A Parallel Multi-Keyword Top-k Search Scheme over Encrypted Cloud Data

Maohu Yang, Hua Dai, Jingjing Bao, Xun Yi and Geng Yang

August 3, 2019

A Parallel Multi-Keyword Top-k Search Scheme over Encrypted Cloud Data [⋆]

Maohu Yang[1], Hua Dai[1,2], Jingjing Bao[1], Xun Yi[3], and Geng Yang[1,2]

[1] Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[2] Jiangsu Security and Intelligent Processing Lab of Big Data, Nanjing 210023, China
[3] Royal Melbourne Institute of Technology University, Melbourne 3001, Australia
`yangmh1234@163.com`, `daihua@njupt.edu.cn`, `jing874444051@163.com`, `xun.yi@rmit.edu.au`, `yangg@njupt.edu.cn`

Abstract. With searchable encryptions in the cloud computing, users can outsource their sensitive data in ciphertext to the cloud that provides efficient and privacy-preserving multi-keyword top-k searches. However, most existing top-k search schemes over encrypted cloud data are the centralize schemes which are limited in large scale data environment. To support scalable searches, we propose a parallel multi-keyword top-k search scheme over encrypted cloud data. In this scheme, the fragment-based encrypted inverted index is designed, which is indistinguishable and can be used for parallel searching. On the basis of such indexes, a Map-Reduce-based distributed computing framework is adopted to implement the parallel multi-keyword top-k search algorithms. Security analysis and experiment evaluation show that the proposed scheme is privacy-preserving, efficient and scalable.

Keywords: Cloud Computing · Inverted Index · Multi-keywords top-k Search · Parallel Computing · Searchable Encryption.

1 Introduction

With the rapid development of computer technology and internet application, data in many areas are growing exponentially, thus the demand for large and scalable storage and computation is becoming urgent. More and more enterprises and individuals outsource their storage and computation to the cloud for using data anytime and anywhere and saving costs of hardware and software [1]. However, while enjoying the benefits of cloud computing, users have to face the risk that sensitive outsourced data could be leaked or abused because cloud service providers can access the data without authorization. Therefore, data owners usually encrypt data before outsourcing [2]. Although encryption preserves the

[⋆] Supported by the National Natural Science Foundation of China under the grant Nos.61872197, 61572263, 61672297 and 61872193; the Postdoctoral Science Foundation of China under the Grand No. 2019M651919; the Natural Research Foundation of NJUPT under the grand No.NY217119; the Natural Science Foundation of Anhui Province under the grant No.1608085MF127; the University Natural Science Foundation of Anhui Province under the grant No.KJ2017A419.

security of data, it also affects the data availability. In this scenario, searchable encryptions (SE) [3–18] that guarantee the security and availability of data have been proposed.

At present, most solutions are based on the vector space model (VSM) and TF-IDF model which extract keywords of documents into "points" in multi-dimensional space and describe the relevance scores between documents and search keywords. The top-k documents are determined by comparing the relevance scores. However, if a scheme calculates the relevance scores between every document and the search keywords, it will cost a large amount of time and computing resources. To improve search efficiency, researchers have provided a variety of schemes. Song et al. [3] proposed the first SE scheme where users need to traverse entire documents while searching, and the search time is proportional to the amount of data set. Goh et al. [4] proposed a search scheme based on the Bloom Filter. Curtmola et al. [5] proposed an efficient SE scheme based on the inverted index, but using this scheme could expose the privacy of keywords.

Cao et al. [6] proposed a new structure to adapt to multi-keyword search, but it's search time increases exponentially when document size grows. Xia et al. [8] proposed a secure and dynamic multi-keyword ranked search scheme which can reduce the large inner product calculation by pruning function. Jiang et al. [9] proposed a secure ciphertext search scheme based on the inverted index, which avoids calculating the relevance scores of irrelevant documents. Chen et al. [10] proposed a method based on data mining, which can achieve linear time complexity with the exponential growth of the document set. Our previous work [11] also proposed a hierarchical agglomerative clustering tree index scheme, which can perform an effective and verifiable ranked search.

