IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON EMERGING APPROACHES TO CYBER SECURITY

Received September 29, 2019, accepted November 2, 2019, date of publication November 7, 2019,
date of current version November 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952163

Lattice-Based Key-Aggregate (Searchable) Encryption in Cloud Storage

YANQING YAO[1,3], ZHENGDE ZHAI[2], JIANWEI LIU[1], AND ZHOUJUN LI[3,4]

[1]Key Laboratory of Aerospace Network Security, Ministry of industry and information technology, School of Cyberspace Science and Technology, Beihang University, Beijing 100191, China
[2]Ant Financial Services Group, Beijing 100020, China
[3]State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[4]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Corresponding author: Yanqing Yao (yaoyq@buaa.edu.cn)

This work was supported in part by NSFC under Grant 61602025, Grant 61972017, Grant 61972018, Grant 61932014, Grant U1636211, and Grant 61170189, in part by the National Key Research and Development Program of China under Grant 2016QY04W0802 and Grant 2017YFB1400702, in part by the Beijing Natural Science Foundation under Grant 4182033, in part by the National Cryptography Development Fund under Grant MMJJ20180215, and in part by the Fund of the State Key Laboratory of Software Development Environment under Grant SKLSDE-2019ZX-17 and Grant SKLSDE-2016ZX-22.

ABSTRACT In cloud storage, selectively sharing encrypted data is becoming increasingly important. One key design challenge is the management of encryption keys. Traditionally, a large quantity of encryption keys have to be managed by the data owner, and an equally large number of keyword trapdoors must be sent to the cloud for the purpose of searching over the shared data, which are cumbersome in terms of secure communication and management. Recently, key-aggregate (searchable) encryption schemes have been introduced to alleviate the problem. However, they were only designed under the Bilinear Diffie-Hellman Exponent assumption in the prior works. Lattice-based key-aggregate (searchable) encryption schemes are valuable, because they have security against quantum computing attacks, average-case to worse-case equivalence as well as simplicity and potential efficiency. Here we propose a key-aggregate encryption scheme and a key-aggregate searchable encryption scheme which are both based on a lattice problem (i.e., the Learning with Errors problem). Some key techniques are employed during the construction of the schemes. A basis delegation algorithm is designed to generate the aggregate key without increasing the lattice dimension. The encryption algorithms of the two schemes are trickily devised to make the encrypted files decryptable or searchable. To overcome the problem of general matrix multiplication failing to satisfy commutative law, a hash function is designed by using diagonalizable matrices to make the encrypted file decryptable and the trapdoor adjustable. We present the schemes' correctness proof, formal security analysis as well as performance analysis, which confirm that they are provably secure and practically efficient. To the best of our knowledge, the former is the first lattice-based key-aggregate encryption scheme and the latter is the first lattice-based key-aggregate searchable encryption scheme. We also demonstrate their application to cloud storage for searchable group data sharing by combining the two schemes.

INDEX TERMS Cloud storage, searchable data sharing, key-aggregate encryption, key-aggregate searchable encryption, lattice-based problem, the learning with errors problem.

I. INTRODUCTION

Recently, could storage has achieved great popularity. Each day, millions of users upload their personal data (e.g., photos, videos) or working data to cloud servers and share them with their friends or colleagues. Cloud storage has attracted wide attention because of its great agility, low cost, and global scalability.

Data privacy is a primary concern in cloud storage. Serious breaches of personal privacy or business secrets may be caused by misbehaving cloud operators or malicious hackers. To address this issue, cryptographic cloud storage [6] has emerged. In this kind of storage, data is encrypted by the data owner before being uploaded to the cloud, and encrypted data can only be retrieved and decrypted by the data owner

The associate editor coordinating the review of this manuscript and approving it for publication was Kuan Zhang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

and authorized users. The decryption keys are only kept by the data owner or the authorized data users and no access to the original data can be achieved by the cloud operator employees. One drawback of traditional cryptographic cloud storage is that it requires the management of a large number of keys. Usually, a data owner has to generate a pair of keys for each file and send the corresponding decryption keys for the files that are to be shared to the intended users. Storage, management and distribution of a large number of keys can be very challenging. Partially inspired by collusion-resistant broadcast encryption scheme [4], Chu *et al.* [5] introduced key-aggregate encryption schemes in which an aggregate key is generated for the authorized user to decrypt all the shared files. Recently, key aggregate signatures [10], [13], [20], [24] also have been studied.

In cryptograhphic cloud storage, another challenging problem is for the authorized user to search the encrypted data and select the encrypted files which contain target keywords. To address the issue, a searchable encryption scheme can be employed. In such a scheme, the data owner encrypts potential keywords and uploads them to the cloud together with encrypted data, and later for retrieving data matching a keyword, the corresponding keyword trapdoor is sent to the cloud by the authorized user to perform keyword search over the encrypted data.

Though searchable encryption schemes have been implemented in cryptographic cloud storage, such schemes also face the problem of cumbersome key management when the number of users and the number of files are very large [8]. When a data owner shares data with his friends, different encryption keys are usually required for different files. Management of those keys can be a real burden. It also implies that a great number of keys have to be securely transmitted to authorized users and securely stored and managed by them, which further complicates the schemes.

Now we recall a range of multi-user searchable encryption schemes and multi-key searchable encryption schemes which are related to our work. In a typical multi-user keyword search scenario, the data owner shares a document with a number of authorized users and each authorized user can perform keyword search over the shared document by providing a trapdoor to the cloud server. Some studies [6], [7], [9], [14], [17]–[19], [27], [28] have combined the single-key with access controls. In literature the main issue is how to decrease the number of trapdoors and shared keys. The concept of multi-key searchable encryption scheme was firstly introduced by Popa and Zeldovich [22] in 2013. In such a scheme, an authorized user sends a single keyword trapdoor to the cloud sever to ask for it to search for the corresponding trapdoor's keyword in files encrypted with multiple keys. Unfortunately, aggregate keys are not considered in such schemes.

Cui *et al.* [8] proposed the concept of key-aggragate searchable encryption scheme and exemplified the concept with a concrete key-aggregate searchable encryption scheme based on the Bilinear Diffie-Hellman Exponent (BDHE) assumption. In such a scheme, a single aggregate key is sent by the data owner to an authorized user, and a single aggregate trapdoor is sent by the authorized user to the cloud server to perform keyword search over the data. It greatly reduces the cost of key management and secure communication.

Key-aggregate encryption scheme and key-aggregate searchable encryption scheme can be applied to searchable group data sharing in cloud storage. In this scenario, there are two requirements for efficient key management. Firstly, a single aggregate key (other than a group of keys) should be sent by the data owner to an authorized data user for sharing any number of files. In addition, a single aggregate trapdoor (other than a group of trapdoors) should be sent by the authorized data user to the cloud server for performing keyword search over any number of shared files. Though the key-aggregate encryption scheme in [5] has inspired the work of [8], the former work only satisfies the first requirement but not the second.

To our knowledge, neither lattice-based key-aggregate encryption scheme nor lattice-based key-aggregate searchable encryption scheme has been known before. It's very valuable to construct such schemes because lattice-based cryptography is resistant to quantum cryptanalysis (as opposed to discrete logarithm problem or factoring problem [25]) and has equivalence between average-case hardness and worst-case hardness. Besides, its conceptual simplicity and potential efficiency are also prominent advantages.

Until now, several lattice-based public key encryption schemes with keyword search have been proposed (e.g., [11], [27], [30]). In addition, Agrawal *et al.* [1] proposed hierarchical identity-based encryption (HIBE) schemes by presenting a technique for delegating a short lattice basis. Moreover, proxy-oriented identity-based searchable encryption schemes from lattices have been introduced to alleviate the data owner's encryption processing burden and to eliminate the cost of managing certificates [29]. However, none of them can be used directly to construct key-aggregate (searchable) encryption schemes mainly because matrix multiplication does not satisfy commutative law in general and it is difficult to design the Encrypt algorithm and the aggregate key (and the aggregate trapdoor) from lattice.

A. OUR CONTRIBUTIONS AND TECHNIQUES

In this paper, we initially investigate the design of key-aggregate encryption scheme based on the Learning with Errors problem. In such a scheme, a message is encrypted under a public-key as well as an identifier (i.e., an index) of the file which the message belongs to. The master secret key is used by the data owner to generate an aggregate decryption key. Any subset of encrypted files is decryptable by the aggregate decryption key. The aggregate decryption key is as compact as a decryption key of a single encrypted file, but aggregates the decryption power for any subset of encrypted files.

However, a key-aggregate encryption scheme can't achieve the goal of searching over the encrypted files. It inspires

us to design a key-aggregate searchable encryption scheme based on the Learning with Errors problem. In such a scheme, a single aggregate key is distributed by the data owner to an authorized user when sharing multiple files, and a single aggregate trapdoor is submitted by the authorized user to perform search queries over the data shared by the data owner.

Note that though the lattice-based key-aggregate encryption scheme provides an insight into the design of lattice-based key-aggregate searchable encryption scheme, the latter requires much more complex mathematical transformation to perform trapdoor generation and keyword matching as well as keyword encryption.

More specifically, our main contributions and techniques are as follows:

- We design a key-aggregate encryption scheme under the hardness assumption of the Learning with Errors problem. As far as we know, this is the first lattice-based key-aggregate encryption scheme until now.
- We design a key-aggregate searchable encryption scheme based on the Learning with Errors problem. To the best of our knowledge, no key-aggregate searchable encryption scheme based on a lattice problem has been proposed in previous literature.
- Partially inspired by the lattice-based hierarchical IBE [1] and public key encryption with keyword search [11], we propose several techniques to design the two schemes.
  - In order to obtain an aggregate key, the set $\mathcal{S}$ of files' indices is mapped to a multiplication of $|\mathcal{S}|$ matrices so that the aggregate key, which is a short lattice basis, can be obtained without increasing the lattice dimension.
  - The encryption algorithms of the two schemes are designed trickily such that the encrypted files can be decrypted or searchable.
  - To deal with the problem that general matrix multiplication fails to satisfy commutative law during the construction, we design a hash function by using diagonalizable matrices and discuss their equivalence property so that the file decryption keys can be derivable from the aggregate key and the trapdoor can be adjustable.
- We prove our schemes' correctness, and present formal security analysis and performance analysis, which confirm that our schemes are provably secure and practically efficient.
- We also present their application to cloud storage for searchable group data sharing by combining the key-aggregate encryption scheme with the key-aggregate searchable encryption scheme.

