in \mathbb{Z}_q^{n\times\overline{m}} \times \mathbb{Z}_q^{\overline{m}\times\overline{m}}$ and parsing $\mathbf{A}_\tau = \mathbf{A}_0\|\mathbf{A}_1\|\ldots\|\mathbf{A}_l$, where $\overline{m} = m + m'$ and $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m''}$ for $i \in \{1, \ldots, l\}$. For a selected identity tag $\tau^* \in \{0,1\}^l$, $\mathcal{B}$ sets $\mathbf{T}_i^{\tau_i^*} = \mathbf{A}_i$ and generates $\mathbf{T}_i^{1-\tau_1^*}$ and $\mathbf{S}_i$ by running TrapGen(q, n, m''). After that, $\mathcal{B}$ executes the remaining algorithms as in Game$_0$. In this case, if there exists an adversary $\mathcal{A}$ who can distinguish **Game**$_0$ and **Game**$_1$, then their exists an algorithm $\mathcal{B}$ can distinguish between $\mathbf{T}_i$ and a truly random matrix from uniform distribution. Note that, in the view of $\mathcal{A}$, $\mathbf{T}_i$ for $i = 1, \ldots, l$ defined above are statistically close to uniform distribution, and thus $\mathcal{A}$ cannot distinguish $\mathbf{T}_i$ with that is chosen randomly in $\mathbb{Z}_q^{n\times m''}$, i.e.

$$\Pr[\mathbf{Game}_0(1^\lambda) = 1] = \Pr[\mathbf{Game}_1(1^\lambda) = 1].$$

Game$_2$ is identical to Game$_1$ except that set $\mathbf{M}_i = \mathbf{A}_{\tau^*} \cdot \mathbf{R}_i^* + \theta_i \cdot \mathbf{B}$ rather than selecting it randomly from $\mathbb{Z}_q^{n\times m}$, where $\theta_i \in \mathbb{Z}_q$ and $\mathbf{R}_i^* \in \{-1, 1\}^{\overline{m}\times\overline{m}}$ are randomly selected at the initialization phase. For the challenge keyword w, make $\mathbf{R}_w = \sum_{i=1}^k w_i \mathbf{R}_i^*$ and $\boldsymbol{z} = (\mathbf{R}_w^*)^\top * \boldsymbol{y} \in \mathbb{Z}_q^{\overline{m}}$ to construct the challenge ciphertext. After that, $\mathcal{B}$ executes the rest algorithms as Game$_0$. In this case, if there exists an adversary $\mathcal{A}$ who can distinguish **Game**$_1$ and **Game**$_2$, then their exists an algorithm $\mathcal{B}$ can distinguish between $\mathbf{M}_i$ and a truly random matrix from uniform distribution. Note that, in the view of $\mathcal{A}$, $\mathbf{M}_i$ for $i = 1, \ldots, k$ defined above are statistically close to uniform distribution, and thus $\mathcal{A}$ cannot distinguish $\mathbf{M_i}$ from the matrix chosen randomly in $\mathbb{Z}_q^{n\times\overline{m}}$, i.e.

$$\Pr[\mathbf{Game}_1(1^\lambda) = 1] = \Pr[\mathbf{Game}_2(1^\lambda) = 1].$$

Game$_3$. **Game**$_3$ is the same as **Game**$_2$ except that an artificial abort event is deployed in the security game. In this case, the difference between **Game**$_3$ and **Game**$_2$ is reflected in the initialization and final guess phase. On the one hand, in the initialization phase, challenger $\mathcal{B}$ needs to choose a random hash function $h \in H$ and keeps it by herself. When she receives the private key on $\tau \neq \tau^*$ and token queries on $w_1, \ldots, w_{q_t}$ from $\mathcal{A}$, she responds to the queries and returns the challenge ciphertext as in Game$_2$. Note that, the challenge keyword here should not be in the query set. On the other hand, for a given tag τ^* and the selected keyword w^*, when the adversary $\mathcal{A}$ outputs the final guess. $\mathcal{B}$ checks if $h(w^*) = 0$ and $h(w_i) \neq 0$ for $i = 1, \ldots, q_t$. If not, $\mathcal{B}$ refreshes b' with a random bit from $\{0, 1\}$ and aborts the game. In addition, $\mathcal{B}$ selects a random bit $\beta \in \{0, 1\}$ such that $\Pr[\beta = 1] = \psi(w^*, w_1, \ldots, w_{q_t})$ where the function $\psi(\cdot)$ is defined in [36]. As all of these challenges are independent from the perspective of $\mathcal{A}$ and the results in [2], we have that

$$\Pr[\mathbf{Game}_1(1^\lambda) = 1] \leq \frac{1}{4q}\Pr[\mathbf{Game}_2(1^\lambda) = 1].$$

Game$_4$ is identical to **Game**$_3$ except the selection of $\mathbf{A}_0$ and $\mathbf{B}$. Instead, $\mathcal{B}$ randomly chooses $\mathbf{A}_0$ from a uniform distribution on $\mathbb{Z}_q^{n\times m}$, and generates $\mathbf{B}$ and the corresponding basis $\mathbf{S_B}$ via the TrapGen algorithm. The remaining parameter settings are the same as those in **Game**$_3$. In this case, $\mathcal{B}$ responds to the private key queries in the same way in **Game**$_3$. While for the token query $w \in \{-1, 1\}^k$ on a given tag τ, let $\mathbf{A}_{\tau,w} =$