However, the above existing schemes need to load the complete indexes into memory at one time for performing search. Because the index size is proportional to the number of documents, when the scale of documents grows to a certain level, the memory will be overflow. Moreover, the indexes of those schemes should be kept in integrity and cannot be segmented, which also limits their scalability. To conquer such limitation, we propose a parallel privacy-preserving top-k search (PPTS) scheme, which can meet the requirements of large scale data. First, we propose a fragment-based encrypted inverted index model. In data preprocessing and outsourcing phase, the indistinguishable fragment-based encrypted inverted indexes are constructed and outsourced to the cloud together with the encrypted documents. In the search phase, the Map-Reduce-based distributed computing framework is adopted and the parallel multi-keyword top-k search algorithms are proposed. After that, we analyze the security of PPTS and perform experiments to evaluate its efficiency.

The contributions of this paper are: 1) We present the fragment-based encrypted inverted index model which is indistinguishable through adding random paddings. 2) By adopting the Map-Reduce-based distributed computing framework, the parallel multi-keyword top-k search algorithms are proposed. 3) We analyze the security and evaluate the search performance. The result shows that the proposed scheme can realize parallel search while preserving data privacy.

2 MODELS AND PROBLEM FORMULATION

2.1 Notations and Preliminaries

- D : The document set, $D = \{d_1, d_2, ..., d_n\}$. $\widetilde{D}$ is the encrypted form.
- n : The number of documents in D.
- W : The dictionary, namely, the set of keywords, denoted as $W = \{w_1, w_2, ..., w_m\}$.
- m : The number of keywords in W.
- Q : The query consisting of a set of the search keywords, $Q=\{w_1, w_2, ..., w_q\}$.
- V_{d_i} : The m-dimensional document vector of d_i. $\widetilde{V_{d_i}}$ is the encrypted form.
- V : The document vector set, $V=\{V_{d_1}, V_{d_2}, ..., V_{d_n}\}$. $\widetilde{V}$ is the encrypted form.
- V_q : The m-dimensional query vector for Q. $\widetilde{V_q}$ is the encrypted form.
- TD : The trapdoor for the search request.
- RS : The result of the search.
- $P_{i,j}$: The posting corresponding to the document d_j containing keyword w_i, $P_{i,j} =< id(d_j), V_{d_j} >$. $\widetilde{P_{i,j}}$ is the encrypted form.
- PL_i : The posting list of keyword w_i, $PL_i = \{P_{i,1}, P_{i,2}, ..., P_{i,\delta}\}$. $\widetilde{PL_i}$ is the encrypted form.
- δ : The number of postings in PL_i.
- ε : The fragmentation parameter.
- F : The fragmented documents of D according to ε, $F = \{F_1, F_2, ..., F_t\}$.
- t : The number of fragments of F.
- β : The number of posting list in F_i.

Vector Space Model(VSM) and TF-IDF Model. The VSM and TF-IDF are widely used in multi-keyword privacy-preserving top-k search [6–13]. The term frequency (TF) refers to the number of times a given keyword or term appears in documents, while the inverse document frequency (IDF) is equal to the total number of documents in the set divided by the number of documents containing a given keyword. VSM is used to convert a given document d_i and search keywords Q into vectors V_{d_i} and V_q. The calculation of those vectors can be referred to [6–13].

Secure Inner Product Operation. This scheme uses the secure inner product operation to calculate the inner product of two encrypted vectors without knowing the plaintext value. The basic idea of this is as follows. Assuming that p and q are two n-dimensional vectors and M is a random $n \times n$-dimensional invertible matrix. M is treated as the secure key. The encrypted form of p and q are denoted as $\widetilde{p}$ and $\widetilde{q}$ respectively, where $\widetilde{p} = pM^{-1}$ and $\widetilde{q} = qM^T$. Then we have $\widetilde{p} \cdot \widetilde{q} = (pM^{-1}) \cdot (qM^T) = pM^{-1}(qM^T)^T = pM^{-1}Mq = p \cdot q$, i.e. $\widetilde{p} \cdot \widetilde{q} = p \cdot q$. Therefore, we have that the inner product of two encrypted vectors equals the inner product of the corresponding two plaintext vectors.

Inverted Index. Inverted index can be used to quickly find those documents containing a given keyword by mapping to improves search efficiency. It consists of dictionary and posting list. The dictionary is a collection of all keywords that appeared in the D. Each index item in inverted index records a keyword and a pointer to the posting list, which is the entry of posting. The posting list records a list of all documents that contain a specified keyword. Each record in the posting list is a posting that describes the information of the document.

