A Personalized Multi-keyword Ranked Search Method Over Encrypted Cloud Data

Xue Tian[1,2], Peisong Shen[1,2], Tengfei Yang[1,2], Chi Chen[1,2], and Jiankun Hu[3(✉)]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing 100093, China
{tianxue,shenpeisong,yangtengfei,chenchi}@iie.ac.cn

[2] School of Cyber Security, The University of Chinese Academy of Sciences, Beijing 100049, China

[3] Cyber Security Lab, School of Engineering and IT, University of New South Wales at the Australian Defence Force Academy, Canberra, Australia
J.Hu@adfa.edu.au

Abstract. Due to data privacy considerations, the data owners usually encrypt their documents before outsourcing to the cloud. The ability to search the encrypted documents is of great importance. Existing methods usually use the keywords to express users' query intention, however it's difficult for the users to construct a good query without the knowledge of document collection. This paper proposes a personalized ciphertext retrieval method based on relevance feedback, which utilizes user interaction to improve the correlation with the search results. The users only need to determine the relevance of the documents instead of constructing a good query, which can greatly improve the users query satisfaction. The selected IEEE published papers are taken as a sample of the experiment. The experimental results show that the proposed method is efficient and could raise the users' satisfaction. Compared with MRSE-HCI method, our method could achieve higher precision rate and equally high efficiency performance.

Keywords: Cloud computing · Ciphertext search · Multi-keyword search
Relevance feedback

1 Introduction

Nowadays the data security issues in the cloud computing environment draws more and more attention. In order to ensure the privacy of personal data, users usually encrypt the documents before uploading them to the cloud server. However, data encryption makes the traditional retrieval mechanism invalid. Ciphertext search has become one of the important research issues in the field of information security.

A series of ciphertext retrieval methods have been proposed for different query requirements and application scenarios, including single keyword, conjunctive keyword and multi-keyword rank retrieval. However, the above methods are difficult to meet the needs of the users' personalized query. This is because these existing methods use the keywords to express users' query intention, which will encounter following three challenges: (I) without knowledge of the targeted domain, it is difficult for the users to submit accurate keywords. (II) The same meaning can be expressed in different words, such as “aircraft” and “plane”. If a user queries the keyword “plane”, then the documents that contain the “aircraft” may not be returned. (III) Different users might have different occupations, interests or cultural backgrounds. Thus, even if they input the same query keywords, the focus of the search results will not be the same. In recent years, some methods such as semantic-based query [1], weighted keyword model [2] and PSRE (a personalized ciphertext search framework) [3] have been put forward to respond to challenge II or challenge III. However, the current query refinement methods all depend on the users to provide the accurate query keywords, which cannot be applied to cope with challenge I.

In order to tackle the above three challenges, we propose a personalized multi-keyword rank search method based on the similar search tree and relevance feedback (MRSE-SSF). The user query personalization is achieved through two-stage. The first stage allows users to submit query keywords and obtains the initial search results. In the second stage, the user picks out the relevant documents in the initial results and the optimized query keywords is automatically calculated and summited. Then the server returns the final search results. The experiment shows that the method improves the users’ satisfaction about the search results.

The contribution of this paper is mainly reflected in the following three aspects:

(1) A new type of ciphertext search method MRSE-SSF is proposed. By adopting the relevant feedback method, MRSE-SSF is used to enhance the user query satisfaction.
(2) The similarity search tree structure is introduced in the ciphertext index building process to improve the query efficiency.
(3) Experimental results demonstrate the effectiveness and efficiency of the proposed method.

2 Related Work

In recent years, many scholars contribute to the extensive study of the ciphertext retrieval. Song et al. [4] first described the ciphertext retrieval problem in 2000; a single-keyword query scheme based on symmetric encryption algorithm is realized by sequential query method. In 2004, Boneh et al. [5] first proposed a single-keyword search method based on the public key encryption (PEKS), which was originally used in encrypted e-mail systems. Nevertheless, it needs a safe channel shared by the data users and servers, which limits the practicality of the method. Later a formal security index structure: Z index is introduced [6]. The index model is realized by a pseudo random function and bloom filter, which is resilient against the adaptive chosen keyword attack. However, Z index does not provide ranking query mechanism. By adding the correlation score in the inverted table, the secure ranked search method [7] is proposed. During the query phase, the cloud server only needs to return the top k related documents matching the query conditions. This method could reduce the bandwidth consumption. However, the above work can only solve the problem of single-keyword ciphertext retrieval.

