IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 23, 2022, accepted April 12, 2022, date of publication April 18, 2022, date of current version April 27, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3168730

An Efficient Search Method Using Features to Match Joint Keywords on Encrypted Cloud Data

LINGBING TAO[1], HUI XU[1], YING SHU[1], AND ZHIXIN TIE[1,2]
[1]School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China
[2]Keyi College, Zhejiang Sci-Tech University, Shaoxing 312369, China

Corresponding author: Zhixin Tie (tiezx@zstu.edu.cn)

This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY13F020043, and in part by the Scientific Research Project of Zhejiang Provincial Department of Education under Project 21030074-F.

ABSTRACT With the continuous improvement of the security of cloud storage, more users upload private data to the cloud. However, a large number of encrypted data using independent keywords to create indexes not only directly increase the storage overhead, but also lead to the decline of search efficiency. Therefore, this paper proposes an efficient search method using features to match joint keywords (FMJK) on encrypted cloud data. This method proposes that each d keywords are randomly selected from the non-duplicated keywords, which are extracted from the documents of the data owner, to generate a joint keyword, and all joint keywords form a keyword dictionary. Each joint keyword is matched with the feature of the document and the query keyword respectively, and the result obtained by the former is regarded as a dimension of the document index, while the result obtained by the latter is regarded as a dimension of the query trapdoor. Finally, the BM25 algorithm is used to calculate the inner product of the document index and the trapdoor, and then sort them and the top k results are returned. Theoretical analysis and experimental results show that the proposed method is more feasible and more effective than the compared schemes.

INDEX TERMS Encrypted cloud data, feature matching, searchable encryption, dimensionality reduction, joint keywords.

I. INTRODUCTION

With the rapid development of science and technology, enterprises or individual users increasingly rely on storing a large number of data documents on cloud servers in order to share data quickly and remotely [1]. However, with the increasing demand, the cost of cloud server storage increases, the efficiency of search decreases, and privacy protection has become a focus of research [2].

In most of the existing cipher-text sorting retrieval methods, KNN (K Nearest Neighbor) technology is used to create indexes supporting cipher-text retrieval [3]–[5]. In the process of massive data encryption search, most of the search encryption schemes have high time complexity and large storage space, which are closely related to the encrypted key, the document index and query request dimensions [6], [7]. Reducing high dimensional data encryption is a solution to improve search efficiency [8]. Some researchers try to study how to enrich the flexibility of retrieval [9], [10], however they still cannot meet the retrieval requirements of a large number of data, and they cannot sort and filter useful data for authorized users. Therefore, in the face of different user needs, it is urgent to find a scheme that can not only guarantee privacy, but also improve retrieval efficiency and ensure query accuracy.

In this paper, we propose an efficient search method using features to match joint keywords (FMJK) on encrypted cloud data based on the MRSE (Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data) scheme [3]. First, it is necessary to extract the features of each document to accurately express its theme. Then each d keywords are randomly selected from the non-duplicated keywords, which are extracted from the documents of the data owner, to generate a joint keyword, and all joint keywords form a keyword dictionary and then the features of each document are matched with the joint keywords to create an index. The authorized user enters the query keywords to match the joint keywords to create a trapdoor, and finally calculates the safe inner product of the trapdoor and the index to return the top k results. The contributions of this article are as follows:

(1) Each d randomly selected keywords form a joint keyword, which matches with the feature of the document to be mapped to one dimension of the index, which reduces the

The associate editor coordinating the review of this manuscript and approving it for publication was Ramakrishnan Srinivasan.

dimension size of the key, the index and the trapdoor, simplifies the matrix operation during encryption, and improves search efficiency.

(2) Through the improved BM25 algorithm [11] to calculate the inner product of the document index and the trapdoor, which not only sorts quickly but also ensures query accuracy.

(3) The randomness of joint keywords and the encryption process of expanding and splitting ensure privacy protection.

The rest of this paper is organized as follows. Section II discusses the related work of searchable technology in cloud computing environment. Section III introduces the system model attack model and research objectives of the searchable encryption process, as well as the explanation of the symbols in this paper. Section IV presents the FMJK scheme in detail. Section V analyzes the FMJK scheme theoretically and experimentally. Finally, the conclusion is drawn in Section VI.

II. RELATED WORK

Remote searching on encrypted data using an untrusted server was first proposed by Dawn et al [12]. Scanning the whole cipher-text file and comparing the cipher-text words, we can confirm the existence of the keyword and even calculate its occurrence times. However, the efficiency of this full-text search method is too low. Curtmola *et al.* [13] proposed an encryption index scheme based on the inverted index, which can achieve the accuracy and effectiveness of the index at the same time. Li *et al.* [14] proposed a flexible multi-keyword query scheme which takes keyword weights and user access history into consideration when generating the query result. Wang *et al.* [15] for the first time proposed a privacy protection scheme that supports both multiple keyword query and fuzzy search. Cao *et al.* [3] proposed a search scheme based on multiple keyword sorting, ranking documents by calculating the inner product score of index vector and request vector. Zhang *et al.* [9] proposed an encryption search based on semantic expansion of central keywords, and used keyword weight method and sub matrix technology to carry out semantic expansion ranking search for central keywords.

