Soft Comput (2017) 21:4829–4841
DOI 10.1007/s00500-017-2684-6

METHODOLOGIES AND APPLICATION

DMRS: an efficient dynamic multi-keyword ranked search over encrypted cloud data

Lanxiang Chen[1,2] · Linbing Qiu[1,2] · Kuan-Ching Li[3,4] · Wenbo Shi[5] · Nan Zhang[1,2]

Published online: 22 June 2017

Abstract With the rise of cloud computing, data owners outsource their data to public cloud servers while allowing users to search the data, aiming at greater flexibility and economic savings. For privacy considerations, private data must be encrypted before outsourcing, and this makes the method of plaintext keyword search infeasible. However, it is critical to enable encrypted data able to be searched. Considering the requirements of practical application scenarios, the function of efficient multi-keyword ranked search and similarity search based on relevance score is necessary for data users. There proposed a number of multi-keyword searchable encryption schemes to try to meet this demand. However, most existing schemes do not satisfy required dynamic update simultaneously. In this paper, a novel and efficient dynamic multi-keyword ranked search scheme improved from traditional secure kNN computation is proposed. The proposed scheme incorporates the similarity measure "coordinate matching" and "inner product similarity" to improve the relevance of search keywords to the relevant cloud files. A reverse data structure is introduced to allow users to perform dynamic operations on document collection, either inserting or deleting. The sparse matrix is used to replace the dense large-scale matrix in index encryption and query vector encryption to improve efficiency. Experiments show that the proposed scheme indeed reduces the overhead of computation and storage compared to MRSE scheme, concurrently guaranteeing privacy and efficiency.

Keywords Cloud computing · Searchable encryption · Multi-keyword · Ranked search · Sparse matrix · Dynamic operation

1 Introduction

In recent years, cloud computing is being considered the most innovative technology, where significant advances in the delivery of information technology and services are designed. More and more individuals and businesses are deciding to outsource local data, e.g., personal health records, financial transactions, e-mails, and government files to cloud server to achieve great flexibility and low management costs.

Although cloud computing has expressive advantages, the outsourcing of private data makes users lose control of their data, so the desire to purchase cloud services becomes very reluctant. Obviously, for the consideration of privacy preserving, sensitive data have to be encrypted before outsourcing to public cloud server. However, it makes the traditional plaintext keyword search method obsolete and results in huge overhead on the encrypted data availability. Apparently, it is quite unrealistic to download and decrypt all data on the client side. Therefore, developing a secure search service on encrypted data is a necessity in today's computing infrastructures.

Communicated by V. Loia.

✉ Lanxiang Chen
lxiangchen@fjnu.edu.cn

1 School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350108, China

2 Key Lab of Network Security and Cryptology, Fujian Normal University, Fuzhou 350108, China

3 Department of Computer Science and Information Engineering (CSIE), Providence University, Taiwan, China

4 School of Computer Science, Guangzhou University, Guangzhou, China

5 School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Hebei, China

Multi-function schemes which are based on fully homomorphic encryption (Gentry 2009) or oblivious RAMs (Goldreich and Ostrovsky 1996) have been proposed to solve these problems. However, the computing overhead presented in these schemes is extremely large. As known, the searchable symmetric encryption (SSE) schemes have advantages on efficiency and functionality. The first SSE scheme was proposed by Song et al. (2000). Recently, more and more researchers focus on variety types of search function, such as single keyword search, multi-keyword Boolean search, similarity search, multi-keyword ranked search. Because of the practicability of multi-keyword ranked search, it is more popular in research work. Unfortunately, most schemes do not support dynamic operation and the performance needs to be further improved.

In this paper, a reverse data structure is utilized to achieve dynamic update of documents in the multi-keyword ranked search over encrypted cloud data. For the improvement of the relevance of search keywords to the relevant cloud files, the similarity measure "coordinate matching" (Singhal 2001) is utilized to capture the relevance of cloud files to the interested keywords. To quantitatively analyze the similarity measure of cloud documents to the search keywords, the "inner product similarity" (Witten and Moffat 1999) was chosen. In the proposed scheme, each document and search keywords are, respectively, mapped to a binary vector in which each bit indicates whether corresponding keyword appears in the document or the search query. As described in privacy-preserving multi-keyword ranked search over encrypted cloud data scheme (MRSE) (Cao et al. 2014), we also use a secure k-nearest neighbor (kNN) algorithm (Wong et al. 2009) to meet the security requirements of index and trapdoor. Besides, we use sparse matrix to replace the dense large-scale matrix in index encryption and query vector encryption. To support data updates, a reverse data structure which is different from the original one is designed. As overall, the contributions of this paper are summarized as:

- A novel and efficient scheme improved from traditional secure kNN computation is proposed,
- The proposed scheme supports dynamic operation on document collection, either inserting or deleting documents,
- Experiments show that the proposed scheme indeed reduces the overhead of computation and storage compared to MRSE scheme, concurrently guaranteeing privacy and efficiency.

The remainder of this paper is organized as follows. Section 2 presents related work, and Sect. 3 describes the system model, threat model, design goals, notations and preliminaries. Section 4 presents the proposed efficient and dynamic multi-keyword ranked search scheme that contains the framework and detailed construction. The security and performance analysis are followed in Sect. 5, and finally, Sect. 6 concludes the paper and directions for future work.

2 Related work

Searchable encryption scheme enables users to perform keyword search on encrypted data without the need to download and decrypt all documents. It can be constructed using either public key cryptography or symmetric key cryptography. As there is a large number of users encompassed with a large number of data in cloud, it is more suitable to utilize searchable symmetric encryption (SSE).

The first SSE scheme was proposed by Song et al. (2000), which was not proven to be secure nor linear search efficient to the length of the document collection, and such limitations were addressed by Goh (2003). However, an inherent problem of using bloom filters is the possibility of false positives. Curtmola et al. (2011) proposed two new constructions SSE-1 and SSE-2. An adaptive security definition and solutions were presented in their scheme. Zou et al. (2017) propose encryption search scheme for content-based image retrieval using comparable encryption and order-preserving encryption technology. Functions of these early schemes only support single keyword search.

Under different threat models, a variety of schemes are proposed, such as single keyword search (Gajek 2016), similarity search (Wang et al. 2014; Tang 2014; Deng et al. 2017; Xhafa et al. 2014), multi-keyword Boolean search (Zhang and Zhang 2011; Poon and Miri 2015) and multi-keyword ranked search (Wang et al. 2012; Zhang et al. 2014; Rane and Ghorpade 2015; Zhangjie et al. 2015; Xia et al. 2016), multi-user search (Kiayias et al. 2016; Ye et al. 2016). Unfortunately, none of these schemes support multi-keyword ranked search and dynamic operations simultaneously.

With the proliferation of cloud environments, as developers and users login with their credentials to manipulate their files, all operations are performed in cloud-based servers. Update operation is essential. The scheme proposed by Song et al. (2000) supported simple dynamic updates, though inefficient. Xia et al. (2016) realized the dynamic update by storing the unencrypted index tree that augments computation overhead. Yang et al. (2012) proposed the solution that combined bloom filter to realize dynamic update. Nevertheless, bloom filter does not satisfy the strict security requirements. The scheme proposed in Mashauri et al. (2015) focused only on the keyword dictionary updating, ignored its computing efficiency however.

