 CMC, vol.55, no.1, pp.37-57, 2018

Verifiable Diversity Ranking Search Over Encrypted Outsourced Data

Yuling Liu[1, *], Hua Peng[1] and Jie Wang[2]

Abstract: Data outsourcing has become an important application of cloud computing. Driven by the growing security demands of data outsourcing applications, sensitive data have to be encrypted before outsourcing. Therefore, how to properly encrypt data in a way that the encrypted and remotely stored data can still be queried has become a challenging issue. Searchable encryption scheme is proposed to allow users to search over encrypted data. However, most searchable encryption schemes do not consider search result diversification, resulting in information redundancy. In this paper, a verifiable diversity ranking search scheme over encrypted outsourced data is proposed while preserving privacy in cloud computing, which also supports search results verification. The goal is that the ranked documents concerning diversification instead of reading relevant documents that only deliver redundant information. Extensive experiments on real-world dataset validate our analysis and show that our proposed solution is effective for the diversification of documents and verification.

Keywords: Cloud security, diversity ranking, relevance, searchable encryption, verifiable search.

1 Introduction

Cloud computing is getting increasing attention from both academic and industry communities as it becomes a major deployment platform of distributed applications, especially for large-scale data management systems. At the big data environment, a number of new technologies have emerged. Time optimization models of multiple knowledge transfers in the big data environment are presented by maximizing the total discounted expected profits (DEPs) of an enterprise [Wu, Zapevalova, Chen et al. (2018)]. Cao et al. [Cao, Zhou, Sun et al. (2018)] propose a novel coverless information hiding method based on MSIM, which utilizes the average value of sub-image's pixels to represent the secret information, according to the mapping between pixel value intervals and secret information.

Besides, in cloud storage, a large number of users are planning to upload their data onto

[1] College of Computer Science and Electronic Engineering, Hunan University, Lushan South Road No. 2, Changsha, Hunan, China, 410082.

[2] Department of Computer Science, University of Massachusetts Lowell, One University Avenue, Lowell, M. A., USA, 01854.

* Corresponding author: Yuling Liu. Email: yuling_liu@126.com.

the public clouds. However, data stored in the cloud may suffer from malicious use by cloud service providers since data owners have no longer direct control over data. Considering data privacy and security, it is a recommended practice for data owners to encrypt data before uploading onto the cloud. Although it protects data security from illegal use and untrusted cloud service providers, it makes data utilization more difficult since many techniques based on plaintext are no longer applicable to cipher text. Therefore, exploring a search technique for encrypted data is extremely urgent.

In the encrypted image retrieval field, the data is represented as a series of pictures. Xia et al. [Xia, Xiong, Vasilakos et al. (2017); Xia, Zhu, Sun et al. (2018)] propose two privacy-preserving content-based image retrieval schemes, which allow the data owner to outsource the image database and CBIR service to the cloud, without revealing the actual content of the database to the cloud server. The two methods improve user experience. Furthermore, in the ciphertext data retrieval field, the first searchable encryption scheme (SSE) was proposed by Song et al. [Song, Wagner and Perrig (2000)]. The paper describes a cryptographic scheme for the problem of searching on encrypted data and provides proofs of security for the resulting crypto systems. But the scheme only support single-keyword search. Golle et al. [Golle, Staddon and Waters (2004)] first proposed the construction of conjunctive keyword searchable encryption and presented two schemes (GSW-1 and GSW-2). In GSW-1, the size of trapdoor is linear with the number of encrypted documents. Although bilinear pairings are used to achieve a trapdoor of constant size, GSW-2 shows large overhead on computation. Ballard et al. [Ballard, Kamara and Monrose (2005)] also constructed two conjunctive keyword search schemes over encrypted data, but their schemes have the same drawbacks as Golle et al. [Golle, Staddon and Waters (2004)]. Cash et al. [Cash, Jarecki, Jutla et al. (2013)] proposed the first sub-linear SSE scheme supporting conjunctive queries for arbitrarily structured data and proved the IND-CKA2 security of the proposed scheme. Xia et al. [Xia, Wang and Sun (2016)] present a secure multi-keyword ranked search scheme over encrypted cloud data, which simultaneously supports dynamic update operations like deletion and insertion of documents. Ahmad et al. [Ahmad and Kumar (2017)] proposed a novel method by combining LSI and hierarchical cluster to get the semantic relation between the results and to reduce the search space respectively. To enrich search semantics, Fu et al. [Fu, Sun, Linge et al. (2014); Fu, Ren, Shu et al. (2016); Fu, Huang, Ren et al. (2017)] adopt different methods achieving semantic search and more smart. Strizhov et al. [Strizhov and Ray (2016)] propose a novel secure and efficient multi-keyword similarity searchable encryption that returns the matching data items in a ranked order manner. These schemes have not considered the query keyword's spelling mistake. Fu et al. [Fu, Shu, Wang et al. (2015); Fu, Wu, Guan et al. (2017)] propose multi keyword fuzzy ranked search scheme that is able to handle spelling mistakes. But they do not support dynamic update and effective rank results. Li et al. [Li, Wu, Yuan et al. (2016)] propose an efficient and effective scheme which support ranked multi-keyword fuzzy search over encrypted data and document dynamic update. However, the above schemes ignore diversity of the result documents. Moreover, it will cost users much time and many resources to filter the real interesting ones among a large quantity of returned files. In the information retrieval (IR), search result diversification approaches have been proposed to produce rankings aimed to satisfy the multiple possible information needs underlying a

query. In general, search result diversification in the Information Retrieval (IR) may be a good choice. However, the scheme cannot be directly applied to searchable encryption schemes due to the lack of consideration of privacy and security.

Besides, majority of work in this area assumes that the cloud server is honest-but-curious. But, in real world, search results may contain corrupted data due to the underlying hardware/software failures and inevitable human errors. Furthermore, the cloud server may return false results in order to save its computation cost or because of the attacks from hackers. So it is very necessary to provide users with a verifiable mechanism to assure the correctness and the completeness of search results.

In this paper, a flexible searchable encryption scheme is proposed, which supports multi-keyword search, diversity ranking and results verification. In view of storage problem, stem segmentation technique is used to extract keyword for each document and to build stem set. This method can obtain higher storage efficiency. To address multi-keyword search, Bloom Filter [Bloom (1970)] is used to build document index. That is to say, each document is expressed as a vector where each dimension value is TF-IDF (term frequency-inverse document frequency) of its corresponding keyword stem. And diversity equilibrium model [Santos, Rodrygo, Macdonald et al. (2010)] is used to establish diversity ranking algorithm. Then cosine measure can be used to compute similarity of one document to the search query. The scheme also supports the verification of search results by exploiting MAC technology.