2.2 System Model

The system model is shown in Fig.1, which is the same as [6–11], [14–16]. It includes three different entities. Data owners(DO) are responsible for constructing fragment-based encrypted inverted indexes ($\widetilde{I}$), and outsourcing the encrypted indexes and documents to the cloud server (CS). CS provide the search service in parallel according to the search request submitted by data users (DU). DU construct a search trapdoor based on its needs and send it to CS, then wait for CS to return the search results.

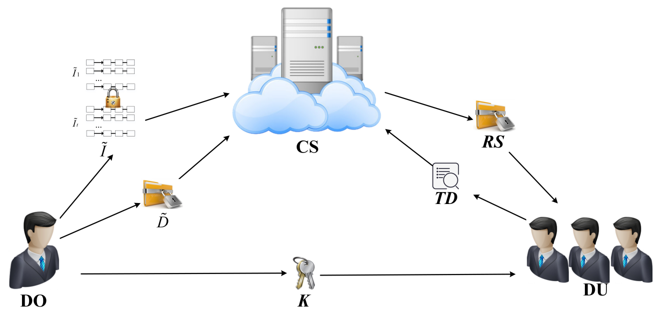

Fig. 1. The system model

2.3 Problem Description

We adopt the “Honest-but-Curious” threat model. In this model, CS honestly and correctly executes instructions in the designated protocol. However, CS can analyze stored data and try to snoop on sensitive information.

The search result of PPTS is represented as RS. V_q is the query vector of Q. V_{d_i} and V_{d_j} respectively represent the document vector of d_i and d_j. Then, RS meets the requirement:

$$|RS| = k \wedge \forall d_i, d_j (d_i \in RS \wedge d_j \in (D - RS)) \rightarrow V_{d_i} \cdot V_q > V_{d_j} \cdot V_q.$$

The PPTS should satisfy three goals. First, the contents are directly seen by CS only include encrypted documents, indexes, and trapdoors, that is, the confidentiality of documents, indexes and trapdoors cannot be leaked. Second, PPTS can handle the search requirements of large document sets in parallel with Map-Reduce parallel search framework. Third, PPTS should fully guarantee the accuracy of search, that is, to improve the efficiency without reducing the accuracy.

2.4 Search Framework

To clearly describe the scheme proposed in this paper, we define a framework for the PPTS scheme. As shown in Fig.2, the search model is composed of five modules: *GenKey*, *Setup*, *BuildIndex*, *GenTrapdoor*, and *Search*.

– *Genkey*: DO generate the key for encryption, and share it with DU.

- *Setup*: DO preprocess the document set D, generate a document vector for each document, and encrypt the D.
- *BuildIndex*: DO fragment the D and then construct an indistinguishable inverted index to provide the CS to perform the search service.
- *GenTrapdoor*: DU generate a trapdoor based on the search keywords.
- *Search*: CS perform the top-k search service in parallel according to the TD and $\widetilde{I}$, and return the RS that satisfy the condition to the DU.

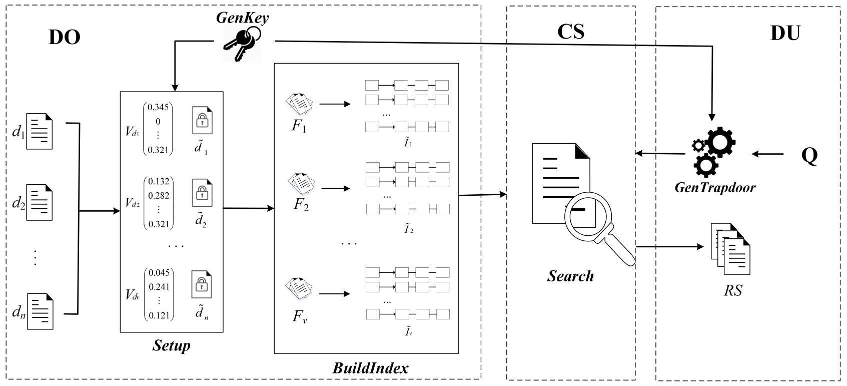

Fig. 2. PPTS Search Framework

3 Parallel Privacy-preserving Top-k Search Scheme

3.1 Fragment-based Encrypted Inverted Indexes Model

Definition 1. *Document Fragmentation.* The document set D is divided into e-qual lengths according to the parameter ε. The generated fragmented documents are denoted as $F = \{F_1, F_2, ..., F_t\}$, satisfying Formula 3 and 4.

$$|F_1| = |F_2| = ... = |F_{t-1}| = \varepsilon, 1 \leq |F_t| \leq \varepsilon \tag{1}$$

$$D = F_1 \cup F_2 \cup ... \cup F_t \tag{2}$$

where $|X|$ represent the number of elements contained in the list or set X.