Cao et al. (2014) proposed a privacy-preserving multi-keyword ranked search over encrypted cloud data (MRSE) scheme. Firstly, depending on whether the keyword appears in the corresponding document or query, it generates the corresponding binary document vector and query vector, respectively. Secondly, two dense matrices are used to encrypt secure index and trapdoor for document vector and query vector, and lastly, the inner product of trapdoor with each subindex indicates the similarity of query keywords and the document. From the above-mentioned description, the efficiency of inner product will be significantly reduced when the number of keyword in the dictionary increases gradually. Besides, since the location of keywords is fixed, the vector structure cannot be modified after the keyword dictionary is generated. To solve this issue, we utilize sparse block matrices (Yang et al. 2012) instead of the original dense matrices in the process of index construction and trapdoor generation. Undoubtedly, this will greatly save the user's computing resources. In addition, we reverse the original vector structure (Mashauri et al. 2015) to adapt to the dynamic update operation of the document. Furthermore, we will combine the advantages of schemes proposed in Yang et al. (2012), Mashauri et al. (2015) to achieve the efficient dynamic multi-keyword ranked search over encrypted cloud data.

3 DMRS definitions

3.1 Problem formulation

System model

As illustrated in Fig. 1, there are three major entities in a cloud service system, listed as the data owner, the data user and the cloud server.

In the scenario of architecture aimed, Data owner intends to outsource a collection of documents D to cloud server. After extracting keywords collection W from D, data owner will first construct an encrypted searchable index I from collection D, so every file in D is encrypted to generate a ciphertext collection C. Next, the encrypted index and the collection C are both outsourced to cloud server concurrently, as the secret keys of trapdoor and file decryption are distributed to authorized data users. Finally, the owner should re-generate the keys after updating his document collection D.

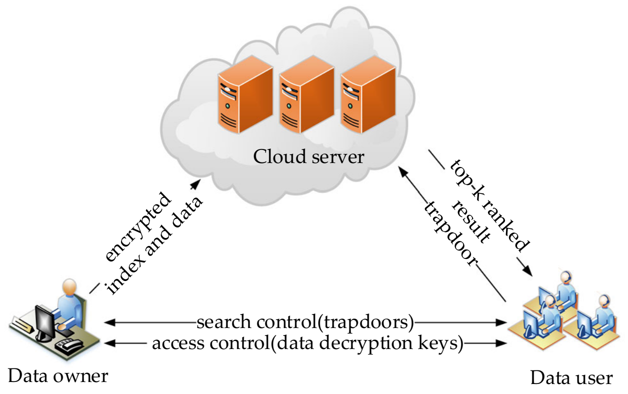


Fig. 1 Architecture of the multi-keyword ranked search over encrypted cloud data

Data users are entities that are authorized to access the documents of the data owner. To search the document collection, data users take some interested keywords as input. A query vector is generated according to the set of query keywords, followed by generating the trapdoor according to search vector. Thereafter, data users make use of keys to decrypt the top-k most relevant encrypted files returned from cloud server.

Cloud server stores the encrypted secure searchable index and the collection of encrypted files. Cloud server has the responsibility to search over the secure index and return the top-k encrypted documents to users. Besides, cloud server updates the encrypted index and encrypted document collection as well when the document collection has been updated by data owner.

Threat model

In the model proposed, the cloud server is considered as "honest-but-curious." That is, the cloud server will act honestly according to the designated protocol and will not distort user data or query results. However, it may learn additional information about the secure index and search requests. As described in MRSE_II scheme (Cao et al. 2014), we only consider those servers with stronger attack ability, e.g., the known background model. In such a model, the cloud server is able to acquire more information. Such data may include the correlation relationship of given trapdoors and the statistical information of document set. With the help of such statistical knowledge, the cloud server could confirm certain keywords by analyzing the uploaded trapdoor, the secure index or the search results.

Design goals

To enable efficient, dynamic and secure multi-keyword ranked search over outsourced encrypted cloud data under the aforementioned model, the design goals of the proposed scheme are listed as:

- *Multi-keyword Ranked Search* the proposed scheme is designed to support multi-keyword search and provide the similarity ranking of search results,
- *Search efficiency* the proposed scheme hammers at achieving higher search efficiency by improving original secure kNN-based MRSE scheme,
- *Dynamic update* in order to increase practicability, the proposed scheme supports the dynamic update of the document set,

- *Privacy preserving* the proposed scheme aims to meet strict privacy requirements. The encrypted index and trapdoor should be generated to not permit any information leakage about keywords and encrypted documents. The keyword privacy and the trapdoor unlinkability should be guaranteed in the search process by cloud server.

3.2 Notations and preliminaries

Notations

In this paper, notations presented in Table 1 are used.

Preliminaries on Secure kNN Computation

The secure k-nearest neighbor computation can be used to encrypt the vectors, and such encrypted vectors can still be utilized to compute the Euclidean distance. The proposed scheme makes use of the inner product similarity as introduced in MRSE (Cao et al. 2014).

The specific encryption process is described as follows. First, the document vector D and the search vector Q are split into two random vectors $\{D', D''\}$ and $\{Q', Q''\}$, respectively, according to the indicator S. Specifically, if the *j-th* bit of S is 0, $D'[j]$ and $D''[j]$ are the same and equal to $D[j]$, while $Q'[j]$ and $Q''[j]$ are set to random numbers so that their sum is equal to $Q[j]$. If the *j-th* bit of S is 1, $Q'[j]$ and $Q''[j]$ are the same and equal to $Q[j]$, while $D'[j]$ and $D''[j]$ are set to random numbers so that their sum is equal to $D[j]$. Then, the two pairs of split vector $\{D', D''\}$ and $\{Q', Q''\}$ are encrypted as $\{M_1^T D', M_2^T D''\}$ and $\{M_1^{-1} Q', M_2^{-1} Q''\}$, respectively, where the $M_1 and M_2$ are two invertible matrices. The formula for the relevance score is as follows.

Table 1 Notation

D	The plaintext collection, denoted as a set of m documents $D = (d_1, d_2, ..., d_m)$
C	The ciphertext collection, stored in cloud server and denoted as $C = (c_1, c_2, ..., c_m)$
C'	The updated documents collection
I	The encrypted searchable index, denoted as $I = (I_1, I_2, ..., I_m)$, where I_i is the subindex built for d_i
W	The dictionary extracted from D, denoted as a collection of n keywords $W = (w_1, w_2, ..., w_n)$
W_R	An updated dictionary that deleting several documents leads to reduction of keywords
W_q	The subset of W and the interested keywords in a search request
T	The trapdoor calculated by data user for the search request W_q
R_q	The top-k-ranked *id* set of documents that containing interested keywords W_q
d_j	A modified document in an adding or deleting operation

$$\begin{aligned}\mathrm{Score}(D, Q) &= (M_1^T D') \cdot (M_1^{-1} Q') + (M_2^T D'') \cdot (M_2^{-1} Q'') \\ &= (M_1^T D')^T M_1^{-1} Q' + (M_2^T D'')^T M_2^{-1} Q'' \\ &= D'^T M_1 M_1^{-1} Q' + D''^T M_2 M_2^{-1} Q'' \\ &= D' \cdot Q' + D'' \cdot Q'' \\ &= D \cdot Q \qquad (1)\end{aligned}$$

The formula demonstrates that the inner product of document vector D and search vector Q can be calculated in encrypted form. However, the original secure kNN algorithm could not be used in our scheme directly. Some modifications on the vector structure should be done to meet the privacy requirements of DMRS. Detailed processes of algorithm for the proposed scheme are shown in Fig. 2. With such improvements, the proposed scheme is able to provide various possible privacy guarantees within the known background models.

