Efficient Multi-keyword Searchable Encryption Based on Multi-input Inner-Product Functional Encryption

Yunong Liang, Zhenfu Cao(✉), Xiaolei Dong, and Jiachen Shen(✉)

Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
51164500072@stu.ecnu.edu.cn, {zfcao,dongxiaolei,jcshen}@sei.ecnu.edu.cn

Abstract. With highly development of cloud computing, data owners wish to outsource their data to clouds for computational and storage resource at a lower price. In order to protect the privacy of sensitive information, they should be encrypted before being uploaded to the cloud server. However, in this way, it is hard to find data in encrypted form according to search criticisms. To solve this problem, searchable encryption has merged. In this paper, we propose a secure and efficient searchable encryption scheme supporting multi-keyword search in 1-to-n setting. The scheme is mainly applicable to the scenes that the number of keywords is limited but the number of files is huge such as sharing a comprehensive knowledge base of a certain field. By tactfully leveraging multi-input inner-product functional encryption, the cloud server is able to complete search processes with search tokens which consist of only two items. It reduces communication and transportation overhead significantly. By using an inverted index structure and super-incremental sequence, our scheme achieves efficient multi-keyword search. In addition, our scheme avoids per-query interaction between the data owner and data users. That is to say, the data owner does not need to stay online waiting for data users to search in his archives. On the other hand, the scheme also achieves partial token privacy, index privacy and token privacy at the same time.

Keywords: Searchable encryption · Multi-keyword · Multi-user
Index privacy · Token privacy

1 Introduction

1.1 Background

As we all know, big data has three outstanding features: large volume, high velocity and high variety. Cloud storage is well designed for big data because of its excellent capability to store large volumes of data, to prepare for high velocity of data generation and to process high variety of data. Cloud computing provides great convenience to users and is one of the most popular technologies at present.

D. Naccache et al. (Eds.): ICICS 2018, LNCS 11149, pp. 377–392, 2018.
https://doi.org/10.1007/978-3-030-01950-1_22

Meanwhile, cloud computing (data outsourcing) raises confidentiality and privacy concerns. Simple encryption can protect data confidentiality easily. However, when data users want to search using some specific keyword to get documents of their interest among massive volumes of data, this becomes a new challenge. In order to search by a particular keyword, the data owner has to decrypt the data first before starting the searching process. It is obviously not practical especially when the volume of data is large. Searchable encryption (SE) [5–7,10,11,15,20] is a cryptographic primitive to address search over ciphertexts. SE allows data users retrieve documents from the cloud server according to some keywords. Searchable encryption has been studied intensively and a mass of keyword search schemes over encrypted cloud data have been proposed.

In the cloud environment, a data owner usually shares his documents with data users. In this paper, we focus on the single-owner/multi-user setting. In this setting, when a data user wants to search over the data owner's documents, he usually needs to ask data owner to produce necessary trapdoor information to help him complete the search. This is to say, the data owner must be online all the time to perform the per-query interaction with data users. However, the primary goal of data owner is to outsource his search services to the cloud server, so we remove the per-query interaction between the data owner and data users in our scheme.

In some practical setting, search with only one keyword may obtain a great quantity of documents and obviously it lacks search precision. Especially when the queried keyword does not accurately describe the documents that data user wants to get. Thus, multi-keyword searchable encryption merged. That is to say, data users can do the research with multi keywords, namely, conjunctive keyword search. Thus, data users can get the documents including all the queried keywords.

Most existing SE schemes assume that the cloud server is honest-but-curious. That is to say, the cloud server may try to find out which keyword the ciphertext is about. Besides, it may also be curious about which keywords data user wants to search. So it is necessary to ensure index privacy and token privacy.

In most literatures, when data users want to search with specific keywords to get their target documents, they have to compute a search token including quite a few items and send it to the cloud server to complete the search process. It may consume a lot of bandwidth and the computing overhead is huge for data users. However, the search token in our scheme consists of only two items by using multi-input inner-product functional encryption. Furthermore, our scheme guarantees index privacy and token privacy simultaneously.

1.2 Design Goal

In this paper, we propose a secure and efficient single-owner/multi-user searchable encryption scheme supporting multi-keyword search. Our design goal is summarized as follows:

1. Our scheme avoids the per-query interaction between the data owner and data users. Namely, the data owner does not need to stay online waiting for data users to search in his archives.
2. By tactfully leveraging multi-input inner-product functional encryption, our scheme allows the cloud server to complete search processes with search tokens which consist of only two items.
3. By using an inverted index structure and super-incremental sequence, our scheme achieves efficient multi-keyword search.
4. Our scheme ensures the correctness of search phase.
5. Our scheme achieves partial token privacy, index privacy and token privacy at the same time.

1.3 Organization

The structure of this paper is as follows:

In Sect. 2, we introduce some related work. Section 3 gives some preliminaries. Then we propose our system model and describe our scheme in detail in Sect. 4 and Sect. 5 respectively. In Sect. 6, we show the correctness and security of our scheme and analyze function and efficiency of our scheme by comparing with other schemes in Sect. 7. In the last section, we summarize this paper.

2 Related Work

2.1 Index

The index structures have an effect on assisting to perfect the scheme. Different index structures have different advantages and disadvantages. Curtmola et al. proposed a searchable encryption scheme in literature [6] based on the inverted index because of its efficiency. Although inverted index structure is efficient on searching, it is not convenient on updating the files. Goh et al. [11] proposed an index structure based on bloom filter. While Chang et al. proposed a vector index in [5].

2.2 Searchable Encryption

The first searchable encryption is proposed by Song et al. [20] in the symmetric key setting. The security notion of searchable encryption was first introduced by Goh [11]. And then, Curtmola et al. [6] presented a stronger security notion, indistinguishability against adaptive chosen-keyword attacks (IND-CKA2). Boneh et al. [7] designed the first searchable encryption with keyword search in the public key setting, but its search efficiency is not fast comparing to the symmetric searchable encryption. All the scheme mentioned above only support single-keyword search.

However, in some settings, a single-keyword may not describe the search precision correctly. Therefore, multi-keyword searchable encryption [2,4,8,12–14,16–19,21,22] has received increasing attention.