At present, more researchers pay attention to the cipher-text retrieval based on multiple keywords. For example, Fu *et al.* [16] proposes a uni-gram segmentation scheme for query keywords that may be spelling less, and improves the retrieval efficiency by considering the weight of keywords. Fu *et al.* [17] replaces traditional keywords with concept map as a knowledge representation, and proposes a privacy preserving intelligence based on concept graph semantic retrieval. Zhao *et al.* [18] introduces user preferences and provides a standardized model to integrate priority ranking and keyword retrieval. Chen *et al.* [19] propose using hierarchical clustering method to realize multiple keyword cipher-text sorting, and uses hash sub-tree structure to verify the integrity of retrieval results.

Although, all the above solutions have improved searchable encryption technology in different directions to varying degrees, however, the cipher-text sorting search technology for multiple keywords needs further research in data privacy, retrieval efficiency and completeness of results.

III. PROBLEM STATEMENT

A. SYSTEM MODEL

As shown in Fig. 1, the model of the searchable encryption system consists of three different entities, which are the cloud server, the data owner, and the data user. The data owner first takes all data files as documents, and extracts keywords from these documents. Then combines each d randomly selected keywords into a joint keyword, and all joint keywords form a keyword dictionary. Thirdly, creates a document index for each document using its exacted keywords and the keyword dictionary and encrypts the document and its encrypted index by means of the key encryption. Finally, the data owner uploads the encrypted documents and their indexes to the cloud server for storage. While querying, the authorized data user inputs multiple keywords, the corresponding keyword trapdoor will be created and submitted to the cloud server. After receiving the search request, the cloud server first calculates the security inner product of the keyword trapdoor and each document index to get the score of each document, and then returns the top k encrypted documents with the highest score to the authorized the data user. After receiving the returned documents, the data user decrypts the encrypted documents using the keys provided by the data owner to obtain the required data information.

B. ATTACK MODEL

In the process of cipher-text searching, the cloud service provider has the characteristics of "honest but curious" [3], that is, the cloud server will perform search operations honestly as well as they may infer and analyze encrypted information based on the uploaded data, the search trapdoors and some additional background knowledge. Therefore, two threat models with different attack capabilities will be considered in the whole cipher-text search process [3]:

Level 1: The cloud server knows the encrypted data sets and the searchable indexes.

Level 2: The cloud server not only knows the encrypted data sets and searchable indexes, but also some background information, including the correlation between trapdoors, some statistical information from the encrypted data sets and searchable indexes.

C. SYMBOL DESCRIPTION

The symbols to be used in this paper and their descriptions are shown as follows:

F : The document set in plaintext, denoted as $F = \{F_1, F_2, \cdots, F_m\}$.

C : The encrypted document set, denoted as $C = \{C_1, C_2, \cdots, C_n\}$.

G : The feature sets that consists of the feature of each document in the document set, denoted as $G = \{g_1, g_2, \cdots, g_m\}$.

W_x : The $x - th$ joint keywords composed of d keywords.

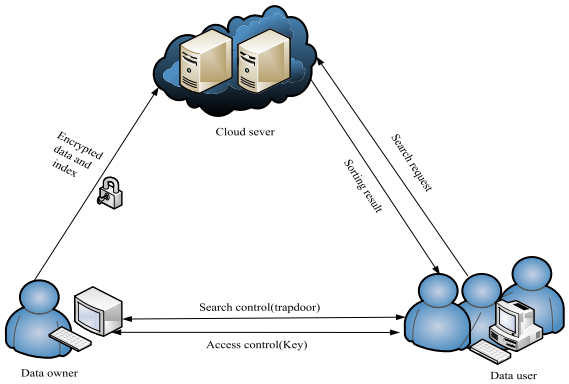

FIGURE 1. Searchable encryption system model.

W : The keyword dictionary, denoted as $W = \{W_1, W_2, \cdots, W_t\}$.

t : The total number of joint keywords in the keyword dictionary.

I : The encrypted index set, $I = \{I_1, I_2, \cdots, I_m\}$.

q : The query keyword set, denoted as $q = \{q_1, q_2, \cdots, q_z\}$.

Q : The query vector.

T : The query trapdoor.

D. PRELIMINARIES

Document feature set: the keywords of a document are extracted according to the word frequency. All keywords of a document form its feature, which accurately represents the theme of the document. m documents' features constitute the document feature set.

Keyword set: the keywords of all documents are extracted, summarized, and de-duplicated to form a keyword set, in which the number of keywords is denoted by n.

Joint keyword: each d keywords are randomly selected from n keywords in the keyword set to make a keyword phrase, namely the joint keyword. The keywords in the joint keyword can be in any order.

Keyword dictionary: all joint keywords are put together to form a keyword dictionary. The number of joint keywords is $t = \lceil n/d \rceil$.