4 The DMRS scheme

In this section, we present the formal definition of DMRS scheme, following with the design and architecture details.

4.1 Framework

Definition 1 (*Efficient dynamic multi-keyword ranked search scheme, DMRS*) A DMRS scheme consists of five polynomial-time algorithms DMRS = (KeyGen, BuildIndex, TrapdoorGen, Search, Update), such that:

$\mathbf{K} \leftarrow$ ***Keygen***$(\mathbf{1^s})$: is a probabilistic key generation algorithm that is executed by data owner to set up the scheme. It takes a security parameter s and returns a secret key $K = \{S, M_1, M_2\}$.

(I,C) $\leftarrow$ ***BuildIndex(K, D)***: it is executed by owner to generate indexes. It takes a secret key K and a document collection D as inputs and returns an index I and ciphertext C.

$\mathbf{T} \leftarrow$ ***TrapdoorGen*** $(\mathbf{K,W_q})$**:** it is executed by user to generate a trapdoor to query keywords that is a subset of dictionary. It takes a secret key K and interested keywords W_q as inputs and returns a trapdoor T.

$\mathbf{R_q} \leftarrow$ ***Search(I,T,k)***: it is executed by server to search for documents in D that contains words W_q. It takes an index I for a collection D, the number of interested documents k and a trapdoor T for query q as inputs, and returns R_q, the top-k ranked *id* set.

$\mathbf{K_e}, \mathbf{K_R}, \mathbf{I_e}, \mathbf{I_R}, \mathbf{C'} \leftarrow$ ***Update***$(\mathbf{K}, \mathbf{d_j})$: it is executed by owner to generate new secret key, indexes and ciphertext for

Keygen(1^s):

1. Generate an $n+u+1$ bit vector S and six pairs of $(n+u+1)\times(n+u+1)$ dimension matrices and the corresponding invertible matrices M_1', $(M_1')^{-1}$, M_2', $(M_2')^{-1}$, M_1'', $(M_1'')^{-1}$, M_2'', $(M_2'')^{-1}$, P_1, $(P_1)^{-1}$, P_2, $(P_2)^{-1}$.

2. Set $M_1=M_1'\cdot P_1\cdot M_1''$ and $M_2=M_2'\cdot P_2\cdot M_2''$, output $K=(S, M_1, M_2)$;

BuildIndex(K, D): For every document d_i:

1. Data owner generates a $1+u+n$ bit vector p_i.

2. The first bit of p_i is set to 1, the $(j+1)$-th entry in p_i where $j\in[1, u]$ is set to a random number $\varepsilon(j)$ by extending the dimension.

3. The last n bit of $p_i[j]$ represent whether the corresponding keyword w_j appears in the document d_i.

4. Split vector p_i to $\{p_i'[j], p_i''[j]\}$: for every bit of p_i: if $(S[j]=1)$, $p_i'[j]+p_i''[j]=p_i[j]$; else $p_i'[j]=p_i''[j]=p_i[j]$.

5. The subindex I_i is built by multiplying $p_i'[j]$ and $p_i''[j]$ with M_1', P_1, M_1'' and M_2', P_2, M_2''.

6. Obtain ciphertext c_i with symmetric encryption algorithm.

7. Upload the collection of ciphertext and encrypted index (C, I) to the server.

TrapdoorGen(K, W_q):

1. Input t keywords of interest, one $n+u+1$ bit binary vector q is generated.

2. The top n bit of $q[j]$ indicates whether keyword $w_j\in W$ is true or false.

3. Select v out of u dummy keywords randomly, the corresponding entries in q are set to 1 and all the remaining bit are set to 0.

4. Split q to $\{q', q''\}$ with vector S: if $(S[j]=1)$, $q'[j]= q''[j] = q[j]$; else $q'[j]+ q''[j]=q[j]$.

5. Get $T=(q'\cdot(M_1'^{-1})^{T}\cdot(P_1^{-1})^{T}\cdot(M_1''^{-1})^{T}, q''\cdot(M_2'^{-1})^{T}\cdot(P_2^{-1})^{T}\cdot(M_2''^{-1})^{T})$.

Search(I, T, k): After receiving the trapdoor T, for every I_i generated from d_i, the cloud server calculate the inner product similarity of T and I_i. Sort the similarity score of result documents set R_q and return it to user.

Fig. 2 The improved kNN-based scheme

the added or deleted document d_j. It takes the original secret key K and the updated document d_j as inputs and returns the new secret key (K_e, K_R), indexes (I_e, I_R) and ciphertext C'.

4.2 The DMRS construction

In this subsection, details on the construction are described. We utilize sparse diagonal matrix (Yang et al. 2012) to improve the kNN-based MRSE scheme and adopt new vector structure (Mashauri et al. 2015) reversed from MRSE scheme. Therefore, the improved scheme can realize efficient and dynamic multi-keyword ranked search under strict privacy requirements.

4.2.1 The improved KNN computation

From the above analysis, we can see that the origin to the cause of low efficiency of MRSE is the utilization of large-scale matrices. Therefore, the sparse matrices are used to replace the original dense matrices, for higher efficiency considerations. The modifications on the original KNN technique are the same as the scheme in Yang et al. (2012). For the consideration of the trade-off between efficiency and security, the invertible matrix M_1 is set as the product of two block diagonal matrix M_1' and a permutation matrix P. That is $M_1 = M_1' \cdot P \cdot M_1''$ where M_1', M_1'' are two block diagonal matrix, and M_1'=*diag*$(A_1, A_2, \dots, A_l)$, A_k is a $n' \times n'$ dimension random matrix, M_1''=*diag*$(B_1, B_2, \dots, B_l)$ and B_k is a $n' \times n'$ dimensions random matrix. The modified invertible matrix M_2 is generated as the same as M_1.

After above processes, cloud server will be hard to analyze the relations between queries and indexes because every keyword could be mixed randomly. Obviously, owing to the introduction of sparse matrix, the number of calculations will be reduced significantly. From the following experimental analysis, these modifications can achieve the balance between efficiency and security. Detailed construction is presented in Fig. 2.

At present, the proposed scheme will not be able to answer for the changes provisionally, even data owner has updated several documents. In the next step, we make use of the new vector structure to support the updates on the collection of documents.

4.2.2 Update operation

In MRSE scheme, the positions of real keywords in document vector and search vector are fixed and it cannot make any improvement to support dynamic updating operations. We can invert the existing structure to acquire the novel dynamic vector structure as the same way presented in Mashauri et al. (2015) and get the new index structure.