Golle et al. [12] first proposed the construction of conjunctive keyword searchable encryption in the single-owner/single-user setting and presented two schemes. In the first scheme, the size of search token is linear with the number of encrypted documents. In the second scheme, the size of search token is constant by using bilinear parings while the computational cost is still not low. In the literatures [14,16,19], conjunctive and disjunctive keyword search are proposed, which make the multi-keyword search semantics get a further extension. Cash et al. [4] proposed the first sublinear symmetric searchable encryption support boolean queries in single-owner/single-user setting and implemented it in a large database [3]. Jarecki et al. [13] extends it to single-owner/multi-user setting, which data owner needs to be online all the time. Besides, the search time mainly depends on the number of files including the least frequency keywords among keywords data user wants to search. That is to say, search efficiency is not high when the keywords data user queries are all high frequency ones.

3 Preliminary

3.1 Notation

We use $s \xleftarrow{R} S$ to denote the operation of uniformly sampling a random element s from a set S. We use PPT to denote a probabilistic polynomial-time algorithm. λ represents the security parameter in this paper. We use lower case boldface italics to denote (column) vectors and upper case boldface italics to denote matrices. For a matrix $\boldsymbol{M}$ over $\mathbb{Z}_q$, we have $[\boldsymbol{M}]_1 := g_1^{\boldsymbol{M}}$ and $[\boldsymbol{M}]_2 := g_2^{\boldsymbol{M}}$, where exponentiation is carried out component-wise. Besides, we use $[n]$ to denote integers no more than n and we use $<\boldsymbol{x}, \boldsymbol{y}>$ to denote inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are column vectors with same dimension.

3.2 Asymmetric Bilinear Groups

Let $\mathcal{PG}$ denote a group generator – an algorithm which takes a security parameter λ as input and outputs a description of prime order groups $\mathbb{G}_1$,$\mathbb{G}_2$,$\mathbb{G}_T$ with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We define $\mathcal{PG}$'s output as $(q, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where q is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order q. g_1, g_2, g_T are the generator of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a map with the following properties:

(1) Bilinearity: $\forall a, b \in Z_q$, $e(g_1^a, g_1^b) = e(g_1, g_2)^{ab}$
(2) Non-degeneracy: $e(g_1, g_2) \neq 1$
(3) Computability: $\forall u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, $e(u, v)$ can be efficiently computed.

3.3 Multi-input Inner-Product Encryption

In inner-product encryption scheme, upon receiving the ciphertext of a vector $\boldsymbol{x}$, only the recipients who have the secret key $k_{\boldsymbol{y}}$ can obtain the inner product $<\boldsymbol{x}, \boldsymbol{y}>$ of $\boldsymbol{x}$ and $\boldsymbol{y}$. While in multi-input inner-product encryption,

only the recipients who have the secret key $k_{\boldsymbol{y}_1,\boldsymbol{y}_2,...\boldsymbol{y}_n}$ and ciphertexts of vector $\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_n$ can obtain the sum of inner product $<\boldsymbol{x}_i,\boldsymbol{y}_i>$, namely, $\sum_{i=1}^{n}<\boldsymbol{x}_i,\boldsymbol{y}_i>$.

We will use the definition of Matrix Decision Diffie-Hellman (MDDH) Assumption in [9].

3.4 Matrix Distribution

Let $k, l \in N$, with $l > k$, we call $\mathcal{D}_{l,k}$ a matrix distribution if it outputs matrices in $Z_q^{l\times k}$ of full rank k in polynomial time. We write $\mathcal{D}_k := \mathcal{D}_{k+1,k}$. Without loss of generality, we assume the first k rows of $\mathbf{A} \xleftarrow{R} \mathcal{D}_{l,k}$ form an invertible matrix. Particularly, we use $\mathcal{U}_{l,k}$ to denote the uniform distribution. $\mathcal{U}_k$ stands for $\mathcal{U}_{k+1,k}$. In this work, we are mostly interested in the uniform matrix distribution $\mathcal{U}_{l,k}$.

3.5 $\mathcal{D}_{l,k}$-Matrix Diffie-Hellman Assumption $\mathcal{D}_{l,k}$-MDDH

Let $\mathcal{D}_{l,k}$ be a matrix distribution. We say that the $\mathcal{D}_{l,k}$-Matrix Diffie-Hellman ($\mathcal{D}_{l,k}$-MDDH) Assumption relative to $\mathcal{PG}$ in $\mathbb{G}_s$ holds if for all PPT adversaries $\mathcal{A}$, there is no non-negligible function Adv. Namely $Adv_{\mathbb{G}_s,\mathcal{A}}^{\mathcal{D}_{l,k}-MDDH} = |Pr[\mathcal{A}(\mathcal{PG},[\boldsymbol{A}]_s,[\boldsymbol{Aw}]_s)=1] - Pr[\mathcal{A}(\mathcal{PG},[\boldsymbol{A}]_s,[\boldsymbol{u}]_s)=1]| = negl(\lambda)$, where the probability is taken over $\boldsymbol{A} \xleftarrow{R} \mathcal{D}_{l,k}$, $\boldsymbol{w} \xleftarrow{R} \mathbb{Z}_q^k$, $\boldsymbol{u} \xleftarrow{R} \mathbb{Z}_q^{k+1}$ and $s \in \{1,2\}$.

Lemma 1. *Among all possible matrix distribution $\mathcal{D}_{l,k}$, the uniform matrix distribution $\mathcal{U}_{l,k}$ is the hardest possible instance. We have $\mathcal{D}_{l,k} - MDDH \Rightarrow \mathcal{U}_{l,k} - MDDH$. For all PPT adversaries $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $Adv_{\mathbb{G}_s,\mathcal{A}}^{\mathcal{U}_{l,k}-MDDH} \leq Adv_{\mathbb{G}_s,\mathcal{B}}^{\mathcal{U}_{k}-MDDH}$.*

Lemma 2. *For $\boldsymbol{A} \xleftarrow{R} \mathcal{U}_{l,k}$, $\boldsymbol{W} \xleftarrow{R} \mathbb{Z}_q^{k\times Q}$, $\boldsymbol{U} \xleftarrow{R} \mathbb{Z}_q^{(k+1)\times Q}$, $s \in \{1,2\}$. $Adv_{\mathbb{G}_s,\mathcal{A}}^{\mathcal{Q}-\mathcal{U}_{l,k}-MDDH} = |Pr[\mathcal{A}(\mathcal{PG},[\boldsymbol{A}]_s,[\boldsymbol{AW}]_s)=1] - Pr[\mathcal{A}(\mathcal{PG},[\boldsymbol{A}]_s,[\boldsymbol{U}]_s)=1]|$. Then, we have for all PPT adversaries $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $Adv_{\mathbb{G}_s,\mathcal{A}}^{\mathcal{Q}-\mathcal{U}_{l,k}-MDDH} \leq Adv_{\mathbb{G}_s,\mathcal{B}}^{\mathcal{U}_{l,k}-MDDH} + \frac{1}{q-1}$.*

4 System Model

In our single-owner/multi-user setting, there are three different kinds of entities: data owner, data user and cloud server. As shown in Fig. 1, the data owner has a collection of files and wants to outsource his search service to the cloud server. The data owner first extracts keywords from the files and constructs inverted indices. It is important to note that our scheme is mainly applicable to the scenes that the number of keywords is limited but the number of files is huge, so the data owner only extracts the most relevant keywords.