Definition 2. *Parameter-(δ, β).* δ is the maximum number of documents containing a certain keyword w_j in a certain fragment F_i. And β is the maximum number of keywords contained in a certain fragment of the F.

$$\begin{cases} \delta = max\{|F_{v,j}| \mid 1 < v < t, 1 < j < m\} \\ \beta = max\{|W_v| \mid 1 < v < t, 1 < j < m\} \end{cases} \tag{3}$$

where $F_{v,j} \subset F_v$ is the document set containing the keywords w_j in fragment F_v, and $W_v \subset W$ is the keywords set contained in fragment F_v.

Definition 3.*Fragment-based Encrypted Inverted Indexes.* $\widetilde{I} = \{\widetilde{I}_1, \widetilde{I}_2, ..., \widetilde{I}_t\}$. Here, $\widetilde{I}_v \in \widetilde{I}$ is an encrypted inverted index corresponding to fragment F_v. Each row in $\widetilde{I}_v$ is $<tag_j, \widetilde{PL}_j>$, corresponding to a keyword w_j. tag_j is a hash-based message authentication code of w_j generated by key c, $tag_j = hash(c, w_j)$. $\widetilde{PL}_j$ is the encrypted posting list of w_j. To protect the private information of keyword frequencies, we make the index $\widetilde{I}_v$ corresponding to a fragment $F_v \in F$ has the same number of rows and the posting list in each row has the same number of posting. Thus, the generated indexes $\widetilde{I}$ indistinguishable. The construction of the index $\widetilde{I}_v$ is given as follows:

1) For any $d_i \in F_{v,j}$, the corresponding posting $\widetilde{P}_{j,i} =< id(d_i), \widetilde{V}_{d_i} >$ is generated to form the $\widetilde{PL}_j$. Here, $id(d_i)$ is the id information of document d_i. If $|F_{v,j}| < \delta$, $\delta - |F_{v,j}|$ different artificial padding $\widetilde{P}_{j,s} =< id(d_s), \widetilde{V}_{d_s}{}' >$ are constructed to add to the $\widetilde{PL}_j$. d_s represents a randomly selected document that satisfies $d_s \in F_v - F_{v,j}$. $V_{d_s}{}'$ represents a randomly generated m-dimensional vector, and the values of each dimension are as follows:

$$V_{d_s}{}'[k] = \begin{cases} 0, & k \neq j \\ rand(min\{V_{d_i}[k]\}), & k = j \wedge d_i \in F_{v,j} \end{cases} \tag{4}$$

2) For any $w_j \in W_v$, the corresponding row $< tag_j, \widetilde{PL}_j >$ is generated according to the above steps to form the $\widetilde{I}_v$. If $|W_v| < \beta$, $\beta - |W_v|$ different artificial rows $<tag_s, \widetilde{PL}_s>$ are constructed to add to the $\widetilde{I}_v$. tag_s is generated by randomly selected keyword w_s, where $w_s \in W - W_v$. $\widetilde{PL}_s$ is composed of δ artificial padding generated by the above steps.

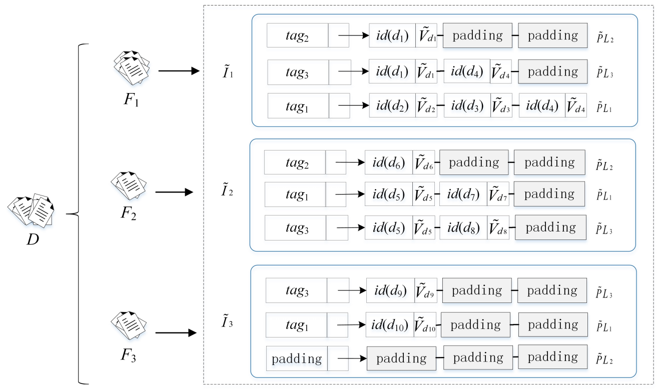
D
F1
F2
F3
Ĩ1
Ĩ2
Ĩ3
tag2 → id(d1) Ṽd1 – padding – padding P̃L2
tag3 → id(d1) Ṽd1 – id(d4) Ṽd4 – padding P̃L3
tag1 → id(d2) Ṽd2 – id(d3) Ṽd3 – id(d4) Ṽd4 P̃L1
tag2 → id(d6) Ṽd6 – padding – padding P̃L2
tag1 → id(d5) Ṽd5 – id(d7) Ṽd7 – padding P̃L1
tag3 → id(d5) Ṽd5 – id(d8) Ṽd8 – padding P̃L3
tag3 → id(d9) Ṽd9 – padding – padding P̃L3
tag1 → id(d10) Ṽd10 – padding – padding P̃L1
padding → padding – padding – padding P̃L2