$(\mathbf{A}_\tau|\mathbf{B} + \sum_{i=1}^{k} w_i\mathbf{A}_i) = (\mathbf{A}_\tau|\mathbf{A}_\tau\mathbf{R}_w + h(w)\mathbf{B})$, where $\mathbf{A}_\tau = \mathbf{A}_0|F_\tau$, $\mathbf{R}_w = \sum_{i=1}^{k} w_i * \mathbf{R}_i^*$ and $h(w) = 1 + \sum_{i=1}^{k} w_i\theta_i \in \mathbb{Z}_q$. Then set $\boldsymbol{st}_w = \text{SampleRight}(\mathbf{A}_0, h(w)\mathbf{B}, \mathbf{R}_w, \mathbf{S}_B, u, \sigma) \in \mathbb{Z}_q^{2(m+m')}$ as the token for query result. Note that, the obtained token $\boldsymbol{st}_w$ is close to the distribution $D_{\Lambda_q^u(\mathbf{A}\tau,w)}$ as in $\mathbf{Game}_3$. From the perspective of $\mathcal{A}$, as $\mathbf{Games}_3$ and $\mathbf{Game}_4$ have the same setting for private key queries, data encryption, and abort conditions, and thus she cannot distinguish them under the existing conditions, i.e.

$$\Pr[\mathbf{Game}_3(1^\lambda) = 1] = \Pr[\mathbf{Game}_4(1^\lambda) = 1].$$

$\mathbf{Game}_5$. $\mathbf{Game}_5$ is the same as $\mathbf{Game}_4$ except that $\mathcal{B}$ selects $(C_0, C_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2(m+m')}$ randomly as the challenge ciphertext. In this case, $\mathcal{A}$ has no advantage to win the game because this challenge ciphertext is always generated randomly from the ciphertext space. Here, we show how to deploy a simulator to prove that $\mathbf{Game}_4$ and $\mathbf{Game}_5$ are computationally indistinguishable to the adversary.

Simulator. Assume that a (t, q_e, q_t, ϵ)-adversary $\mathcal{A}$ for our scheme exists. From the adversary, we construct a challenger $\mathcal{B}$ that solves LWE with probability at least ϵ' and in polynomial time for a target tag τ^*.

Setup. $\mathcal{B}$ samples the entries of LWE instance $(\boldsymbol{u_i}, v_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ randomly for $i = 0, \ldots, \overline{m}$, then makes the following assignment:

(1) Assemble $\mathbf{A}_{\tau^*}$ by the last m samples above, i.e., let $\mathbf{A}_{\tau^*} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\overline{m}})$.
(2) Generate matrices $\mathbf{A}_0, \mathbf{T}_1^0, \mathbf{T}_1^1, \ldots, \mathbf{T}_l^0, \mathbf{T}_l^1$ as $\mathbf{Game}_1$. Construct the remainder of the public parameters, namely $\mathbf{M}_1, \ldots, \mathbf{M}_k$ and $\mathbf{B}$ as $\mathbf{Game}_3$.
(3) Send $sp = (\mathbf{A}_0, \mathbf{T}_1^0, \mathbf{T}_1^1, \ldots, \mathbf{T}_l^0, \mathbf{T}_l^1, \mathbf{M}_1, \ldots, \mathbf{M}_k, \mathbf{B}, \boldsymbol{u}_0)$ to $\mathcal{A}$.

Phase 1. When adversary $\mathcal{A}$ asks the private key for a tag τ. $\mathcal{B}$ answers the query as follows:

-Derive Queries. Considering q_e queries for the private key of a tag $\tau \neq \tau^*$, $\mathcal{B}$ answers the query as follows:

(1) Generate $F(\tau) = (\mathbf{T}_1^{\tau_1}|\ldots|\mathbf{T}_l^{\tau_l})$, then compute $A_\tau = (\mathbf{A}_0|F(\tau))$. Since $\tau \neq \tau^*$, without loss of generality, we assume that the i-th bit of τ^* and τ are different. As $\mathbf{T}_i^{\tau_i}$ is generated by TrapGen algorithm, $\mathcal{B}$ knows the basis $\mathbf{S}_i^{\tau_i}$ for $\Lambda_q^\perp(\mathbf{T}_i^{\tau_i})$.
(2) Respond to the private key query of τ by running $sk_\tau \leftarrow \text{RandBasis}(\text{ExtBasis}(\mathbf{S}_i^{\tau_i}, \mathbf{A}_\tau))$.
(3) Finally, send the answer sk_τ to $\mathcal{A}$.

-Token Queries. When $\mathcal{A}$ asks for the token of a keyword w under the target identity tag τ^*, If $h(w) = 0$, abort the game and output a random bit $b' \in \{0, 1\}$. Otherwise, $\mathcal{B}$ answers the query as follows:

(1) Invoke TrapGen algorithm to obtain $\mathbf{B} \in \mathbb{Z}_q^{n\times\overline{m}}$ with a trapdoor $\mathbf{S}_B$ for $\Lambda_q^\perp(\mathbf{B})$.
(2) For $1 \leq i \leq k$, choose $\mathbf{R}_i^* \in \{-1, 1\}^{\overline{m}\times\overline{m}}$ randomly, and also choose k random scalars $\theta_i \in \mathbb{Z}_q$ for $i = 1, \ldots, k$.
(3) Let $\mathbf{M}_i = \mathbf{A}_{\tau^*} \cdot \mathbf{R}_i^* + \theta_i \cdot \mathbf{B} \in Z_q^{n\times\overline{m}}$, so $A_w = (A_{\tau^*}|\sum_{i=1}^{k} w_i\mathbf{M}_i + \mathbf{B}) = (\mathbf{A}_{\tau^*}|\mathbf{A}_{\tau^*}\mathbf{R}_w + h(w)\mathbf{B})$ where $\mathbf{R}_w = \sum_{i=1}^{k} w_i\mathbf{R}_i^* \in \mathbb{Z}_q^{\overline{m}\times\overline{m}}$ and $h(w) = 1 + \sum_{i=1}^{k} \theta_i w_i$.
(4) Set $\boldsymbol{st}_w \leftarrow \text{SampleRight}(\mathbf{A}_{\tau^*}, h_w, \mathbf{R}_w, \mathbf{S}_B, u_0, \sigma)$ be the token of user τ^* on keyword w, and send it to $\mathcal{A}$.