And then, the data owner outsources encrypted indices and encrypted files to the cloud server. Besides, the data owner sends partial token and search-authorized secret key to each legitimate data user, with which data user is able

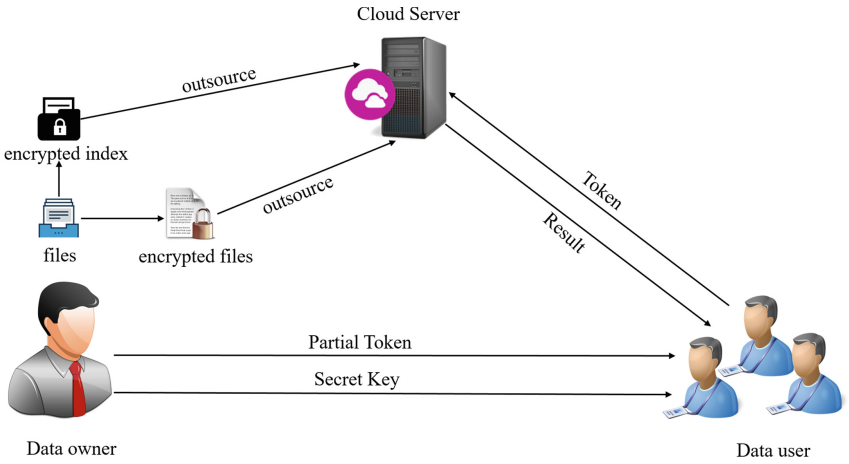

Fig. 1. System model

to generate search token about the keywords he wants to search. When a data user performs a search query, he sends the search token to the cloud server. With the search token and encrypted indices, the cloud server finally returns target documents to the data user.

Formally, our multi-keyword searchable encryption is a tuple of six polynomial-time algorithms $\pi = (Setup, Enc, PartialTokenGen, ClientKGen, TokenGen, Search)$

- $Setup(1^\lambda) \rightarrow (pp, msk)$: is a probabilistic algorithm that the data owner takes security parameter 1^λ as input and generates system master key msk and public parameter pp.
- $Enc(pp, F, W, DU) \rightarrow (C_W, C_F, C_{Indices}, C_{List})$: is a probabilistic algorithm that the data owner takes public parameter pp, a document collection $F = \{f_1, f_2, ..., f_n\}$, keyword dictionary $W = \{w_1, w_2, ..., w_m\}$ which is public and a set of legitimate data users DU as input and generate encrypted keywords C_W, encrypted files C_F and encrypted indices $C_{Indices}$. The file encryption is executed by using some simple symmetric encryption due to efficiency concerns. Besides, the data owner generates an encrypted list C_{List} about data users and their corresponding information.
- $PartialTokenGen(pp, msk, \xi) \rightarrow pt$: is a probabilistic algorithm that the data owner takes pp, msk to generate partial token pt for each legitimate data user $\xi \in DU$, with which and his search-authorized private key, a data user can generate search tokens for the keywords he wants to search.
- $ClientKGen(pp, msk, \xi) \rightarrow sk$: is a probabilistic algorithm that the data owner takes pp and msk as input and generates different search-authorized private key sk for each legitimate data user $\xi \in DU$.

- $TokenGen(sk, pt, Q) \rightarrow token$: is a deterministic algorithm that the data users use their private key sk and partial-token pt to produce search tokens $token$ for the keyword set Q they want to query.
- $Search(token, C_W, C_F, C_{Indices}, C_{List}) \rightarrow RST$: is a deterministic algorithm that the cloud server uses search token $token$ to search over encrypted indices $C_{Indices}$. Then it downloads the matched encrypted files RST and returns them to the data user.

5 Construction

In this section, we will introduce our multi-keyword searchable encryption scheme in detail.

– $Setup(1^\lambda)$: Given a bilinear group $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where q is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order q. g_1, g_2, g_T are generators of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively. Randomly select a matrix $\boldsymbol{A}$ from $\mathbb{Z}_q^{3\times 2}$ of full rank, namely randomly select a matrix $\boldsymbol{A}$ from $\mathcal{U}_2$, randomly choose a matrix $\boldsymbol{M}$ from $\mathbb{Z}_q^{3\times 3}$, $\boldsymbol{V}$ from $\mathbb{Z}_q^{2\times 3}$, and randomly select m vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_m$ from $\mathbb{Z}_q^2$. Let $\varepsilon = (Setup, Enc, KGen, Dec)$ be a public-key encryption scheme, where $Setup$ is a public key generation algorithm, Enc is an encryption algorithm, $KGen$ is a secret key generation algorithm and Dec is a decryption algorithm. $pk_{server} \leftarrow \varepsilon.Setup(1^\lambda)$ is public key of the cloud server. Then, output public parameter pp=$(g_1, g_2, g_T, q, [\boldsymbol{A}]_1, pk_{server})$ and master secret key $msk = (\boldsymbol{M}, \boldsymbol{V}, \{\boldsymbol{z}_i\}_{i\in[m]})$

– $Enc(pp, F, W, DU)$: Choose a super-incremental sequence $\alpha_1, \alpha_2, ..., \alpha_m \in (0, \frac{\log_{g_T} q}{2})$, that is, for $i \in [m]$,

$$\alpha_i > \alpha_1 + \alpha_2 + ... + \alpha_{i-1}$$

For each keyword $w_i \in W$, use a pseudo random substitution to map i to j, let $\boldsymbol{x}_i = (w_i, 1, r) \in \mathbb{Z}_q^3$, where r are randomly chosen from $\mathbb{Z}_q$, choose $\boldsymbol{y}_i = (y_{i_1}, y_{i_2}, 1) \in \mathbb{Z}_q^3$ such that $\alpha_j = <\boldsymbol{x}_i, \boldsymbol{y}_i>$. Besides, choose different r_ξ for each data user $\xi \in DU$, where r_ξ is randomly chosen from $\mathbb{Z}_q$. Record $\{\boldsymbol{x}_i\}_{i\in[m]}$, $\{\boldsymbol{y}_i\}_{i\in[m]}$ and $List = \{\xi, r_\xi\}_{\xi\in DU}$. Then, we compute $C_W = \{{g_T}^{\alpha_1}, {g_T}^{\alpha_2}, ..., {g_T}^{\alpha_m}\}$ as the ciphertext of keywords W, compute C_F as the ciphertext of files F with some symmetric encryption algorithm and generate encrypted indices $C_{Indices} = \{{g_T}^{\alpha_j}, Id_{w_i}\}_{j\in[m]}$. Id_{w_i} means a set of file identifiers of files which include keywords w_i. Besides, we use pk_{server} to compute $C_{List} = \varepsilon.Enc(List)$. Finally, the data owner sends (C_W, C_F, $C_{Indices}$, C_{List}) to cloud server.