Fig. 3. Example of $\widetilde{I}$

We take an example to explain the above definitions. We assume $D = \{d_i | i = 1, ..., 10\}$ and $W = \{w_1, w_2, w_3\}$. D is divided to $F_1 = \{d_1, d_2, d_3, d_4\}$, $F_2 = \{d_5, d_6, d_7, d_8\}$, $F_3 = \{d_9, d_{10}\}$. Then, $\delta = 3$ and $\beta = 3$ are calculated. Finally, the indexes $\widetilde{I} = \{\widetilde{I}_1, \widetilde{I}_2, \widetilde{I}_3\}$ are generated as shown in Fig. 3.

3.2 Data Preprocessing and Outsourcing

The data preprocessing and outsourcing of PPTS are mainly performed by DO which include three algorithms: *GenKey*, *Setup* and *BuildIndex*.

$K \leftarrow GenKey(1^{\lambda})$: On input a security parameter λ, the key generation algorithm output the key K. DO randomly generate the key sk, $c \in \{0,1\}^{\lambda}$, an m-dimensional vector S and two $m \times m$ invertible matrices M_1, M_2. Finally, the key $K = (sk, c, S, M_1, M_2)$ is formed. K is shared between DO and DU but is private to CS.

$(\widetilde{V}, \widetilde{D}) \leftarrow Setup(D)$: On input the document set D, this algorithm output the encrypted document vector set $\widetilde{V}$ and encrypted document set $\widetilde{D}$. DU encrypts document d_i into $\widetilde{d_i}$ using sk. Then DU generate document vector V_{d_i} according to VSM and TF-IDF models. The key S is used to split the document vector V_{d_i} into $V^{'}_{d_i}$ and $V^{''}_{d_i}$ according to the following formula, and then the reversible matrices M_1 and M_2 are used to encrypt V_{d_i} to $\widetilde{V_{d_i}} = (M_1^T V^{'}_{d_i}, M_2^T V^{''}_{d_i})$. Finally, the generated $\widetilde{d_i}$ and $\widetilde{V_{d_i}}$ are added to $\widetilde{D}$ and $\widetilde{V}$ respectively.

$$\begin{cases} V^{'}_{d_i}[j] = V^{''}_{d_i}[j] = V_{d_i}[j], & S[j] = 0 \\ V^{'}_{d_i}[j] + V^{''}_{d_i}[j] = D_i[j], & S[j] = 1 \end{cases} \tag{5}$$

Algorithm 1: $BuildIndex(K, \widetilde{V}, D, \varepsilon)$

```
1  Calculate the value of Parameter-$(\delta, \beta)$;
2  for each $F_v \in F$ do
3      for each $w_j \in W_v$ do
4          for each $d_i \in F_{v,j}$ do
5              add $\widetilde{P}_{j,i} =< id(d_i), \widetilde{V}_{d_i} >$ to $\widetilde{PL}_j$;
6          end
7          while $|\widetilde{PL}_j| < \delta$ do
8              add artificial padding $\widetilde{P}_{j,s} =< id(d_s), \widetilde{V}_{d_s}' >$ to $\widetilde{PL}_j$;
9          end
10         add $< tag_j, \widetilde{PL}_j >$ to $\widetilde{I}_v$;
11     end
12     while $|\widetilde{I}_v| < \beta$ do
13         add artificial row $< tag_s, \widetilde{PL}_s >$ to $\widetilde{I}_v$ ;
14     end
15     add $\widetilde{I}_v$ to $\widetilde{I}$;
16 end
17 return $\widetilde{I}$
```

$\widetilde{I} \leftarrow BuildIndex(K, D, \widetilde{V}, \varepsilon)$: This algorithm is run by DU to generate encrypted indexes. Its inputs are the key K, the document set D, the encrypted document vector set $\widetilde{V}$, and the fragmentation parameter ε, and output the $\widetilde{I}$. Procedures of this algorithm is shown in Algorithm 1 where DO first divides D into fragments and then builds an encrypted index for each fragment. Since the operations on each fragment are exactly the same after fragmented, the data preprocessing stage can be executed in parallel. Finally, DO outsource the $\widetilde{D}$ and $\widetilde{I}$ to CS.