Correlation score of multiple keywords and documents: For each morpheme $W_{xj}(j = 1, 2, \cdots, d)$ in the $x - th$ joint keywords $W_x(x = 1, 2, \cdots, t)$ calculates the weighted summation of the correlation score between the morpheme W_{xj} and the document $F = \{F_1, F_2, \cdots, F_m\}$. The morpheme W_{xj} is different importance to a document. Generally, the higher the frequency of occurrence, the more important it is to this document, but the length of a document will affect the importance of this morpheme to this document, that is, W_{xj} is for a long document is less important than a short document. Therefore, we can use (1) to calculate the relevance score $R(W_x, F_i)$ of the $x - th$ joint keywords $W_x(x = 1, 2, \cdots, t)$ and the document $F_i(i = 1, 2, \cdots, m)$.

$$R(W_x, F_i) = \sum_{j=1}^{d} \frac{f_j \times (k_1 + 1)}{f_j + k_1 \times \left(1 - b + b \times {}^{dl}/_{avgdl}\right)} \quad (1)$$

where k_1, b is the adjustment factor, the function of parameter b is to adjust the impact of the length of the document on the importance. They are set according to experience, generally $k_1 = 2, b = 0.75$. f_j is the frequency of W_{xj} in document F_i, dl is the length of document F_i, $avgdl$ is the average length of m documents. At the same time, for each morpheme q_i in the query keyword set q, the more documents contain morpheme q_i, the less important q_i is. Because when many documents contain morpheme q_i, the document discrimination of q_i is very low. Therefore, we can use (2) to calculate the correlation weight between the query keywords and documents.

$$IDF(q) = \sum_{i=1}^{N_q} \log\left(\frac{m - y(q_i) + 0.5}{y(q_i) + 0.5}\right) \quad (2)$$

where N_q is the number of the keywords in q, m is the number of all documents, and $y(q_i)$ is the number of documents containing the keyword q_i.

Search relevance score: Usually the BM25 algorithm [11] can be used as a searchable relevance score, but this paper uses the safe inner product similarity to judge the result ranking output, thus the BM25 algorithm is slightly modified to fit the framework of this paper. We can use (3) to calculate the score of document F_i when submitting query request Q.

$$Score(Q, F_i) = IDF(q) \times R(W_x, F_i) \quad (3)$$

IV. THE PROPOSED FMJK METHOD

In this section, we will introduce the proposed FMJK method in detail, which includes six steps.

A. DOCUMENT PROCESSING

For each document $F_i(i = 1, 2, \cdots, m)$, the data owner extracts its feature keywords to form the feature representation g_i, which can accurately represent the theme of the document F_i. The features of m documents constitute the document feature set G.

B. CREATING A KEYWORD DICTIONARY

In this stage, the keywords of all documents will be extracted. All non-duplicated keywords are put together to generate a keyword set, which contains n keywords, thus the dimension size of the keyword set is n. At this time, the random algorithm is used to combine each d randomly selected keywords from the keyword set into a joint keyword W_x, and all joint keywords form a keyword dictionary W, thus, the dimension of the keyword dictionary W is $t = \lceil n/d \rceil$.

C. GENERATING THE KEY

At this stage, the data owner executes the key generation algorithm to generate two random invertible matrices M_1, M_2 and the partition indicator S. Here, M_1, M_2, which will be used for encrypting the created index of each document and the query trapdoor vector, are $(t + u + 1) \times (t + u + 1)$ random invertible matrices, and $S \in \{0, 1\}^{t+u+1}$, where t is the size of the keywords dictionary W, and $(u + 1)$ is

Algorithm 1 The Score Between the Feature and the Joint Keywords

Require:
g_i : The i-th feature keywords in G;
X : The keywords of the i-th joint keywords W_x in W.
F_i : The i-th document in F.
Ensure:
$h_{x,b}$: The b-th keyword in $g_i \cap X$;
FR : The Correlation score of $h_{x,b}$;
$R(h_{x,b}, F_i)$: The weighted score calculated by (1).
$FR = 0$;
For each keyword $h_{x,b}$ in $g_i \cap X$ do
$FR = FR + R(h_{x,b}, F_i)$
End for
Return FR;

the extended dimension. Thus, the key can be expressed as $K = \{S, M_1, M_2\}$.

D. CREATING THE INDEX

Step 1: Creating an initial index P_i for each document $F_i(i = 1, 2, \cdots, m)$. The data owner creates a t dimensional index vector P_i for each document. The value of each dimension of P_i is the weighted sum of the relevance scores, which is the weight between the document features of the document and the joint keywords in the dictionary. The scoring algorithm is shown in Algorithm 1.

Step 2: Expanding the dimensions of the index P_i. The index P_i is expanded from t dimensions to $(t + u + 1)$ dimensions. The values of the first t dimensions remain the original value, and the values of the $(t+1)-th$ to $(t+u)-th$ dimensions are set to any random number $\varepsilon_i(i = 1, 2, \cdots, u)$, which must obey the same uniform distribution $U(\mu-\mathrm{c}, \mu+\mathrm{c})$. The value of $(t + u + 1) - th$ dimension is set to a constant 1.