Because our schemes are based on a lattice problem, they have security against quantum computing attacks and average-case to worst-case hardness equivalence. In addition, they have the merits of conceptual simplicity and potential efficiency.

B. ORGANIZATION

In Section *II*, we review some notations, concepts and lemmas that will be used in the paper. In Section *III*, we review the concept of key-aggregate encryption scheme, propose a concrete construction of key-aggregate encryption scheme based on the Learning with Errors problem, and analyze its correctness and security. Then in Section *IV* we review the key-aggregate searchable encryption scheme framework, design a concrete key-aggregate searchable encryption scheme based on the Learning with Errors problem, and analyze its correctness and security. In Section *V*, we show the performance analysis of our schemes. In Section *VI*, we show how the combination of the two schemes is applied to cloud storage for searchable group data sharing. Finally, we conclude the paper in Section *VII*.

II. PRELIMINARIES

For a positive integer n, denote $[n] \stackrel{def}{=} \{1, 2, \ldots, n\}$. For $x \in \mathbb{R}$, let $\lfloor x \rceil$ be the closest integer to x. Consider an ordered set $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ in $\mathbb{R}^m$, the elements of which are linearly independent (column) vectors, the norm of $\mathbf{S}$ is defined by $\|\mathbf{S}\| \stackrel{def}{=} \max_{j\in[k]} \|\mathbf{s}_j\|$, and denote $\tilde{\mathbf{S}}$ as the Gram-Schmidt orthogonalization of $\mathbf{S}$. For any $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, denote $\Lambda_q^{\perp}(\mathbf{A}) \stackrel{def}{=} \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\cdot\mathbf{x} = \mathbf{0} \ mod\ q\}$ and $\Lambda_q^{\mathbf{u}}(\mathbf{A}) \stackrel{def}{=} \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\cdot\mathbf{x} = \mathbf{u} \ mod\ q\}$.

For a distribution Ψ over the set D, let $d \stackrel{\Psi}{\leftarrow} D$ represent sampling d according to Ψ over the set D. In particular, let $d \stackrel{U}{\leftarrow} D$ represent sampling d according to the uniform distribution over the set D.

For a real number $\alpha \in (0, 1)$ and a prime q, let X be a normal random variable, the mean and standard deviation of which are 0 and $\alpha/\sqrt{2\pi}$ respectively, and let $\overline{\Psi}_\alpha$ be the distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil \ mod\ q$. Let $s_R \stackrel{def}{=} \sqrt{n\log q}\cdot\omega(\sqrt{\log m})$. Denote $\mathcal{D}_{\mathbb{Z}^m,s_R}$ as the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter s_R and center $\mathbf{0}$, then it can be seen that $\mathcal{D}_{\mathbb{Z}^m,s_R}(\mathbf{z}) \stackrel{def}{=} \frac{\exp(-\pi\|\mathbf{z}\|^2/s_R^2)}{\sum_{\mathbf{x}\in\mathbb{Z}^m}\exp(-\pi\|\mathbf{x}\|^2/s_R^2)}$ for every $\mathbf{z} \in \mathbb{Z}^m$.

A matrix $\mathbf{R} \in \mathbb{Z}^{m\times m}$ is called $\mathbb{Z}_q$-invertible if $\mathbf{R}\ mod\ q$ is invertible as a matrix in $\mathbb{Z}_q^{m\times m}$. Let $\mathcal{D}_{m\times m}$ be the distribution over matrices in $\mathbb{Z}^{m\times m}$ defined as $(\mathcal{D}_{\mathbb{Z}^m,s_R})^m$ under the assumption that the resulting matrix is $\mathbb{Z}_q$-invertible. Further, let $\mathcal{D}'_{m\times m}$ be the distribution over matrices in $\mathbb{Z}^{m\times m}$ defined as $(\mathcal{D}_{\mathbb{Z}^m,s_R})^m$ under the assumption that the resulting matrix is $\mathbb{Z}_q$-invertible and diagonalizable in $\mathbb{Z}^{m\times m}$.

Definition 1 (See [23]): Consider a positive integer n, a prime q, and a distribution λ over $\mathbb{Z}_q$. For a vector $\mathbf{s} \in \mathbb{Z}_q^n$, denote $A_{\mathbf{s},\lambda}$ as the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a uniformly random vector $\mathbf{a}$ from $\mathbb{Z}_q^n$ and outputting $(\mathbf{a}, \mathbf{a}^T\mathbf{s} + b)$ where b is a sample drawn from the distribution λ. The $(\mathbb{Z}_q, n, \lambda) - LWE$ (Learning with Errors) problem is to distinguish between the distribution $A_{\mathbf{s},\lambda}$ (for constant random secret key $\mathbf{s} \in \mathbb{Z}_q^n$) and the uniform distribution

over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *conditioned on oracle access to any desired* $m = poly(n)$ *samples.*

Regev [23] showed that if there exists an efficient (and possibly quantum) algorithm for solving the $(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)-$LWE problem where $\alpha \in (0, 1)$ and $q > 2\sqrt{n}/\alpha$, then an efficient quantum algorithm can be employed to approximate the decision version of the shortest vector problem and the shortest independent vectors problem, to within some factors (i.e., $\tilde{O}(n/\alpha)$) in the l_2 norm.

Lemma 1 (See [1], [12]): Let $\mathbf{y} \overset{\overline{\Psi}_\alpha^m}{\leftarrow} \mathbb{Z}_q^m$ *and let* $\mathbf{e}$ *be a certain vector in* $\mathbb{Z}^m$*. Then the quantity* $|\mathbf{e}^T\mathbf{y}|$ *which is an integer in* $[0, q-1]$ *has the following property. With all but negligible probability in m,*

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

The proof can be found in the counterpart of Lemma 8.2 in [12]. The above lemma will be used to show that our decryption can work correctly.

Lemma 2 (See [1], [2]): Let $q \geq 3$ *be odd. There exists a probabilistic polynomial-time (PPT for short) algorithm* $\mathsf{GenBasis}(1^n; 1^m; q)$ *that, for* $poly(n)-$*bounded* $m \geq Cn \log q$*, where C is a fixed constant with* $C > 1$*, generates* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{T} \in \mathbb{Z}^{m\times m}$ *satisfying the following properties.*

(1) *The statistical distance between the distribution* $\mathbf{A}$ *and uniform distribution is within negl(n).* $\mathbf{T}$ *is a basis of* $\Lambda_q^\perp(\mathbf{A})$ *with* $\|\widetilde{\mathbf{T}}\| \leq \widetilde{L} = O(\sqrt{n \log q})$.

(2) *Suppose a* $\mathbb{Z}_q$*-invertible matrix* $\mathbf{R} \in \mathbb{Z}^{m\times m}$ *is sampled from* $\mathcal{D}_{m\times m}$ *(or a product of such), and s satisfies* $s > \|\widetilde{\mathbf{T}}\| \cdot s_R\sqrt{m} \cdot \omega(\log^{3/2} m)$ *(or* $s > \|\widetilde{\mathbf{T}}\| \cdot (s_R\sqrt{m} \cdot \omega(\log^{1/2} m))^l \cdot \omega(\log m)$ *if* $\mathbf{R}$ *is a product of l matrices sampled from* $\mathcal{D}_{m\times m}$*). Then there exists an efficient algorithm* $\mathsf{BasisDel}(\mathbf{A}, \mathbf{R}, \mathbf{T}, s)$ *that generates a basis of* $\Lambda_q^\perp(\mathbf{AR}^{-1})$*, which is denoted as* $\mathbf{T}'$.

(3) *Assume that* $\mathbf{T}_{ar}$ *is a basis of* $\Lambda_q^\perp(\mathbf{AR}^{-1})$ *with* $\|\widetilde{\mathbf{T}_{ar}}\| < s/\omega(\sqrt{\log m})$*. Then the distribution* $\mathbf{T}'$ *is statistically close to the distribution* $\mathsf{RandBasis}(\mathbf{T}_{ar}, s)$*, where* $\mathsf{RandBasis}(\mathbf{T}_{ar}, s)$ *is a PPT algorithm that outputs a basis* $\mathbf{T}''$ *such that* $\|\widetilde{\mathbf{T}''}\| \leq s\sqrt{m}$ *with overwhelming probability.*

The above lemma shows that (1) a matrix $\mathbf{A}$ statistically close to uniform distribution over $\mathbb{Z}_q^{n\times m}$ and a short basis can be produced by a PPT algorithm $\mathsf{GenBasis}(1^n; 1^m; q)$, (2) there exists a basis delegation algorithm without increasing the dimension of the underlying matrix, and (3) the algorithm $\mathsf{BasisDel}$ generates a random basis of $\Lambda_q^\perp(\mathbf{AR}^{-1})$ which is statistically close to a random basis of $\Lambda_q^\perp(\mathbf{AR}^{-1})$ whose Gram-Schmidt norm is bounded by a function of $\|\widetilde{\mathbf{T}}\|$.

Lemma 3 (See [12]): Assume that $q \geq 3$ *and* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *where* $m > n$*. Assume that* $\mathbf{T}$ *is a basis of* $\Lambda_q^\perp(\mathbf{A})$ *and* $\tau \geq \|\widetilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log m})$*. Then for* $\mathbf{u} \in \mathbb{Z}_q^n$*,*

(1) $Pr[\mathbf{d} \sim \mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\tau} : \|\mathbf{d}\| > \tau\sqrt{m}] \leq negl(n)$;

(2) *there exists a PPT algorithm* $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{u}, \tau)$ *which outputs* $\mathbf{d} \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$ *sampled from a distribution that is statistically close to* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\tau}$*, whenever* $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ *is not empty.*

Assume that the hash function H is defined as follows.

$$H : \{0, 1\}^* \to \mathbb{Z}^{m\times m} : i \mapsto H(i) \sim \mathcal{D}'_{m\times m}.$$

Namely, the output $H(i)$ is distributed as $\mathcal{D}'_{m\times m}$ over the choice of the random oracle H.