In short, the contributions are summarized as follows:

1) Firstly, a diversity ranking search scheme over encrypted outsourced data while preserving privacy is proposed. Diversity ranking search indicate that the search results are as diverse as possible. By doing so, information redundancy is reduced.
2) Secondly, stem segmentation is used to build stem set for each document. It saves vast storage space.
3) Thirdly, the scheme also achieves the verification of search results by exploiting MAC technology. HMAC-SHA1 is selected when building bloom filter. It ensures the security of scheme.
4) Fourthly, theory analysis and experimental results on the real-world dataset show our proposed schemes are efficient and feasible.

The rest of this paper is organized as follows. Section 2, some related research are discussed. Section 3 presents the system model, threat model and our design goals and then briefly describes some notations and background knowledge used in this paper. Section 4 depicts the basic design of our scheme and detail. Section 5 depicts security analysis and performance evaluation. Finally, the paper concludes with some suggestions for future work.

2 Related work

2.1 Verifiable search

In the semi-honest-but-curious model, the cloud server may return incorrect results to users. To resist such threats, Pang et al. [Pang and Mouratidis (2008)] propose a scheme for checking the validity of search results based on the authentication structure of Merkle

hash tree. But the scheme is unable to protect the search privacy which is quite important in the cloud environment. Wang et al. [Wang, Cao, Ren et al. (2012)] construct a single keyword search scheme with search result verification using the hash chain. Lu [Lu (2012)] achieve query authentication using the verification method in plaintext field. The method is not suitable for encrypted cloud data. Kurosawa et al. [Kurosawa and Ohtaki (2012)] formally define the security against active adversaries including privacy as well as reliability, and propose the first UC-secure verifiable single-keyword search scheme. The communication overhead and verification cost grow linearly with the cardinality of document collection. Sun et al. [Sun, Liu, Lou et al. (2015)] exploit a bilinear-map accumulator tree as the authenticated data structure and presented an efficient verifiable conjunctive keyword search scheme.

However, the verification mechanism works at the expense of leaking all document lists that match each keyword in one query. In the literature, Miao et al. [Miao, Ma, Wei et al. (2017)] design a verifiable scheme without secure channel to assure data integrity and availability, but it is unable to return sorted results. And the above verification mechanisms lack the diversity ranking verification.

2.2 Diversity ranking in the information retrieval

In the Information Retrieval, many diversity ranking search schemes are proposed. Xia et al. [Xia, Xu, Lan et al. (2016)] propose to model the novelty of a document with a neural tensor network. Instead of manually defining the similarity functions or features, the method automatically learns a nonlinear novelty function based on the preliminary representation of the candidate document and other documents. Sundaresan [Sundaresan (2015)] propose a method for diversity ranking search based on aspect affinity includes collecting user search queries, parsing the collected user search queries for aspect phrases, identifying aspect metadata for the aspect phrases, creating a ranked index list of aspects from the aspect metadata. Ren et al. [Ren, Chen, Ma et al. (2016)] propose a novel User Session Level Diversification (UserLD) approach based on the observation that a query's subtopics are implicitly reflected by the search intents in different user sessions. Xu et al. [Xu, Li, Zhang et al. (2016)] propose a new web page ranking algorithm after analyzing the link diversity and content features distribution of the web pages. In this method, the web pages ranking score is calculated by the TrustRank method combining web pages links diversity and the web pages content features. Li et al. [Li, Liu, Liu et al. (2015)] propose a novel semantic-based approach to achieve the diversity-aware retrieval of Electronic Medical Records. But the scheme cannot be directly applied to searchable encryption schemes due to the lack of consideration of privacy and security. In this paper, a diversity equilibrium model is used to diversity ranking in the encrypted data. Moreover, hinge concept is used to reduce the times of distance calculation at diversity selection. In this case, the efficiency of diversity ranking is improved.

3 Problem formulation

3.1 System model

A complete system model in cloud computing should involve three different entities: the data owner, the data user and the cloud server. The scheme's system model is shown in Fig. 1.

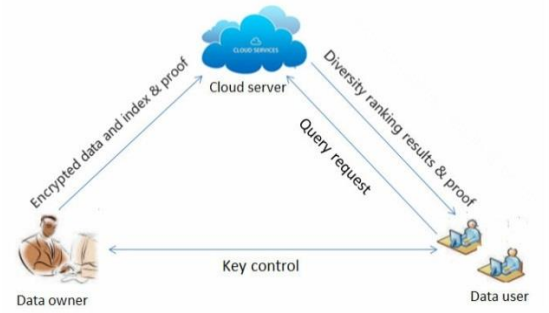

Figure 1: Model of the verifiable diversity ranking search over encrypted cloud data

Firstly, data owner build index and validation set for data, also encrypt them and data where uploaded to the cloud server. Secondly, to search for the interesting files, the data user should create a search request. Thirdly, the encrypted search query by key control mechanism, e.g. broadcast encryption, will be sent to the cloud. Upon receiving the search request from the authorized user, the cloud server will conduct designated search operation over the index and send back the relevant encrypted documents, which have been well ranked by the cloud server according to some diversity ranking criteria. And the cloud server returns the most relevant encrypted documents as well as the verification proof. Fourthly, the data user can verify the validity of the search outcome by the proof from the cloud server. If it is valid, the data user locally decrypts the received ciphertexts with a secret key; otherwise, rejects them.

3.2 Notations

- *C*-The plaintext document collection, denoted as $C=\{C_1,\ C_2, \ldots,\ C_N\}$
- *E*-The encrypted document collection
- *h*-The set of hash functions for building index
- *BF*-An index Bloom filter
- *l*-The number of hash
- key_{doc}-The key to encrypt document before outsourcing
- *qbf*-A set of query Bloom filters for a query request
- $W_{t,d}$-The weight of *t* keyword in the document *d*
- *RD*-Resulting documents collection, denoted as $RD=\{rd_1,\ rd_2, \ldots,\ rd_S\}$
- *P*-The hinge documents collection
- μ-Balance parameters
- Er-The ranked result documents
- *addr*-The hash value of keyword
- *tag*-Message authentication code

3.3 Threat model

In this work, we consider the honest-but-curious model which is commonly used in the existing works [Yu, Wang, Ren et al. (2010); Vimercati, Foresti, Jajodia et al. (2007)]. Specifically, the cloud server is not completely trusted and will act in an 'honest' manner and follow our proposed protocol in general. At the same time, the cloud server is 'curious' to infer as much secret information as possible from encrypted documents, index stored on it and messages received during the service. In our scheme, the data owner and the authorized user are trusted. We also assume that all the communication channels between the data owner/authorized users and cloud server are secured by existing security protocols such as SSL, TLS, etc. Based on the system model in Fig. 1, we consider the following two types of attacks.

1) **Known Ciphertext Attack Model (KCAM):** In this model, the attacker only masters the encrypted documents and the retrieval index at cloud, which are outsourced from the data owner. This model is the most basic attack model. All the cloud storage applications are subjected to this security threat.
2) **Known Plaintext Attack Model (KPAM):** This is a stronger attack model. In this model, besides the encrypted documents and the index, the attacker masters more information including the generation mechanisms of retrieval index and query requests, even part of the plaintext of the original documents. In this case, the attacker could use the known index/request generation mechanism with document/word frequency and other document statistical information to deduce/identify some private information.