– $PartialTokenGen(pp, msk, \{\boldsymbol{x}_i\}_{i\in[m]}, \xi)$: For each legitimate data user $\xi \in DU$, the data owner randomly chooses different $\boldsymbol{s}_{\xi,i} \in \mathbb{Z}_q^2$ and $r_{pt} \in \mathbb{Z}_q$. Let $\boldsymbol{r}_{\xi_{pt}} = (0, 0, r_{pt})$, and compute partial-token as follows.

$$[\boldsymbol{c}_i]_1 = [\boldsymbol{A}\boldsymbol{s}_{\xi,i}]_1 \tag{1}$$

$$[\boldsymbol{c}_i^{'}]_1 = [\boldsymbol{MAs}_{\xi,i} + \boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}]_1 \tag{2}$$

$$[\boldsymbol{c}_i^{''}]_1 = [\boldsymbol{VAs}_{\xi,i} + \boldsymbol{z}_i]_1 \tag{3}$$

Send the partial-token $pt = \left([\boldsymbol{c}_i]_1, [\boldsymbol{c}_i^{'}]_1, [\boldsymbol{c}_i^{''}]_1\right)_{i\in[m]}$ to the data user ξ by a secure channel.

– $ClientKGen(pp, msk, \{\boldsymbol{y}_i\}_{i\in[m]}, \xi, r_\xi)$: For each legitimate user $\xi \in DU$, the data owner randomly chooses different $\boldsymbol{r}_{\xi_1} \in \mathbb{Z}_q^2$, let $\boldsymbol{r}_{\xi_{sk}} = (0, r_\xi - r_{pt}, 0) \in \mathbb{Z}_q^3$ and compute secret key as follows.

$$\boldsymbol{d}_i = \boldsymbol{M}^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + \boldsymbol{V}^T\boldsymbol{r}_{\xi_1} \tag{4}$$

$$Z_i = <\boldsymbol{z}_i, \boldsymbol{r}_{\xi_1}> \tag{5}$$

Send the secret key $sk = \left(\{[\boldsymbol{d}_i]_2, [Z_i]_T, [\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}]_2\}_{i\in[m]}, [\boldsymbol{r}_{\xi_1}]_2\right)$ to the data user ξ by a secure channel.

– $TokenGen(sk, pt, Q)$: With partial-token pt, secret key sk and the keywords set $Q = \{w_{q_1}, w_{q_2}, ..., w_{q_t}\} \subseteq W$ to be searched, data users compute search tokens as follows. We use $e([\boldsymbol{X}]_1, [\boldsymbol{Y}]_2)$ to denote $[\boldsymbol{X}^T\boldsymbol{Y}]_T$.

$$st = \prod_{i=1}^{t} \frac{e([\boldsymbol{c}_{q_i}^{'}]_1, [\boldsymbol{y}_{q_i} + \boldsymbol{r}_{\xi_{sk}}]_2) \cdot e([\boldsymbol{c}_{q_i}^{''}]_1, [\boldsymbol{r}_{\xi_1}]_2) / e([\boldsymbol{c}_{q_i}]_1, [\boldsymbol{d}_{q_i}]_2)}{[Z_i]_T} \tag{6}$$

Data users send the search tokens $token = (st, \varepsilon.Enc(t))$ corresponding to the keywords they want to search to the cloud server.

– $Search(token, C_W, C_F, C_{Indices}, C_{List})$: When the cloud server receives a search token, it first decrypts $\varepsilon.Enc(t)$ to get t and retrieves real search token $rst = g_T^{\sum_{i=1}^{t} <\boldsymbol{x}_{q_i}, \boldsymbol{y}_{q_i}>}$ by t and $[r_\xi]_T$ corresponding to user identity ξ and then determines whether g^{α_m} less than the real search token rst, if so, return $\perp$, which means that the search token is illegal and there is no corresponding keywords. Otherwise, by using binary search, the cloud server determines whether there is a k satisfying $g^{\alpha_k} \leq rst \leq g^{\alpha_{k+1}}$, if so, it means that the keyword corresponding to g^{α_k} is one of the keyword the data user wants to search. Then, it calcautes $rst = rst/g^{\alpha_k}$ and repeats the above steps until rst equals to one. Pseudo code is showed in Algorithm 1. Finally, cloud server takes all the file identifiers that contain the keywords to be searched and then returns the ciphertexts of the corresponding file to the data user.

6 Correctness and Security

6.1 Correctness

We now show the correctness of the search phase.

Algorithm 1. Search Process

if $g^{\alpha_m} \leq rst$ **then**
  return $\perp$
else
  $int\ low = 1, high = m;$
  while $low \leq high \&\& rst == 1$ **do**
    $mid = (low + high)/2$
    if $g^{\alpha_{mid+1}} \leq rst$ **then**
      $low = mid + 1$
    else if $g^{\alpha_{mid}} > rst$ **then**
      $high = mid - 1$
    else $g^{\alpha_{mid}} \leq rst \leq g^{\alpha_{mid+1}}$
      the keyword corresponding to $g^{\alpha_{mid}}$ is one of the target keyword
      $rst = rst/g^{\alpha_{mid}}$
      $high = mid - 1$
      $low = 1$
    end if
  end while
end if