Similar to [15], [26] that defined the concept of diagonalizable matrix in $\mathbb{R}^{m\times m}$, we say that a square matrix $\mathbf{H} \in \mathbb{Z}^{m\times m}$ is $\mathbb{Z}_q$-diagonalizable whenever there is a $\mathbb{Z}_q$-invertible matrix $\mathbf{P}$ such that $\mathbf{P}^{-1}\mathbf{HP}$ is a diagonal matrix in $\mathbb{Z}^{m\times m}$, and we have the following theorem.

Theorem 1: A square matrix $\mathbf{H} \in \mathbb{Z}^{m\times m}$ *is* $\mathbb{Z}_q$*-diagonalizable if and only if* $\mathbf{H}$ *has m linearly independent eigenvectors in* $\mathbb{Z}_q^m$.

Proof: "$\Longrightarrow$" Assume that there is a $\mathbb{Z}_q$-invertible matrix $\mathbf{P}$ such that $\mathbf{P}^{-1}\mathbf{HP}$ is a diagonal matrix in $\mathbb{Z}^{m\times m}$. Let $\mathbf{P} = (\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_m)$. Then

$$\begin{aligned}
&\mathbf{H}(\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_m)\\
&= (\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_m)\begin{bmatrix}\lambda_1 & 0 & \cdots & 0 & 0\\ 0 & \lambda_2 & \cdots & 0 & 0\\ 0 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 0 & \lambda_m\end{bmatrix}\\
&= (\lambda_1\mathbf{p}_1\ \lambda_2\mathbf{p}_2\ \ldots\ \lambda_m\mathbf{p}_m)
\end{aligned}$$

Therefore, $\mathbf{Hp}_i = \lambda_i\mathbf{p}_i$ holds for each $i \in \{1, 2, \ldots, m\}$. Since $\mathbf{P}$ is a $\mathbb{Z}_q$-invertible matrix, $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m$ are m linearly independent eigenvectors of $\mathbf{H}$.

"$\Longleftarrow$" Assume that $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m \in \mathbb{Z}_q^m$ are m linearly independent eigenvectors of $\mathbf{H}$. Then for every $i \in \{1, 2, \ldots, m\}$, we have that $\mathbf{Hp}_i = \lambda_i\mathbf{p}_i$. Therefore,

$$\begin{aligned}
&\mathbf{H}(\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_m)\\
&= (\mathbf{Hp}_1\ \mathbf{Hp}_2\ \cdots\ \mathbf{Hp}_m)\\
&= (\lambda_1\mathbf{p}_1\ \lambda_2\mathbf{p}_2\ \cdots\lambda_m\mathbf{p}_m)\\
&= (\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_m)\begin{bmatrix}\lambda_1 & 0 & \cdots & 0 & 0\\ 0 & \lambda_2 & \cdots & 0 & 0\\ 0 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 0 & \lambda_m\end{bmatrix}
\end{aligned}$$

For simplicity, let $\mathbf{P} = (\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_m)$. Then

$$\mathbf{HP} = \mathbf{P}\begin{bmatrix}\lambda_1 & 0 & \cdots & 0 & 0\\ 0 & \lambda_2 & \cdots & 0 & 0\\ 0 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 0 & \lambda_m\end{bmatrix}$$

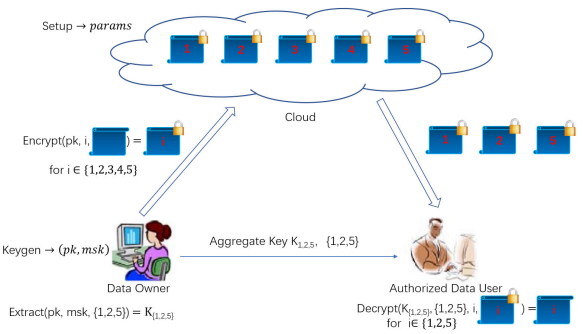


FIGURE 1. Framework of key-aggregate encryption scheme.

That is,

$$\mathbf{P}^{-1}\mathbf{H}\mathbf{P} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \lambda_m \end{bmatrix}$$

Therefore, $\mathbf{H} \in \mathbb{Z}^{m\times m}$ is $\mathbb{Z}_q$-diagonalizable. □

In the reminder of the paper, we introduce a function G as follows.

For any sample $\mathbf{H}$ from the distribution $\mathcal{D}'_{m\times m}$, compute n linearly independent eigenvectors $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{Z}_q^m$ of the matrix $\mathbf{H}$ such that $\mathbf{H}\mathbf{p}_i = \lambda_i \mathbf{p}_i$ for each $i \in \{1, 2, \ldots, m\}$. Let $\mathbf{P} = (\mathbf{p}_1\ \mathbf{p}_2\ \ldots\ \mathbf{p}_m)$. Assume that $\mathbf{P}$ is sampled from the distribution $\mathcal{D}'_{m\times m}$. Let

$$G(\mathbf{H}) = \mathbf{P}^{-1}\mathbf{H}\mathbf{P} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \lambda_m \end{bmatrix}$$

It can be easily seen that $G(\mathbf{H})$ is still $\mathbb{Z}_q$-invertible.

III. KEY-AGGREGATE ENCRYPTION SCHEME

Key-aggregate encryption scheme was firstly proposed by Chu *et al.* [5] in 2014. In a key-aggregate encryption scheme, a message is encrypted not only under the data owner's public key but also under an identifier (i.e., an index) of the file which the message belongs to, an aggregate decryption key for a flexibly chosen set of ciphertexts is generated by the master secret key holder (in other words, the data owner) and sent to authorized users through a secure channel such as secure emails or secure devices. In the scheme, any encrypted file whose index is in $\mathcal{S}$ can be decrypted by the authorized user but any encrypted file whose index is outside the set $\mathcal{S}$ remains confidential.

A. KEY-AGGREGATE ENCRYPTION FRAMEWORK

Recall the key-aggregate encryption framework [5] (also see Figure 1) as follows.

- **Setup**$(1^n, N)$: This algorithm is initially run by the data owner or cloud sever. On input of a security parameter 1^n and the number of files N, it outputs the public parameters **params**. For simplicity, the public parameters are omitted from the input of the following four algorithms.
- **Keygen**: This algorithm is run by the data owner to generate one pair (**pk**, **msk**) that consists of public key **pk** and master secret key **msk**.
- **Encrypt**(**pk**, i, b): This algorithm is run by the data owner. On input of a public key **pk**, i denoting the index of the file the message b belongs to, and the message b, it outputs a ciphertext $\mathcal{C}$. Note that the index i and the encrypted file can be sent to the cloud server.
- **Extract**(**pk**, **msk**, $\mathcal{S}$): This algorithm is run by the data owner. On input of the public key **pk**, the master secret key **msk**, and a set $\mathcal{S}$[1] of files' indices, it outputs the aggregate key $\mathbf{K}_\mathcal{S}$. Then the aggregate key $\mathbf{K}_\mathcal{S}$ and the set $\mathcal{S}$ are sent by the data owner to the authorized user.
- **Decrypt**($\mathbf{K}_\mathcal{S}$, $\mathcal{S}$, i, $\mathcal{C}$): This algorithm is run by the authorized user. On input an aggregate key $\mathbf{K}_\mathcal{S}$, a set $\mathcal{S}$ of the files' indices, i denoting the index of the file which the decryption of the ciphertext $\mathcal{C}$ belongs to, and $\mathcal{C}$, it outputs the decrypted result b if $i \in \mathcal{S}$.

Correctness of the above scheme demands that for any integer N, any set $\mathcal{S} \subseteq \{1, 2, \ldots, N\}$, any security parameter 1^n, any index $i \in \mathcal{S}$, and any message b, it holds that

$$\Pr[\mathsf{Decrypt}(\mathbf{K}_\mathcal{S}, \mathcal{S}, i, \mathcal{C}) = b : \mathbf{params} \leftarrow \mathsf{Setup}(1^n, N), (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Keygen}, \mathcal{C} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, i, b), \mathbf{K}_\mathcal{S} \leftarrow \mathsf{Extract}(\mathsf{pk}, \mathsf{msk}, \mathcal{S})] = 1 - negl(1^n, N).$$

B. KEY-AGGREGATE ENCRYPTION SCHEME BASED ON THE LWE PROBLEM

Partially inspired by the construction of hierarchical IBE scheme based on the LWE problem [1], we propose a key-aggregate encryption scheme as follows.

Scheme 1:

- **Setup**$(1^n, N)$: This algorithm takes as input a security parameter 1^n, the number of files N, and derives the public parameters $\mathbf{params} = (N, q, m, n, \widetilde{L}, H, G, \overline{s_R}, \alpha)$ where the prime $q = poly(n)$, $m \geq 2n\log q$, $\widetilde{L} = O(\sqrt{n\log q})$, $s_R = \sqrt{n\log q}\cdot\omega(\sqrt{\log m})$, and $\overline{s_R} = s_R\cdot\sqrt{m}\cdot\omega(\log^{1/2} m) = \sqrt{mn\log q}\cdot\omega(\log m)$.
- **Keygen**: Invoke $\mathsf{GenBasis}(1^n; 1^m; q)$ to output a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a short basis of $\Lambda_q^\perp(\mathbf{A})$, denoted as $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m\times m}$. Generate a uniformly random vector $\mathbf{u}_0 \in \mathbb{Z}_q^n$. Let $\mathsf{pk} = (\mathbf{A}, \mathbf{u}_0)$, and $\mathsf{msk} = \mathbf{T}_\mathbf{A}$.
- **Encrypt**(**pk**, i, b):
  - Let $\mathbf{F}_i = \mathbf{A}[G(H(i))]^{-1} \in \mathbb{Z}_q^{n\times m}$.

[1] In the remainder of the paper, we assume that there exist at least two elements in $\mathcal{S}$ for simplicity.