3.4 Design goals

Our design should achieve the following goals:

1) Diversity Ranking Search: The goal is that the ranked documents concerning diversification instead of relevant documents that only deliver redundant information.
2) Search Results Verification: The scheme can verify the authentication of search results by checking whether all the returned ranking documents remain unmodified, whether unqualified documents are returned and whether results documents are ranked.
3) Privacy Preserving: Our scheme should not leak any privacy under our carefully defined security model. In the searching phase, we are concerned with privacy requirements: keyword privacy, index confidentiality, query confidentiality.

3.5 Preliminaries

Stem Segmentation: In this paper, stem segmentation is a process of linguistic normalisation, in which the variant forms of a word are reduced to a common form. A stemmer for English, for example, should identify the string "acute" (and possibly "acumem", "acupuncture", etc.) as based on the root "acu", "automation", "autobiography", and "autosuggestion", as based on "auto". On the other hand, "argue", "argued", "argues", "arguing", and "argus" reduce to the stem "argu" (illustrating the case where the stem is not itself a word or root). It is adopted to save storage space and improve search efficiency.

Bloom Filter： Bloom filter is a kind of data structure with very high space efficiency. It

makes use of the m-bit array to represent a document, and can determine whether a keyword belongs to the document. It is initially set to 0 in all positions. A bloom filter uses l independent hash functions $h_1, h_2, \ldots, h_l$, with range $\{0, 1, \ldots, l\text{-}1\}$. These hash functions map the data to a random number uniform over the range $\{0, \ldots, l\text{-}1\}$. For each keyword $w \in W$, the bits $h_i(w)$ $(1 \le i \le l)$ are set to 1. To check if an item y is in W, we check whether all $h_i(y)$ $(1 \le i \le l)$ are set to 1. If not, then obviously y is not a member of W. If all $h_i(y)$ are set to 1, we assume that y is in W, at times, there are wrong with some probability. Hence, a bloom filter may yield a false positive, where it suggests that an element y is in W even though it is not. For many applications, this is acceptable as long as the probability of a false positive is sufficiently small.

Keyword Weight: Keywords are used to summarize document content. In order to express keyword's significance to the document, we adopt the most widely statistical measurement "*TF*×*IDF*", where *TF* (term frequency) is the occurrence of the term appearing in the document, and *IDF* (inverse document frequency) is usually obtained by dividing the total number of document collection by the number of documents containing the term. Specially, *TF* represents the importance of the term within a document and *IDF* indicates the importance or degree of distinction within the whole document collection. Here we calculate the keyword weight with the formula below:

$$Weight_{w,C} = tf_{C,w} \times \log_2 idf_w \quad (1)$$

Where $tf_{C,w}$ is the *TF* of the term w in the document C. idf_w is the *IDF* of the term w.

4 Verifiable diversity ranking search scheme

4.1 Framework

The processing flow of the scheme over the encrypted data shows in Fig. 2. While the cloud server begins to provide the storage services, the data owner and cloud server set up the global system parameters (h, m, *keys*) to initialize the cloud storage system (**Setup**). Before outsourcing documents, the data owner carries out data processing (**Documents Processing**). Then, the data owner uses the hash functions and MAC functions to build index (**Building Index**). Finally the data owner uploads the encrypted documents and the corresponding index to the cloud.

When an authorized user wants to query the data in the cloud with certain query words, the user generates the query Bloom filters using the key key_{index} and the hash functions (**Generate Query**). After the cloud server receives the query request, it executes the retrieval over the index (**Search**). Then, the cloud server ranks the results with diversity algorithm (**Diversity Ranking**). To prevent returning inaccurate search results, once receiving the results, data user tests their correctness and completeness (**verify**).

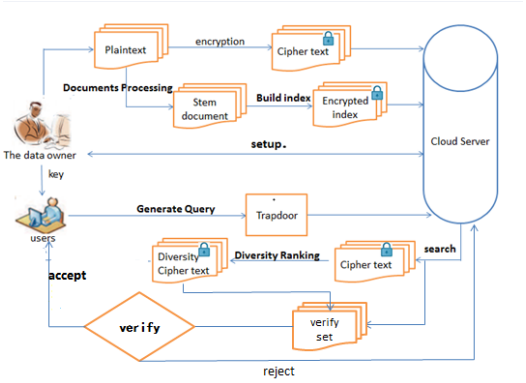

Figure 2: Framework of the verifiable diversity ranking search

4.2 Setup

In this phase, firstly, the cloud server and data owner set up the global system parameters (h, m, *keys*) to initialize the system. h represents a set of hash functions (h_1, h_2, ... , h_l), h_i:{0, 1}$*\rightarrow$[1, l]($1 \le i \le l$). m is the bit length of bloom filter. To protect data privacy, the data owner utilizes the symmetric encryption algorithm to encrypt the documents before outsourcing. The *keys* {key_{doc} and key_i, ($1 \le i \le l$)} of encrypted index are stored by data owner. key_{doc} is used to encrypt the original document. key_i, ($1 \le i \le l$) are used in the hash function. That is to say, in the process of building bloom filter, key_i($1 \le i \le l$) are used to map keyword stem set. Secondly, the data owner can distribute the *keys* to the authorized users through secure communication channels.

4.3 Documents processing

Original document is encrypted before uploading to the cloud server. Therefore, the data owner needs to build document index. But, before building index, the data owner must extract keywords for each document. We first extract keywords from C to build a keyword stem set W={w_1, w_2, ..., w_n}. We apply the stem segmentation to ascertain the root of the word. For example, for the following set of words: “walk”, “walks”, “walking” and “walked” all have a similar meanings, but they also display certain distinctions. In this case, if we query the keyword “walking”, but the keyword in index is “walk”, the probability of finding the keyword “walking” is low because the distance between “walk” and “walking” is too large. In fact, the data owner is to denote the keyword with the same root into the same form. The data owner can confirm the root word and find the corresponding files. Meanwhile, the method save storage space and improve search efficiency. Finally, for the constructed stem set, we compute the weight between the files and stems.

4.4 Building index

At the above subsection, the data owner collects all the keywords stem and calculates their relevance between the files and stems. Then the data owner uses the hash functions h_i ($1 \le i \le l$) and the key *keys* {key_{doc} and key_i, ($1 \le i \le l$)} to generate bloom filter bf[j] =

$\{ bf_1[j], bf_2[j], \ldots, bf_N[j]\}$, $1 \le j \le m$, for the document C_j, $1 \le j \le N$. In this process, for each keyword, the bits h_i (W) $(1 \le i \le l)$ are set to the weight of keyword $Weight_{C,w}$.

$$bf_k[j](1 \le j \le m,\ 1 \le k \le N) = \begin{cases} Weight_{C,w} & if\, h_i(W||key_i) = j(1 \le i \le l) \\ 0 & else \end{cases} \quad (2)$$

For each bloom filter, the data owner builds verification set, *V1* and *V2*. These verifiable sets are used to check the authentication of search results.