3.3 Map-Reduce-based Top-k Search

The Map-Reduce-based top-k search phase of PPTS is performed by DU and CS. DU generate a search trapdoor TD and submit it to CS. CS perform the Map operation according to TD to obtain the k documents most relevant to each fragment, and then perform *Reduce* operation to merge and rank the previously acquired documents to generate the final top-k results. This phase mainly contains two polynomial-time algorithms: *GenTrapdoor* and *Search*.

$\boldsymbol{TD \leftarrow GenTrapdoor(K, Q, k)}$: This algorithm takes a plaintext query containing the key K, the search keyword set Q, and the number of documents to be returned k, and outputs the encrypted query as a trapdoor TD. Its goal is to protect the keyword information in the query from CS. The construction process of TD as the following steps:

1) The query vector V_q is constructed according to Q. If $w_i \in Q$, the IDF of w_i is stored in $V_q[i]$, otherwise, the value of $V_q[i]$ is 0. Then, according to the following formula, V_q is split into two vectors $V_q^{'}$ and $V_q^{''}$. Finally, $V_q^{'}$ and $V_q^{''}$ are encrypted with reversible matrices M_1 and M_2 to obtain the encrypted query vector $\widetilde{V_q} = (M_1^{-1}V_q{'}, M_2^{-1}V_q{''})$.

$$\begin{cases} V_q^{'}[j] + V_q^{''}[j] = V_q[j], & S[j] = 0 \\ V_q^{'}[j] = V_q^{''}[j] = V_q[j], & S[j] = 1 \end{cases} \tag{6}$$

2) The hash-based message authentication code tag_i of w_i is calculated and constitutes the set $T = \{tag_i \mid tag_i = hash(c, w_i) \wedge w_i \in Q\}$.

3) Output $TD = (T, \widetilde{V}_q, k)$.

Algorithm 2: $Search.Map(\widetilde{I}_v, TD)$

```
1  for each $< tag_j, \widetilde{PL}_j > \in \widetilde{I}_v$ do
2      if $tag_j \in T$ then
3          for each $\widetilde{P}_{j,i} \in \widetilde{PL}_j$ do
4              if $Score(\widetilde{V}_{d_i}, \widetilde{V}_q) > minScore\{RS_i\}$ then
5                  if $|RS_i| = k$ then
6                      Delete the document with the lowest relevance score in $RS_i$;
7                  end
8                  add $< id(d_i), Score(\widetilde{V}_{d_i}, \widetilde{V}_q) >$ to $RS_i$;
9              end
10         end
11     end
12 end
13 return $RS_i$
```

$\boldsymbol{RS \leftarrow Search(\widetilde{I}, TD)}$: When CS receives the trapdoor TD, it performs the top-k search in parallel on the basis of the indexes $\widetilde{I}$, and then returns the result encrypted documents. The standard Map-Reduce model is adopted to find the top-k relevant documents. In the Map stage, local top-k result is obtained in each fragment. In the Reduce phase, all local top-k results are merged to obtain the global top-k result is calculated. Detailed procedures are shown in Algorithm 2 and 3.

Algorithm 3: $Search.Reduce(RS_1, RS_2, ..., RS_t)$

```
1 for $each\ RS_v$ do
2     if $RS_v.Score(\widetilde{V}_{d_i}, \widetilde{V}_q) > minScore\{RS\}$ then
3         if $|RS| = k$ then
4             Delete the document with the lowest relevance score in $RS$;
5         end
6         Obtain the $\widetilde{d}_i$ according to $id(d_i)$, and add it to the $RS$;
7     end
8 end
9 return $RS$
```

According to the structure of index and the the top-k search algorithms, we have that each encrypted inverted index for a fragment is independent and the top-k search follows the Map-Reduce model. Thus, the proposed search scheme is scalable. It means that, when the volume of outsourced data grows, the search efficiency can be preserved by adding servers.