In order to express the user's query intention comprehensively, multiple keywords search method are proposed, including the conjunctive keyword retrieval [8] and multi-keyword rank search method (MRSE) [9]. The conjunctive keyword method demands the results contain all the query keywords. The MRSE method returns the top k most relevant documents through the implementation of secure *k* nearest neighbor algorithm. The query response time of the MRSE method grows with the increase of the document collection, which is difficult to adapt to the demand of the massive data in the cloud computing. Many articles [10–12] are devoted to improving the efficiency of MRSE. The MRSE-HCI method [11] introduces the dynamic hierarchical clustering algorithm to the MRSE, which has an excellent efficiency performance.

However, the methods above cannot solve the three challenges mentioned in sector 1, which includes inaccurate query keywords, synonyms in the query keywords and different focus of the search results. As to challenge II, some scholars proposed semantic-based query, which have implemented the query expansion of query keywords through the semantic dictionary (such as Wordnet) [1]. For the challenge III, the literature [2] allowed users to customize the keyword weight, in order to distinguish the query intention. [3] proposed a personalized ciphertext search framework PRSE. According to the user's search history, a scoring mechanism is established to express the user's personal search preferences. The existing query improvement methods rely on the user to provide accurate query keywords, which is not applicable in case of challenge I. To summarize, the current methods cannot solve all the above three challenges simultaneously.

3 Background and Related Definitions

This section describes the background of the ciphertext retrieval method. Section 3.1 gives the basic model of the personalized search method over encrypted cloud data. Section 3.2 describes the security model of the method. Section 3.3 presents the similarity search tree and the K-MEDOIDS algorithm used in the index building process. The evaluation criteria and main symbol definitions complete this section.

3.1 System Model

The system model includes three entities: the data owner, the data user and the cloud server, as illustrated in Fig. 1. The data owner encrypts and outsources the massive document collection as well as the index built on them to the cloud server. In addition, the data owner entitles the data user to access the documents. Hence, the data user could search over the encrypted documents after authorization. (The authorization and access control are not discussed in this paper.) Meanwhile the cloud server provides the cloud storage and computing resources for the ciphertext retrieval. Once received the encrypted query request from the data user, the cloud server begins to search the index and returns the top k most relevant documents. So far, it is the classic system model of ciphertext retrieval. This paper introduces query refinement to the system model, as shown in step 3 and step 4 of Fig. 1. The user identifies the relevant documents to refine the keywords, and the cloud server returns the final search results.

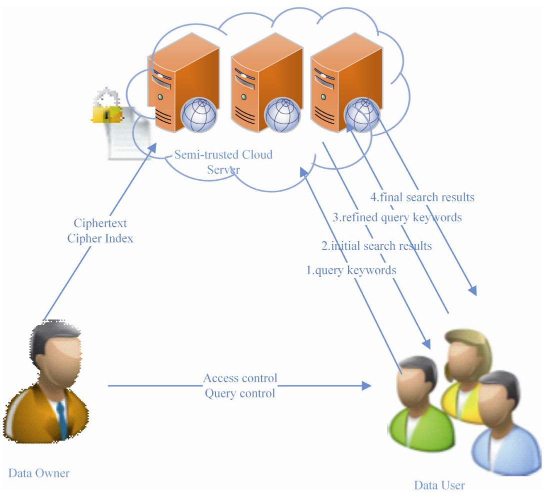

Fig. 1. The system model of personalized ciphertext retrieval

3.2 Threat Model

This paper assumes that the data owner and the data user are trustworthy, while the cloud server is semi-trusted. That means the cloud server would perform the data user's request truthfully but might be curious about the documents' content. According to the information obtained by the cloud server, two threat models are discussed here [9].