Step 3: Dividing the extended index randomly. The extended index is randomly split into two indexes P_i' and P_i'' according to the value of the split indicator S, the value of their $j - th(j = 1, 2, \cdots, t + u + 1)$ dimension are assigned according to (4):

$$\begin{cases} P_i'[j] = P_i''[j] = P_i[j] & for\ S[j] = 0 \\ P_i'[j] + P_i''[j] = P_i[j], \quad P_i'[j] \neq P_i''[j] & for\ S[j] = 1 \end{cases} \tag{4}$$

Step 4: The split indexes P_i' and P_i'' are encrypted respectively. At this time, it is necessary to encrypt the indexes P_i' and P_i'' by using the generated symmetric keys M_1, M_2, and get the encrypted index of each document $I_i = \{P_i'^T M_1, P_i''^T M_2\}$.

E. GENERATING THE TRAPDOOR

Step 1: Creating a query request Q. A t dimensional query request vector Q will be created when an authorized user enters keywords to query. The value of each dimension of P_i is the weighted sum of the relevance scores, which is the weight between the query keywords and the joint keywords in the dictionary. The scoring algorithm is shown in Algorithm 2.

Step 2: Expanding the dimensions of query request Q. The dimension of Q will be expanded from t dimensions to

Algorithm 2 The Score Between Query Keywords and the Joint Keywords

Require:
q : The query keyword set;
X : The original keywords of the x-th joint keywords W_x in W.
Ensure:
$h_{k,x}$: The k-th keyword in $q \cap X$;
CS : The Correlation score of $h_{k,x}$;
$IDF(h_{x,b})$: The weighted score calculated by (2).
$CS = 0$;
For each keyword $h_{k,x}$ in $q \cap X$ do
$CS = CS + IDF(h_{k,x})$
End for
Return CS;

$(t + u + 1)$ dimensions. The query request now becomes $\bar{Q}$. v numbers are randomly selected from $(t + 1)$ to $(t + u)$ to generate a set V. The value of the $j-th(j = 1, 2, \cdots, t+u+1)$ dimension of the query request $\bar{Q}$ is assigned according to (5).

$$\begin{cases} \bar{Q}[j] = Q[j] & for\ j = 1, 2, \cdots, t \\ \bar{Q}[j] = r & for\ j \in V \\ \bar{Q}[j] = 0 & for\ j \in [t + 1, t + u],\ j \notin V \\ \bar{Q}[j] = e & for\ j = t + u + 1 \end{cases} \tag{5}$$

where r and e are random numbers.

Step 3: Dividing the extended query request randomly. The extended query request is randomly split into two query vectors Q' and Q'' according to the value of the split indicator S, the value of their $j - th(j = 1, 2, \ldots, t + u + 1)$ dimension is assigned according to (6).

$$\begin{cases} Q'[j] = Q''[j] = \bar{Q}[j] & for\ S[j] = 1 \\ Q'[j] + Q''[j] = \bar{Q}[j], Q'[j] \neq Q''[j] & for\ S[j] = 0 \end{cases} \tag{6}$$

Step 4: The split query request vectors Q' and Q'' are encrypted respectively. At this time, it is necessary to encrypt the query request vectors Q' and Q'' by using the generated symmetric keys M_1, M_2, and get the trapdoor $T = \{M_1^{-1}Q', M_2^{-1}Q''\}$.

F. QUERY

When the cloud server receives the trapdoor uploaded by the authorized user, it starts to execute the search algorithm, where $R = Search(I, T, k)$. The specific algorithm process is as follows:

After receiving trapdoors, the cloud server get the correlation score of the search according to (7), and return the top k documents with the highest score. At this time, the user gets the encrypted documents, and then he decrypts these documents to get information.

$$\begin{aligned} I_i \bullet T &= \{P_i'^T M_1, P_i''^T M_2\} \bullet \{M_1^{-1}Q', M_2^{-1}Q''\} \\ &= P_i'^T \bullet Q' + P_i''^T \bullet Q'' \\ &= r(I_i Q + \sum_{k \in V} \varepsilon_k) + e \\ &= r(Score(w, F_i) + \sum_{k \in V} \varepsilon_k) + e \end{aligned} \tag{7}$$

V. PERFORMANCE ANALYSIS

A. EFFICIENCY ANALYSIS

In the process of searching encrypted documents, the efficiency of the search is generally related to the dimensions of creating keys, indexes, and trapdoors, while their dimensions depend on the dimension of the keyword dictionary. In order to reduce the dimensionality of the keyword dictionary without affecting the accuracy of the search, the FMJK scheme proposes that each d keywords form a joint keyword, which matches with the feature of the document to be mapped to one dimension of the index. Then the algorithm process of creating indexes and trapdoors saves high-dimensional key matrix calculation time and the encryption time, which improves the search speed.