Challenge. After the queries abort, $\mathcal{A}$ sends two keywords w_0 and w_1 to $\mathcal{B}$, $\mathcal{B}$ checks if $h(w^*) = 0$. If it is, abort the game and output a random $b' \in \{0, 1\}$. Otherwise, $\mathcal{B}$ generates the challenge ciphertext under the target identity τ^* as follows:

(1) Let
$$\boldsymbol{v}^* = \begin{bmatrix} v_1 \\ \vdots \\ v_{\overline{m}} \end{bmatrix} \in \mathbb{Z}_q^{\overline{m}}$$
and $C_0^* = v_0 \in \mathbb{Z}_q$.
(2) $\mathcal{B}$ chooses $w^* \in \{w_1, w_2\}$ randomly, and sets
$$\mathbf{R}_{w^*} = \sum_{i=1}^{k} w_i^*\mathbf{R}_i^*$$
for $1 \leq i \leq k$, then computes
$$C_1^* = \begin{bmatrix} \boldsymbol{v}^* \\ (\mathbf{R}_{w^*})^\top v^* \end{bmatrix} \in \mathbb{Z}_q^{2(m+m')}.$$
(3) Finally, $\mathcal{B}$ chooses a bit $b \in \{0, 1\}$. If $b = 0$, send the ciphertext $C^* = (C_0^*, C_1^*)$ to the adversary. Otherwise, choose a random $C = (C_0, C_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2(m+m')}$ and send $C = (C_0, C_1)$ to the adversary.

Phase 2. The simulator repeats the same steps as in Phase 1 with the restriction that $\mathcal{A}$ cannot query for challenge keywords w_1 and w_2.

Guess. After all the queries, $\mathcal{A}$ outputs a guess b' of b by executing the steps below:

Artificial abort: The challenger checks if $h(w^*) = 0$ and $h(w_i) \neq 0$ for $i = 1, \ldots, q_t$, where w^* is the target message and $w^* \notin \{w_1, \ldots, w_{q_t}\}$. If not, it returns a random bit b' from $\{0, 1\}$ and aborts the game. Otherwise, $\mathcal{B}$ outputs the guess as the answer to the LWE challenge which she tries to solve.

Note that if the LWE oracle is pseudorandom, then $\mathbf{A}_w = (\mathbf{A}_{\tau^*}|\mathbf{R}_{w^*})$ because of $h(w^*) = 0$, and for some random noise vector $\boldsymbol{y} \in \mathbb{Z}_q^{m+m'}$ distributed as $\overline{\Psi}_\alpha^{m+m'}$, we get that $\boldsymbol{v}^* = \mathbf{A}_{\tau^*}^\top s + \boldsymbol{y}$. Therefore
$$C_1^* = \begin{bmatrix} \mathbf{A}_{\tau^*}^\top s + \boldsymbol{y} \\ (\mathbf{A}_{\tau^*}\mathbf{R}_w^*)^\top s + \mathbf{R}_w^{*\top}\boldsymbol{y} \end{bmatrix} = \mathbf{A}_{\tau^*}{}^\top s + \begin{bmatrix} \boldsymbol{y} \\ (R_w^*)^\top\boldsymbol{y} \end{bmatrix}$$

Above C_1^* is a valid part of challenge ciphertext C^*. Again $C_0^* = \boldsymbol{u}_0^\top s + x$ ia also a valid part of challenge ciphertext C_0. Therefore (C_0^*, C_1^*) is a valid challenge ciphertext.

If LWE oracle is random oracle, v_0 is uniform in $\mathbb{Z}_q$ and $\boldsymbol{v}^*$ is uniform in $\mathbb{Z}_q^{m+m'}$. Therefore, the challenge ciphertext is always uniform in $\mathbb{Z}_q \times \mathbb{Z}_q^{2(m+m')}$.

Let $[p_{min}, p_{max}]$ be the probability interval of artificial abort which does not happen in trapdoor queries. From the simulation above and Lemma 6.2, we have that the value $|p_{max} - p_{min}|$ is at least $\frac{q_t}{q^2}$, and thus it is required to make q large enough to achieve a negligible function $negl(n)$, where $q_t \leq q/2$, p_{min} is at least $\frac{1}{2q}$. Therefore, the advantage of $\mathcal{B}$ in solving LWE can be obtained as:

$$\begin{aligned} Adv_{\mathcal{B}}^{LWE} &\geq \frac{p_{min}}{2}|\Pr[b = b'] - 1/2| + \frac{1}{2}(p_{max} - p_{min}) \\ &\geq \frac{p}{4q} + negl(n) \end{aligned}$$

That is to say

$$|\Pr[\mathbf{Game}_4(1^\lambda)=1]-\Pr[\mathbf{Game}_5(1^\lambda)=1]| \le \mathrm{Adv}^{\mathrm{LWE}}_{\mathcal{B}}.$$

To sum up, all the above arguments show that the given security theorem holds. □

7 IMPLEMENTATION

This section describes the implementation of each algorithm in our scheme. The first part introduces how to realize the TrapGen algorithm to find a short basis of lattice for a given matrix and then generate the master key. The second part introduces how to use Extbasis and RandBasis to extend a basis and matrix to a higher dimensional basis and matrix that support the user tag, which is used to derive the private key. The third part introduces several sampling algorithms to efficiently compute a search token.