4 SECURITY ANALYSIS

This chapter mainly elaborates PPTS from two aspects of security and efficacy. Security is to analyze the confidentiality of documents, indexes, and trapdoors. Efficacy is to analyze the scalability of PPTS, and prove that it has the capability of parallel search and can store large document sets.

Theorem 1. *PPTS satisfies privacy requirements.*

Proof. First, the symmetric encryption algorithm is used to encrypt documents in PPTS, which can protect the privacy of documents when the key is not leaked. Second, the security of the $\widetilde{I}$ is guaranteed by random mapping of keywords, filling redundant values and matrix encryption. Because of the characteristics of the hash-based message authentication code, attackers cannot recover keyword information according to the codes. The document vectors and query vectors are encrypted by matrix encryption technology. Therefore, it can be fully proved that the indexes and trapdoors of PPTS are confidential. In addition, the problem of correlation between similar trapdoors can be solved by randomly adding redundant values to the search trapdoors.

Theorem 2. *PPTS has parallel execution capability.*

Proof. First, the indexes $\widetilde{I}$ is designed according to the parallel computing framework Map-Reduce, that is, both Map and Reduce phases can be executed in parallel. Second, HDFS as a distributed file system can store large data. On the Hadoop cluster, DO do not need to pay attention to the details of data storage and transmission. It only needs to submit the $\widetilde{D}$ and the $\widetilde{I}$ to Hadoop to provide a secure, stable and effective search service for DU, which has very high practicability. Also, the execution capability can be linearly improved by server expansions, providing almost unlimited processing power.

Theorem 3. *The accuracy and privacy of search are not affected by artificial paddings and rows.*

Proof. We assume that $\widetilde{P}_{j,s} =< id(d_s), \widetilde{V}_{d_s}' >$ is an artificial padding added to $\widetilde{PL}_j$ corresponding to the keyword w_j. $\forall d_i \in F_{v,j}$ satisfies the following equation:

$$Score(V_{d_s}', V_q) = V_{d_s} \cdot V_q = rand(min\{V_{d_i}[j]\}) \times V_q[j] < Score(V_{d_i}', V_q) \quad (7)$$

Therefore, the relevance score corresponding to the padding must be lower than any document in $F_{v,j}$. When searching, DU only focuses on the documents with the highest relevance score, so the added paddings will not affect the accuracy of the search results. By adding paddings and rows, the posting list corresponding to each keyword is equal in length and to each fragment equal in width. Therefore, it is impossible to judge whether it is artificial padding or row based on the length of the posting list. Because $d_s \notin F_{v,j} \wedge d_s \in F_v$, d_s has uniqueness and indistinguishability in posting list PL_j. As a result, the added paddings will not affect the privacy of the search results.

5 PERFORMANCE EVALUATION

To evaluate the performance of PPTS, we implement it on the Hadoop platform and compared time cost with SPTS. Here, SPTS is a sequential privacy-preserving top-k search scheme running on a single server. In other words, SPTS use one server to search each $\widetilde{I}_v \in \widetilde{I}$ sequentially. We extent the New York Times Dataset [19] to generate our experimental dataset which has 3,600,000 documents and 228,623 keywords are extracted. We implement the schemes using Java in Hadoop platform with three servers. Each server has 3.2GHz, 8-core CPU, 16G memory and 1T hard disk. Default parameters are $n = 3,600,000, |Q| = 15, k = 5$, and $\varepsilon = 30,000$ which are the number of documents, search keywords, search documents and fragmentation parameter respectively.

In the following experiments, we evaluate the time cost of searches where one of the parameters n, k, and $|Q|$ changes and the other parameters adopt the default values. The results are shown in Fig.4-6.

Fig.4-6 all show that the proposed PPTS outperforms SPTS in the time cost of ranked searches, and the former saves at least 80% of the time cost compared with the latter. The reason is that both PPTS and SPTS are based on the inverted index. In the inverted index, the number of candidate posting corresponding to search keywords is positively correlated with the number of documents and search keywords, but is not affected by the value of search documents. As the number of documents or search keywords increases, more resources are needed to calculate the relevance score. SPTS only use a single server with limited processing power, which can possibly reach its processing bottleneck and make the search speed slower and slower. However, PPTS use multiple servers to perform the search at the same time, and the task pressure is shared on multiple servers, so the time cost will not increase too much.