(1) Building V1

V1 is used to verify if the retrieved documents satisfy all the query keywords. Each entry in *V1* corresponds to a keyword stem and consists of two fields << *addr*, *tag* >>. The set is described at Fig. 3.

$addr_1$	$tag1_1$
$addr_2$	$tag1_2$
$addr_3$	$tag1_3$
$addr_4$	$tag1_4$
$addr_5$	$tag1_5$
⋮	⋮

Figure 3: *V1* set

The field $addr\{addr_i=h(w_i), 1 \le i \le n\}$ stores the output of a hash function about a keyword stem, which is used to locate an entry in *V1*. The field *tag* stores the verifiable information which is used to check the authentication of search keywords stem. And $tag1_i=MAC(addr_i,\ CID_j)$, $(1 \le i \le n,\ 1 \le j \le N)$. CID_j is the unique identifier of document CID_j. For each document, there is a matching set where each element is the identifier of a document containing the keyword stem.

(2) Building V2

V2 should verify if the retrieved documents is tampered. The set is described at Fig. 4.

EID_1	$tag2_1$
EID_2	$tag2_2$
EID_3	$tag2_3$
EID_4	$tag2_4$
EID_5	$tag2_5$
⋮	⋮

Figure 4: *V2* set

We need to compute the authentication information for each encrypted document and upload them to the cloud server. We encrypt the *C* and *CID*. The encrypted document and *CID* is represented *E* and *EID*. $tag2_j=MAC\ (EID_j,\ E_j)$, $(1 \le j \le N)$. In the verification

stage, users calculate *MAC* (EID_j, E_j). If they are not tampered, the *MAC* (EID_j, E_j)= $tag2_j$.

Finally, the data owner encrypts the document with key_{doc} and uploads the encrypted document, the bloom filters and verification sets to the cloud server.

4.5 Generate query

When an authorized user wants to search files at cloud, the user provides some original query keywords like using web search engine application. Then, the query keywords are processed. In fact, the user is to denote the keywords with the same root into the same form. That is to say, users use stem segmentation to deal with keywords. Given a set of *t* query keyword stems $Q=\{q_1, q_2, ..., q_t\}$, If the user does not give the weight of query keyword stems, the weight of every query keyword stem is set '1'. Afterwards, the query bloom filters *qbf* [j], $1 \le j \le m$, is generated for all the query keyword stems using the set of hash functions h_i ($1 \le i \le l$) in the Eq. (3). Finally, the user submits *qbf* to the cloud server as the query requirement.

$$qbf\ [j](1 \le j \le m) = \begin{cases} 1 & if h_i(Q||key_i) = j(1 \le i \le l) \\ 0 & else \end{cases} \quad (3)$$

4.6 Search

The cloud server uses "secure inner product" [Cao, Wang, Li et al. (2011)] to compute the relevance scores of each document as the following Eq. (4). If the relevance score is greater than *T*, the document is selected as result document. In the scheme, the *T* is 0, because the scheme need search all similar documents.

When calculate the similarity value of the document to the query or between two documents, two vectors are used: the index bloom filter *bf* and the query bloom filter *qbf*.

$$\mathrm{rel}(bf, qbf) = \frac{\sum_{i=1}^{m} bf_i \times qbf_i}{\sqrt{\sum_{i=1}^{m} (bf_i)^2} \times \sqrt{\sum_{i=1}^{m} (qbf_i)^2}} \quad (4)$$

4.7 Diversity ranking

At the above subsection, cloud server obtains a series of result documents, $RD=\{rd_1, rd_2, \ldots, rd_s\}$. Then, cloud server need to rank the results. In this paper, a diversity ranking equation is proposed according Santos et al. [Santos, Rodrygo, Macdonald et al. (2010)]. This equation is used to calculate diversity score.

$$\mathrm{dscore} = (1 - \mu)\mathrm{rel}(\mathrm{bf}, \mathrm{qbf}) + \mu(1 - divMax(rbf, pbf)) \quad (5)$$

Where a bloom filter *bf* of documents *C* is scored with respect bloom filter *qbf* of a query *q* based on a linear combination of relevance (*rel(bf, qbf)*) and diversity ($divMax$*(qbf, bf)*), with the interpolation parameter μ trading off between the relevance and diversity. *rbf* is bloom filter of result document *RD* and *pbf* is bloom filter of hinge document set *P*. And we use Eq. (4) to calculate *rel (bf, qbf)*. In addition, we use hinge concept to reduce the times of distance calculation at diversifying selection. The distance formula *d(rbf, pbf)* is adopted at building hinge document set. If the distance between rbf_i and pbf_i is greater than a threshold value of distance, rd_i is selected.

$$d(rbf, pbf) = \sqrt{\sum (pbf_i - rbf_i)^2} \quad (6)$$

At Algorithm 1, the hinge document set *P* is calculated. Firstly, let *P* be an empty set. For all results document *RD*, rd_1 is the first document to be searched and rd_1 is added to *P*. Secondly, cloud server calculate distance of remainder result document $rd_i(1 \le i \le s)$ and each hinge document set *P*. If the distance is greater than the threshold *st*, cloud server adds rd_i to *P*.

Algorithm 1: **calculating the hinge document set**

Input: RD, resulting documents; st, the threshold value of distance;

Output: P, the hinge document set

1 for (i=1; RD.size,i++) {

2 if (P==null) P = {rd_1}

3 else if (d(rbf_i, pbf_j)≥st,$\forall_{p_j} \in P$)

4 P={rd_i}

5 }

The *divMax (rbf, pbf)* is described in the Algorithm 2. For all hinge document sets *P*, cloud server calculate relevance score of result document $rd_i(1 \le i \le s)$ and each hinge document set. The equation is *rel*(*rbf*, *pbf*). Then, cloud server selects a maximum score *mscore*, but the score is not equal to 1.

Algorithm 2: **divMax (rbf, pbf)**

Input: rbf, the bloom filter of result documents; pbf, the bloom filter of hinge document set

Output: mscore, maximum relevance score

1 for (i=1; P.size,i++){

2 if (mscore ==0 && rel(rbf_i, pbf_i) != 1) mscore == rel(rbf_i, pbf_i)

3 else if (mscore <= rel(rbf_i, pbf_i) && rel(rbf_i, pbf_i) != 1)

4 mscore == rel(rbf_i, pbf_i)

5 }

Cloud server uses the final diversity score *dscore* to rank the result documents. Then, cloud server builds *V3*. *V3* is consists of fields << *EID*, *tag3* >>. $tag3_i$=*MAC* (num_i, EID_i), $(1 \le i \le s)$. num_i is sequence number of diversity ranking.

4.8 Verify

The data user verifies the validity of the returned encrypted documents as follows.