Known Ciphertext Model. The cloud server is merely aware of the encrypted data in the cloud. That is, the encrypted documents, the encrypted index and the encrypted query vector.

Known Background Model. The cloud server not only knows all the information the first model mentioned, but also acquires the statistical information of the dataset such as the document/keyword frequency.

3.3 Preliminaries

In this paper, similar search tree is adopted as the index storage structure, and a clustering algorithm is used in the process of index building.

Similarity Search tree (SS tree) [13], which is the deformation of the R tree, is constructed from bottom to top. The upper node is covering all elements of the lower nodes of hyper sphere. Each node contains a center point and radius. If the node is a leaf node, the center is the document vector value; if the node is the intermediate node, the center is the hyper sphere's center.

The K-MEDOIDS algorithm [14] is a commonly used clustering algorithm. It mainly divides n data objects into K clusters. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal. It can be viewed as the most centrally located point in the cluster.

3.4 Evaluation Standard

(1) **Similarity Function**
This paper calculates the dissimilarity rather than similarity between documents. Euclidean distance is adopted to measure the differences between pairs of documents. The documents' vectors such as x and y are first normalized. Then calculate the Euclidean distance of the vectors as Eq. (1).

$$\begin{aligned} D(x, y) &= \sqrt{|x|^2 - 2x \bullet y + |y|^2} \\ &= \sqrt{2 - 2x \bullet y} \end{aligned} \tag{1}$$

(2) **Retrieval Result Evaluation**
The users' satisfaction can be quantified by the search precision. The precision stands for the true correlation rate in the retrieval top k documents, which can be calculated as Eq. (2).

$$P = \frac{the\ number\ of\ relevant\ documents\ in\ the\ results}{the\ number\ of\ retrieval\ documents} \tag{2}$$

(3) **Information Leakage Evaluation**
This paper uses the same information leakage model as [9]. The top k documents are returned by the cloud server, and we try to leak retrieval order information to the cloud server as little as possible. The evaluation method of information leakage is explained below.

$$P_k = \sum_{i=1}^{k} |c_i - c_i'|/k \tag{3}$$

c_i stands for the order of the file i in the top k retrieval. c_i' stands for the real position of file i without random value. The larger value of P_k is, the better the real retrieval results conceals and the less the information leaks.

3.5 Notations

Table 1 provides the notations in this paper.

Table 1. Notation.

D	The plaintext document collection, $D = \{d\mid d_1,d_2,\ldots,d_m\}$, where m is the number of documents
D_r	The relevant plaintext document collection
D_{nr}	The irrelevant plaintext document collection
C	The encrypted document collection, $C = \{c\mid c_1,c_2,\ldots,c_m\}$, where m is the number of documents
W	The dictionary. There are n keywords, $W = \{w\mid w_1,w_2,\ldots,w_n\}$
$D(O_i,O_j)$	The distance between the O_i and O_j
Q_w	The query vector
Q_{wopt}	The optimized query vector
R_{Qw}	The radius of the query sphere
R_n	The radius of sphere in the similarity search tree
min-SS	The minimum number of members contained in the intermediate nodes of the SS tree
max-SS	The maximum number of members contained in the intermediate nodes of the SS tree
u	The number of the increased dimension to enhance the security of document vector
v	The number of dimension randomly chosen from the u dimension

4 MRSE-SSF Method

The section describes the personalized multi-keyword ranked search method based on similarity search tree and relevance feedback (MRSE-SSF). In addition, the specific algorithms of each step are introduced separately.

4.1 MRSE-SSF Method

The MRSE-SSF method consists of five algorithms: (i) a key generation algorithm Keygen (1^n, u), (ii) an encrypted index building method Build-index (D, SK), (iii) a trapdoor producing algorithm Trapdoor (Q_w, SK), (iv) a query request construction algorithm Query (T_w, k, I) and (v) a query refinement algorithm Feedback (D_r, D_{nr}).