The data user first calculates search token as follows:

$$
\begin{aligned}
st &= \prod_{i=1}^{t} \frac{e([\boldsymbol{c}_{q_i}^{'}]_1, [\boldsymbol{y}_{q_i} + \boldsymbol{r}_{\xi_{sk}}]_2) \cdot e([\boldsymbol{c}_{q_i}^{''}]_1, [\boldsymbol{r}_{\xi_1}]_2) / e([\boldsymbol{c}_{q_i}]_1, [\boldsymbol{d}_{q_i}]_2)}{[Z_i]_T} \\
&= \prod_{i=1}^{t} \frac{{g_T}^{<\boldsymbol{c}_{q_i}^{'}, \boldsymbol{y}_{q_i} + \boldsymbol{r}_{\xi_{sk}}>} \cdot {g_T}^{<\boldsymbol{c}_{q_i}^{''}, \boldsymbol{r}_{\xi_1}>} / {g_T}^{<\boldsymbol{c}_{q_i}, \boldsymbol{d}_{q_i}>}}{[Z_i]_T} \\
&= \prod_{i=1}^{t} \frac{{g_T}^{<\boldsymbol{M}\boldsymbol{A}\boldsymbol{s}_{\xi,qi} + \boldsymbol{x}_{qi} + \boldsymbol{r}_{\xi_{pt}}, \boldsymbol{y}_{q_i} + \boldsymbol{r}_{\xi_{sk}}> + <\boldsymbol{V}\boldsymbol{A}\boldsymbol{s}_{\xi,qi} + \boldsymbol{z}_{qi}, \boldsymbol{r}_{\xi_1}>}}{{g_T}^{<\boldsymbol{z}_{qi}, \boldsymbol{r}_{\xi_1}> + <\boldsymbol{A}\boldsymbol{s}_{\xi,qi}, \boldsymbol{M}^T(\boldsymbol{y}_{q_i} + \boldsymbol{r}_{\xi_{sk}}) + \boldsymbol{V}^T \boldsymbol{r}_{\xi_1}>}} \\
&= \prod_{i=1}^{t} {g_T}^{<\boldsymbol{x}_{q_i} + \boldsymbol{r}_{\xi_{pt}}, \boldsymbol{y}_{q_i} + r_{\xi_{sk}}>} \\
&= {g_T}^{\sum_{i=1}^{t} <\boldsymbol{x}_{q_i}, \boldsymbol{y}_{q_i}> + r_\xi}
\end{aligned} \tag{7}
$$

When the cloud server receives a search token, it first decrypts $\varepsilon.Enc(t)$ to get t and retrieves real search token $rst = {g_T}^{\sum_{i=1}^{t} <\boldsymbol{x}_{q_i}, \boldsymbol{y}_{q_i}>}$ by t and $[r_\xi]_T$ according to user's identity ξ. Because $\alpha_1, \alpha_2, ..., \alpha_m \in (0, \frac{\log_{g_T} q}{2})$ is a super-incremental sequence, we have that $\alpha_i > \alpha_1 + \alpha_2 + ... + \alpha_{i-1}$. Thus, ${g_T}^{\alpha_i} > {g_T}^{\alpha_1 + \alpha_2 + ... + \alpha_{i-1}}$. Because of the ciphertext of keywords ${g_T}^{\alpha_j} = {g_T}^{<\boldsymbol{x}_{qi}, \boldsymbol{y}_{qi}>}$, it means that the product of the cipherhext of keywords that data user wants to search equals to the real search token. When the cloud server retrieves ${g_T}^{\sum_{i=1}^{t} <\boldsymbol{x}_{qi}, \boldsymbol{y}_{qi}>}$, it determines whether there is a k satisfying ${g_T}^{\alpha_k} \leq rst < {g_T}^{\alpha_{k+1}}$ or not. Obviously, the keywords corresponding to ${g_T}^{\alpha_{k+1}}, ..., {g_T}^{\alpha_m}$ can not be the target keyword. If keyword corresponding to ${g_T}^{\alpha_k}$ is not the target keyword, the keywords corresponding to ${g_T}^{\alpha_1}, ..., {g_T}^{\alpha_{k-1}}$ must be the target keywords, namely, ${g_T}^{\alpha_1 + ... + \alpha_{k-1}} = rst$. However, according to the super-incremental sequence, we

know that ${g_T}^{\alpha_1+\ldots+\alpha_{k-1}} < rst$, that is to say, the keywords corresponding to ${g_T}^{\alpha_1}, ..., {g_T}^{\alpha_{k-1}}$ cannot be the target keywords. Therefore, we know that the keyword corresponding to ${g_T}^{\alpha_k}$ is one of the target keywords.

6.2 Security

For the files, ciphertexts C_F are semantic security by adopting symmetric encryption, such as AES. Then a probabilistic polynomial-time adversary cannot get any useful information from C_F with non-negligible probability. For the keyword, we have:

Partial-Token Privacy. Partial-Token Privacy means that a probabilistic polynomial-time adversary cannot get any useful information from the partial-token. That is to say, assuming that an adversary gets one item of the partial-token from a legitimate data user ξ, he could not know which keywords the item is about.

- Setup: The challenger plays a role as the system and runs $Setup()$, then it keeps master key msk.
- Init: The challenger runs $Enc()$ to get different reasonable $\{\boldsymbol{x}_i\}_{i\in[m]}$, $\{\boldsymbol{y}_i\}_{i\in[m]}$ and $\{\xi, r_\xi\}_{\xi\in DU}$, with which it can run $ClientKGen()$ and $PartialTokenGen()$.
- Query Phase1: The adversary adaptively queries sk and pt about different ξ for polynomial times. The challenger runs $PartialTokenGen()$ and $ClientKGen()$ algorithm and returns $pt \longleftarrow PartialTokenGen(pp, msk, \{\boldsymbol{x}_i\}_{i\in[m]}, \xi)$ and $sk \longleftarrow ClientKGen(pp, msk, \{\boldsymbol{y}_i\}_{i\in[m]}, \xi, r_\xi)$.
- Challenge phase: The adversary randomly selects two keywords w_{i_0} and w_{i_1} and submits the identity ξ he wants to challenge with the restriction that ξ has not queried before. The challenger flips a coin to select $\beta \longleftarrow \{0, 1\}$ and then runs $PartialTokenGen()$ algorithm. The challenger returns $PartialTokenGen(pp, msk, \boldsymbol{x}_{i\beta}, \xi, r_\xi)$ to the adversary.
- Query Phase 2: The adversary executes queries as Phase1 did.
- Guess: Finally, the adversary gives a guess β' of β and wins the game if $\beta' = \beta$. We can define the advantage of adversary winning the game is $|Pr[\beta' = \beta] - \frac{1}{2}|$.

Theorem 1. *If an adversary wins the game mentioned above with a non-negligible advantage, there is an adversary $\mathcal{B}$ can break MDDH assumption.*

Proof. Specific proofs are detailed in the Appendix A. □

Index Privacy. Index privacy means that a probabilistic polynomial-time adversary cannot get any useful information from encrypted keyword C_w. In other words, the cloud server cannot determine which keyword the ciphertext is for.