– For an index $i \in \{1, 2, \ldots, N\}$ and a message bit $b \in \{0, 1\}$, choose a uniformly random vector $\mathbf{s} \xleftarrow{U} \mathbb{Z}_q^n$, pick small noise values $x \xleftarrow{\bar{\Psi}_\alpha} \mathbb{Z}_q$ and $\mathbf{y} \xleftarrow{\bar{\Psi}_\alpha^m} \mathbb{Z}_q^m$. Generate the ciphertext as follows.

$$\mathcal{C} = (\mathbf{c}_0, \mathbf{c}_1) = (\mathbf{u}_0^T\mathbf{s} + x + b\lceil\frac{q}{2}\rceil, \mathbf{F}_i^T\mathbf{s} + \mathbf{y}).$$

- $\mathsf{Extract}(\mathsf{pk}, \mathsf{msk} = \mathbf{T_A}, \mathcal{S})$:
  - Let $\mathbf{R}_\mathcal{S} = \prod_{i\in\mathcal{S}} [G(H(i))] \in \mathbb{Z}^{m\times m}$.
  - Let $\mathbf{F}_\mathcal{S} = \mathbf{A}\mathbf{R}_\mathcal{S}^{-1} \in \mathbb{Z}_q^{n\times m}$.
  - The aggregate key is computed as $\mathbf{K}_\mathcal{S} \leftarrow \mathsf{BasisDel}(\mathbf{A}, \mathbf{R}_\mathcal{S}, \mathbf{T_A}, s_\mathcal{S})$, which is a short random basis of $\Lambda_q^\perp(\mathbf{F}_\mathcal{S})$. (It should be noted that $s_\mathcal{S} > \|\widetilde{\mathbf{T}_\mathbf{A}}\| \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m)$ by Lemma 2.)
- $\mathsf{Decrypt}(\mathbf{K}_\mathcal{S}, \mathcal{S}, i, \mathcal{C} = (\mathbf{c}_0, \mathbf{c}_1))$: If $i \notin \mathcal{S}$, then output $\perp$. Otherwise, the following steps should be taken.
  - Let $\mathbf{R}_\mathcal{S} = \prod_{j\in\mathcal{S}} [G(H(j))]$.
  - Let $\mathbf{F}_\mathcal{S} = \mathbf{A}\mathbf{R}_\mathcal{S}^{-1}$ and $\mathbf{R}_{\mathcal{S}-\{i\}} = \prod_{j\in\mathcal{S}, j\neq i} [G(H(j))]$.
  - Compute $\mathbf{K}_i \leftarrow \mathsf{BasisDel}(\mathbf{F}_\mathcal{S}, \mathbf{R}_{\mathcal{S}-\{i\}}^{-1}, \mathbf{K}_\mathcal{S}, \widetilde{s_{\mathcal{S}-\{i\}}})$. (It should be noted that $\widetilde{s_{\mathcal{S}-\{i\}}} > \|\widetilde{\mathbf{K}_\mathbf{S}}\| \cdot (\overline{s_R})^{|\mathcal{S}|-1} \cdot \omega(\log m)$ according to Lemma 2.)
  - Compute $\mathbf{F}_i = \mathbf{A}[G(H(i))]^{-1}$.
  - Evaluate $\mathbf{d}_i \leftarrow \mathsf{SamplePre}(\mathbf{F}_i, \mathbf{K}_i, \mathbf{u}_0, \tau)$. (It should be noted that $\tau \geq \|\widetilde{\mathbf{K}_i}\| \cdot \omega(\sqrt{\log m})$ by Lemma 3.) Hence, $\mathbf{F}_i\mathbf{d}_i = \mathbf{u}_0$.
  - Compute $b = \mathbf{c}_0 - \mathbf{d}_i^T\mathbf{c}_1$. If $|b| > \frac{q}{4}$, then output 1. Otherwise, output 0.

Remark 1: In the Decrypt algorithm of the above scheme, in order to compute $\mathbf{K}_i \leftarrow \mathsf{BasisDel}(\mathbf{F}_\mathcal{S}, \mathbf{R}_{\mathcal{S}-\{i\}}^{-1}, \mathbf{K}_\mathcal{S}, \widetilde{s_{\mathcal{S}-\{i\}}})$, commutative law between matrix multiplication is needed, but in general matrix multiplication fails to satisfy commutative law. To overcome the difficulties, we introduced a hash function H and function G by using diagonalizable matrices so that the file decryption keys can be derivable from the aggregate key.

C. CORRECTNESS

To keep the correctness of the scheme, the parameters should be as follows.

Let $s_\mathcal{S} \overset{def}{=} \widetilde{L} \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m)$. By Lemma 2 (1), $\|\widetilde{\mathbf{T}_\mathbf{A}}\| \leq \widetilde{L} = O(\sqrt{n\log q})$. Hence, $s_\mathcal{S} \geq \|\widetilde{\mathbf{T}_\mathbf{A}}\| \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m)$. Since $\mathbf{K}_\mathcal{S} \leftarrow \mathsf{BasisDel}(\mathbf{A}, \mathbf{R}_\mathcal{S}, \mathbf{T_A}, s_\mathcal{S})$, we get that $\|\widetilde{\mathbf{K}_\mathcal{S}}\| \leq s_\mathcal{S}\sqrt{m}$ with overwhelming probability by Lemma 2 (2) and (3) where

$$\begin{aligned} s_\mathcal{S}\sqrt{m} &= \widetilde{L} \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m) \cdot \sqrt{m} \\ &= O(n\log q) \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m) \cdot \sqrt{m} \\ &= O(n\log q) \cdot (n\log q)^{\frac{|\mathcal{S}|}{2}} \cdot (\sqrt{m} \cdot \omega(\log m))^{|\mathcal{S}|+1}. \end{aligned}$$

Let $\widetilde{s_{\mathcal{S}-\{i\}}} \overset{def}{=} s_\mathcal{S}\sqrt{m} \cdot (\overline{s_R})^{|\mathcal{S}|-1} \cdot \omega(\log m)$. Since $\mathbf{K}_i \leftarrow \mathsf{BasisDel}(\mathbf{F}_\mathcal{S}, \mathbf{R}_{\mathcal{S}-\{i\}}^{-1}, \mathbf{K}_\mathcal{S}, \widetilde{s_{\mathcal{S}-\{i\}}})$, we get that $\|\widetilde{\mathbf{K}_i}\| \leq \widetilde{s_{\mathcal{S}-\{i\}}}\sqrt{m}$ with overwhelming probability by Lemma 2 (2) and (3).

Let $\tau \overset{def}{=} \widetilde{s_{\mathcal{S}-\{i\}}}\sqrt{m} \cdot \omega(\sqrt{\log m})$. Since $\mathbf{d}_i \leftarrow \mathsf{SamplePre}(\mathbf{F}_i, \mathbf{K}_i, \mathbf{u}_0, \tau)$, we get that $\|\mathbf{d}_i\| \leq \tau\sqrt{m}$ with overwhelming probability by Lemma 3.

Let $n^\delta > \lceil\log q\rceil = O(\log n)$, $m = \lceil 6\, n^{1+\delta}\rceil = O(n\log n)$, and $q = m^5 \cdot \omega(\log^5 n)$. (See Section 4.2 of [1] for the detailed analysis.)

Correctness :

It can be easily seen that

$$\begin{aligned} \mathbf{c}_0 - \mathbf{d}_i^T\mathbf{c}_1 &= \mathbf{u}_0^T\mathbf{s} + x + b\lfloor\frac{q}{2}\rfloor - \mathbf{d}_i^T(\mathbf{F}_i^T\mathbf{s} + \mathbf{y}) \\ &= x + b\lfloor\frac{q}{2}\rfloor - \mathbf{d}_i^T\mathbf{y}. \end{aligned}$$

Since $q \geq 5\tau(m+1)$, $\alpha \leq \frac{1}{\tau\sqrt{m+1}\cdot\omega(\sqrt{\log n})}$, and $\chi = \bar{\Psi}_\alpha^m$, just as discussed in the proof of Theorem 7.1 in [12], with overwhelming probability, we have that $|x - \mathbf{d}_i^T\mathbf{y}| \leq \frac{q}{5}$.

- When $b = 0$, with overwhelming probability we have

$$|\mathbf{c}_0 - \mathbf{d}_i^T\mathbf{c}_1| = |x - \mathbf{d}_i^T\mathbf{y}| < \frac{q}{4}.$$

- When $b = 1$, with overwhelming probability we have

$$|\mathbf{c}_0 - \mathbf{d}_i^T\mathbf{c}_1| \geq \lceil\frac{q}{2}\rceil - |x - \mathbf{d}_i^T\mathbf{y}| > \lceil\frac{q}{2}\rceil - \frac{q}{4} = \frac{q}{4}.$$

D. SECURITY ANALYSIS

Theorem 2: Assume that $\mathcal{A}$ is a PPT adversary that attacks the above scheme, when H is modeled as a random oracle. Let Q_H is the number of H queries made by $\mathcal{A}$ and N be the number of files. Then there is a PPT algorithm $\mathcal{B}$ that decides LWE such that

- *If $\mathcal{A}$ is a selective adversary (**INDr-sID-CPA**) with advantage ϵ, then $\epsilon \leq$ **LWE-adv**$[B]$.*
- *If $\mathcal{A}$ is an adaptive adversary (**INDr-ID-CPA**) with advantage ϵ, then $\epsilon \leq$ **LWE-adv**$[B] \cdot (NQ_H^N) + negl(1^n, N)$ where **LWE-adv**$[B]$ is with respect to the parameters $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$ in Scheme 1.*

Proof: The proof is similar to that of Theorem 5 in [1]. It is omitted for simplicity. □

IV. KEY-AGGREGATE SEARCHABLE ENCRYPTION SCHEME

Consider the case where any group of selected files can be shared with any group of users. In traditional searchable encryption schemes, the desired flexibility requires different encryption keys to be used for different files. However, this implies that it's necessary for the data owner to securely send to authorized users a large quantity of keys for encryption and search if necessary, and it's necessary for the authorized users to securely manage the received keys, and submit an equally large number of keyword trapdoors to the cloud for the purpose of searching over the shared data. It's impractical from the aspects of secure communication, storage, and complexity.