From the perspective of complexity, the dimension of the key during encryption in the MRSE scheme is n, which is the total number of keywords in the keyword dictionary. Because the dimension needs to be expanded to increase data security, the $u + 1$ is the expanded dimension. At this time, the key is two $(t + u + 1) \times (t + u + 1)$ dimensional invertible matrix M_1, M_2, then the time complexity of m document encryption is $O(m(n+u)^2)$, the time complexity of generating trapdoor is $O((n+u)^2)$. The proposed FMJK scheme changes the keywords of the dictionary. Each d (d greater than 1) keywords form a joint keyword, and the dimension of the keyword dictionary becomes $t = \lceil n/d \rceil$. The time complexity of creating an index and trapdoor can be approximately expressed as $O(m(n+du)^2/d^2)$, $O((n+du)^2/d^2)$ respectively, thus the encryption time of the FMJK scheme is reduced and the speed is improved. At the same time, the query algorithm process adopts the vector inner product sum method, and the query time is related to the number of documents and the size of the keyword dictionary. The query time complexity of the MRSE scheme is $O(m(n + u))$, according to the FMJK scheme of this paper, its query time complexity is similar to $O(m(n + du)/d)$, thus, due to d is a positive integer greater than 1, the efficiency of the query has also been improved.

B. PRIVACY ANALYSIS

The security of the key. The cloud server may reversely deduce the main content information of the document according to the index information, however the general data document and the index encryption are traditional symmetric encryption technology, and the key is a $(t+u+1)\times(t+u+1)$ dimensions matrix, which is generated randomly. Under the attack model with known background, the cloud server knows the index encryption process $I_i = \{P_i'^T M_1, P_i''^T M_2\}$. For the i-th document $F_i(i = 1, 2, \cdots, m)$, the process can create the following equation:

$$\begin{cases} P_i'^T M_1 = \tilde{P}_i'^T \\ P_i''^T M_2 = \tilde{P}_i''^T \end{cases} \tag{8}$$

For the i-th document $F_i(i = 1, 2, \cdots, m)$, based on the part D of section IV, we can know that P_i' and P_i'' are derived from P_i which is the index of the document $F_i(i = 1, 2, \cdots, m)$, and after dimension expanding, dimensions of the index P_i is $(t + u + 1)$. Thus the equation number is $2(t+u+1)$, however, P_i' and P_i'' have $2(t+u+1)$ unknowns, and M_1, M_2 have $2(t + u + 1)^2$ unknowns, thus the total unknown number is $2(t + u + 1) + 2(t + u + 1)^2$. For m documents, there will be produced $2m(t + u + 1)$ equations, and the unknown numbers are $2m(t + u + 1) + 2(t + u + 1)^2$. Thus the cloud server does not have enough information to get the key.

Keyword and index information security. The cloud server cannot know the total number of keywords, thus in the process of generating the keyword dictionary, each d keywords are randomly formed into a joint keywords to make the dictionary more random and improve privacy protection. At the same time, in the process of creating index, the document feature keyword set is matched with the joint keywords to generate a weighted score, which protects the keyword information. And the expansion part of the indexes introduces a group of random number $\varepsilon_1, \varepsilon_2, \cdots \varepsilon_u$. Because of the cloud server will leakage the content of the documents based on the information about document indexes, thus, the $\varepsilon_1, \varepsilon_2, \cdots \varepsilon_u$ are unequal random numbers to protect the privacy of the indexes. And each random number $\varepsilon_i(i = 1, 2, \cdots u)$ conforms to the same standard normal distribution. The larger the value of the standard deviation δ, the more difficult it is for the cloud server to find the information of the index vector according to the method of statistical analysis, but considering the accuracy of the search, the value of δ is reasonably selected.

Security of query information and uncorrelation of trapdoor. In the process of generating query vector by query keywords in the scheme, the query keyword set is matched with the joint keywords in the dictionary to calculate the score, which protects the security of query information. In addition, the random numbers r and e are introduced into the extended dimension of trapdoors to confuse the public. Even if the same query keywords are input, different trapdoors will be generated, i.e., the independence, which prevents the cloud server from analysing the query information according to the statistical information of query keywords and the correlation score.

C. EXPERIMENT ANALYSES

The RFC (Request For Comments) [20] data is selected as the experimental data set. The experimental system is implemented in the Java language, and the experimental environment is Window10 server, with an INTEL quad-core 3540 (2.66GHz) processor and 8G memory.

The first experiment is about the choice of the value of the parameter d. This experiment takes 6000 documents as an example to test the time comparison of creating index, generating trapdoor, and query while $d = 1, 2, \cdots, 10$. The experimental results are shown in Fig. 2, 3 and 4.

Fig. 2, 3 and 4 show that the time efficiency of creating the indexes, the trapdoors and the query decreases and tends to balance gradually while parameter d increase.

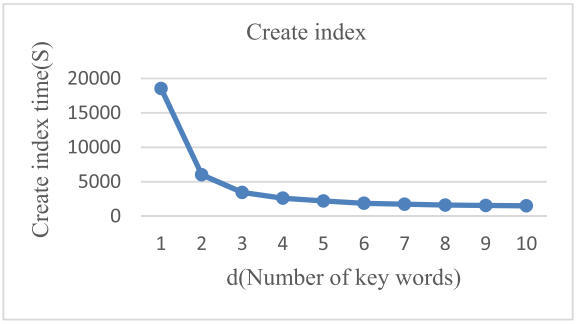

FIGURE 2. The index creation time changes with the value of the parameter *d*.