- Setup: The challenger plays a role as the system and runs $Setup()$, then it keeps master key msk.
- Query Phase1: The adversary adaptively queries the ciphertext of keyword w for polynomial times, and get $C_w \longleftarrow Enc(pp, w)$.
- Challenge phase: The adversary randomly selects two keywords w_{i_0} and w_{i_1} which have not queried before, and sends them to the challenger. The challenger flips a coin to select $\beta \longleftarrow \{0, 1\}$ and then returns $C_w \longleftarrow Enc(pp, w_{i\beta})$ to the adversary.
- Query Phase 2: The adversary continues to query the ciphertext of keyword w as Phase1 did with the restriction that w is neither w_{i_0} nor w_{i_1}.
- Guess: Finally, the adversary gives a guess $\beta^{'}$ of β and wins the game if $\beta^{'} = \beta$. We can define the advantage of adversary winning the game is $|Pr[\beta^{'} = \beta] - \frac{1}{2}|$.

Theorem 2. *If an adversary wins the game mentioned above with a non-negligible advantage, our scheme is secure with index privacy.*

Proof. If the adversary wants to know whether C_w is about w_{i_0} or w_{i_1}, he will analyze the $C_w = {g_T}^{<\boldsymbol{x}_{i\beta}, \boldsymbol{y}_{i\beta}>} = {g_T}^{\alpha_j}$. Because α_j is less than $\frac{\log_{g_T} q}{2}$, he could get $<\boldsymbol{x}_{i\beta}, \boldsymbol{y}_{i\beta}>$ by logarithmic operation. However, $\boldsymbol{y}_{i\beta}$ is kept secret by the challenger, so that it is impossible for the adversary to get $\mathbf{x}_{i\beta}$ and has no chance to get keyword $w_{i\beta}$. Therefore, the adversary can not know whether β equals to 0 or 1. □

Token Privacy. Token privacy means that given a search token, a probabilistic polynomial-time adversary cannot learn which keyword the search token is for. Namely, the cloud server cannot know which keyword the data user queries.

- Setup: The challenger plays a role as the system and runs $Setup()$, then it keeps master key msk.
- Init: The challenger runs $Enc()$ to get reasonable $\{\boldsymbol{x}_i\}_{i\in[m]}$, $\{\boldsymbol{y}_i\}_{i\in[m]}$ and $\{\xi, r_\xi\}_{\xi\in DU}$, with which it can run $ClientKGen()$ to obtain secret key sk. Besides, it runs $PartialTokenGen()$ to get partial token pt.
- Query Phase1: The adversary with ξ adaptively queries search token of keyword w for polynomial times, and get $st \longleftarrow Token(sk, pt, w)$.
- Challenge phase: The adversary randomly selects two keywords w_{i_0} and w_{i_1} which have not queried before, and sends them to the challenger. The challenger flips a coin to select $\beta \longleftarrow \{0, 1\}$ and then runs $TokenGen()$ algorithm. The challenger returns $st \longleftarrow Token(sk, pt, w)$ to the adversary.
- Query Phase 2: The adversary continues to query search token of keyword w as Phase1 did with the restriction that w is neither w_{i_0} nor w_{i_1}.
- Guess: Finally, the adversary gives a guess $\beta^{'}$ of β and wins the game if $\beta^{'} = \beta$. We can define the advantage of adversary winning the game is $|Pr[\beta^{'} = \beta] - \frac{1}{2}|$.

Theorem 3. *If an adversary wins the game mentioned above with a non-negligible advantage, our scheme is secure with token privacy.*

Proof. If the adversary wants to know whether st is about w_{i_0} or w_{i_1}, he will analyze the $st = g_T{}^{<\boldsymbol{x}_{i\beta},\boldsymbol{y}_{i\beta}>+r_\xi}$. For the cloud, although he can calculate $g_T{}^{<\boldsymbol{x}_{i\beta},\boldsymbol{y}_{i\beta}>}$ and get $<\boldsymbol{x}_{i\beta},\boldsymbol{y}_{i\beta}>$ by logarithmic operation, the cloud has no way to get $\boldsymbol{y}_{i\beta}$. Therefore, it is incapable of getting $\boldsymbol{x}_{i\beta}$ and unable to get keyword $w_{i\beta}$. □

7 Functionality and Efficiency

We compare our scheme with the work in [13,22]and the first scheme of work in [12] in Table 1. From the table, we can see that the size of ciphertext is linear with the number of keywords m in both literature [13] and our scheme. While in literature [22], the ciphertext size is linear with the product of the number of keywords m and the number of files n. And in literature [12], the ciphertext size is linear with the number of keywords data owner extracts. We can easily find that the size of the search token is constant only in our scheme. Obviously, our scheme could significantly reduce the communication and transportation overhead, especially when the number of data users is large and the query frequency is high. Besides, by using an inverted index structure and super-incremental sequence, our scheme achieves efficient multi-keyword search, which is illustrated in Table 1. In addition, our scheme avoids the per-query interaction between data owner and data users. That is to say, the data owner does not need to stay online waiting for data users to search in his archives. Furthermore, our scheme supports multi-keyword search in single-owner/multi-user setting.

Table 1. Calculation overhead

	[12]	[13]	[22]	Ours
Ciphertext size	$o(n\alpha)$	$o(m)$	$o(mn)$	$o(m)$
Token size	$o(n+t)$	$o(ct)$	$o(m)$	2
Search time	$o(m)$	$o(ct)$	$o(wm\log_2 n)$	$o(t\log_2 m)$
Inverted index	×	√	×	√
Multi-keyword	√	√	√	√
Multi-user	×	√	×	√
Non interaction*	N/A	×	N/A	√

*: the interaction between data owner and data users whenever data users perform search queries.
n: the number of files.
m: the number of keywords.
t: the number of keywords data user wants to search.
c: the number of files including the least frequency keywords among keywords data user wants to search.
w: the number of files including the keywords data user wants to search.
α: the number of keyword in each file, which its fixed in [13].

8 Conclusion

In our scheme, search tokens have only two items by tactfully leveraging multi-input inner-product functional encryption, which reduces communication and transportation overhead significantly. The use of inverted index structure and super-incremental sequence makes the multi-keyword search process efficient. In addition, our scheme avoids the per-query interaction between data owner and data users. That is to say, data owner does not need to stay online waiting for data users to search in his archives. What is more, our scheme ensures the correctness of search process and protects the privacy of keywords and plaintext files.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China (Grant No.61632012, 61672239, and U1509219) and in part by "the Fundamental Research Funds for the Central Universities". Zhenfu Cao and Jiachen Shen are the corresponding authors.