7.1 TrapGen Implementation

Many lattice encryption applications [2, 6] use the TrapGen algorithm to obtain a pair of matrices $(\mathbf{A}, \mathbf{S})$ as their public/private keys, where $\mathbf{S}$ is a short basis of lattice $\Lambda^{\perp}(\mathbf{A})$. However, few studies show how to implement it. A recent study [4] only brings a rough introduction to the flow of the algorithm. To realize the algorithm, we propose a customized gaussian elimination method, which provides a general way to solve the linear equations problem over finite fields, and can directly be used to construct a Hermite normal form matrix and its basis.

In details, our method first addresses the linear equations problem over the prime finite field $\mathbb{Z}_q$; that is, find a set of integer solutions x satisfying that $\mathbf{Ax} = \mathbf{0} \mod q$. Without loss of generality, suppose that $rank(\mathbf{A}) = n$, then the number of the basic system of solutions is $m - n$. A generic solution is to produce the reduced row echelon form of the coefficient matrix $\mathbf{A}$ via gaussian elimination method and partial pivoting, and get a basic solution, i.e, for each row of A, do

$$\mathbf{A}(k, j:n) = \mathbf{A}(k, j:n) - \mathbf{A}(k, j) \cdot \mathbf{A}(i, j:n) \tag{6}$$

one by one to eliminate elements below the diagonal and to reduce the elements on the diagonal to 1, where $\mathbf{A}(i, j:n) = \mathbf{A}(i, j:n)/\mathbf{A}(i, j)$. Inspired by the gaussian elimination method [31], to ensure that the solution is an element on the lattice, our solution modifies equation (6) by replacing the division operation in the real number field of the algorithm with the modular inversion on $\mathbb{Z}_q$, i.e., do

$$\mathbf{A}(k, j:n) = \mathbf{A}(k, j:n) - \mathbf{A}(k, j) \cdot \mathbf{A}(i, j:n) \mod q$$

where $\mathbf{A}(i, j:n) = \mathbf{A}(i, j:n) \cdot \mathbf{A}(i, j)^{-1}$. Let $\mathbf{A'}$ denote the upper triangular matrix of $\mathbf{A}$ after gaussian elimination, and then the fundamental solution of linear equations $\mathbf{Ax} = \mathbf{0} \mod q$ can be described as an $m \times (m-n)$-matrix S' that satisfies

$$S'(i,j) = \begin{cases} -\mathbf{A'}(i, n+j), & \text{if } 1 \le i \le n, 1 \le j \le m-n \\ 1, & \text{if } n+1 \le i \le m, j = i-n \\ 0, & \text{else} \end{cases}$$

where each column of S' is a solution of $\mathbf{Ax} = \mathbf{0} \mod q$ and these $m - n$ solutions are linearly independent.

From the structure of the above solution, it can be partitioned into upper and lower blocks, where the lower block is an identity matrix of order $(m-n) \times (m-n)$. Here we call the variable corresponding to the value of "1" in this identity matrix as free variables. Certainly the first $m - n$ variables can also be selected as free variables to construct the solution. Then the new solution obtained is also a set of linearly independent vectors and the upper block is an identity matrix, which is more suitable for constructing the basis of the lattice. With this set of solutions, the next stage is how to extend it to an m-dimensional one, because the basis of $\Lambda^{\perp}(\mathbf{A})$ is an $m \times m$-dimension matrix. This process is straightforward, i.e., adding n column vectors like $\boldsymbol{s}_i = q \times \boldsymbol{e}_i = (0, \ldots, 0, q, 0, \ldots, 0)$ after the $\mathbf{S'}$ for $m - n + 1 \le i \le m$, where $\boldsymbol{e}_i$ is a column vector whose i-th coordinate is "1", and the rest are "0". Now the matrix $\mathbf{S} = (\mathbf{S'}|\boldsymbol{s}_{m-n+1}|\cdots|\boldsymbol{s}_m)$ is a basis of the lattice $\Lambda^{\perp}(\mathbf{A})$ in $\mathbb{Z}^{m \times m}$. Finally, the rest operations for generating a short basis of $\Lambda^{\perp}(\mathbf{A})$ are based on Alwen et al.'s work [4], i.e. converting the above matrix into a Hermite normal form matrix to construct a short basis.

7.2 ExtBasis Implementation

As mentioned in Section 5, we employ both ExtBasis(·) and Randbasis(·) algorithms [10] to derive the private key for a given identity tag. Here, ExtBasis (·) provides a short basis of the form

$$\mathbf{S}^{+} = \begin{pmatrix} \mathbf{S} & \mathbf{Q} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

where $\mathbf{Q}$ satisfies $\mathbf{A}_0\mathbf{Q} = -F_\tau \in \mathbb{Z}_q^{n \times m'}$. Therefore, the problem of extending a basis for lattice $\Lambda^{\perp}(\mathbf{A}_\tau)$ is equivalent to find an arbitrary matrix $\mathbf{Q}$ that satisfies $\mathbf{A}_0\mathbf{Q} \mod q = -F_\tau$. Unlike the previous one in the Initialization stage, it is much more simpler. For each column vector $-\boldsymbol{f}_i$ of F_τ, we only need to find one solution $\boldsymbol{q}_i$ for $\mathbf{A}_0\mathbf{x} = -\boldsymbol{f}_i \mod q$ and $\mathbf{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n)$. Note that the RandBasis(·) algorithm is directly used to randomize the basis for the user.