1. User receive *V1*, *V2*, *V3, EID* and result documents E_{r_j}, *j=1,2,…,n.*
2. Decrypt the EID. It is represented *CID*. And compute *MAC* ($addr_{querykeyword}, CID_j$). For all *CID* of result documents, check if *MAC* ($addr_{querykeyword}, CID_j$)=$tag1$. If not, output "reject".
3. Compute *MAC* (EID_j, E_{r_j}). Parse the *MAC* set in *V2* as {$tag2$}. For all C_{r_j}, check if *MAC* (EID_j, E_{r_j})=*tagT*. If not, output "reject".
4. $numrec_j$, $(1 \le j \le s)$ is the sequence number of received documents. Compute

$MAC(EID_j, numrec_i)$. Searching in the $V3$, if exist mismatch, output "reject". For all E_{r_j}, check if $MAC(EID_j, numrec_j)$=$tag3$. If not, output "reject".

5 Performance and security analysis

In this section, performance analysis of our proposed search scheme over encrypted data is presented. All the algorithms mentioned are implemented in the paper on a 2.10 GHZ AMD processor, Windows 8.1 operating system with a RAM of 8 GB. In the experiments, we choose a publicly available real dataset: the RFC [1-7000] and get the root of every keyword with a well-known stemming technique called Porter Stemming Algorithm [Porter (2006)]. In the experiments, we use the keyed hash function HMAC-SHA1 [Bellare and Krawczyk (1996)] to build the Bloom filter. And the numbers of key in HMAC-SHA1 are 2 and 3 (l=2 and l=3).

The performance of the scheme is evaluated by the time of index construction, index storage, searching time, recall rate, diversified evaluation and query precision.

5.1 Index construction

The index construction contains four steps: keyword extraction, calculating TF-IDF, building bloom filter and building verification set. Given the document set constructed by using bloom filter, the time cost of index construction for the basic scheme is measured. It is obvious that the time cost of the index construction is mainly affected by the number of documents in the dataset. Each entry in the bloom filter is associated to a keyword in the keyword set. To get an encrypted entry of bloom filter, the data owner needs hash function. The computation complexity of building the bloom filter is $O(m)$, where m represents the size of keyword set. Each array stores the identifiers of all documents containing the associated keyword. The time cost of generating each array varies from one keyword to another keyword. Fig. 5 shows the time consumption to generate the bloom filter with different sizes of the documents. The time is approximately linear. In the index construction, the step of building bloom filter is the major computation and takes up most of the time. The other two steps (keyword extraction and calculating TF-IDF) are quite efficient. Fig. 6 shows the time of verification set construction, including three set: *V1*, *V2* and *V3*. The data owner needs to compute MAC. From the Fig. 6, because of verification set construction, time of index construction has increased so much. But it basically conforms to actual requirement.

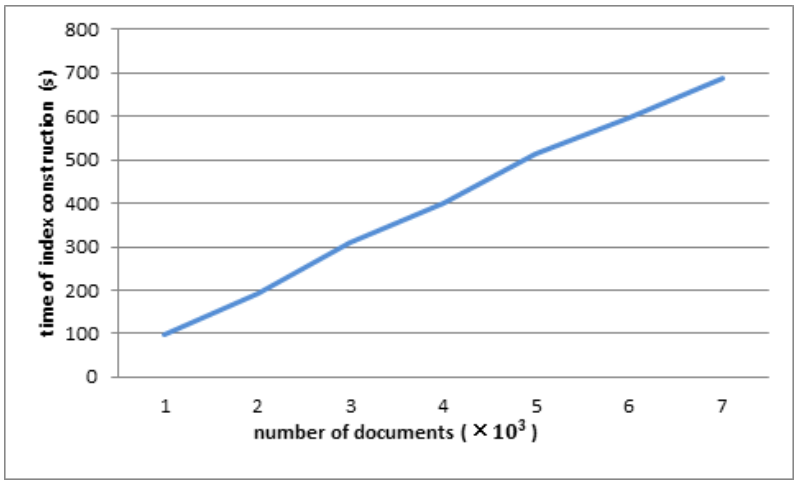

Figure 5: Time of Bloom filter Construction

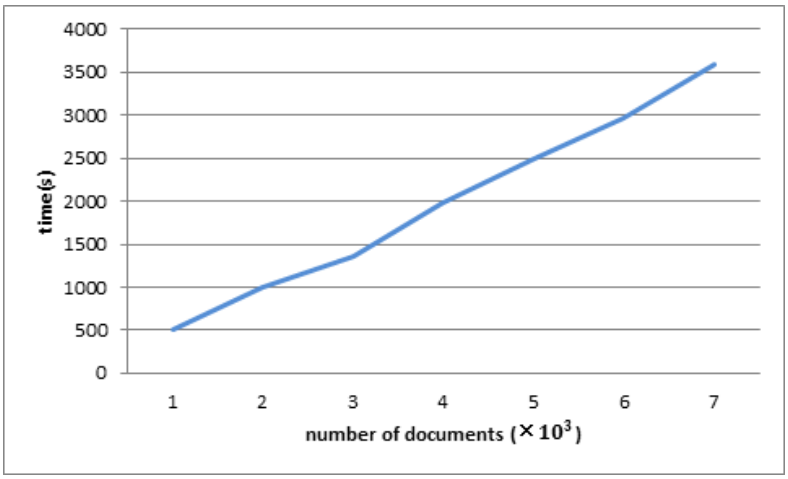

Figure 6: Time of verification set Construction

5.2 Index storage

In the cloud computing environment, the storage space is an important problem. In this paper, stem segmentation technique is used to extract keyword for each document and to build stem set. It greatly saves storage space. Fig. 7 shows the storage overhead of stem segmentation and keyword extract for the different sizes.

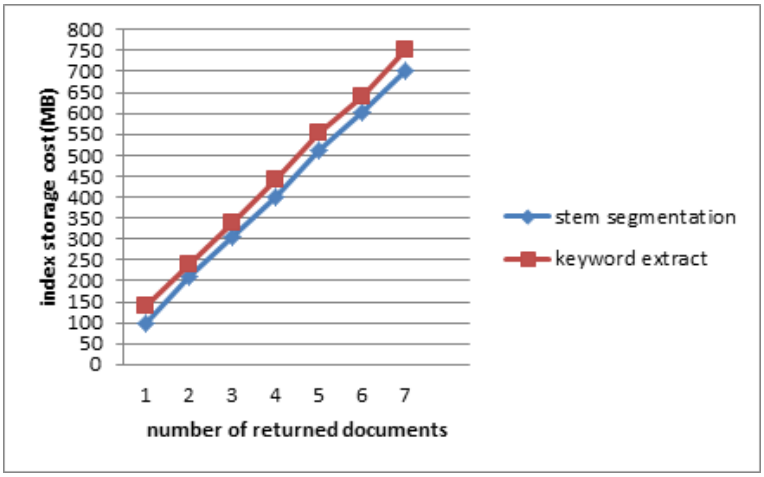

Figure 7: Index storage cost

5.3 Search

In this section, the performance of search scheme is evaluated. The search process consists of two steps: retrieving the documents that match the estimated keyword and sorting the results to acquire the diversity. The first part, its search complexity is proportional to the number of documents containing query keyword. In this part, an important parameter is m (the length of the bloom filter). In the scheme, m=20000. The remaining part, its search complexity is proportional to the diversity ranking. Fig. 8 shows the search time for the scheme. The experiment compares the time consumption of different documents. In short, the relationship between number of documents and search time are approximately exponential.