- **Keygen (1^n, u):** The data owner randomly generates a splitting indicator vector S which has $(n + u + 1)$ bits and two reversible $(n + u + 1) \times (n + u + 1)$ matrices M_1 and M_2. And the symmetric key is defined as $SK = \{S, M_1, M_2\}$.
- **Build-index (D, SK):** The data owner assigns a document vector $D[i]$ to each document. The weight of keyword w_j in the document is recorded in $D[i][j]$ $(0 < j < n)$ [15]. The index I is constructed according to the index building algorithm. The algorithm's details will be discussed in Sect. 4.3. Then we extend the center vectors in the index from n bits to $(n + u + 1)$ bits. Particularly, the $(n + j)$ $(0 < j < u + 1)$ bit is assigned to a random value R_j and the last $(n + u + 1)$ bit equals 1. Next, based on

the splitting indicator vector S, the center vector is cut into two parts $D[i][j]'$ and $D[i][j]''$ and the index $I = \{I', I''\}$. If S_i is 1, $D[i][j]'$ and $D[i][j]''$ are random values, and the sum is $D[i][j]$; if S_i is 0, then $D[i][j]' = D[i][j]'' = D[i][j]$. Last, the index I is encrypted by the matrices M_1 and M_2, that is, $I = \{M_1^T I', M_2^T I''\}$ and is outsourced to the cloud.

- **Trapdoor (Q_w, SK):** The query vector Q_w has $(n + u + 1)$ bits. The first n-bits value of Q_w is decided by whether the keyword occurred in the dictionary. If occurred, the corresponding position $Q_w[i]$ is 1; if not, the position is 0. Then we randomly choose v from u keywords, and assign the relevant position to 1, and the else is 0. The last bit of Q_w is a random value t. The vector Q_w is then transformed by a random value q, that is $Q_w = (q \bullet Q_w(n + u), t)$. Afterwards Q_w is split according to vector S, and the process is reverse to the previous split. If S_i equals 1, $Q_w[i]' = Q_w[i]'' = Q_w[i]$; if S_i equals 0, $Q_w[i]'$ and $Q_w[i]''$ are random values and the sum is $Q_w[i]$. Last, the matrices M_1 and M_2 encrypted Q_w' and Q_w''. The trapdoor is a triple, $T_w = \{M_1^{-1} Q_w', M_2^{-1} Q_w'', R_{Qw}\}$ and R_{Qw} stands for the radius of the query sphere.
- **Query (T_w, k, I):** During the query process, in order to improve the search efficiency, the cloud server performs the search hierarchically rather than iterating each document. On the basis of T_w, the cloud server firstly computes the relationship between the query sphere and the super sphere in the root note, and finds the sphere which has the max intersection with query sphere. Then search the next layer until the leaf nodes. Last, calculate the distance between the leaf nodes' document vector and the query sphere's document vector, and return the top k document list.
- **Feedback (D_r, D_{nr}):** The data user divides the result set into the relevant document set D_r and the irrelevant document set $D_{nr,}$ and calculates the optimized query keyword vector Q_{wopt}.

Then generate a new trapdoor $T_w' =$ Trapdoor (Q_{wopt}, SK) based on the refined query keyword vector Q_{wopt}, and execute the query Query (T_w', k, I) again with the trapdoor T_w'.

4.2 MRSE-SSF Index Structure Construction Algorithm

In the second step of the MRSE-SSF method, the indexing structure is constructed. The similar search tree is introduced into the index structure, and the hierarchical structure of the document set is represented by the tree structure. The index construction is divided into the following steps:

Step 1: Set the minimum number of nodes m and the maximum number of nodes M in the similarity search tree.
Step 2: Call the K-MEDOIDS clustering algorithm and establish the N_0 minimum sphere.
Step 3: Form a new sphere by uniting the x$(m \leq x \leq M)$ spheres which are close. Call the K-MEDOIDS algorithm, and set k = 1. Then calculate the center point and the radius of the new sphere.
Step 4: Repeat step 3, choose the union result of which the increased volume is the minimum, output the N_i sphere.
Step 5: Set the abovementioned N_i sphere as one layer of similarity search tree.

Step 6: Repeat step 3, 4 and 5, until the number of sphere N_i less than m.
Step 7: The last N_i sphere is the root node of the similarity search tree.