The concept of key-aggregate searchable encryption scheme has been proposed by Cui *et al.* [8]. In the scheme, every keyword is encrypted not only under the public key, but

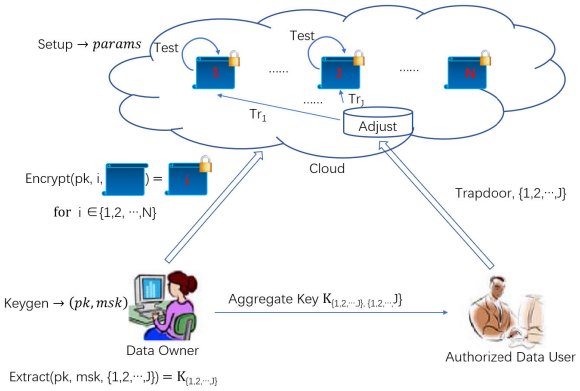


FIGURE 2. Framework of key-aggregate searchable encryption.

also under the identifier of the file, the data owner only needs to generate and send one aggregate key for a large number of shared files to an authorized user, and the authorized user only needs to submit one aggregate trapdoor to the cloud server.

A. KEY-AGGREGATE SEARCHABLE ENCRYPTION SCHEME FRAMEWORK

Recall the key-aggregate searchable encryption scheme framework [8] which consists of seven algorithms (also see Figure 2) as follows.

- $\mathsf{Setup}(1^n, N)$: This algorithm is initially run by the data owner or cloud sever. On input of a security parameter 1^n and the number of files N, it outputs the public parameters **params**. For simplicity, the public parameters are omitted from the input of the following six algorithms.
- Keygen: This algorithm is run by the data owner to produce one pair $(\mathsf{pk}, \mathsf{msk})$ that consists of a public key pk and a master secret key msk.
- $\mathsf{Encrypt}(\mathsf{pk}, i, w)$: This algorithm is run by the data owner. For each file index i and keyword w, it generates the keyword ciphertext $\mathcal{C}_i$.
- $\mathsf{Extract}(\mathsf{pk}, \mathsf{msk}, \mathcal{S})$: This algorithm is run by the data owner. On input of the data owner's public key pk, master secret key msk and a set $\mathcal{S}$ of files' indices, it outputs the aggregate key $\mathbf{K}_{\mathcal{S}}$. Then the aggregate key $\mathbf{K}_{\mathcal{S}}$ and the set $\mathcal{S}$ are sent by the data owner to the authorized user.
- $\mathsf{Trapdoor}(\mathsf{pk}, \mathbf{K}_{\mathcal{S}}, \mathcal{S}, w)$: This algorithm is run by the authorized user. On input of the data owner's public key pk, the aggregate searchable encryption key $\mathbf{K}_{\mathcal{S}}$, the set $\mathcal{S}$ of files' indices, and a keyword w, it outputs an aggregate trapdoor $\mathbf{Tr}$. Then the aggregate trapdoor $\mathbf{Tr}$ and the set $\mathcal{S}$ are sent to the cloud server by the authorized user.
- $\mathsf{Adjust}(i, \mathcal{S}, \mathbf{Tr})$: This algorithm is run by the cloud server. On input of the target file's index i, the set $\mathcal{S}$ of files' indices, and the aggregate trapdoor $\mathbf{Tr}$, it outputs a trapdoor $\mathbf{Tr}_i$ for the target file with index $i \in \mathcal{S}$.
- $\mathsf{Test}(i, \mathcal{S}, \mathbf{Tr}_i, \mathcal{C}_i)$: This algorithm is run by the cloud server. On input of the target file's index i, the set $\mathcal{S}$ of files' indices, the trapdoor $\mathbf{Tr}_i$ for the target file with index $i \in \mathcal{S}$, and the keyword ciphertext $\mathcal{C}_i$, it outputs true (or "1") or false (or "0") to denote whether the file with index i contains the keyword w.

B. KEY-AGGREGATE SEARCHABLE ENCRYPTION SCHEME BASED ON THE LWE PROBLEM

Now we propose a key-aggregate searchable encryption scheme based on the LWE problem as follows. As far as we know, this is the first lattice-based key-aggregate searchable encryption scheme.

Scheme 2:

- $\mathsf{Setup}(1^n, N)$: This algorithm takes as input a security parameter 1^n and the number of files N, and produces the public parameters $\mathbf{params} = (N, q, m, n, H_0, H, G, \widetilde{L}, \overline{s_R}, \alpha)$ where the prime $q = poly(n)$, the dimension $m \geq 2n \log q$, the hash function $H_0 : \{0,1\}^* \to \mathbb{Z}_q^n$, $\widetilde{L} = O(\sqrt{n \log q})$, and $\overline{s_R} = s_R \cdot \sqrt{m} \cdot \omega(\log^{1/2} m) = \sqrt{mn \log q} \cdot \omega(\log m)$.
- Keygen: Invoke $\mathsf{GenBasis}(1^n; 1^m; q)$ to output a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a short basis of $\Lambda_q^{\perp}(\mathbf{A})$, denoted as $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$. Let $\mathsf{pk} = \mathbf{A}$, $\mathsf{msk} = \mathbf{T_A}$.
- $\mathsf{Encrypt}(\mathsf{pk}, i, w)$: Compute $\mathbf{u} = H_0(w)$ and $\mathbf{R}_i = G(H(i))$. Select a uniformly random vector $\mathbf{s} \xleftarrow{U} \mathbb{Z}_q^n$. Choose small noise values $x_i \xleftarrow{\Psi_\alpha} \mathbb{Z}_q$, and $\mathbf{y}_i \xleftarrow{\Psi_\alpha^m} \mathbb{Z}_q^m$. Compute

$$\mathbf{c}_{i1} = \mathbf{u}^T \cdot \mathbf{s} + x_i, \ \mathbf{c}_{i2} = (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s}, \ \mathbf{c}_{i3} = (\mathbf{R}_i^{-1})^T \cdot \mathbf{y}_i.$$

  Output the keyword ciphertext $\mathcal{C}_i = (\mathbf{c}_{i1}, \mathbf{c}_{i2}, \mathbf{c}_{i3})$.
- $\mathsf{Extract}(\mathsf{pk}, \mathsf{msk} = \mathbf{T_A}, \mathcal{S})$:
  - Let $\mathbf{R}_{\mathcal{S}} = \prod_{i \in \mathcal{S}} G(H(i)) \in \mathbb{Z}^{m \times m}$.
  - Let $\mathbf{F}_{\mathcal{S}} = \mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1} \in \mathbb{Z}_q^{n \times m}$.
  - The aggregate key is computed as $\mathbf{K}_{\mathcal{S}} \leftarrow \mathsf{BasisDel}(\mathbf{A}, \mathbf{R}_{\mathcal{S}}, \mathbf{T_A}, s_{\mathcal{S}})$, which is a short random basis of $\Lambda_q^{\perp}(\mathbf{F}_{\mathcal{S}})$. (It should be noted that $s_{\mathcal{S}} > \|\widetilde{\mathbf{T}_\mathbf{A}}\| \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m)$ by Lemma 2.)

  Then $(\mathbf{K}_{\mathcal{S}}, \mathcal{S})$ is sent to the authorized user by the data owner, and hence the keyword search right is delegated to the authorized user.
- $\mathsf{Trapdoor}(\mathsf{pk}, \mathbf{K}_{\mathcal{S}}, \mathcal{S}, w)$:
  - Compute $\mathbf{u} = H_0(w)$.
  - Let $\mathbf{R}_{\mathcal{S}} = \prod_{i \in \mathcal{S}} G(H(i)) \in \mathbb{Z}^{m \times m}$.
  - Let $\mathbf{e} \leftarrow \mathbf{SamplePre}(\mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1}, \mathbf{K}_{\mathcal{S}}, \mathbf{u}, \tau')$. (It should be noted that $\tau' \geq \|\widetilde{\mathbf{K}_{\mathcal{S}}}\| \cdot \omega(\sqrt{\log m})$ by Lemma 3.) Let the trapdoor $\mathbf{Tr} = \mathbf{e}$.

  Then $(\mathbf{Tr}, \mathcal{S})$ is sent to the cloud server by the authorized user.
- $\mathsf{Adjust}(i, \mathcal{S}, \mathbf{Tr})$: Let $\mathbf{R}_j = G(H(j))$ for each $j \in \mathcal{S} - \{i\}$. Compute $\mathbf{Tr}_i = (\prod_{j \in \mathcal{S} - \{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}$.
- $\mathsf{Test}(i, \mathcal{S}, \mathbf{Tr}_i, \mathcal{C}_i)$: Let $\mathbf{R}_j = G(H(j))$ for each $j \in \mathcal{S}$. Compute $\delta_i = \mathbf{c}_{i1} - \mathbf{Tr}_i^T \cdot \mathbf{c}_{i2} - \mathbf{Tr}_i^T \cdot \mathbf{pub} \cdot \mathbf{c}_{i3}$, where

$\mathbf{pub} = (\prod_{j\in\mathcal{S}} \mathbf{R}_j)^T$. (It should be noted that the value **pub** can be computed only once for efficiency.) If $\delta_i < \frac{q}{4}$, it outputs true (or "1"). Otherwise, it outputs false (or "0").

Remark 2: The construction of $\mathsf{Encrypt}(\mathsf{pk}, i, w)$ is tricky. In fact, we have attempted two approaches below to construct the Encrypt algorithm. The values $\mathbf{u}$, $\mathbf{R}_j$ for each $j \in \mathcal{S}$, $\mathbf{s}$, x_i and $\mathbf{y}_i$ are the same as the counterparts in the above construction.

- **Approach 1**: If the ciphertext is defined as $\mathbf{c}_{i1} = \mathbf{u}^T \cdot \mathbf{s} + x_i$ and $\mathbf{c}_{i2} = (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s} + \mathbf{y}_i$, then

$$\begin{aligned}\delta_i &= \mathbf{c}_{i1} - \mathbf{Tr}_i^T \cdot \mathbf{c}_{i2}\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - [(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \\ &\quad \cdot [(\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s} + \mathbf{y}_i]\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - \mathbf{u}^T \cdot \mathbf{s} - [(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot \mathbf{y}_i\\ &= x_i - [(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot \mathbf{y}_i,\end{aligned}$$

  where $\mathbf{e}^T \cdot (\prod_{j\in\mathcal{S}} \mathbf{R}_j^{-1})^T \cdot \mathbf{A}^T = \mathbf{u}^T$.