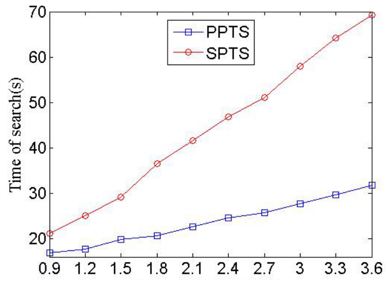

Fig. 4. Number of documents n $(\times 10^6)$

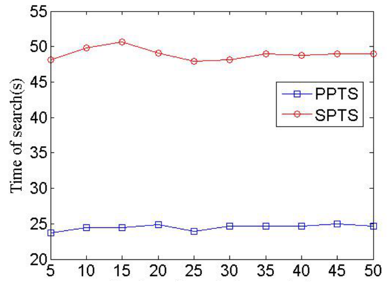

Fig. 5. Number of search documents k

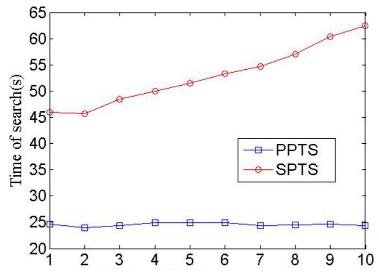

Fig. 6. Number of search keywords $|Q|$

6 CONCLUSION

In this paper, we propose a parallel privacy-preserving top-k search scheme over encrypted cloud data. In this scheme, the fragment-based encrypted inverted index is designed, which is indistinguishable and can be used for parallel searching. On the basis of such indexes, the Map-Reduce-based distributed computing framework is adopted and the parallel multi-keyword top-k search algorithms are proposed. Security analysis and experiment evaluation show that the proposed scheme is privacy-preserving, efficient and scalable.

References

1. L. M. V. González, L. Rodero-Merino, J. Caceres, and M. A. Lindner, "A break in the clouds: towards a cloud definition," *Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.
2. S. Kamara and K. E. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security, FC 2010 Workshops, RLCPS, WECSR, and WLC 2010, Tenerife, Canary Islands, Spain*, January 2010, pp. 136–149.

3. D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA*, May 2000, pp. 44–55.
4. E. Goh, "Secure indexes," *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.
5. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA*, 2006, pp. 79–88.
6. N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, April 2011, pp. 829–837.
7. W. Sun, B. Wang, N. Cao, M. Li, and W. Lou, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, 2014.
8. Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, 2016.
9. X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Inf. Sci.*, vol. 403, pp. 22–41, 2017.
10. C. Chen, X. Zhu, P. Shen, J. Hu, and S. Guo, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, 2016.
11. X. Zhu, H. Dai, X. Yi, G. Yang, and X. Li, "MUSE: an efficient and accurate verifiable privacy-preserving multikeyword text search over encrypted cloud data," *Security and Communication Networks*, vol. 2017, pp. 1 923 476:1–1 923 476:17, 2017.
12. Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.
13. X. Ge, J. Yu, C. Hu, H. Zhang, and R. Hao, "Enabling efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *IEEE Access*, vol. 6, pp. 45 725–45 739, 2018.
14. C. Guo, R. Zhuang, C. Chang, and Q. Yuan, "Dynamic multi-keyword ranked search based on bloom filter over encrypted cloud data," *IEEE Access*, vol. 7, pp. 35 826–35 837, 2019.
15. W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China*, May 2013, pp. 71–82.
16. Y. Yang, Y. Zhan, J. Liu, X. Liu, F. Yuan, and S. Zhong, "Chinese multi-keyword fuzzy rank search over encrypted cloud data based on locality-sensitive hashing," *J. Inf. Sci. Eng.*, vol. 35, no. 1, pp. 137–158, 2019.
17. R. Zhang, R. Xue, T. Yu, and L. Liu, "Dynamic and efficient private keyword search over inverted index-based encrypted data," *ACM Trans. Internet Techn.*, vol. 16, no. 3, pp. 21:1–21:20, 2016.
18. H. Wang, X. Dong, and Z. Cao, "Secure and efficient encrypted keyword search for multi-user setting in cloud computing," *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 32–42, 2019.
19. B. D, "New york times dataset[db/ol]," `http://developer.nytimes.com/docs`, 2018.