Remark. The ExtBasis algorithm takes the same lattice basis to generate the extended basis for the tag-matrix, and the algorithm involves a similar process to solve linear equations with the same coefficient matrix. Therefore, our scheme can be further improved to distribute the keys for different users in batches. In particular, the gaussian elimination method over matrix $(\mathbf{A}_0|F_{\tau^1}|\ldots|F_{\tau^n})$ is executed to find the corresponding solution for equation $\mathbf{A}_0 x = \mathbf{h}_i$, where each $\mathbf{h_i}$ is the column of the block $(F_{\tau^1}|\ldots|F_{\tau^n})$.

7.3 SampleLeft Implementation

The main challenge of implementing search is to sample a desired vector for $\boldsymbol{u}_0$ by SampleLeft algorithm with the keyword w and user's private key sk_τ. According to the requirements of the algorithm, the goal is to output a random vector $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ sampled from a distribution statistically close to the distribution $D_{\Lambda_q^u(\mathbf{A}_{\tau,w}),\sigma}$, where $\mathbf{A}_{\tau,\mathbf{w}} = (A_\tau|G(w))$. In particular, the sample algorithm proceeds as follows:

- SampleLeft. First, choose a vector $\boldsymbol{e}_2$ which is distributed statistically close to $D_{\mathbb{Z}_{m_1},\sigma}$ randomly as $\boldsymbol{e}_2$. Then execute $\boldsymbol{e}_1 = \mathrm{SamplePre}(\mathbf{A}_\tau, sk_\tau, \boldsymbol{y})$, and $\boldsymbol{y} = \boldsymbol{u} - (G(w) \cdot \boldsymbol{e}_2) \in \mathbb{Z}_q^n$. Let $\boldsymbol{e} = (\boldsymbol{e}_1, \boldsymbol{e}_2)$ as the sample result.
- SamplePre. For the given vector $\boldsymbol{y}$ matrix $\mathbf{A}_\tau$ and its basis sk_τ, call $\mathrm{SampleD}(\mathbf{A}_\tau, sk_\tau)$ to sample a vector $\boldsymbol{x'}$, so $\mathbf{A}\boldsymbol{x'} = \mathbf{0}$. Find a special solution $\boldsymbol{x}_0$ for $\mathbf{A}\boldsymbol{x} = \boldsymbol{y}$ and output $\boldsymbol{x} \leftarrow (\boldsymbol{x}_0 + \boldsymbol{x'})$. It is easy to verify that $\boldsymbol{y} = \mathbf{A}_\tau \boldsymbol{x}$.

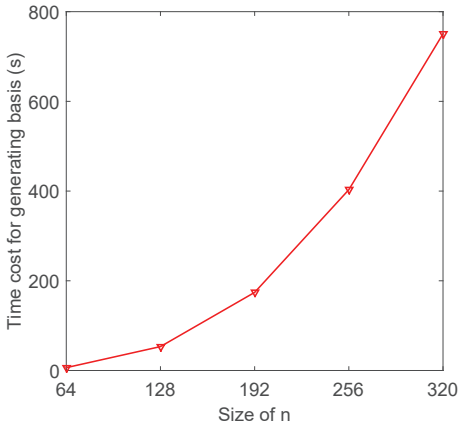


Figure 1: Performance of basis generation

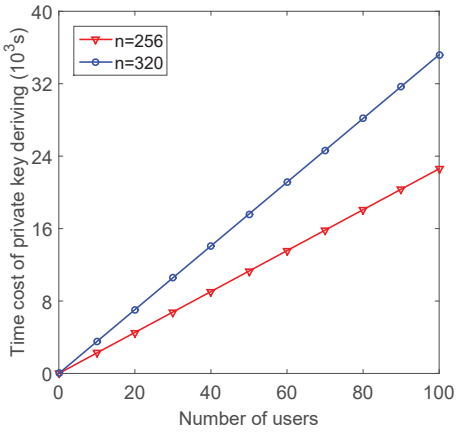


Figure 2: Performance of private key derivation

- SampleD. Let $\boldsymbol{v}_n = 0, \boldsymbol{c}_n = c$. Perform $c_i' = (\boldsymbol{c}_i \cdot \tilde{\boldsymbol{b}}_i^\top)/(\tilde{\boldsymbol{b}}_i \cdot \tilde{\boldsymbol{b}}_i^\top)$, $s_i' = s/\|\boldsymbol{b}_i\|$, $z_i = \text{SampleZ}(\mathbb{Z}, s_i, c_i)$, $\boldsymbol{c}_{i-1} = \boldsymbol{c}_i - z_i\boldsymbol{b}_i$ and $\boldsymbol{v}_{i_1} = \boldsymbol{v}_i + z_i\boldsymbol{b}_i$ while $i = 0$. Output $\boldsymbol{v}_0$ as the result.