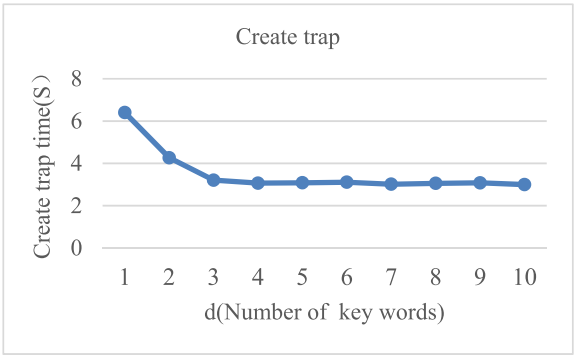

FIGURE 3. The trapdoor creation time changes with the value of the parameter *d*.

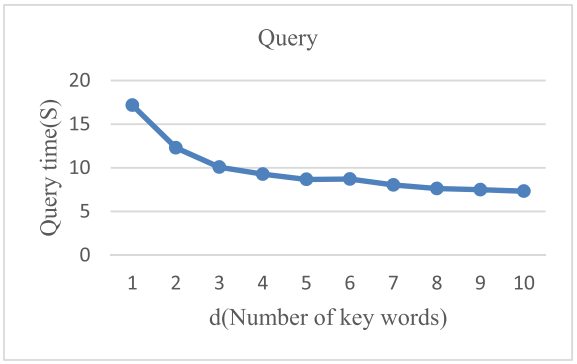

FIGURE 4. The time of query changes with the value of parameter *d*.

The index creation time and its value decreasing with each increase 1 in the parameter d (Number of key words) are listed in Table 1. As shown in the table, the index creation time while $d = 2$ is 67.58% less than that of while $d = 1$, and the index creation time while $d = 3$ is 42.62% less than that of while $d = 2$, and the index creation time while $d = 10$ is only 3.04% less than that of while $d = 9$. From Fig 3, 4, it can be found that the trapdoor creation time and the time of query have similar trends. Thus the later experiment will take $d = 7$ to do the rest of experiments.

In the second experiment, we will take the FMJK scheme compared with the schemes CRQM (A Novel Category Group Index Mechanism for Efficient Ranked Search of Encrypted Cloud Data) [8] and the MRSE to test the time change and accuracy of creating indexes, generating trapdoors and query when the number of documents is 1000, 2000, . . . , 6000 with the extension dimensions $u = 100$.

TABLE 1. The index creation time and its decreasing value as parameter *d* (Number of key words) is increased by 1 each time.

d (NUMBER OF KEY WORDS) IS INCREASED BY 1 EACH TIME

parameter d (Number of key words)	CREATE INDEX TIME(S)	decreasing value as parameter d (Number of key words) is increased by 1	Reduction percent as parameter d (Number of key words) is increased by 1
1	18544.184	/	/
2	6012.599	12531.585	67.58%
3	3449.954	2562.645	42.62%
4	2616.190	833.764	24.17%
5	2223.261	392.929	15.02%
6	1884.314	338.947	15.25%
7	1741.000	143.314	7.61%
8	1617.795	123.205	7.08%
9	1558.896	58.899	3.64%
10	1511.496	47.400	3.04%

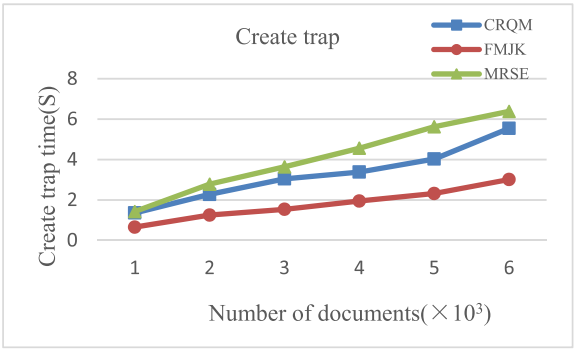

FIGURE 5. Comparison of the three schemes in the time of creating index.

The experimental results are shown in Fig. 5, 6 and 7. We used the number of documents as a variable. In order to compare the fairness and accuracy of the results, the extension dimensions u of the three schemes is equal and the number of query keywords is set to 10.

Fig. 5 shows that the index creation time of the three schemes increases linearly with the number of documents increasing. Because the larger the number of documents, the larger the total number of keywords extracted. Moreover, the dimension of generating key and creating index is related to the dimension size of the keywords dictionary. The larger the dimension, the larger the time of index creation and encryption, thus, the time increases linearly when creating index.

The FMJK scheme reduces the dimension of the keyword dictionary from the beginning, thus, the process of dimension expansion, segmentation and encryption of the index has been greatly improved. While the CRQM scheme reduces the dimension after the initial index is created, it is not different from the MRSE. Therefore, it can be seen that the FMJK scheme takes less time to create index, and the larger the number of documents, the more obvious the improvement.