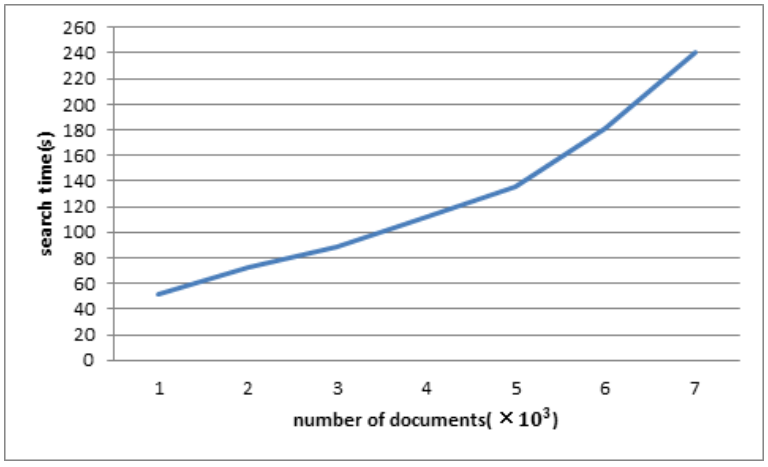

Figure 8: Search time of the scheme (For the different size of dataset)

5.4 Recall rate

We use the recall rate [Song, Wang, Wang et al. (2016)] to check the full rate of the document. For a query, D_{match} represents the documents which correctly match it in a searchable encryption system. And D_{return} represents the whole documents returned from the server.

$$\text{Recall} = \frac{\mathrm{D}_{match} \cap \mathrm{D}_{return}}{\mathrm{D}_{match}} \times 100\% \tag{7}$$

By experiment, we prove that our scheme could achieve 100% query recall rate and find all the encrypted documents which satisfy the user's query.

5.5 Diversified evaluation

In this section, we adopt three commonly diversified evaluation: ERR-IA@K [Chapelle, Metlzer, Zhang et al. (2009)], α-n DCG@K [Clarke, Kolla, Cormack et al. (2008)], MAP -IA [Hersh, Cohen and Yang (2005); Tomlinson (2006)]. These are standard evaluation practice in TREC (the Text Retrieval Conference). The *K* is 20 (the number of document). First of all, we figure up these evaluations on the basis of the original resulting document. Secondly, we calculate these evaluations on the basis of the resulting document of diversification. We compare their data in the Tab. 2.

Table 1: The description of the evaluation

	ERR-IA@20	α-n DCG@20	MAP-IA
computing method	$\frac{1}{K}\sum_{i=1}^{K}\frac{1}{rank_i}$	$\sum_{i=1}^{K}\frac{number_{doc}}{\log(1+rank)}$	$\sum_{i=1}^{Nq}\frac{P_i(r)}{Nq}$

Rank is the result document's location. $P_i(r)$ is the precision of query *i* when the recall ratio is *r*. Nq is the number of query.

Table 2: Diversification evaluation's comparison

	ERR -IA@20	α-n DCG@20	MAP-IA
Original resulting document	0.201341	0.26512	0.04104
resulting document of diversification	0.303201	0.39054	0.05013

From Tab. 2, the ERR-IA@20 is improved 50.6% and α-n DCG@20 is improved 47.3%. MAP-IA is improved 22.1%. It is observed that our scheme has obvious improvement of performance in the aspect of diversity.

5.6 Query precision

The query precision rate indicates the ratio of exactly relevant documents to all the return documents as illustrated in Eq. (8). The query precision of our scheme is mainly related to the false positive probability by the Bloom filters, so we discuss the false positive probability to analyze the query precision in our scheme.

$$\text{Precision} = \frac{\mathrm{D}_{match}}{\mathrm{D}_{return}} \times 100\% \tag{8}$$

In this scheme, we evaluate the query precision with different h (hash function) and m (width of Bloom filter) values. In the experiment, m=15000 and m=20000, h=3 and h=4, μ=0.3. The μ is obtained by the experiment test Fig. 9. From Fig. 3, when μ is 0.3, all evaluations are better. In Fig. 10, at the m=15000, the query precision respectively are 85% and 91%. At m=20000, the query precision respectively are 89% and 95%.

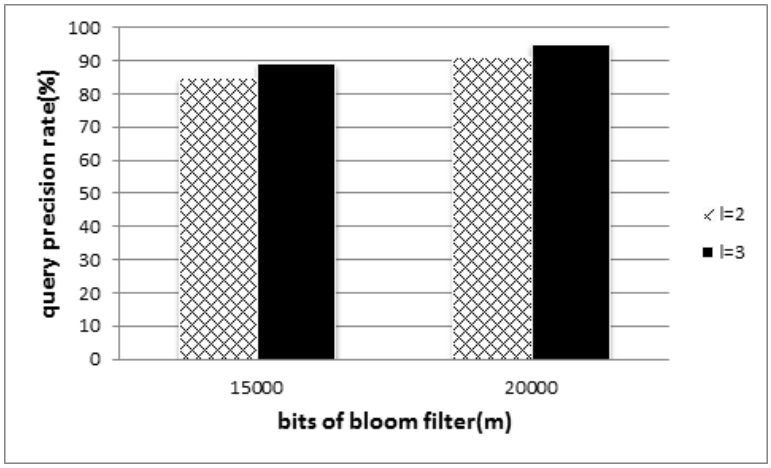

Figure 9: The evaluation of different μ

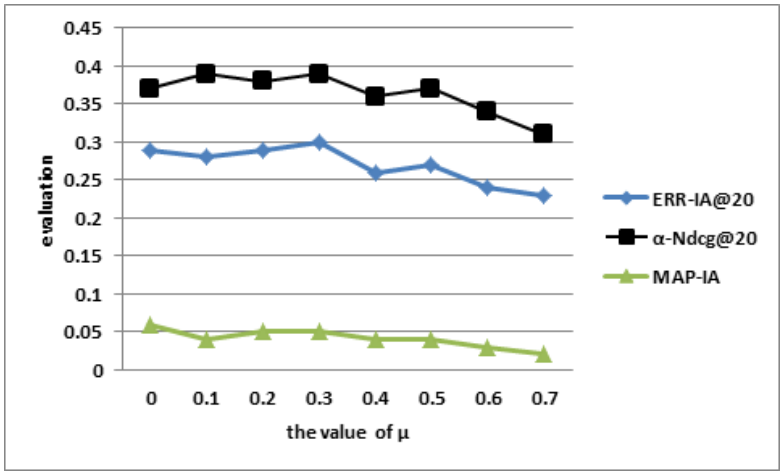

Figure 10: Query precision

5.7 Verification efficiency

In the scheme, after receiving the returned results and the proof set from the cloud server, the user needs to compute a MAC and examines whether the calculated value is equal to value of proof set. The MAC takes the concatenate of the query keyword and the returned documents as input. As Fig. 11 shows, when t is constant, the verification time is linear with k, where k is the number of returned documents. When the value t increases, the change of time is not very noticeable. When the number of relevant documents desired by

the user is up to 105 and the number of query keywords is 4, the verification time only needs less than 10 ms.