8 EXPERIMENT

All the experiments[1] for our scheme are conducted on a Windows 10 laptop with Core i5 CPU 2.5 GHz and 8G memory. Considering that most of the operations here are linear operations of matrices and sampling operations, we use Matlab to implement our scheme for proof of concept, because Matlab provides rich APIs for matrix operations and their underlying implementations are also highly optimized. To better understand the performance of our proposed scheme, we generate synthetic data to report the cost and scalability of each operation. Following the parameter setting of existing work [3, 6, 21], we set the parameter of LWE $n = 256, m = 9753, lm'' = 256, q = 4093$ and $n = 320, m = 13133, lm'' = 256, q = 8191$, where n is the security parameter.

8.1 Experiment Result

8.1.1 Initialization. The Initialize algorithm is to find a short basis of lattice $\Lambda^\perp(\mathbf{A})$ for an arbitrary matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ as the master secret key [4]. Fig.1 shows the time cost for generating a valid basis for different size of n. Note that other related parameters are chosen according to the parameter setting given in Section 5.2. Again, the requirement is that, all these selected parameters in our experiment are sufficient to ensure the LWE assumption holds. From Fig.1, we can see that the average time cost increases with the growth of n. For example, it takes about 6.34 s to generate a pair of matrix and its corresponding lattice basis for $n = 64$ and 750.62 s for $n = 320$.

8.1.2 Key Derivation. The Key derive algorithm is to generate a private key for a given user, and the user can be identified via their tag. In this experiment, we use a 1024-bit string to denote the tag, which is then transferred to a tag-matrix. Fig.2 shows the time cost for private key generation for 100 users over different basis size. This cost is proportional to the number of users. As seen, it spends about 22606 s and 35203 s on generating private keys for all 100 users while $n = 256$ and $n = 320$. Note that further evaluations in Section 8.2 demonstrate that our proposed batch optimization can further improve the efficiency of the private key generation.

[1]The source code is published at: https://github.com/Raycrypto/Multi-writer-PEKS

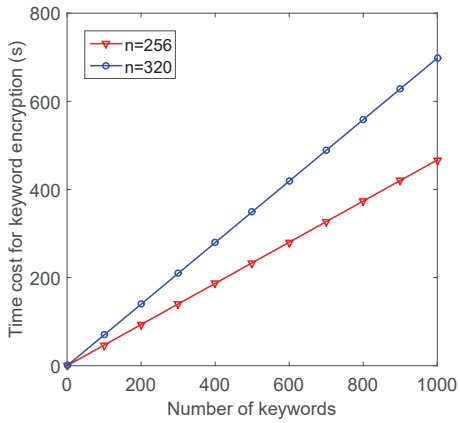


Figure 3: Performance of data encryption

8.1.3 Data Encryption. Fig.3 depicts the time cost for encrypting 1000 keywords. This process consists of $2k+3$ matrix multiplicative transformations and additions, two vector inner product operations, and two matrix multiplications. As seen, this cost is proportional to the number of keywords. From Fig. 3, it takes about 467 s to encrypt 1000 keywords for $n = 256$ and 698 s for $n = 320$.

8.1.4 Search. The search algorithm includes the token generation and search (token matching) operation. The user first uses her private key to generate the token for the keyword she expects to search and sends it to the server. Then the server performs search by checking the token with the encrypted keywords. Fig.4 records the time of performing search test over 1000 encrypted keywords after removing the token generation cost, which is given later in Table 4-b. For the search operation, the whole process only contains one inner product calculation for two vectors. Fig. 4 shows that it takes 13.4 ms to perform search matching over 1000 keywords for $n = 256$ and 16.98 ms for $n = 320$.

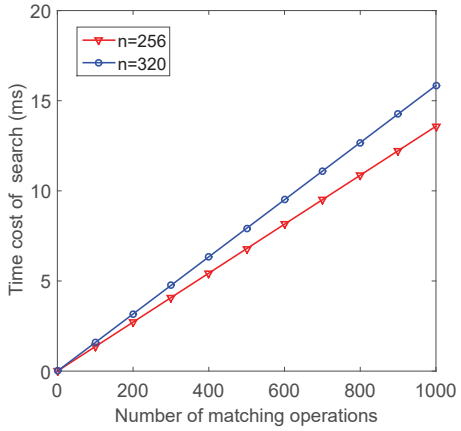
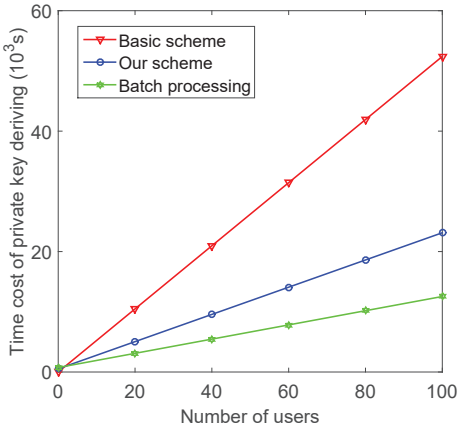
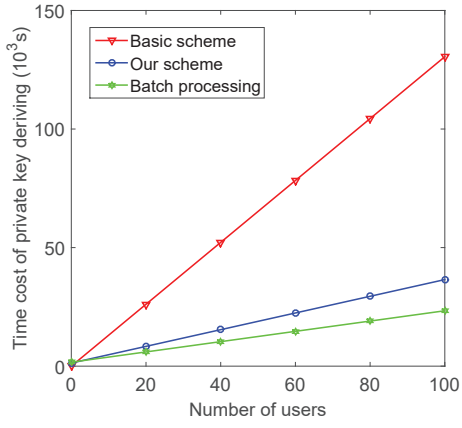


Figure 4: Performance of search



(a) Time cost for $n = 256$



(b) Time cost for $n = 320$

Figure 5: Comparison of Setup and KeyGen

8.2 Comparison with Prior Art

To demonstrate the advantage of our scheme, we perform comparison of the storage overhead and time cost with a basic LWE PEKS scheme proposed by Behnia et al. [6] (denoted as basic scheme). This scheme does not leverage identity for key generation and generates a pair of public and private keys for each user. For the sake of fairness, we conduct the experiment under the same parameters setting as [6]. The tags and keywords are randomly generated with the size of 128 bits. Note that the scheme is not implemented in [6] and only simulated results on a high-end machine (with better capabilities than ours) are reported in that paper. Hence, the comparison on the time cost is based on the evaluation of our implementation and the simulated results reported in [6].