The new index structure is detailed in Mashauri et al. (2015), in which the first position is the constant 1 and u dummy keywords are following; the last positions are used to indicate the presence or absence of $n$real keywords. It is utilized in DMRS scheme, and it will enable the proposed scheme to support updating operations. If there is any keyword added to W, e.g., w_j, then the value of corresponding index vector $p[1+u+n+j]$ is set to 1; otherwise, it is set to 0. When the user requests to search some keywords, the search vector is generated in the same way as the procedure of index vector construction. Detailed processes of updating operations are illustrated as follows.

Updating document collection will lead to the change of dictionary size. Next, data owner should update the keys for generating secure index set and trapdoor. Besides, the t updated documents will be extracted and will derive the number r of deletion or addition of keywords. The detailed algorithms of keys update are shown in Fig. 3.

(1) KeyExtend: According to the number r of extracted keywords, this algorithm generates a r bit binary vector, two pairs of $r \times r$ dimension matrices and the corresponding reversible matrices (S_{re}, M'_{1e}, $(M'_{1e})^{-1}$, M'_{2e}, $(M'_{2e})^{-1}$, M''_{1e}, $(M''_{1e})^{-1}$, M''_{2e}, $(M''_{2e})^{-1}$). The new generated matrices are added to the original matrices (M'_1, M''_1, M'_2, M''_2, $(M'_1)^{-1}$, $(M''_1)^{-1}$, $(M'_2)^{-1}$, $(M''_2)^{-1}$) and get the two pairs of modified matrices (M_{1e}, M_{2e}, $(M_{1e})^{-1}$, $(M_{2e})^{-1}$) by block diagonal technique (Ishai et al. 2006). Then, all elements of S_{re} will be copied to S and will obtain a $n+u+r$+1 bit vector S_e. Details of this process are shown in Fig. 3.

(2) KeyReduce: First, it generates four $(n+u+1-r) \times (n+u+1-r)$ dimension random permutation matrices (P_{1R}, P_{2R}, $(P_{1R})^{-1}$, $(P_{2R})^{-1}$). Next, the elements of the last r rows and r columns in the original matrices M'_1, $(M'_1)^{-1}$, M'_2, $(M'_2)^{-1}$, M''_1, $(M''_1)^{-1}$, M''_2, $(M''_2)^{-1}$ will be deleted and will derive four pairs of $(n+u+r+1) \times (n+u+r+1)$ dimension matrices (M'_{1R}, $(M'_{1R})^{-1}$, My'_{2R}, $(M'_{2R})^{-1}$, M''_{1R}, $(M''_{1R})^{-1}$, M''_{2R}, $(M''_{2R})^{-1}$). Then, compute $M_{1R} = M'_{1R} \cdot P_{1R} \cdot M''_{1R}$, $M_{2R} = M'_{2R} \cdot P_{2R} \cdot M''_{2R}$, $(M_{1R})^{-1} = (M'_{1R})^{-1} \cdot (P_{1R})^{-1} \cdot (M''_{1R})^{-1}$, $(M_{2R})^{-1} = (M'_{2R})^{-1} \cdot (P_{2R})^{-1} \cdot (M''_{2R})^{-1}$. Finally, this algorithm traverses the new dictionary and the old one. If a keyword exists in the both dictionaries, then the certain bit is copied into the new splitting vector S_R and omitted otherwise.

The addition or deletion of the documents also requires updating the indexes. The detailed algorithms of indexes update are shown in Fig. 4.

(3) IndexExtend: Index recreation is influenced by the number r of modified keywords. First, the algorithm extends the r entries following the last entry of document vector p for the original document. Then, call the algorithm BuildIndex(K, D) to get all document vectors p_{ie}, the new index I_e. Finally, subciphertext set is generated and added into the original one.

(4) IndexReduce: Firstly, it generates the new dictionary W_R extracted from the updated document set. Next, this algorithm will call BuildIndex(K, D) to generate the updated indexes I_R, and finally, delete the subindex I_j and the ciphertext c_j, $j \in (0, r]$. Details of this process are shown in Fig. 4.

5 The security and performance analysis

The security and performance analysis of the proposed scheme is given in this section.

5.1 The precision and privacy

Precision

As proposed in MRSE, the random insertion of dummy keywords has negative influence on the similarity score of documents to queried keywords. The procedure of dealing with dummy keyword inserting in our scheme is as the same as MRSE. Therefore, we only focus on the influence on precision and efficiency resulted from the different way of splitting large-scale matrix.

From the above introduction, the accuracy may be affected by the order of the block matrix. To improve the search efficiency, the large-scale matrix can be split into several small-scale block matrix. However, it may result that less keywords are mixed. On the contrary, the order of block matrix is larger; then, the efficiency is reduced, but security is guaranteed. The order of block matrix is decided by

```
KeyExtend($K$, $r$):
    1. Retrieve the original key $K$;
    2. Generate a $r$ bit vector $S_{re}$, two pairs of $r \times r$ dimension matrices and the corresponding invertible matrices $M_{1e}'$, $(M_{1e}')^{-1}$, $M_{2e}'$, $(M_{2e}')^{-1}$, $M_{1e}''$, $(M_{1e}'')^{-1}$, $M_{2e}''$, $(M_{2e}'')^{-1}$;
    3. Generate a $(n+u+r+1) \times (n+u+r+1)$ dimension unit matrix $E$;
    4. Compute $EM_1' = diag(M_{1e}', M_1')$, $EM_1'' = diag(M_{1e}'', M_1'')$, $(EM_1')^{-1} = diag((M_{1e}')^{-1}, (M_1')^{-1})$, $(EM_1'')^{-1} = diag((M_{1e}'')^{-1}, (M_1'')^{-1})$, $EM_2' = diag(M_{2e}', M_2')$, $EM_2'' = diag(M_{2e}'', M_2'')$, $(EM_2')^{-1} = diag((M_{2e}')^{-1}, (M_2')^{-1})$, $(EM_2'')^{-1} = diag((M_{2e}'')^{-1}, (M_2'')^{-1})$, where $diag()$ denotes the block diagonal operation [25];
    5. Copy the value of every position in $S$ and $S_{re}$ into a new vector $S_e$ in order;
    6. Randomly upset the elements of $E$ by column and get four random permutation matrices ($P_{1e}$, $P_{2e}$, $(P_{1e})^{-1}$, $(P_{2e})^{-1}$);
    7. Set four $(n+u+r+1) \times (n+u+r+1)$ dimension matrices respectively as follows.
        $M_{1e} = EM_1' \cdot P_{1e} \cdot EM_1''$, $M_{2e} = EM_2' \cdot P_{2e} \cdot EM_2''$,
        $(M_{1e})^{-1} = (EM_1')^{-1} \cdot (P_{1e})^{-1} \cdot (EM_1'')^{-1}$, $(M_{2e})^{-1} = (EM_2')^{-1} \cdot (P_{2e})^{-1} \cdot (EM_2'')^{-1}$;
    8. Output the new key $K_e$ with 5-tuple as $\{S_e, M_{1e}, M_{2e}, (M_{1e})^{-1}, (M_{2e})^{-1}\}$.

KeyReduce($K$, $r$):
    1. Retrieve the original key $K$;
    2. Delete the last $r$ rows and $r$ columns elements of $M_1'$, $(M_1')^{-1}$, $M_2'$, $(M_2')^{-1}$, $M_1''$, $(M_1'')^{-1}$, $M_2''$, $(M_2'')^{-1}$ and get the $(n+u+1-r) \times (n+u+1-r)$ dimension matrices $M_{1R}'$, $(M_{1R}')^{-1}$, $M_{2R}'$, $(M_{2R}')^{-1}$, $M_{1R}''$, $(M_{1R}'')^{-1}$, $M_{2R}''$, $(M_{2R}'')^{-1}$;
    3. Generate a $(n+u+1-r) \times (n+u+1-r)$ dimension unit matrix $E$, then randomly upset the elements of $E$ by column and get four random permutation matrices ($P_{1R}$, $P_{2R}$, $(P_{1R})^{-1}$, $(P_{2R})^{-1}$);
    4. For each keyword in $W$ and updated dictionary $W_R$:
        if ($w$ exists in $W$ and $W_R$ simultaneously)
            copy the value of corresponding bit for $w$ from $S$ to $S_R$;
        else skip the bit position;
    5. Set four $(n+u+1-r) \times (n+u+1-r)$ dimension matrices respectively as follows:
        $M_{1R} = M_{1R}' \cdot P_{1R} \cdot M_{1R}''$, $M_{2R} = M_{2R}' \cdot P_{2R} \cdot M_{2R}''$,
        $(M_{1R})^{-1} = (M_{1R}')^{-1} \cdot (P_{1R})^{-1} \cdot (M_{1R}'')^{-1}$, $(M_{2R})^{-1} = (M_{2R}')^{-1} \cdot (P_{2R})^{-1} \cdot (M_{2R}'')^{-1}$;
    6. Output the new key $K_R$ with 5-tuple as $\{S_R, M_{1R}, M_{2R}, (M_{1R})^{-1}, (M_{2R})^{-1}\}$.
```