A Proof of Theorem1

Proof. The proof of theorem1 consists of six games. The transitions between contiguous games are summarized in Table 2. $([\boldsymbol{c}_i]_1, [\boldsymbol{c}_i{}']_1, [\boldsymbol{c}_i{}'']_1)$ is computed by $PartialTokenGen()$. $[\boldsymbol{d}_i]_2$ and $[Z_i]_T$ are part of sk computed by $ClientKGen()$. We use $\boldsymbol{u} \xleftarrow{R} \mathbb{Z}_q^3 \backslash Span(\boldsymbol{A})$ and $\boldsymbol{a}^{\perp} \xleftarrow{R} \mathbb{Z}_q^3$ so that $\boldsymbol{A}^T\boldsymbol{a}^{\perp} = \boldsymbol{0}$ and $<\boldsymbol{u}, \boldsymbol{a}^{\perp}> = 1$. To analyze Game3, we consider the selective variant of the game: Game3*. Then we prove Game3* through using an information-theoretic argument via Game4*. Both Game3* and Game4* are selective security.

The concrete proof is similar to [1]. Here we just simply outline each game.

Let $\mathcal{A}$ be a PPT adversary, and let $\lambda \in N$ be the security parameter. We define

$$Adv_t(\mathcal{A}) := Pr[Game_t(1^{\lambda}, \mathcal{A}) = 1]\, t \in \{0, 1, 2, 3, 3^*, 4^*\}$$

- Game 0: is the partial-token privacy game.
- Game 1: using $\mathcal{U}_k$-MDDH, we change the distribution of the vectors $[\mathbf{c}_i]_1$ computed by $PartialTokenGen()$. This change depends on the fact that:
  - The distributions $\{\boldsymbol{s}_{\xi,i}\}_{i\in[m]}$ and $\{\boldsymbol{s}_{\xi,i} + \boldsymbol{s}\}_{i\in[m]}$, where $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_q^2$, $\mathbf{s}_{\xi,i} \xleftarrow{R} \mathbb{Z}_q^2$ are identically distributed.
  - By $\mathcal{U}_k$-MDDH assumption, we can switch $([\boldsymbol{A}]_1, [\boldsymbol{A}\mathbf{s}]_1)$ to $([\boldsymbol{A}]_1, [\boldsymbol{u}]_1)$, where $\boldsymbol{A} \xleftarrow{R} \mathcal{U}_2$, $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_q^2$, $\boldsymbol{u} \xleftarrow{R} \mathbb{Z}_q^3$.
  - The uniform distributions over $\mathbb{Z}_q^3$ and $\mathbb{Z}_q^3 \setminus Span(\boldsymbol{A})$ are $\frac{1}{q}$-close.
- Game 2: using an information theoretic argument, we change the way how the vectors $[\boldsymbol{c}_i{}'']_1$ and $[\boldsymbol{d}_i]_2$ are computed by $PartialTokenGen()$ and

ClientKGen() respectively. This relies on the fact that the distributions $\boldsymbol{V}$ and $\boldsymbol{V} - \boldsymbol{z}_i(\boldsymbol{a}^{\perp})^T$ are identical. Thus, we have

$$\begin{aligned}[\boldsymbol{c}_i^{''}]_1 &= [(\boldsymbol{V} - \boldsymbol{z}_i(\boldsymbol{a}^{\perp})^T)(\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}) + \boldsymbol{z}_i]_1 \\ &= [\boldsymbol{V}(\mathbf{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}) - \boldsymbol{z}_i(\boldsymbol{a}^{\perp})^T\boldsymbol{u} + \boldsymbol{z}_i]_1 \\ &= [\boldsymbol{V}(\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u})]_1 \end{aligned} \tag{8}$$

$$\begin{aligned}\boldsymbol{d}_i &= \boldsymbol{M}^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + (\boldsymbol{V}^T - \boldsymbol{a}^{\perp}\boldsymbol{z}_i^T)\boldsymbol{r}_{\xi_1} \\ &= \boldsymbol{M}^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + \boldsymbol{V}^T\boldsymbol{r}_{\xi_1} - \boldsymbol{a}^{\perp}<\boldsymbol{z}_i, \boldsymbol{r}_{\xi_1}> \end{aligned} \tag{9}$$

- Game 3: we switch $\{[\boldsymbol{r}_{\xi_1}]_2, [<\boldsymbol{z}_i, \boldsymbol{r}_{\xi_1}>]_2\}_{i\in[m]}$ to $\{[\boldsymbol{r}_{\xi_1}]_2, [\widetilde{z_i}]_2\}_{i\in[m]}$ for all calls to *ClientKGen*(), where $\widetilde{z_1}, \widetilde{z_2}, ..., \widetilde{z_m} \xleftarrow{R} \mathbb{Z}_q$. This is justified by the distributions $[\boldsymbol{r}_{\xi_1}^T||<\boldsymbol{z}_1, \boldsymbol{r}_{\xi_1}>||...||<\boldsymbol{z}_m, \boldsymbol{r}_{\xi_1}>]_2 \in \mathbb{G}_2^{1\times(2+m)}$ and $[\boldsymbol{r}_{\xi_1}^T\boldsymbol{U}^T]_2$, where $\boldsymbol{U} \xleftarrow{R} \mathcal{U}_{2+m,2}$ are identical. According to $\mathcal{U}_{2+m,2}$-MDDH, we know that $[\boldsymbol{r}_{\xi_1}^T\boldsymbol{U}^T]_2$ is indistinguishable from a random vector over $\mathbb{G}_2^{1\times(2+m)}$ of the form $[\boldsymbol{r}_{\xi_1}^T||\widetilde{z_1}||...||\widetilde{z_m}]_2$.
- Game 3*: is the selective variant of Game3, in other words, any adversary playing this game has to commit its challenge queries w_{i_0} and w_{i_1} beforehead.
- Game 4*: is similar to Game3*, excep t $[\boldsymbol{c}_i^{'}]_1$ and $[\boldsymbol{Z}_i]_T$ computed by *PartialTokenGen*() and *ClientKGen*() respectively. The transform is true because of the fact $\left\{\widetilde{z_i} - <\boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}, \boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}>\right\}_{i\in[m]}$ and $\{\widetilde{z_i}\}_{i\in[m]}$ are identically distributed. Besides, $\boldsymbol{M}$ and $\boldsymbol{M} - \boldsymbol{x}_i(\boldsymbol{a}^{\perp})^T$ are identically distributed.