4.3 Query Algorithm

The core idea of the query algorithm is to find the node in the index tree that has the largest intersection with the query hypersphere [13]. The positional relationship between the index tree hypersphere and the query hypersphere can be divided into three situations: contained, intersected and disjoint. Suppose that R_{Qw} represents query hypersphere radius, which is set in the T_w, and R_n denotes hypersphere radius in the similarity search tree. O_{Qw} stands for the center of the query vector that is the same as Q_w, and O_n represents the centroid value of the node in the similarity search tree.

The method of determining intersection, as shown in Eq. (4),

$$(R_{Q_w} + R_n) > D(O_{Q_w}, O_n) > |R_{Q_w} - R_n| \tag{4}$$

The method of determining containment, as shown in Eq. (5),

$$D(O_{Q_w}, O_n) < |R_{Q_w} - R_n| \tag{5}$$

The method of determining disjoint, as shown in Eq. (6).

$$D(O_{Q_w}, O_n) > (R_{Q_w} + R_n) \tag{6}$$

During the query process, we set the state flag to record the positional relationship for the above value. The state flag is in the range within four values: {unmarked, disjoint, intersected, contained}, and the initial value is unmarked.

The specific steps of the query algorithm are as follows:

Step 1: The server first calculates the relationship between the query sphere and the root nodes of each sphere in the index tree. Then get the maximum intersection between the query sphere with a certain sphere in root nodes.
Step 2: According to last step's result, continue to go down one layer of nodes to find the closet sphere to the query sphere.
Step 3: Repeat step 2, until go down to the layer of leaf nodes. Calculate the distance between the leaf nodes and the query sphere's center O_{Qw} and choose the nearest node. Then return the top k nearest documents in that node to the user.

4.4 Relevance Feedback Algorithm

This paper adopts the Rocchio algorithm [16] as the feedback algorithm in the last step of the MRSE-SSF method. The algorithm is to implement the relevance feedback in the vector space model. The basic principle of the algorithm is to find an optimal query vector Q_{wopt}, which has the largest similarity between the relevant documents and the minimum degree of similarity between the unrelated documents. Use D_r to represent the relevant document vector set, and denotes the irrelevant document vector set with D_{nr}, the optimal query vector Q_{wopt} can be expressed as:

$$Q_{wopt} = \arg\max_{Q_w}[\mathrm{sim}(Q_w, \mu(\mathrm{D_r})) - \mathrm{sim}(Q_w, \mu(\mathrm{D_{nr}}))] \tag{7}$$

In the above formula, *u(D)* represents the center of the document vector set, which can be called a centroid, and is calculated as follows. The value of the centroid's each element is the mean of all the corresponding vectors' elements.

$$\mu(\mathrm{D}) = \frac{1}{|D|}\sum_{\vec{d_j}\in D}\vec{d_j} \tag{8}$$

The optimal query vector is calculated as follows. The center of initial query vector is moved by a quantity that is the amount of difference between the centroid of the relevant document and the centroid of the irrelevant document.

$$\begin{aligned} Feedback\left(D_r, D_{nr}\right) &= Q_{wopt} \\ &= Q_w + [\mu(D_r) - \mu(D_{nr})] \\ &= Q_w + [\frac{1}{|D_r|}\sum_{\vec{d_j}\in D_r}\vec{d_j} - \frac{1}{|D_{nr}|}\sum_{\vec{d_j}\in D_{nr}}\vec{d_j}] \end{aligned} \tag{9}$$

5 Security Analysis

5.1 Known Ciphertext Model

Under the known ciphertext model, the adversary can obtain the corresponding ciphertext information including the encrypted documents, the encrypted index and the encrypted query vector, but the encryption key is confidential. This method is based on the adaptation and security enhancement of the secure kNN method [17].