Fig. 3 Algorithms of key update

the size of keyword dictionary. For the experimental time savings, the number of documents in the collection and the sizes of dictionary is set to 1000. Figure 5 shows the comparison to different influences on search precision with the different order o of smaller-scale matrix in the proposed scheme. As described above, increasing the dimension of the block matrix can further improve the accuracy, but affect the performance as well. Therefore, the proposed scheme provides users with a trade-off between performance and accuracy.

Privacy

The DMRS scheme is improved based on MRSE_II scheme (Cao et al. 2014). In the process of the secure index construction and the trapdoor generation, several large-scale

```
IndexExtend($K_e$, $r$):
    1. For each original document, extend the $r$ entries following the last entry of document vector $p$;
    2. Generate a series of $1+u+n+r$ bit vectors $p_{ie}$ according to the new keyword dictionary for all documents;
    3. Call the algorithm BuildIndex($K$, $D$) and get the new indexes $I_e$;
    4. Encrypt $d_j$, $j \in (0, t]$ by utilizing symmetric encryption mechanism;
    5. Output the new collections of ciphertext and index ($I_e$, $C'$).
IndexReduce($K_R$, $r$):
    1. Delete the $r$ reduced keywords and generate updated dictionary $W_R$;
    2. Delete the sub-index $I_j$ for the $t$ deleted document $d_j$, $j \in (0, t]$;
    3. Delete the ciphertext $c_j$, $j \in (0, r]$;
    4. For each updated document:
        Generate a $1+u+n-r$ bit document vector $\boldsymbol{p_R}$ according to $W_R$ for the remaining document set;
    5. For the remaining documents:
            Call BuildIndex($K$, $D$) to generate the new indexes $I_R$;
    6. Output the new collections of ciphertext and index ($I_R$, $C'$).
```

Fig. 4 Algorithms of index update

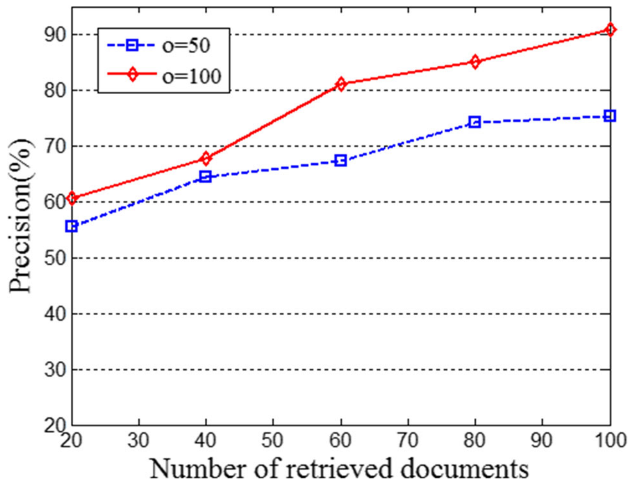


Fig. 5 The precision of search with different order o of smaller-scale matrix

dense matrices were utilized to encrypt document vectors and search vectors in MRSE scheme. Also, the system parameter w is chosen to determine the number of dummy entries in data and query vectors.

Theorem 5.1 *The DMRS is secure against scale analysis attack if the MRSE_II scheme is secure against scale analysis attack under the known background model.*

Proof In the proposed scheme, for the search performance considerations, the large scale of matrices is divided into smaller scale of block diagonal matrices. As described in the proposed scheme, two random block matrices orderly multiply a random permutation matrix. Consequently, different terms will also be mixed and it increases the index and trapdoor attack difficulty from the cloud server. Thus, the keyword privacy and the trapdoor unlinkability will be reinforced if the secret keys are kept well. As description of MRSE, there is a trade-off between security and precision. Lower accuracy will enhance the security of sorting results, though it may induce the result that user is unable to obtain the possible interested documents. Hence, this scheme can also provide users with a trade-off parameter to meet the different needs of accuracy and privacy.

In ranked search, access pattern is the rank order of the search results. There are several effective methods, e.g., private information retrieval (PIR) technique (Ishai et al. 2006) can be used to protect the access pattern. And due to the fact that the cloud server is in charge of most computations, secure hardware can be used to protect the privacy of access pattern. As presented in MRSE, in order to meet the privacy requirements, the overall rank privacy P_k is processed in the same way. Therefore, the proposed scheme also can provide a balance between precision and rank privacy like MRSE. □