We build a PPT adversary $\mathcal{B}_1$ so that

$$Adv_0(\mathcal{A}) - Adv_1(\mathcal{A}) \leq Adv_{\mathbb{G}_1,\mathcal{B}_1}^{\mathcal{U}_k - MDDH}(\lambda) + \frac{1}{q}$$

Because of information theoretic argument, we know that

$$Adv_1(\mathcal{A}) = Adv_2(\mathcal{A})$$

Table 2. Sequence of games for the proof of partial-token privacy

Game	$\boldsymbol{c}_i$	$\boldsymbol{c}_i^{'}$	$\boldsymbol{c}_i^{''}$	$\boldsymbol{d}_i$	Z_i
0	$\boldsymbol{A}\boldsymbol{s}_{\xi,i}$	$M\boldsymbol{c}_i + \boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}$	$V\boldsymbol{c}_i + \boldsymbol{z}_i$	$M^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + V^T\boldsymbol{r}_{\xi_1}$	$< \boldsymbol{z}_i, \boldsymbol{r}_{\xi_1} >$
1	$\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}$	$M\boldsymbol{c}_i + \boldsymbol{x}_i + vr_{\xi_{pt}}$	$V\boldsymbol{c}_i + \boldsymbol{z}_i$	$M^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + V^T\boldsymbol{r}_{\xi_1}$	$< \boldsymbol{z}_i, \boldsymbol{r}_{\xi_1} >$
2	$\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}$	$M\boldsymbol{c}_i + \boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}$	$V\boldsymbol{c}_i$	$M^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + V^T\boldsymbol{r}_{\xi_1} - \boldsymbol{a}^{\perp} < \boldsymbol{z}_i, \boldsymbol{r}_{\xi_1} >$	$< \boldsymbol{z}_i, \boldsymbol{r}_{\xi_1} >$
3	$\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}$	$M\boldsymbol{c}_i + \boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}$	$V\boldsymbol{c}_i$	$M^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + V^T\boldsymbol{r}_{\xi_1} - \boldsymbol{a}^{\perp}\widetilde{z_i}$	$\widetilde{z_i}$
3*	$\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}$	$M\boldsymbol{c}_i + \boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}$	$V\boldsymbol{c}_i$	$M^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + V^T\boldsymbol{r}_{\xi_1} - \boldsymbol{a}^{\perp}\widetilde{z_i}$	$\widetilde{z_i}$
4*	$\boldsymbol{A}\boldsymbol{s}_{\xi,i} + \boldsymbol{u}$	$M\boldsymbol{c}_i + \boldsymbol{r}_{\xi_{pt}}$	$V\boldsymbol{c}_i$	$M^T(\boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}}) + V^T\boldsymbol{r}_{\xi_1} - \boldsymbol{a}^{\perp}\widetilde{z_i}$	$\widetilde{z_i} - < \boldsymbol{x}_i + \boldsymbol{r}_{\xi_{pt}}, \boldsymbol{y}_i + \boldsymbol{r}_{\xi_{sk}} >$

There exists a PPT adversary $\mathcal{B}_3$ so that the

$$Adv_2(\mathcal{A}) - Adv_3(\mathcal{A}) \leq Adv^{\mathcal{U}_k - MDDH}_{\mathbb{G}_2, \mathcal{B}_3}(\lambda) + \frac{1}{q-1}$$

Using complexity leveraging, we build a PPT adversary $\mathcal{B}_{3*}$ so that

$$Adv_3(\mathcal{A}) \leq m(m-1) \cdot Adv_{3*}(\mathcal{B}_{3*})$$

For all adversaries $\mathcal{A}$, $Adv_{3*}(\mathcal{A}) = Adv_{4*}(\mathcal{A})$.

In Game 4*, from Table II, we can easily find that the partial token of $w_{i\beta}$ of ξ is only associate with vector $\mathbf{s}_{\xi,i\beta}$. However, $\boldsymbol{s}_{\xi,i\beta}$ is randomly choosen from $\mathbb{Z}_q^k$. That is to say, $\boldsymbol{A}\boldsymbol{s}_{\xi,i0}$ and $\boldsymbol{A}\boldsymbol{s}_{\xi,i1}$ are statistically indistinguishable. Thus, $Adv_{4*}(\mathcal{A}) = 0$. Therefore, we obtain that

$$Adv_0(\mathcal{A}) \leq Adv^{\mathcal{U}_k - MDDH}_{\mathbb{G}_1, \mathcal{B}_1}(\lambda) + Adv^{\mathcal{U}_k - MDDH}_{\mathbb{G}_2, \mathcal{B}_3}(\lambda) + \frac{2}{q-1}$$

Using $\mathcal{U}_{l,k} - MDDH$ assumption in $\mathbb{G}_1$, $\mathbb{G}_2$, we know that $Adv_0(\mathcal{A})$ is negligible in λ.

References

1. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings, pp. 601–626 (2017)
2. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005). https://doi.org/10.1007/11602897_35
3. Cash, D., et al.: Dynamic searchable encryption in very-large databases: data structures and implementation. In: Network and Distributed System Security Symposium (2014)
4. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_20
5. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
6. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
8. Dhumal, A.A., Jadhav, S.: Confidentiality-conserving multi-keyword ranked search above encrypted cloud data. Procedia Comput. Sci. **79**, 845–851 (2016)

9. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8
10. Fu, Z., Wu, X., Wang, Q., Ren, K.: Enabling central keyword-based semantic extension search over encrypted outsourced data. IEEE Trans. Inf. Forensics Secur. **12**(12), 2986–2997 (2017)
11. Goh, E.J.: Secure indexes. IACR Cryptology ePrint Archive 2003, 216 (2003)
12. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_3
13. Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Outsourced symmetric private information retrieval. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 875–888 (2013)
14. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
15. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_21
16. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
17. Li, H., Liu, D., Dai, Y., Luan, T.H., Shen, X.S.: Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage. IEEE Trans. Emerg. Top. Comput. **3**(1), 127–138 (2015)
18. Li, H., Yang, Y., Luan, T., Liang, X., Zhou, L., Shen, X.: Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. IEEE Trans. Dependable Secure Comput. **13**(3), 312–325 (2016)
19. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_27
20. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
21. Hwang, Y.H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73489-5_2
22. Zhu, X., Dai, H., Yi, X., Yang, G., Li, X.: MUSE: an efficient and accurate verifiable privacy-preserving multikeyword text search over encrypted cloud data. Secur. Commun. Netw. **2017**(2), 1–17 (2017)