Fig. 6 shows that the CRQM and the FMJK create trapdoors faster than the MRSE, which is due to the CRQM and the FMJK reduce trapdoors dimension, which is the same as creating index. From the results, the FMJK scheme is the

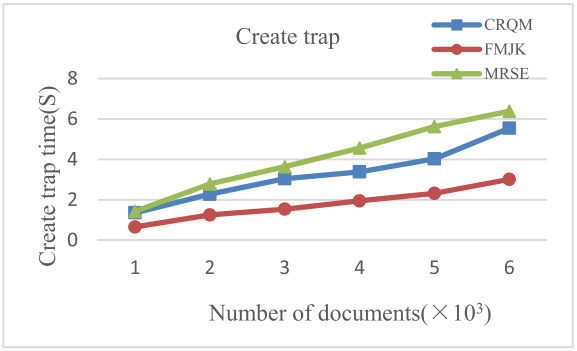

FIGURE 6. Comparison of the three schemes in the time of creating trapdoor.

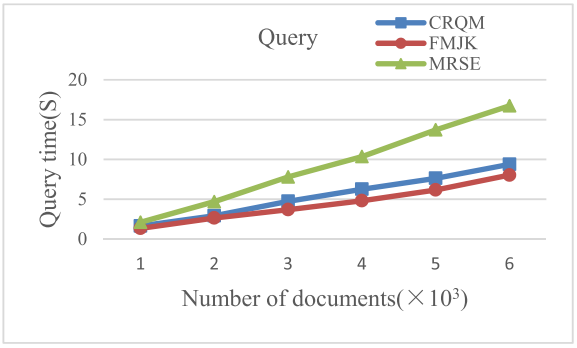

FIGURE 7. Comparison of the three schemes in the time of query.

fastest, because each 7 keywords constitute a joint keyword, which greatly reduces the dimension of creating trapdoors.

Fig. 7 shows that the change of search query time as the number of documents increases, taking 10 documents returned as an example. The CRQM and the FMJK schemes are faster than the MRSE. In general, the FMJK is slightly faster than the CRQM. This is due to the larger the number of documents, the dimensions of the indexes and the trapdoors increase. However, in order to speed up the search efficiency, the CRQM and the FMJK schemes dimensions are reduced.

Another advantage of the FMJK is that it ensures the accuracy of the query. The query accuracy is represented by a, which is defined as follows:

$$a = L_m / L_n \tag{9}$$

where L_n is expressed as number of documents returned during query, L_m is expressed as number of documents containing query keywords.

Due to the PCA dimension reduction in the CRQM scheme, some components will be lost in the process of extracting principal components, which will affect the accuracy of query. Although the FMJK scheme combines multiple keywords into a joint keyword, it reduces the dimension, however, when creating indexes and trapdoors, the weighted score is obtained by matching the features of each document with the joint keywords, thus, there is no loss of precision in the query results. The CRQM scheme uses a threshold value to reduce the loss of principal components, and the threshold value in this experiment is 0.95. When the total number of documents is 5000, the accuracy of different number of

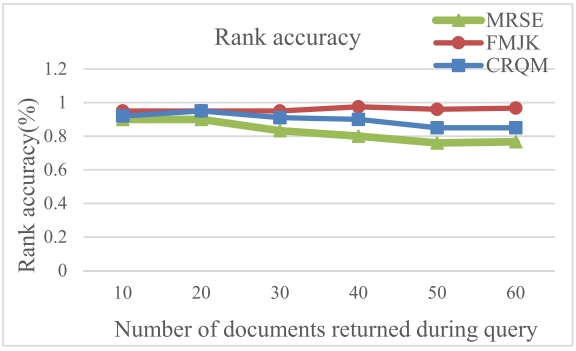

FIGURE 8. Comparison of the three schemes in query accuracy.

TABLE 2. Comparison of the CRQM and the FMJK in the storage space of indexes and trapdoors.

Number of documents	CRQM(indexes) (MB)	FMJK(indexes) (MB)	CRQM(trapdoors) (KB)	FMJK(trapdoors) (KB)
1000	7.81	7.80	16.00	8.00
2000	22.30	15.62	16.00	8.00
3000	40.10	23.40	16.00	8.00
4000	62.50	31.20	16.00	8.00
5000	78.10	39.00	16.00	8.00
6000	94.00	47.00	16.00	8.00

retrieved documents is used to test the query accuracy of the three schemes. As shown in Fig. 8, comparison of query accuracy of the three schemes, the accuracy of the FMJK scheme is better than that of the CRQM scheme, while the accuracy of the CRQM scheme is better than that of MRSE scheme. The accuracy of all three schemes is higher than 75%, however the FMJK scheme is higher than the MRSE scheme. The FMJK scheme is 5% higher than the MRSE scheme in the worst case and 20% higher in the best case.

The CRQM and FMJK schemes need to store the encrypted indexes and trapdoors. Each dimension of the indexes and trapdoors occupies a certain storage space. In order to verify that the FMJK scheme reduces the storage space, the indexes and trapdoors storage experiments are compared with the CRQM scheme when the number of documents is 1000, 2000, . . . , 6000. The experimental results are shown in Table 2.