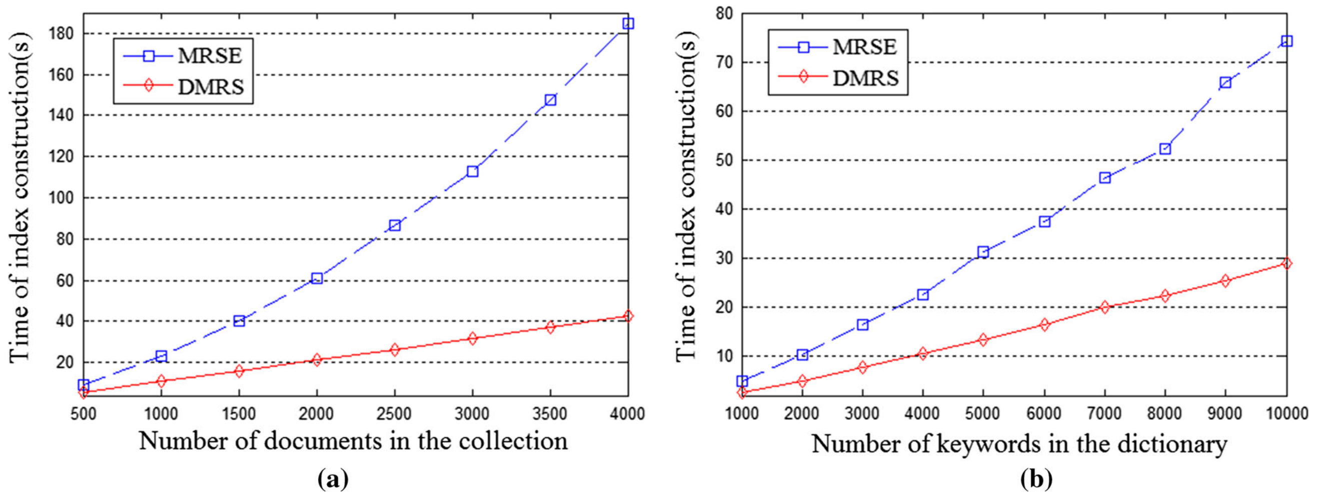


Fig. 6 The overhead of index construction. **a** For different size of dataset with the same dictionary, $m = 4000$. **b** For different size of dictionary with the same document dataset, $n = 1000$

5.2 The performance analysis

In this section, we analyze the performance of the proposed DMRS scheme. We demonstrate a thorough experimental evaluation of the scheme in a simulation environment. Experiments are implemented on a computer with Intel core i5-M480 2.67GHz processor, 6 GB memory, 1GB AMD Radeon HD 6400M graphics card, running Win7 (64 bit) system. Gauss matrix, invertible matrix and matrix multiplication are achieved by using MATLAB. The performance of our scheme is compared with MRSE scheme (Cao et al. 2014).

Computing overhead

From the above description, we can deduce that the overhead of the proposed scheme is minimal. The series of binary vectors and random matrices to denote dataset and keys, respectively, are randomly generated, and computing overhead of index construction, trapdoor generation, search and update operations will be compared and analyzed next.

1) Index construction

The process of index construction is done within two steps. The former one is to map the keyword dictionary extracted from each document to a data vector, while the latter is the encryption of a set of data vectors. The dimension of data vector directly determines the time of mapping or encrypting, and the dimension of vector is determined by the size of the dictionary. The time of generating whole index is related to the number of user's documents. Figure 6a shows that given the same dictionary ($n = 4000$), time of index construction for the two schemes increases linearly with the increasing size of dataset. Figure 6b demonstrates that, given the same number of files ($m = 1000$), DMRS consumes much less time than MRSE on constructing indexes. DMRS and MRSE are sensitive to the size of keyword dictionary for index construction. Despite it, DMRS shows better performance in both situations.

2) Trapdoor Generation

Compared with index construction, trapdoor generation consumes relatively less time. Figure 7a shows that the time of generating a trapdoor in MRSE is greatly affected by the number of keywords in the dictionary. Figure 7b demonstrates that the time of trapdoor generation in MRSE scheme is about 80% larger than DMRS scheme. The difference of overheads to generate trapdoors is mainly caused by larger scale of matrices and smaller scale of matrices in MRSE and DMRS, respectively. Additionally, it reveals that the size of dictionary has little effect on the trapdoor generation.

3) Search

In fact, search process executed by the cloud server is composed of computing and ranking similarity scores for documents dataset. Figure 8 shows that the search overhead depends on the size of documents set, and the number of keywords in a query has low influence. Figure 8b demonstrates that the proposed scheme is about 5 times faster than MRSE in different numbers of search keywords.

4) Update

As presented in MRSE, the keys and indexes will be regenerated when the size of dictionary is changed. In this set of experiments, we compare the time overhead on the key and index updates in DMRS with the time to regenerate the keys and the time to reconstruct the set of the document indexes in MRSE when the number of keywords extracted from the update documents is changed.

Figure 9 shows that the overhead of index construction and key extension increases, encompassed with the number of keywords extracted from the newly added documents.

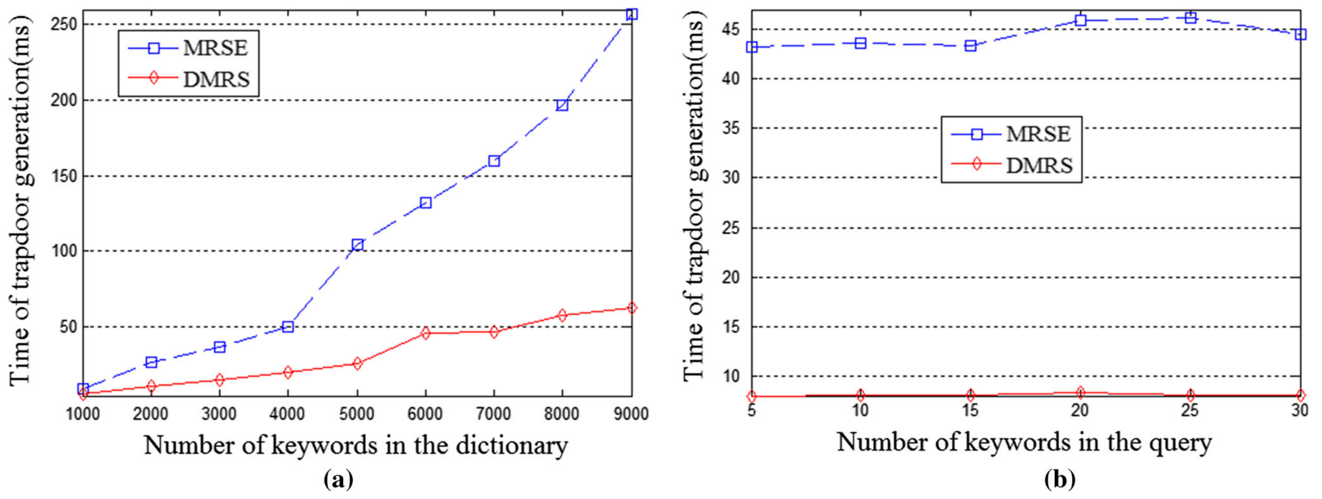


Fig. 7 The overhead of trapdoor generation. **a** For different sizes of dictionary within the same query keywords, $q = 20$. **b** For different numbers of query keywords within the same dictionary, $n = 4000$