  It's difficult for us to show an upper bound for $|\delta_i| = |x_i - [(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot \mathbf{y}_i|$.
- **Approach 2**: In order to reduce the length of the ciphertext, we have attempted to define the ciphertext as $\mathcal{C}_i = (\mathbf{c}_{i1}, \mathbf{c}_{i2}) = (\mathbf{u}^T \cdot \mathbf{s} + x_i,\ (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s} + (\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j)^T \cdot \mathbf{y}_i)$. Unfortunately, $\mathcal{S}$ must be introduced as an input of the algorithm Encrypt, which is weaker than the construction model of [8], because the data owner would lose the ability to share any set of files.

Remark 3: In the Test algorithm of Scheme 2, in order to compute $\delta_i = \mathbf{c}_{i1} - \mathbf{Tr}_i^T \cdot \mathbf{c}_{i2} - \mathbf{Tr}_i^T \cdot \mathbf{pub} \cdot \mathbf{c}_{i3}$, commutative law between matrix multiplication is needed, but in general matrix multiplication fails to satisfy commutative law. To overcome the difficulties, we introduced a hash function H and function G by using diagonalizable matrices so that the trapdoor is adjustable and the Test algorithm holds.

C. CORRECTNESS

To keep the correctness of the scheme, the parameters should be as follows.

Let $s_{\mathcal{S}} \overset{def}{=} \widetilde{L} \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m)$. By Lemma 2 (1), $\|\widetilde{\mathbf{T}_\mathbf{A}}\| \le \widetilde{L} = O(\sqrt{n \log q})$. Hence, $s_{\mathcal{S}} \ge \|\widetilde{\mathbf{T}_\mathbf{A}}\| \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m)$. Since $\mathbf{K}_{\mathcal{S}} \leftarrow \mathsf{BasisDel}(\mathbf{A}, \mathbf{R}_{\mathcal{S}}, \mathbf{T}_\mathbf{A}, s_{\mathcal{S}})$, we get that $\|\widetilde{\mathbf{K}_{\mathcal{S}}}\| \le s_{\mathcal{S}}\sqrt{m}$ with overwhelming probability by Lemma 2 (2) and (3), where

$$\begin{aligned}s_{\mathcal{S}}\sqrt{m} &= \widetilde{L} \cdot (\overline{s_R})^{|\mathcal{S}|} \cdot \omega(\log m) \cdot \sqrt{m}\\ &= O(n \log q) \cdot (n \log q)^{\frac{|\mathcal{S}|}{2}} \cdot (\sqrt{m} \cdot \omega(\log m))^{|\mathcal{S}|+1}.\end{aligned}$$

Let $\tau' \overset{def}{=} s_{\mathcal{S}}\sqrt{m} \cdot \omega(\sqrt{\log m})$. Since $\mathbf{e} \leftarrow \mathbf{SamplePre}(\mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1}, \mathbf{K}_{\mathcal{S}}, \mathbf{u}, \tau')$, we get that $\|\mathbf{e}\| \le \tau'\sqrt{m}$ with overwhelming probability by Lemma 3, where

$$\tau' = O(n \log q) \cdot (n \log q)^{\frac{|\mathcal{S}|}{2}} \cdot (\sqrt{m})^{|\mathcal{S}|+1} \cdot (\omega(\log m))^{|\mathcal{S}|+\frac{3}{2}}.$$

Let $n^\delta > \lceil \log q \rceil = O(\log n)$, $m = \lceil 6\, n^{1+\delta} \rceil = O(n \log n)$, and $q = m^5 \cdot \omega(\log^5 n)$. (See Section 4.2 of [1] for the detailed analysis.)

Correctness :

If $(\mathbf{c}_{i1}, \mathbf{c}_{i2}, \mathbf{c}_{i3}) = \mathsf{Encrypt}(\mathsf{pk}, i, w)$, and $\mathbf{Tr}_i = \mathsf{Adjust}(i, \mathcal{S}, \mathbf{Tr})$ where $\mathbf{Tr}$ is the trapdoor generated by the algorithm $\mathsf{Trapdoor}(\mathbf{K}_{\mathcal{S}}, \mathcal{S}, w)$, then

$$\begin{aligned}\delta_i &= \mathbf{c}_{i1} - \mathbf{Tr}_i^T \cdot \mathbf{c}_{i2} - \mathbf{Tr}_i^T \cdot \mathbf{pub} \cdot \mathbf{c}_{i3}\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - [(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s}\\ &\quad - [(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot (\prod_{j\in\mathcal{S}} \mathbf{R}_j)^T \cdot (\mathbf{R}_i^{-1})^T \cdot \mathbf{y}_i\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - \mathbf{e}^T \cdot (\prod_{j\in\mathcal{S}} \mathbf{R}_j^{-1})^T \cdot \mathbf{A}^T \cdot \mathbf{s} - \mathbf{e}^T \cdot \mathbf{y}_i\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - \mathbf{u}^T \cdot \mathbf{s} - \mathbf{e}^T \cdot \mathbf{y}_i\\ &= x_i - \mathbf{e}^T \cdot \mathbf{y}_i.\end{aligned}$$

Since $q \ge 5\tau'(m+1)$, $\alpha \le \frac{1}{\tau'\sqrt{m+1} \cdot \omega(\sqrt{\log n})}$, and $\chi = \bar{\Psi}_\alpha^m$, just like what have been discussed in the proof of Theorem 7.1 in [12], we have that $|\delta_i| < \frac{q}{5}$ with overwhelming probability.

The correctness proof above means that each user knowing the aggregate key can perform a successful keyword search.

D. SECURITY ANALYSIS

In what follows, we will prove several security properties of our scheme.

Theorem 3: Even if the cloud server colludes with a malicious authorized user, a keyword search can not be performed by them over any file which is not in the scope of the authorized user's aggregate key.

Proof: Under the assumption, an attacker $\mathcal{A}$ may possess the knowledge of both a malicious authorized user and a curious cloud server. $\mathcal{A}$ aims to perform keyword search over a file which is not in the scope of his (or her) aggregate key. Assume that the index of the target file is i. Without loss of generality, assume that i is contained in a subset $\mathcal{S}'$ and the attacker $\mathcal{A}$ has the aggregate key with the file index set $\mathcal{S}$. After receiving the trapdoor $\mathbf{Tr} = \mathbf{e}$ where $\mathbf{e} \leftarrow \mathbf{SamplePre}(\mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1}, \mathbf{K}_{\mathcal{S}}, \mathbf{u}, \tau')$, the attacker $\mathcal{A}$ may take the target set $\mathcal{S}'$, where $i \in \mathcal{S}'$ but $i \notin \mathcal{S}$, as the input of the Adjust algorithm in order to generate the adjusted trapdoor. Correspondingly, assume that $\mathbf{Tr}_i = (\prod_{j\in\mathcal{S}'-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}$. We discuss in two cases below.

- If the value **pub** is generated by the wrong set $\mathcal{S}'$, then

$$\begin{aligned}\delta_i &= \mathbf{c}_{i1} - \mathbf{Tr}_i^T \cdot \mathbf{c}_{i2} - \mathbf{Tr}_i^T \cdot \mathbf{pub} \cdot \mathbf{c}_{i3}\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - [(\prod_{j\in\mathcal{S}'-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s}\end{aligned}$$

$$- [(\prod_{j\in\mathcal{S}'-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot (\prod_{j\in\mathcal{S}'} \mathbf{R}_j)^T \cdot (\mathbf{R}_i^{-1})^T \cdot \mathbf{y}_i$$
$$= \mathbf{u}^T \cdot \mathbf{s} + x_i - \mathbf{e}^T \cdot (\prod_{j\in\mathcal{S}'} \mathbf{R}_j^{-1})^T \cdot \mathbf{A}^T \cdot \mathbf{s} - \mathbf{e}^T \cdot \mathbf{y}_i$$

- If the value **pub** is generated by the original set $\mathcal{S}$, then

$$\begin{aligned}\delta_i &= \mathbf{c}_{i1} - \mathbf{Tr}_i^T \cdot \mathbf{c}_{i2} - \mathbf{Tr}_i^T \cdot \mathbf{pub} \cdot \mathbf{c}_{i3}\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - [(\prod_{j\in\mathcal{S}'-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s}\\ &\quad - [(\prod_{j\in\mathcal{S}'-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}]^T \cdot (\prod_{j\in\mathcal{S}} \mathbf{R}_j)^T \cdot (\mathbf{R}_i^{-1})^T \cdot \mathbf{y}_i\\ &= \mathbf{u}^T \cdot \mathbf{s} + x_i - \mathbf{e}^T \cdot (\prod_{j\in\mathcal{S}'} \mathbf{R}_j^{-1})^T \cdot \mathbf{A}^T \cdot \mathbf{s} - \mathbf{e}^T \cdot\\ &\quad (\prod_{j\in\mathcal{S}'-\{i\}} \mathbf{R}_j^{-1})^T \cdot (\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1})^T \cdot \mathbf{y}_i\end{aligned}$$

Since $\mathbf{e} \leftarrow \mathbf{SamplePre}(\mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1}, \mathbf{K}_{\mathcal{S}}, \mathbf{u}, \tau')$, it can be easily seen that $(\mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1})\cdot\mathbf{e} = \mathbf{u}$, that is, $(\mathbf{A}\prod_{j\in\mathcal{S}} \mathbf{R}_j^{-1})\cdot\mathbf{e} = \mathbf{u}$. Since the attacker $\mathcal{A}$ does not have the aggregate key of the file index set $\mathcal{S}'$, we get that $\mathbf{R}_{\mathcal{S}} \neq \mathbf{R}_{\mathcal{S}'}$ from the **Extract** algorithm. (It can be easily seen by reduction to absurdity.) Correspondingly, $\prod_{j\in\mathcal{S}} \mathbf{R}_j^{-1} \neq \prod_{j\in\mathcal{S}'} \mathbf{R}_j^{-1}$.