As can be seen from Table 2, the storage metrics of the FMJK scheme is significantly lower than those of the CRQM, with approximately 50% reduction in the best case. As for the storage cost of the indexes, the CRQM dimension reduction method does not want the loss of principal components to be too serious, so the threshold is required to strictly control the dimension reduction, and the cost is higher. In terms of the storage cost of trapdoors, the trapdoors generated by the query keywords are not affected by the number of documents in storage space.

VI. CONCLUSION

In this paper, we propose an efficient search method using features to match joint keywords (FMJK) on encrypted cloud data. This method proposes that each d keywords are randomly selected from the non-duplicated keywords, which are extracted from the documents of the data owner, to generate a

joint keyword, and all joint keywords form a keyword dictionary, which greatly reduces the dimension of the keywords dictionary. Because the dimension of creating indexes and trapdoors is related to the dimension of keyword dictionary, it also reduces the dimension of the key, the indexes and the trapdoors, which improves the search efficiency. In the process of creating each dimension of the indexes and the trapdoors, the document features and query keywords are accurately matched with the joint keywords in the keywords dictionary to get a weighted score, which ensures the accuracy of the query. And the reduction of dimension also reduces the overhead of storage space occupied by the indexes and the trapdoors. The theoretical analysis and experimental results show that the proposed method is more feasible and more effective than the compared schemes.

REFERENCES

[1] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 6, pp. 1083–1095, Nov./Dec. 2016.

[2] Y. Yang, H. Lin, X. Liu, W. Guo, X. Zheng, and Z. Liu, "Blockchain-based verifiable multi-keyword ranked search on encrypted cloud with fair payment," *IEEE Access*, vol. 7, pp. 140818–140832, 2019.

[3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837.

[4] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2009, pp. 139–152.

[5] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Trans. Commun.*, vol. E98.B, no. 1, pp. 190–200, 2015.

[6] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Jan. 2016.

[7] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, Nov. 2014.

[8] L. Liu and Z. Liu, "A novel fast dimension-reducing ranked query method with high security for encrypted cloud data," *Chin. J. Electron.*, vol. 29, no. 2, pp. 344–350, Mar. 2020.

[9] W. Zhang, Y. Lin, and G. Qi, "Catch you if you misbehave: Ranked keyword search results verification in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 74–86, Mar. 2015.

[10] Z. Guan, X. Liu, L. Wu, J. Wu, R. Xu, J. Zhang, and Y. Li, "Cross-lingual multi-keyword rank search with semantic extension over encrypted data," *Inf. Sci.*, vol. 514, pp. 523–540, Apr. 2020.

[11] M. Murata, H. Nagano, R. Mukai, K. Kashino, and S. Satoh, "BM25 with exponential IDF for instance search," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1690–1699, Oct. 2014.

[12] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2000, pp. 44–55.

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Jan. 2011, doi: 10.3233/JCS-2011-0426.

[14] R. Li, Z. Xu, W. Kang, K. C. Yow, and C.-Z. Xu, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Gener. Comput. Syst.*, vol. 30, no. 1, pp. 179–190, Jan. 2014, doi: 10.1016/j.future.2013.06.029.

[15] J. Wang, H. Ma, T. Qiang, L. Jin, H. Zhu, S. Ma, and X. Chen, "A new efficient verifiable fuzzy keyword search scheme," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 3, no. 4, pp. 61–71, Dec. 2012.

[16] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Jul. 2016, doi: 10.1109/TIFS.2016.2596138.

[17] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017, doi: 10.1109/TIFS.2017.2692728.

[18] R. Zhao, H. Li, Y. Yang, and Y. Liang, "Privacy-preserving personalized search over encrypted cloud data supporting multi-keyword ranking," in *Proc. 6th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2014, pp. 1–6.

[19] C. Chen, X. Zhu, P. Shen, J. Hu, S. Guo, Z. Tari, and A. Y. Zomaya, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.

[20] *RFC Index*. Accessed: May 25, 2020. [Online]. Available: https://www.rfc-editor.org/rfc-index-100a.html

LINGBING TAO received the B.S. degree in electronic engineering from Zhejiang University, Zhejiang, China, in 2002. He is an Experimentalist with the School of Information Science and Technology, Zhejiang Sci-Tech University (ZSTU), Zhejiang. His current research interests include network security, e-commerce, and ubiquitous computing.

HUI XU received the B.S. degree in computer science and technology from Hubei Polytechnic University, Hubei, China, in 2018. She is currently pursuing the M.S. degree in computer technology with the School of Information Science and Technology, Zhejiang Sci-Tech University (ZSTU), Zhejiang, China. Her research interests include network security and community mining.

YING SHU received the B.S. degree in electronic information engineering from Jimei University, Xiamen, China, in 2020. She is currently pursuing the M.S. degree in electronics and communication engineering with the School of Information Science and Technology, Zhejiang Sci-Tech University (ZSTU), Zhejiang, China. Her research interests include data mining and network security.

ZHIXIN TIE received the Ph.D. degree in computer science from Zhejiang University, China, in 2000. He is currently an Associate Professor with the School of Information Science and Technology, Zhejiang Sci-Tech University (ZSTU), China. His current research interests include mobile agent systems, embedded systems, community mining, data mining, and power automation systems.