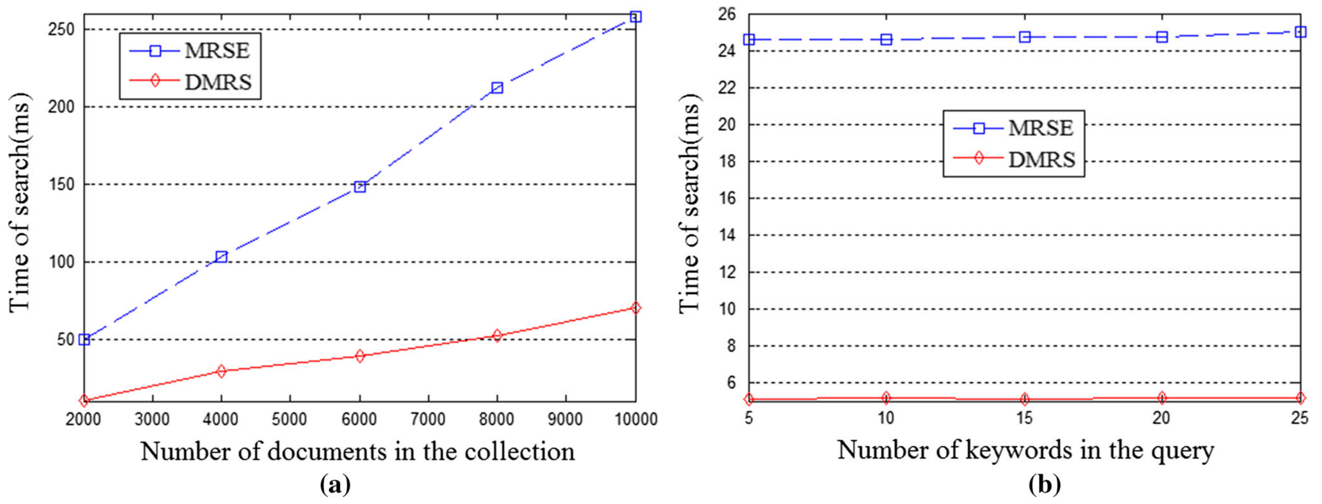


Fig. 8 Time overhead of search. **a** For the same query keywords in different sizes of dataset, $q = 20$, $n = 2500$. **b** For different numbers of query keywords in the same dataset, $m = 1000$, $n = 2500$

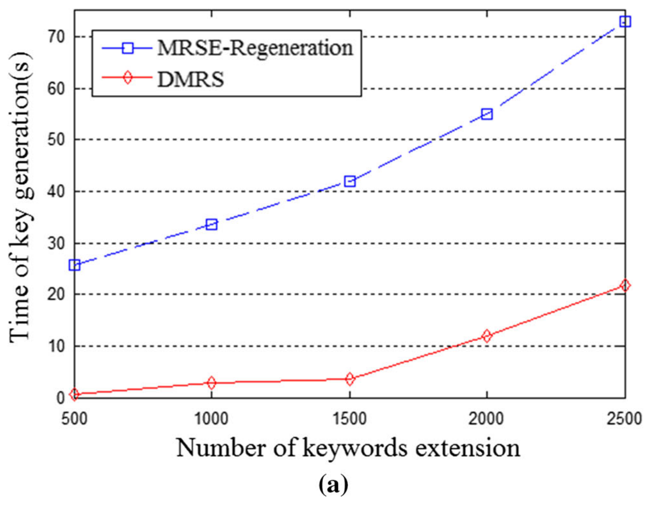
Besides, the extension performance of the proposed scheme is better than MRSE. This tremendous time savings benefits from that DMRS uses block diagonal matrix design, and the presented algorithms support reusing the original set of indexes and keys.

Figure 10 indicates that the overhead of index construction and key reduction decreases with the number of keywords extracted from the newly deleted documents. As shown in Fig. 10a, the reduction performance of DMRS is more advantageous when the dictionary size is very large. In other words, the more keywords are deleted, the more times of calculation for each original matrix. While Fig. 10b depicts, the time overhead of index update in DMRS is smaller than MRSE.

From Figs. 9 and 10, the updates performance of DMRS is better than the MRSE. This is mainly due to the fact that the proposed scheme can reuse the original keys and index. The proposed scheme does not make any modifications when the number of keywords is changed. However, in MRSE, the keys should be regenerated and the indexes should be reconstructed when the size of dictionary is changed.

Storage overhead

From the above experimental analysis we can see clearly that the time overhead is minimal. The cloud server only stores the ciphertexts and the secure index. The size of the index is mainly determined by the size of document collection and the number of keywords. As can be seen from Table 2, the index sizes of two schemes grow linearly with the number of keywords. Obviously, DMRS scheme takes up less space.



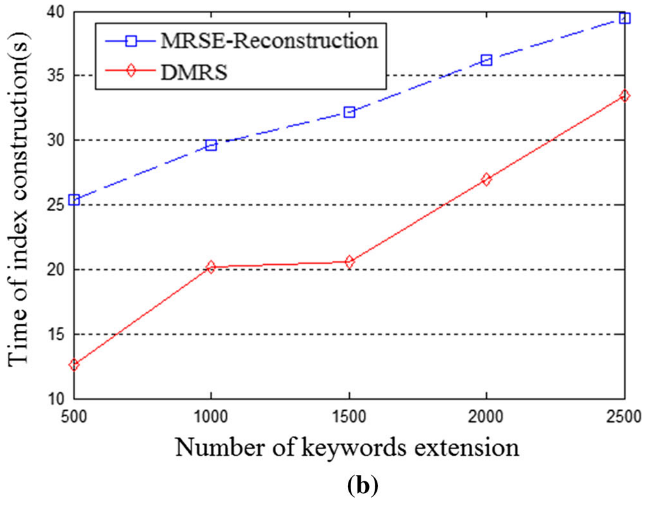
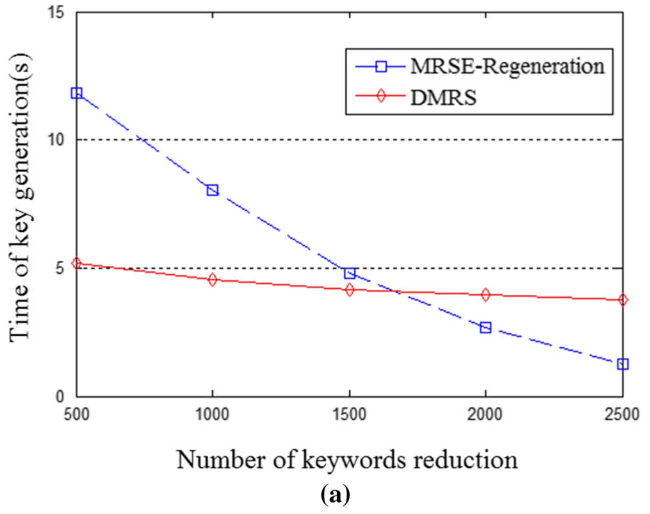


Fig. 9 Time overhead of extension with the same sizes of dictionary and document collection, m=1000, n=4000. **a** The time overhead of key extension. **b** The time overhead of index construction



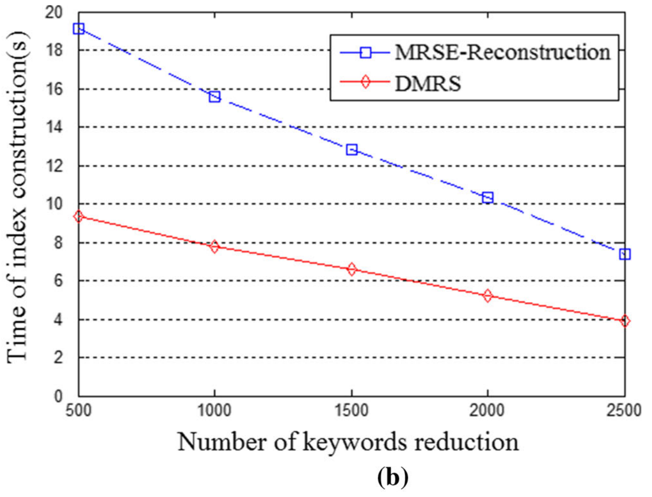


Fig. 10 Time overhead of reduction with the same sizes of dictionary and document collection, $m = 1000$, $n = 4000$. **a** The time overhead of key reduction. **b** The time overhead of index construction

Table 2 Size of index with $m = 1000$

Size of dictionary	2000	4000	6000	8000	10,000
MRSE(MB)	30.5	61	91.6	122.1	152.6
DMRS(MB)	30.2	60.4	90.6	120.9	151.1

Table 3 Size of trapdoor with $m = 1000$

Size of dictionary	2000	4000	6000	8000	10,000
MRSE (KB)	31.3	62.5	93.8	125	156.3
DMRS (KB)	30.9	61.9	92.8	123.8	154.7

The trapdoor for query keywords is generated according to the keywords dictionary and sent to the cloud server. Therefore, the size of trapdoor is mainly determined by the size of dictionary. From Table 3, the storage overhead of trapdoor is smaller than the index, and it is sensitive to the number of keywords such as the secure index.