Similar to the analysis of Section 3.1 in [11], from the randomness of $\mathbf{u}$, $\mathbf{e}$, $\mathbf{s}$, and $\mathbf{R}_j$ for each $j \in \mathcal{S}'$, it can be seen that $\mathbf{u}^T \cdot \mathbf{s} - \mathbf{e}^T \cdot (\prod_{j\in\mathcal{S}'} \mathbf{R}_j^{-1})^T \cdot \mathbf{A}^T \cdot \mathbf{s}$ is a random vector in $\mathbb{Z}_q$. Namely, $|\delta_i|$ is random over $[0, q]$. Correspondingly, the **Test** algorithm will output false (or "0") with probability $\frac{3}{4}$ for any file with index $i \notin \mathcal{S}$. □

The following theorem says that even if the attacker has obtained an aggregate key for a set of files, given another set of files, the attacker can't generate a new aggregate key.

Theorem 4: A new aggregate key can not be produced by the attacker for any new set of files from the known aggregate key.

Proof: Assume that the attacker $\mathcal{A}$ has an aggregated key $\mathbf{K}_{\mathcal{S}}$ with the index set $\mathcal{S}$ of the files from a data owner. $\mathcal{A}$ may try to generate a new aggregate key for a new set $\mathcal{S}'$ of files of the same data owner. In order to generate the new key, from the **Extract** algorithm, the owner's master secret key $\mathbf{T_A}$ should be obtained. Thus $\mathcal{A}$ is unable to produce the new aggregate key $\mathbf{K}_{\mathcal{S}'}$. □

Assume that the cloud server is curious and the authorized users are malicious. The cloud server knows the stored keyword ciphertexts, the submitted trapdoor, etc. The malicious authorized user may own an aggregate key $\mathbf{K}_{\mathcal{S}}$, and he (or she) can perform keyword search over a set $\mathcal{S}$ of files. We propose the following theorem.

Theorem 5: Query privacy can be achieved by our key-aggregate searchable encryption scheme.

Proof: Our scheme can be proved to achieve query privacy from three claims below.

Claim 1: The keyword in a query can not be determined by an attacker from the submitted trapdoor.

Proof: Assume that an attacker $\mathcal{A}$ knows the submitted trapdoor $\mathbf{Tr} = \mathbf{e}$ and the file set $\mathcal{S}$, where $\mathbf{e} \leftarrow \mathbf{SamplePre}(\mathbf{A}\mathbf{R}_{\mathcal{S}}^{-1}, \mathbf{K}_{\mathcal{S}}, H(w), \tau')$, and tries to determine the keyword in a query. $\mathcal{A}$ may succeed only when the aggregate key $\mathbf{K}_{\mathcal{S}}$ can be guessed by $\mathcal{A}$. In order to compute $\mathbf{K}_{\mathcal{S}}$, the attacker must obtain the data owner's master secret key $\mathbf{T_A}$, which is kept secret. Thus, the attacker can't achieve this goal. □

Claim 2: The keyword in a file can not be determined by an attacker from the related public information and the stored keyword ciphertext.

Proof: Assume that an attacker $\mathcal{A}$ may try to learn the keyword with the knowledge of the public parameters $\mathbf{params} = (N, q, m, n, H_0, H, G, \widetilde{L}, \overline{s_R}, \alpha)$, the public key $\mathsf{pk} = \mathbf{A}$, and the stored keyword ciphertext $\mathcal{C}_i = (\mathbf{c}_{i1}, \mathbf{c}_{i2}, \mathbf{c}_{i3}) = (\mathbf{u}^T \cdot \mathbf{s} + x_i, (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s}, (\mathbf{R}_i^{-1})^T \cdot \mathbf{y}_i)$. $\mathcal{A}$ may try to do two kinds of attacks below.

- The attacker $\mathcal{A}$ may try to obtain the value $\mathbf{s}$ from the known $\mathbf{c}_{i2} = (\mathbf{A}\mathbf{R}_i^{-1})^T \cdot \mathbf{s}$ and $\mathbf{c}_{i3} = (\mathbf{R}_i^{-1})^T \cdot \mathbf{y}_i$. However, as $\mathbf{A}$, $\mathbf{s}$ and $\mathbf{R}_i = H(i)$ for each i are all uniformly random in the random oracle model, $\mathcal{A}$ can't compute the value $\mathbf{s}$.
- The attacker $\mathcal{A}$ may try to compute $H_0(w)$, because when $\mathcal{A}$ obtains the value $H_0(w)$, $\mathcal{A}$ will determine whether the keyword is in the file. Note that $\mathbf{u}$ is uniformly random in the random oracle model. From the LWE assumption, $\mathbf{c}_{i1}$ is uniformly random over $\mathbb{Z}_q$. Thus, the attacker $\mathcal{A}$ can't achieve this goal.

In brief, the attacker can't determine a keyword from the related public information and the stored keyword ciphertext. □

Claim 3: The keyword in a query can not be determined by an attacker from the **Adjust** *algorithm.*

Proof: Recall that the output of the **Adjust** algorithm is $\mathbf{Tr}_i = (\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1}) \cdot \mathbf{e}$, which is obtained by multiplying the trapdoor $\mathbf{Tr} = \mathbf{e}$ by $(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1})$. Since $(\prod_{j\in\mathcal{S}-\{i\}} \mathbf{R}_j^{-1})$ can be easily computed by the attacker, the algorithm provides no help for the an attacker $\mathcal{A}$ to determine a keyword from the trapdoor. Combining the above fact with Claim 1, we get the result. □

□

V. PERFORMANCE ANALYSIS

In cloud storage systems, the main factors that affect the performance of the key-aggregate encryption scheme and the key-aggregate searchable encryption scheme are communication overhead and computational complexity [16]. In this section, we provide performance analysis of our schemes and other related ones.

A. COMMUNICATION OVERHEAD

For the key-aggregate encryption scheme, the communication overhead mainly depends on the ciphertext size and the

TABLE 1. Communication overhead comparison of key-aggregate encryption schemes.

Schemes	[CCT+14]	Scheme 1
Ciphertext size	$2L_{\mathbb{G}} + L_{\mathbb{G}_T}$	$(m+1)\log q$
Aggregate key size	$L_{\mathbb{G}}$	$s_{\mathcal{S}}\sqrt{m}$
Hardness Assumption	BDHE	LWE

TABLE 2. Communication overhead comparison of key-aggregate searchable encryption schemes.

Schemes	[CLW16]	Scheme 2
Ciphertext size	$2L_{\mathbb{G}} + L_{\mathbb{G}_T}$	$(2m+1)\log q$
Aggregate key size	$L_{\mathbb{G}}$	$s_{\mathcal{S}}\sqrt{m}$
Trapdoor size	$L_{\mathbb{G}}$	$\tau'\sqrt{m}$
Hardness Assumption	BDHE	LWE

aggregate key size. For the key-aggregate searchable encryption scheme, the communication overhead mainly relies on the ciphertext size, the aggregate key size, and the trapdoor size.

For simplicity, we specify some notations as follows. Assume that the bilinear map in [5], [8] is uniformly defined as $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$. Denote $L_{\mathbb{G}}$ as the bit-length of an element in the group $\mathbb{G}$. Denote $L_{\mathbb{G}_T}$ as the bit size of an element in group $\mathbb{G}_T$. The upper bound of the Gram-Schmidt norm of the aggregate key is used to represent the size of the aggregate key. Additionally, the aggregate key size and the trapdoor size of our schemes shown in Tables 1 and 2 hold with overwhelming probability by the analysis in Subsection III-C and Subsection IV-C.

The communication overhead comparison between our key-aggregate encryption scheme and that of [5] is listed in Table 1. The communication overhead comparison between our key-aggregate searchable encryption scheme and that of [8] is listed in Table 2. Assume that $L_{\mathbb{G}} = 512$, $L_{\mathbb{G}_T} = 1024$, $m = 371$, $n = 23$, and $q = 257$. Then the ciphertext size of the scheme in [5] is 2048 bits, while the ciphertext size of Scheme 1 here is approximately 2976 bits. The ciphertext size of the scheme in [8] is 2048 bits, while the ciphertext size of Scheme 2 here is approximately 5944 bits. Thus the communication overhead of our schemes are comparable to the ones in prior work. Moreover, our schemes are designed based on the LWE assumption, for which no quantum algorithms are currently known, but previous constructions are secure under the hardness of the BDHE problem, which can be solved by efficient quantum algorithms [25].

B. COMPUTATIONAL COMPLEXITY

Now we analyze the computational complexity of our key-aggregate encryption scheme and key-aggregate searchable encryption scheme. In practice, for a sufficiently large number of files with indices set $\widetilde{\mathcal{S}}$ and each $i \in \widetilde{\mathcal{S}}$, $G(H(i))$ can be efficiently pre-computed and stored in a database for direct invocation in the two schemes.

TABLE 3. Computational complexity of Scheme 1.

Algorithms	Computational complexity
Keygen	probabilistic polynomial-time
Encrypt	$O(n^2 \log n)$
Extract	efficient
Decrypt	efficient

1) COMPUTATIONAL COMPLEXITY OF SCHEME 1

For the key-aggregate encryption scheme, the computational complexity mainly depends on the computational costs of the **Keygen**, **Encrypt**, **Extract**, and **Decrypt** algorithms, which are analyzed respectively as follows.