From the perspective of the storage, Table 4-(a) shows that the sizes of the private key and the token in our scheme and the basic scheme are comparable. For example, when $n = 256$, the size of private key and token are 547 MB and 113 KB in our scheme, while these are 519 MB, 110 KB in the basic scheme. However, the storage cost of the keyword ciphertext in our scheme is much smaller than the basic scheme. In the case of $n = 256$, the size of a keyword ciphertext in our scheme is 113 KB, while it is 9086 KB in the basic scheme, which is about 80× larger than us. The reason is that we improve the structure of the ciphertext by removing the redundant components of their ciphertext. This also affects the computation cost, which will be given later.

Table 4: Comparison with the scheme in [6]

Scheme		Private key	Token	Ciphertext
[6]	$n = 256$	519 MB	110KB	9086 KB
	$n = 320$	942 MB	148 KB	29692 KB
Ours	$n = 256$	547 MB	113 KB	113 KB
	$n = 320$	988 MB	151 KB	151 KB

(a) space cost comparison

Scheme		Encrypt	TokGen	Search
[6]	$n = 256$	2115.12 ms	814.9 ms	643 ms
	$n = 320$	3727 ms	1509 ms	1214 ms
Ours	$n = 256$	467 ms	6126 ms	13.4 ms
	$n = 320$	698 ms	8362 ms	16.98 ms

(b) time cost comparison

In terms of time cost, we report the time cost of each algorithm in our scheme and compare it with the simulated cost in the basic scheme. From Fig.5, we can find that in the basic scheme, it takes them almost 524s and 1305s to generate a pair of public key for $n = 256$ and $n = 320$ respectively, and the time to generate public/private key pairs is proportional to the number of users. While in our scheme, it takes 728s and 1667s to generate the private key for the first user but average 252.2s and 365.2s for 100 users. Compared to the average time spent, our scheme is 2× and 3.5× faster than theirs. This result is due to the fact that we only use the TrapGen algorithm once to generate a short basis of lattice, and then use this basis to generate private keys for users. However, the above process needs to repeat 100 times in the basic scheme, and the cost will expand rapidly with an increasing number of users. In addition, as the ExtBasis algorithm shares the same input, master key, our proposed optimization technique can compute the extended basis in a batch. The corresponding results can be seen in Fig. 5, which improves the performance. With the batch processing technique,

key derivation can achieve a speed up of is 1.8× and 1.3× compared to our original algorithm.

In the Table 4, our TokenGen algorithm takes more time to generate the search token, because the new identity feature introduces more calculations. But for each keyword, one may execute the token generation algorithm once and cache this token locally for subsequent use. While for the efficiency of encryption and search operations, both algorithms are 80× and 190× faster than the basic scheme for $n = 256$ and 320 respectively.

9 CONCLUSION

In this paper, we design and implement a lattice-based searchable encryption scheme from LWE for applications in multi-writer scenarios. In particular, our scheme leverages the philosophy of identity-based encryption and can support users to encrypt data directly via their identities. This alleviates the bottleneck of key management in LWE based searchable encryption schemes. We conduct detailed correctness and security analysis for our scheme and perform a series of experiments to evaluate the efficiency of each function. There are still issues remained to be solved, i.e., how to reduce the size of private key and how to improve the performance of key derivation and token generation. Accordingly, we will explore and design schemes constructed from Ring-LWE [24] or Module-LWE [20]. Adapting these variants of LWE would further improve the efficiency but might introduce tradeoff on security. We leave them as our future work.

ACKNOWLEDGMENTS

This work was supported in part by the Fundamental Research Funds for the Central Universities (No.30918012204), Research Grants Council of Hong Kong under Grant CityU 11276816, Grant CityU 11212717, and Grant CityU C1008-16G, in part by the Innovation and Technology Commission of Hong Kong under ITF Project ITS/168/17, in part by the National Natural Science Foundation of China under Grant 61572412.

REFERENCES

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. 2008. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *Journal of Cryptology* 21, 3 (2008), 350–391.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. 2010. Efficient Lattice (H)IBE in the Standard Model. In *Proc. of EUROCRYPT*. 553–572.

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

[4] Joël Alwen and Chris Peikert. 2011. Generating Shorter Bases for Hard Random Lattices. *Theory Computing System* 48, 3 (2011), 535–553.

[5] Daniela Becker, Jorge Guajardo, and Karl-Heinz Zimmermann. 2018. Revisiting Private Stream Aggregation: Lattice-Based PSA. In *Proc. of NDSS*.

[6] Rouzbeh Behnia, Muslum Ozgur Ozmen, and Attila Altay Yavuz. 2018. Lattice-Based Public Key Searchable Encryption from Experimental Perspectives. *IEEE Trans. on Dependable and Secure Computing* (2018).