6 Conclusions and future work

In this paper, we present a novel scheme that it supports efficient and dynamic multi-keyword ranked search over encrypted cloud data. Like MRSE scheme, the efficient similarity measure of "coordinate matching" is chosen to effectively obtain the relevance of outsourced documents to the queried keywords, and “inner product similarity” is used for the analysis of similarity measure. Besides, we use block sparse diagonal matrix and permutation matrix to improve search speed and ensure the privacy requirement in this proposed scheme. The new and secure index structure is constructed to realize dynamic operation. Evaluations on the security and efficiency of the proposed scheme under different settings presented indicate that the DMRS scheme introduces lower overhead on both computation and communication than MRSE scheme.

There are still many challenge problems in SSE schemes, as the most of SSE schemes mainly consider the challenge from the cloud server. Actually, there are many secure challenges in a multi-user scheme. Fine-grained access authorization and the revocation of the user are big challenges. In the future works, we will try to improve the SSE scheme to handle these challenge problems. In addition, we will explore supporting other multi-keyword semantics (e.g., weighted query) over encrypted data, integrity check of rank order in search result and privacy guarantees in the untrusted server model.

Acknowledgements This work was supported by the Natural Science Foundation of China (Nos. 61602118, 61572010 and 61472074), Fujian Normal University Innovative Research Team (No. IRTL1207), Natural Science Foundation of Fujian Province (No. 2015J01240), Science and Technology Projects of Educational Office of Fujian Province (No. JK2014009) and Fuzhou Science and Technology Plan Project (No. 2014-G-80).

Compliance with ethical standards

Conflict of interest Lanxiang Chen declares that she has no conflict of interest. Linbing Qiu declares that he has no conflict of interest. Kuan-Ching Li declares that he has no conflict of interest. Wenbo Shi declares that he has no conflict of interest. Nan Zhang declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Cao N, Wang C, Li M, Ren K, Lou W (2014) Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans Parallel Distrib Syst 25(1):222–233

Curtmola R, Garay J, Kamara S, Ostrovsky R (2011) Searchable symmetric encryption: improved definitions and efficient constructions. J Comput Secur 19(5):895–934

Deng Z, Li K, Li K, Zhou J. A multi-user searchable encryption scheme with keyword authorization in a cloud storage. Future Gener Comput Syst. doi:10.1016/j.future.2016.05.017

Gajek S (2016) Dynamic symmetric searchable encryption from constrained functional encryption. In: Proceedings of the cryptographers’ track at the RSA conference, pp 75–89

Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of symposium on the theory of computing (STOC), pp 169–178

Goh E-J (2003) Secure indexes. Cryptology ePrint Archive: Report 2003/216

Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious RAMs. J ACM 43(3):431–473

Ishai Y, Kushilevitz E, Ostrovsky R, Sahai A (2006) Cryptography from anonymity. In: Proceedings of the 47th annual IEEE symposium on foundations of computer science, pp 239–248

Kiayias A, Oksuz O, Russell A, Tang Q, Wang B (2016) Efficient encrypted keyword search for multi-user data sharing. In: Proceedings of European symposium on research in computer security (ESORICS), pp 173–195

Mashauri D, Li R, Han H, Gu X , Xu Z, Xu CZ (2015) Adaptive multi-keyword ranked search over encrypted cloud data. In: Proceedings of the international conference on collaborative computing: networking, applications and worksharing, pp 3–13

Poon HT, Miri A (2015) An efficient conjunctive keyword and phase search scheme for encrypted cloud storage systems. In: Proceedings of IEEE 8th international conference on cloud computing, pp 508–515

Rane DD, Ghorpade VR (2015) Multi-user multi-keyword privacy preserving ranked based search over encrypted cloud data. In: Proceedings of the international conference on pervasive computing (ICPC), pp 1–4

Singhal A (2001) Modern information retrieval: a brief overview. IEEE Comput Soc Tech Comm Data Eng Bull 24(4):35–43

Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceedings of IEEE symposium on security and privacy, pp 44–55

Tang Q (2014) Nothing is for free: security in searching shared and encrypted data. IEEE Trans Inf Forensics Secur 9(11):1943–1952

Wang C, Cao N, Li J, Ren K, Lou W (2012) Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans Parallel Distrib Syst 23(8):1467–1479

Wang B, Yu S, Lou W, Hou YT (2014) Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In: Proceedings of IEEE conference on computer communications, pp 2112–2120

Witten IH, Moffat A (1999) Managing gigabytes: compressing and indexing documents and images, second edition. Morgan Kaufmann

Wong WK, Cheung DW, Kao B, Mamoulis N (2009) Secure kNN computation on encrypted databases. In: Proceedings of ACM SIGMOD international conference on management of data, pp 139–152

Xhafa F, Wang J, Chen X, Krause PJ (2014) An efficient PHR service system supporting fuzzy keyword search and fine-grained access control. Soft Comput 18(9):1795–1802

Xia Z, Chen L, Sun X, Liu J (2016) A multi-keyword ranked search over encrypted cloud data supporting semantic extension. Int J Multimed Ubiquitous Eng 11(8):107–120

Xia Z, Wang X, Sun X, Wang Q (2016) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst 27(2):340–352

Yang C, Zhang W, Xu J, Xu J, Yu N (2012) A fast privacy-preserving multi-keyword search scheme on cloud data. In: Proceedings of the international conference on cloud and service computing (CSC), pp 104–110

Ye J, Wang J, Zhao J, Shen J, Li K-C. Fine-grained searchable encryption in multi-user setting. Soft Comput. doi:10.1007/s00500-016-2179-x

Zhang B, Zhang F (2011) An efficient public key encryption with conjunctive-subset keywords search. J Netw Comput Appl 34(1):262–267

Zhangjie F, Sun X, Qi Liu L, Zhou JS (2015) Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans Commun 98(1):190–200

Zhang W, Xiao S, Lin Y, Zhou T, Zhou S (2014) Secure ranked multi-keyword search for multiple data owners in cloud computing. In: Proceedings of the 44th annual IEEE/IFIP international conference on dependable systems and networks, pp 276–286

Zou Q, Wang J, Ye J, Shen J, Chen X (2017) Efficient and secure encrypted image search in mobile cloud computing. Soft Comput. doi:10.1007/s00500-016-2153-7