- By Lemma 2, we get that **GenBasis**$(1^n; 1^m; q)$ is a probabilistic polynomial-time algorithm. Therefore, **Keygen** is run in probabilistic polynomial-time.
- Since a) $G(H(i))^{-1}$ is a diagonal matrix from the fact that $G(H(i))$ is a diagonal matrix, b) the computational complexity of multiplication between a $n \times m$ matrix and a diagonal $m \times m$ matrix is $O(mn)$, c) the operations in the **Encrypt** algorithm includes the inverse of a diagonal matrix, multiplication between a matrix and a diagonal matrix, matrix-vector multiplication, and vector addition, in which multiplication between a matrix and a diagonal matrix is the most time-consuming operation, we conclude that the encryption time is $O(mn)$ per message bit, that is, $O(n^2 \log n)$ per message bit (by the fact that $m = \lceil 6\, n^{1+\delta} \rceil = O(n \log n)$ from the Subsection III-C). In comparison, the encryption time of the identity-based encryption scheme proposed by Bohen et al. [3] is $O(n^4)$ per message bit (in which n is the size of the master public modulus $N = pq$).
- Since a) the computational complexity of the first two steps in the **Extract** algorithm is $O(|\mathcal{S}|m + mn)$ by the fact that $G(H(i))$ is a diagonal matrix for each $i \in \mathcal{S}$, and b) **Basisdel**$(\mathbf{A}, \mathbf{R}_{\mathcal{S}}, \mathbf{T}_{\mathbf{A}}, s_{\mathcal{S}})$ is an efficient algorithm by Lemma 2 (2), we get that the **Extract** algorithm is efficient.
- Since a) **Basisdel**$(\mathbf{F}_{\mathcal{S}}, \mathbf{R}^{-1}_{\mathcal{S}-\{i\}}, \mathbf{K}_{\mathcal{S}}, \widetilde{s_{\mathcal{S}-\{i\}}})$ is an efficient algorithm by Lemma 2 (2), b) **SamplePre**$(\mathbf{F}_i, \mathbf{K}_i, \mathbf{u}_0, \tau)$ is a probabilistic polynomial-time algorithm by Lemma 3, c) the computational complexity of the remaining four steps in the **Decrypt** algorithm is $O(|\mathcal{S}|m + 2m + mn)$, we get that the **Decrypt** algorithm is efficient.

In summary, the computational complexity of the algorithms in our key-aggregate encryption scheme is as follows (see Table 3).

2) COMPUTATIONAL COMPLEXITY OF SCHEME 2

For the key-aggregate searchable encryption scheme, the computational complexity mainly includes the computational

TABLE 4. Computational complexity of Scheme 2.

Algorithms	Computational complexity
Keygen	probabilistic polynomial-time
Encrypt	$O(n^2 \log n)$
Extract	efficient
Trapdoor	probabilistic polynomial-time
Adjust	$O(\|\mathcal{S}\| \cdot n \log n)$
Test	$O(n \log n)$

costs of Keygen, Encrypt, Extract, Trapdoor, Adjust, and Test algorithms. The hash function H_0 can be pre-computed and stored in a database in the scheme. Similar to the analysis in the first scheme, we get the computational complexity of the algorithms in our key-aggregate searchable encryption scheme as follows (see Table 4).

VI. APPLICATION TO CLOUD STORAGE

In this section, we study how our schemes can be applied to cloud storage for searchable group data sharing by combining the key-aggregate encryption scheme with the key-aggregate searchable encryption scheme.

Assume that the Setup algorithms in the two schemes are run by the public cloud server (e.g., Dropbox), and all authorized users have registered. To upload a file with index i, the data owner runs Encrypt in Scheme 1 to encrypt the file and Encrypt in Scheme 2 to encrypt the keywords. Without loss of generality, assume that the file ciphertext is $\mathcal{C}_f$ and the keyword ciphertext is $\mathcal{C}'$. The data owner uploads $(i, \mathcal{C}, \mathcal{C}')$ to the cloud. In order to share some files having indices set $\mathcal{S}$ with an authorized user, the data owner sends a first aggregate key generated in Scheme 1 and a second aggregate key generated in Scheme 2 along with $\mathcal{S}$ through a secure channel to the authorized user. In order to retrieve the files including a target keyword, the authorized user runs the Trapdoor algorithm by using the second aggregate key to output the keyword trapdoor and sends it with $\mathcal{S}$ to the cloud server. Then the cloud server will run the Adjust algorithm to get the trapdoor for each file whose index is in $\mathcal{S}$ and run the Test algorithm to do keyword search. Afterwards, the cloud server will return the encrypted files which includes the target keyword to the authorized user. When getting the encrypted files, the authorized user will run the Decrypt algorithm to decrypt the files by using the first aggregate key.

VII. CONCLUSION

In this paper, based on the Learning with Errors problem, we designed a key-aggregate encryption scheme and a key-aggregate searchable encryption scheme for the first time. We proved their correctness, and presented formal security analysis and performance analysis, which confirm that our schemes are provably secure and practically efficient. We also studied their application to cloud storage for searchable group data sharing by combining the two schemes together. Our lattice-based schemes have some advantages such as their security against quantum computing attacks, average-case to worst-case equivalence as well as simplicity and potential efficiency. However, in our construction of key-aggregate encryption scheme, only one bit message is encrypted at a time; in our construction of key-aggregate searchable encryption scheme, if an authorized user wants to query over files shared by multiple owners, multiple trapdoors should be sent to the cloud server. In future, for the first construction, we will attempt to extend single-bit encryption to multi-bit encryption; for the second construction, we will attempt to reduce the number of trapdoors in the multi-owners setting. Moreover, we will try to design the above schemes which are leakage-resilient in future.

REFERENCES

[1] S. Agrawal, D. Boneh, and X. Boyen, "Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE," in *Proc. CRYPTO*, 2010, pp. 98–115.
[2] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," *Theory Comput. Syst.*, vol. 48, pp. 535–553, Apr. 2011.
[3] D. Boneh, C. Gentry, and M. Hamburg, "Space-efficient identity based encryptionwithout pairings," in *Proc. FOCS*, 2007, pp. 647–657.
[4] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Proc. CRYPTO*, vol. 3621, 2005, pp. 258–275.
[5] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 468–477, Feb. 2014.
[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[7] X. Chen, J. Li, X. Huang, J. Li, Y. Xiang, and D. S. Wong, "Secure outsourced attribute-based signatures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3285–3294, Dec. 2014.
[8] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2374–2385, Aug. 2016.
[9] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Secur.*, vol. 19, no. 3, pp. 367–397, 2011.
[10] H. Du, Q. Wen, and S. Zhang, "An efficient certificateless aggregate signature scheme without pairings for healthcare wireless sensor network," *IEEE Access*, vol. 7, pp. 42683–42693, 2019.
[11] C. Gu, Y. Guang, Y. Zhu, and Y. Zheng, "Public key encryption with keyword search from lattices," *Int. J. Inf. Technol.*, vol. 19, no. 1, pp. 1–10, 2013.
[12] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. STOC*, 2008, pp. 197–206.
[13] C. Gentry, A. O'Neill, and L. Reyzin, "A unified framework for trapdoor-permutation-based sequential aggregate signatures," in *Proc. PKC PTII*, 2018, pp. 34–57.
[14] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2014.
[15] Accessed: Aug. 5, 2019. [Online]. Available: https://wenku.baidu.com/view/6039a0b231b765ce050814f6.html?from=search&from_page=view&from_mod=download
[16] J. Li, C. F. Jia, Z. Liu, J. Li, and M. Li, "Survey on the searchable encryption," *J. Softw.*, vol. 26, no. 1, pp. 109–128, 2015.
[17] J. Li and K. Kim, "Hidden attribute-based signatures without anonymity revocation," *Inf. Sci.*, vol. 180, no. 9, pp. 1681–1689, 2010.
[18] J. Li, J. Li, X. Chen, C. Jia, and Z. Liu, "Efficient keyword search over encrypted data with fine-grained access control in hybrid cloud," in *Proc. 6th Int. Conf. Netw. Syst. Secur.*, 2012, pp. 490–502.
[19] Z. Liu, Z. Wang, X. Cheng, C. Jia, and K. Yuan, "Multi-user searchable encryption with coarser-grained access control in hybrid cloud," in *Proc. 4th Int. Conf. Emerg. Intell. Data Web Technol.*, 2013, pp. 249–255.
[20] X. Lu, W. Yin, Q. Wen, Z. Jin, and W. Li, "A lattice-based unordered aggregate signature scheme based on the intersection method," *IEEE Access*, vol. 6, pp. 33986–33994, 2018.

[21] M. Noroozi and Z. Eslami, "Public-key encryption with keyword search: A generic construction secure against online and offline keyword guessing attacks," *J. Ambient Intell. Humanized Comput.*, 2019, doi: 10.1007/s12652-019-01254-w.
[22] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," Cryptol. ePrint Arch., Tech. Rep. 2013/508, 2013. Accessed: Aug. 5, 2019. [Online]. Available: https://eprint.iacr.org/2013/508.pdf
[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 34-1–34-40, 2009.
[24] P. V. Reddy and P. Gopal, "Identity-based key-insulated aggregate signature scheme," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, pp. 303–310, Jul. 2017.
[25] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," in *Proc. FOCS*, 1994, pp. 124–134.
[26] E. F. Wang and S. M. Shi, *Advanced Algebra (Third Version)*. Beijing, China: Higher Education Press, vol. 2013, pp. 290–301.
[27] D. Wu, X. Wang, and Q. Gan, "Public key encryption with keyword search from lattices in multiuser environments," *Math. Problems Eng.*, vol. 2016, Oct. 2016, Art. no. 6549570.
[28] F. Zhao, T. Nishide, and K. Sakurai, "Multi-user keyword search scheme for secure data sharing with fine-grained access control," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2012, pp. 406–418.
[29] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Inf. Sci.*, vol. 494, pp. 193–207, Aug. 2019.
[30] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial Internet of Things," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2914117.
[31] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Inf. Sci.*, vol. 472, pp. 223–234, Jan. 2019.



YANQING YAO received the Ph.D. degree in computer science from Beihang University, China, in 2015. From 2012 to 2014, she was a Visiting Student with New York University, USA. She is currently an Assistant Professor with Beihang University. Her research interests include cryptography and information security.



ZHENGDE ZHAI received the Ph.D. degree in information security from the Institute of Software of software, Chinese Academy of Sciences, China, in 2008. He is currently a Security Expert with the Ant Financial Services Group. His research interest includes system security.



JIANWEI LIU received the Ph.D. degree in communication engineering from Xidian University, China, in 1998. He is currently a Professor and the Dean of School of Cyber Science and Technology, Beihang University. His current research interests include cryptographic protocol design, security on wireless and mobile networks, computer network security, and cryptography.



ZHOUJUN LI received the B.S. degree in computer science from Wuhan University, China, in 1984, and the M.S. and Ph.D. degrees in computer science from the National University of Defense Technology, China, in 1986 and 1999, respectively. He is currently a Professor with Beihang University and a part-time Professor with Shenzhen University. His research interest includes information security.