[7] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004. Public Key Encryption with Keyword Search. In *Proc. of EUROCRYPT*. 506–522.

[8] Dan Boneh and Brent Waters. 2007. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Proc. of TCC*. 535–554.

[9] Ahmad Boorghany and Rasool Jalili. 2014. Implementation and Comparison of Lattice-based Identification Protocols on Smart Cards and Microcontrollers. *IACR Cryptology ePrint Archive* (2014), 78.

[10] David Cash, Dennis Hofheinz, and Eike Kiltz. 2009. How to Delegate a Lattice Basis. *IACR Cryptology ePrint Archive* 2009 (2009), 351.

[11] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. 2012. Bonsai Trees, or How to Delegate a Lattice Basis. *Journal of Cryptology* 25, 4 (2012), 601–639.

[12] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2011. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security* 19, 5 (2011), 895–934.

[13] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *Proc. of STOC*. 197–206.

[14] Philippe Golle, Jessica Staddon, and Brent R. Waters. 2004. Secure Conjunctive Keyword Search over Encrypted Data. In *Proc. of ACNS*. 31–45.

[15] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. 2001. NSS: An NTRU Lattice-Based Signature Scheme. In *Proc. of EUROCRYPT*. 211–228.

[16] Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2013. Outsourced symmetric private information retrieval. In *Proc. of ACM CCS*. 875–888.

[17] Jonathan Katz and Vinod Vaikuntanathan. 2009. Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices. In *Proc. of Asiacrypt*. 636–652.

[18] Elena Kirshanova. 2014. Proxy Re-encryption from Lattices. In *Proc. of PKC*. 77–94.

[19] Shangqi Lai, Sikhar Patranabis, Amin Sakzad, Joseph K. Liu, Debdeep Mukhopadhyay, Ron Steinfeld, Shifeng Sun, Dongxi Liu, and Cong Zuo. 2018. Result Pattern Hiding Searchable Encryption for Conjunctive Queries. In *Proc. ACM CCS*. 745–762.

[20] Adeline Langlois and Damien Stehlé. 2015. Worst-case to Average-case Reductions for Module Lattices. *Design Codes Cryptography* 75, 3 (2015), 565–599.

[21] Richard Lindner and Chris Peikert. 2011. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *Proc. of CT-RSA*. 319–339.

[22] Zhe Liu, Reza Azarderakhsh, Howon Kim, and Hwajeong Seo. 2017. Efficient Software Implementation of Ring-LWE Encryption on IoT Processors. *IEEE Trans. on Computers* PP, 99 (2017), 1–1.

[23] Vadim Lyubashevsky. 2012. Lattice Signatures without Trapdoors. In *Proc. of Asiacrypt*. 738–755.

[24] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On Ideal Lattices and Learning with Errors over Rings. *Journal of ACM* 60, 6 (2013), 43:1–43:35.

[25] Daniele Micciancio and Oded Regev. 2013. Lattice-based Cryptography. *Lecture Notes in Computer Science* 4117, 1-2 (2013), 131–141.

[26] T Monz, D Nigg, E. A. Martinez, M. F. Brandl, P Schindler, R Rines, S. X. Wang, I. L. Chuang, and R Blatt. 2016. Realization of a scalable Shor Algorithm. *Science* 351, 6277 (2016), 1068–1070.

[27] Chris Peikert et al. 2016. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science* 10, 4 (2016), 283–424.

[28] Oded Regev. 2006. *Lattice-Based Cryptography*. Springer Berlin Heidelberg. 131–141 pages.

[29] Oded Regev. 2009. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of ACM* 56, 6 (2009), 34:1–34:40.

[30] Cédric Van Rompay, Refik Molva, and Melek Önen. 2018. Secure and Scalable Multi-User Searchable Encryption. In *Proc. of SCC@AsiaCCS 2018*. 15–25.

[31] Tateaki Sasaki and Hirokazu Murao. 1982. Efficient Gaussian Elimination Method for Symbolic Determinants and Linear Systems. *ACM Trans. on Mathematical Software* 8, 3 (1982), 277–289.

[32] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. 2007. Multi-Dimensional Range Query over Encrypted Data. In *Proc. of IEEE S&P*. 350–364.

[33] Peter W. Shor. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proc. of IEEE FOCS*. 124–134.

[34] Shifeng Sun, Joseph K. Liu, Amin Sakzad, Ron Steinfeld, and Tsz Hon Yuen. 2016. An Efficient Non-interactive Multi-client Searchable Encryption with Support for Boolean Queries. In *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*. 154–172.

[35] Shifeng Sun, Xingliang Yuan, Joseph K. Liu, Ron Steinfeld, Amin Sakzad, Viet Vo, and Surya Nepal. 2018. Practical Backward-Secure Searchable Encryption from Symmetric Puncturable Encryption. In *Proc. of the ACM CCS*. 763–780.

[36] Brent Waters. 2005. Efficient Identity-Based Encryption Without Random Oracles. In *Proc. of EUROCRYPT*. 114–127.

[37] Brent R Waters, Dirk Balfanz, Glenn Durfee, and Diana K Smetters. 2004. Building an Encrypted and Searchable Audit Log. In *Proc. of NDSS*.

[38] Lei Xu, Shifeng Sun, Xingliang Yuan, Joseph K Liu, Cong Zuo, and Chungen Xu. 2019. Enabling Authorized Encrypted Search for Multi-Authority Medical Databases. *IEEE Transactions on Emerging Topics in Computing* (2019).