The encryption key of the method consists of two parts, which are the $(n + u + 1)$ bits vector S and $(n + u + 1) \times (n + u + 1)$ of the reversible matrix (M_1, M_2). For simplicity and without loss of generality, we assume that $u = 0$. That is, do not add random values (add random values could increase the security of the method). Since the vector S is unknown, the document vectors $D[i]'$ and $D[i]''$ can be regarded as two random $(n + 1)$ dimensional vectors. In order to solve the linear equations constructed by the encrypted document data, i.e., $C_i' = M_1^T D[i]'$ and $C_i'' = M_2^T D[i]''$ $(0 \le \mathrm{i} \le \mathrm{m})$, we have $2m(n + 1)$ unknowns in m document vectors, and $2(n + 1)(n + 1)$ unknowns in the matrix (M_1, M_2). Because we merely have $2m(n + 1)$ equations, less than the number of unknowns, there is not enough information to solve the document vector or matrix (M_1, M_2).

Similarly, the query vectors Q_w and the optimized query vector Q_{wopt} can be inferred likewise. So the method is safe under the known ciphertext model.

5.2 Known Background Model

In this model, the cloud server not only knows the encrypted information, but also has the ability to analyze the query and results. Then we view the query irrelevance and keyword security in details.

The Query Irrelevance. Due to introducing the random value to the query vector generation process, the trapdoor is generated differently each time. Thus, the cloud server could not associate the query vector with the initial files. Three processes contribute to the randomness of the query vector.

1. Randomly choose v bits from the u bits and assign random values to the v bits.
2. Generate a random value q to normalize the $(n + u)$ bits of the query vector.
3. The last bit of the query vector is set to the random value.

The Keyword Security. When data users are searching the documents, they tend to hide their query keywords from the cloud server. The trapdoors are usually used to protect the query keywords. In the known background model, the cloud server can infer the keywords by the word frequency information (including the number of documents including the keywords). In this paper, the keyword encryption method adopts the secure inner product calculation method [18, 19]. Because the encryption method is proved to meet the keyword security [18], the encryption method proposed in this paper conforms to the keyword security.

By setting the value of u, the user can confuse the relationship between the query and the search results, and increase the difficulty of using the statistical analysis to guess the keywords. However, the increase in the value of u will also reduce the accuracy of the query, therefore, this is the balance between the keyword security and the query accuracy.

6 Performance Analysis

To test MRSE-SSF model's performance on the real dataset, we have established the experimental platform to verify the accuracy and efficiency of the retrieval. The testing platform is built on the Intel Core i7 3.40GZ Linux server, and the data set uses the selected papers published in the last 10 years of IEEE conference, which are 4196 documents and 8341 keywords. Figure 2 shows the efficiency of the MRSE-SSF method, the MRSE-HCI method and the MRSE method.

In Fig. 2(a), when the number of the document collection is exponentially increased, the query response time of MRSE method is exponential, and the query response time of MRSE-SSF and MRSE-HCI is linear. The query time of MRSE-SSF, MRSE and MRSE-HCI is stable in Fig. 2(b), when the number of retrieved documents varies. In Fig. 2(c), when the number of query keywords changes, the MRSE-SSF method still has a great advantage over MRSE.

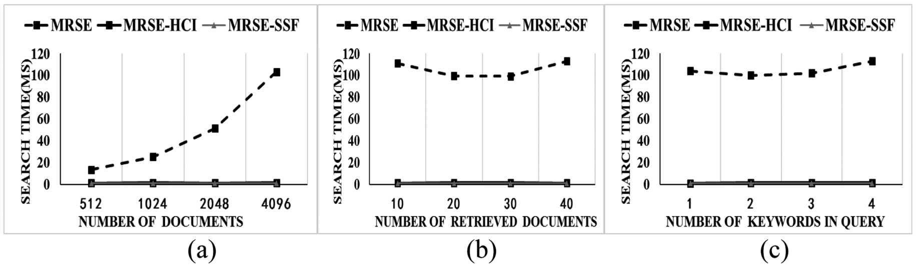

Fig. 2. Search efficiency (a) search time for different number of documents (b) search time for different number of retrieved documents (c) search time for different number of keywords

Figure 3(a) shows the comparison of the MRSE-SSF, MRSE and MRSE-HCI methods in terms of precision. As can be seen from the picture, the precision of MRSE-SSF is improved compared to MRSE-HCI.