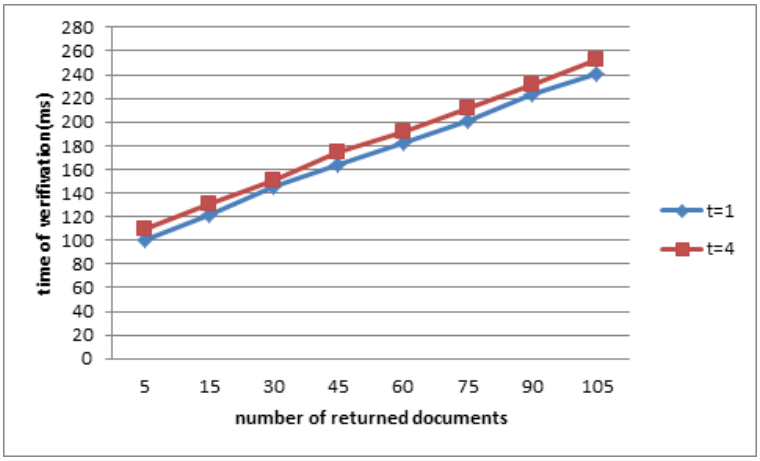

Figure 11: Verification time of the scheme with different number of returned documents

5.8 Security analysis

In this subsection, we discuss the security analysis of our scheme under two different security attack models introduced: Known Cipher text Attack Model (KCAM) and Known Plaintext Attack Model (KPAM). KCAM represents the application scenario in which the potential attackers do not have any background knowledge, while KCAM is widely appeared at the private data cloud services, such as email, online storage systems. KPAM might represent cloud application services for some public data, for example, personal health records (PHR), voter registration database, etc. In KPAM, the attackers have the capability to master part of the plaintext data through legal or illegal ways. We analyze the data privacy and the query privacy in different attack models. Then we discuss the strategies to strengthen our scheme under these different security scenarios.

5.8.1 Data privacy

In our scheme, the cloud server stores and processes three types of information including the encrypted documents C, the index I, and the query requests Q from the authorized cloud user. C is encrypted by the data owner using the symmetric encrypted algorithms. Meanwhile, the key key_{doc} for encrypting documents is grasped by the data owner and the authorized users. Therefore, based on the security of encryption algorithm, the attacker is unable to break the data privacy through attacking C without key_{doc} under both attack models. And then, we analyze the security threats brought by the index I under the different attack models.

Under the KCAM model, the attackers only master the index and the encrypted documents in the cloud. Because we use bloom filter to map the keywords, and the hash function is HMAC-SHA1 [Bellare and Krawczyk (1996)]. It is difficult to be cracked. Above all, it is irreversible. The attackers cannot guess the corresponding position. Therefore, through the analysis, we can think that the data privacy in our scheme is guaranteed under the KCAM model.

Under the KPAM model, the attackers master the plaintext information of some documents. In our scheme, we choose h hash function $h_i(1 \le i \le l)$ and m bits Bloom filter. l is the number of keys. So it is not only a hash function, but also multiple encrypted. In the worst-case scenario under the KPAM model, attacker masters the hash functions h and steals the data owner's key key_i through the illegal ways. Then A can compute the Bloom filters bf_j for a document d's all single words, and guesses some encrypted documents probably containing these words. This attack requires A to master a large scale of documents to do statistical analysis. Therefore, if the attacker does not have key_i, our scheme is secure. And if the number of documents mastered by the attacker is not too much, our scheme is secure under the KPAM model even if the attacker steals the key_i.

5.8.2 Query privacy

Query privacy requires that the cloud server and the attacker who is able to monitor the channel between the user. And cloud cannot know the user's interests from the query request.

In the scheme, the authorized user generates the query words qw. Then the authorized user calls $h(qw||key_i)$ to output the query Bloom filters QBF and submits QBF to the cloud server. If attacker does not master key_i, attacker must answer the one-way hash functions h to guess the query word in QBF. Attacker guesses the query keyword from a query bloom filter in QBF with the probability no more than $q_h/2^{\mathrm{m}}$. Thus, the query privacy of our scheme in these scenarios is guaranteed. Considering the worst-case scenario under the KPAM model, attacker masters the hash functions h and steals the data owner's key key_i through the illegal ways. In this scenario, A can break the security privacy by the 'dictionary' attacks. So, our scheme can protect the query privacy unless the key_i has leaked out.

6 Conclusions

In this paper, we address the problem of verifiable result diversification search over encrypted cloud data while preserving privacy in cloud computing. We present a scheme with Bloom Filter index structure and diversity equilibrium model that it allows the authorized user to execute the diversified retrieval over the encrypted documents at cloud. At the same time, the scheme also supports results verification. Considering the security, we build Bloom filter with the help of HMAC-SHA1. But, some efficient indexing structures are not used. And the index Bloom filters upload directly to the cloud server. If the index Bloom filter is executed secondary encryption, the scheme has better security. Finally, the performance of the proposed schemes is analyzed in detail, including the time of index construction, time of verification set construction, index storage, the time of search, recall rate, diversified evaluation, query precision and verifiable efficiency, by the experiment on real-world dataset. The results show that the proposed solution is very efficient and effective for the diversity ranking of documents.

As our ongoing work, we will continue to research on the security risks and efficiency.

Acknowledgement: This work is supported, in part, by the National Natural Science Foundation of China under grant numbers 61103215; in part, by Hunan Provincial Natural Science Foundation of China.

References

Ahmad, S.; Kumar, P. S. (2017): An efficient privacy-preserving multi-keyword ranked search over encrypted data in cloud computing. *India Conference*, pp. 1-6.

Ballard, L.; Kamara, S.; Monrose, F. (2005): Achieving efficient conjunctive keyword searches over encrypted data. *International Conference on Information and Communications Security*, vol. 3783, pp. 414-426.

Bellare, M.; Ran, C.; Krawczyk, H. (1996): Keying hash functions for message authentication. *International Cryptology Conference on Advances in Cryptology*, vol.1109, pp. 1-15.

Bloom, B. H. (1970): Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, vol. 13, no. 7, pp. 422-426.

Cao, N.; Wang, C.; Li, M.; Ren, K.; Lou, W. (2011): Privacy-preserving multi-keyword ranked search over encrypted cloud data. *INFOCOM*, 2011 *Proceedings IEEE*, vol. 25, pp. 829-837.