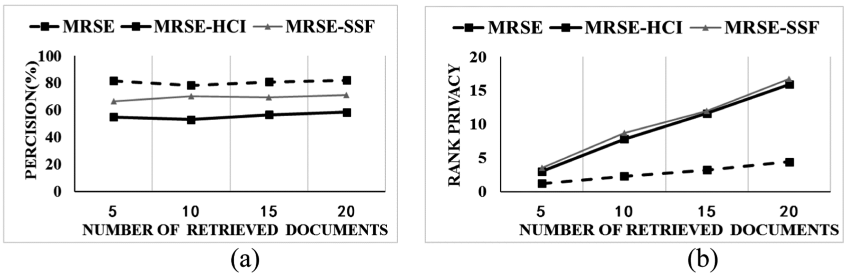

Fig. 3. (a) Precision (b) Rank privacy

Figure 3(b) shows the comparison of MRSE-SSF, MRSE and MRSE-HCI methods on the information leakage of the sorting results. The figure shows that the MRSE-SSF rank privacy is higher than MRSE, which means it protects privacy better.

In sum, the MRSE-SSF is more effective and has more privacy protection than the MRSE method, which has the similar advantages as the MRSE-HCI method. At the same time, the search results' precision of MRSE-SSF method is higher than the MRSE-HCI method, which is the focus of our paper's contribution.

7 Conclusions

In order to improve the quality of search results, this paper proposes a personalized multi-keyword ciphertext retrieval method based on relevance feedback. After the first stage of the query, the user feedbacks the interested documents in the results. Next, an optimized query vector is generated to further improve the search results. This method can provide the data users with personalized search results, which are more in line with the query intent of the users.

References

1. Fu, Z., Sun, X., Ji, S., Xie, G.: Towards efficient content-aware search over encrypted outsourced data in cloud. In: IEEE INFOCOM 2016-the 35th Annual IEEE International Conference on Computer Communications, pp. 1–9. IEEE, April 2016
2. Li, H., Yang, Y., Luan, T.H., Liang, X., Zhou, L., Shen, X.S.: Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. IEEE Trans. Dependable Secure Comput. **13**(3), 312–325 (2016)
3. Fu, Z., Ren, K., Shu, J., Sun, X., Huang, F.: Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE Trans. Parallel Distrib. Syst. **27**(9), 2546–2559 (2016)
4. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy, S&P 2000, pp. 44–55. IEEE (2000)
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
6. Goh, E.J.: Secure indexes. IACR Cryptology ePrint Archive 2003, 216 (2003)
7. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: 2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS), pp. 253–262. IEEE, June 2010
8. Zhang, B., Zhang, F.: An efficient public key encryption with conjunctive-subset keywords search. J. Netw. Comput. Appl. **34**(1), 262–267 (2011)
9. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings IEEE INFOCOM, pp. 829–837. IEEE Press, New York (2011)
10. Tian, X., Zhu, X., Shen, P., Chen, C., Zou, H.: Efficient ciphertext search method based on similarity search tree. J. Softw. **27**(6), 1566–1576 (2016). (in Chinese)
11. Chen, C., Zhu, X., Shen, P., Hu, J., Guo, S., Tari, Z., Zomaya, A.Y.: An efficient privacy-preserving ranked keyword search method. IEEE Trans. Parallel Distrib. Syst. **27**(4), 951–963 (2016)
12. Chen, C., Zhu, X., Shen, P., Hu, J.: A hierarchical clustering method for big data oriented ciphertext search. In: 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 559–564. IEEE, April 2014
13. White, D.A., Jain, R.: Similarity indexing with the SS-tree. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 516–523. IEEE, February 1996
14. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis, vol. 344. Wiley, Hoboken (2009)
15. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann, Burlington (1999)
16. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, vol. 1, no. 1, p. 496. Cambridge University Press, Cambridge (2008)
17. Wong, W.K., Cheung, D.W.L., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 139–152. ACM, June 2009

18. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 71–82. ACM, May 2013
19. Shen, P., Chen, C., Tian, X., Tian, J.: A similarity evaluation algorithm and its application in multi-keyword search on encrypted cloud data. In: Military Communications Conference, Milcom 2015, pp. 1218–1223. IEEE (2015)