Cao, Y.; Zhou, Z. L.; Sun, X. M.; Gao, C. Z. (2018): Coverless information hiding based on the molecular structure images of material. *Computers, Materials & Continua,* vol. 54, no. 2, pp. 197-207.

Cash, D.; Jarecki, S.; Jutla, C.; Krawczyk, H.; Ros¸ M. C. et al. (2013): Highly-scalable searchable symmetric en- cryption with support for boolean queries. *Advances in Cryptology-CRYPTO* 2013, pp. 353-373.

Chapelle, O.; Metlzer, D.; Zhang, Y.; Grinspan, P. (2009): Expected reciprocal rank for graded relevance. *ACM Conference on Information and Knowledge Management*, vol. 43, pp. 621-630.

Clarke, C. L. A.; Kolla, M.; Cormack, G. V.; Vechtomova, O.; Ashkan, A. et al. (2008): Novelty and diversity in information retrieval evaluation. *International ACM SIGIR Conference on Research and Development in Information Retrieval*, vol. 4, pp. 659-666.

Fu, Z.; Huang, F.; Ren, K.; Weng, J.; Wang, C. (2017): Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data. *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 8, pp. 1874-1884.

Fu, Z.; Ren, K.; Shu, J.; Sun, X.; Huang, F. (2016): Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel & Distributed Systems*, vol. 27, no. 9, pp. 2546-2559.

Fu, Z.; Shu, J.; Wang, J.; Liu, Y.; Lee, S. (2015): Privacy-preserving smart similarity search based on simhash over encrypted data in cloud computing. *Journal of Internet Technology*, vol. 16, no. 3, pp. 453-460.

Fu, Z.; Sun, X.; Linge, N.; Zhou, L. (2014): Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query.

IEEE Transactions on Consumer Electronics, vol. 60, no. 1, pp. 164-172.

Fu, Z.; Wu, X.; Guan, C.; Sun, X.; Ren, K. (2017): Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics & Security*, vol. 11, no. 12, pp. 2706-2716.

Golle, P.; Staddon, J.; Waters, B. (2004): Secure conjunctive keyword search over encrypted data. *ACNS 04: International Conference on Applied Cryptography and Network Security*, vol. 3089, pp. 31-45.

Hersh, W. R.; Cohen, A. M.; Yang, J. (2005): TREC 2005 genomics track overview. *Fourteenth Text Retrieval Conference*, pp. 14-25.

Kurosawa, K.; Ohtaki, Y. (2012): UC-secure searchable symmetric encryption. *International Conference on Financial Cryptography and Data Security*, vol. 7397, pp. 285-298.

Li, F.; Wu, C.; Yuan, X.; Zhang, W.; Jiang, J. (2016): Multi-keyword ranked fuzzy search over encrypted data in cloud supporting dynamic update. *Journal of Computational & Theoretical Nanoscience*, vol. 13, no. 12, pp. 9705-9709.

Li, J.; Liu, C.; Liu, B.; Mao, R.; Wang, Y. et al. (2015): Diversity-aware retrieval of medical records. *Computers in Industry*, vol. 69, no. C, pp. 81-91.

Lu, Y. (2012): Privacy-preserving logarithmic-time search on encrypted data in cloud. *Proceedings of the 19th Annual Network & Distributed System Security Symposium*.

Miao, Y.; Ma, J.; Wei, F.; Liu, Z.; Wang, X. A. et al. (2017): Vcse: Verifiable conjunctive keywords search over encrypted data without secure-channel. *Peer-to-Peer Networking and Applications*, vol. 10, no. 4, pp. 995-1007.

Pang, H. H.; Mouratidis, K. (2008): Authenticating the query results of text search engines. *VLDB Endowment*.

Porter, M. F. (2006): The porter stemming algorithm. *ResearchGate.*

Ren, P.; Chen, Z.; Ma, J.; Wang, S.; Zhang, Z. et al. (2016): User session level diverse reranking of search results. *Neurocomputing*, no. 274.

Santos, R. L. T.; Macdonald, C.; Ounis, I. (2010): Selectively diversifying web search results. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 1179-1188.

Song, D. X.; Wagner, D.; Perrig, A. (2000): Practical techniques for searches on encrypted data. *IEEE Symposium on Security & Privacy*, pp. 0044.

Song, W.; Wang, B.; Wang, Q.; Peng, Z.; Lou, W. et al. (2016): A privacy-preserved full-text retrieval algorithm over encrypted data for cloud storage applications. *Journal of Parallel & Distributed Computing*, no. 99, pp. 14-27.

Strizhov, M.; Ray, I. (2016): Secure multi-keyword similarity search over encrypted cloud data supporting efficient multi-user setup. *IIIA-CSIC*.

Sun, W.; Liu, X.; Lou, W.; Hou, Y. T.; Li, H. (2015): Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. *Computer Communications*, pp. 2110-2118.

Sundaresan, N. (2015): Search ranking diversity based on aspect affinity. *United States*

Patent Application.

Tomlinson, S. (2006): Early precision measures: Implications from the downside of blind feedback. *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 705-706.

Vimercati, S. D. C. D.; Foresti, S.; Jajodia, S.; Paraboschi, S.; Samarati, P. (2007): Over-encryption: Management of access control evolution on outsourced data. *International Conference on Very Large Data Bases*, vol. 299, pp. 123-134.

Wang, C.; Cao, N.; Ren, K.; Lou, W. (2012): Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on Parallel & Distributed Systems*, vol. 23, no. 8, pp. 1467-1479.

Wu, C. R.; Zapevalova, E.; Chen, Y. W.; Li, F. (2018): Time optimization of multiple knowledge transfers in the big data environment. *Computers, Materials & Continua*, vol. 54, no. 3, pp. 269-285.

Xia, L.; Xu, J.; Lan, Y.; Guo, J.; Cheng, X. (2016): Modeling document novelty with neural tensor network for search result diversification. *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 395-404.

Xia, Z.; Wang, X.; Sun, X. (2016): A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel & Distributed Systems*, vol. 27, no. 2, pp. 340-352.

Xia, Z.; Xiong, N. N.; Vasilakos, A. V.; Sun, X. (2017): Epcbir: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing. *Information Sciences*, no. 387, pp. 195-204.

Xia, Z.; Zhu, Y.; Sun, X.; Qin, Z.; Ren, K. (2018): Towards privacy-preserving content-based image retrieval in cloud computing. *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 276-286.

Xu, G.; Li, X.; Zhang, Z.; Xu, L. (2016): Web spam detection based on link diversity and content Features. *Neurocomputing*, vol. 10, no. 7, pp. 363-372.

Yu, S.; Wang, C.; Ren, K.; Lou, W. (2010): Achieving secure, scalable, and fine-grained data access control in cloud computing. *Proceedings-IEEE INFOCOM*, vol. 29, no. 16, pp. 1-9.