IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 13, 2020, accepted June 2, 2020, date of publication June 15, 2020, date of current version June 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002236

A Novel Feature Matching Ranked Search Mechanism Over Encrypted Cloud Data

LIANGGUI LIU AND QIUXIA CHEN
College of Information Science and Technology, Zhejiang Shuren University, Hangzhou 310015, China

Corresponding author: Lianggui Liu (felix@zjsru.edu.cn)

This work was supported in part by the National Natural Science Foundation of China and Civil Aviation Administration of China under Grant U1533133, in part by the National Natural Science Foundation of China under Grant 61002016 and Grant 61711530653, in part by the Zhejiang Provincial Natural Science Foundation under Grant LQ18F030006, in part by the China Scholarship Council under Grant 201708330439, and in part by the Startup research funding of Zhejiang Shuren University under Grant 2020R001.

ABSTRACT Encrypted search technology has been studied extensively in recent years. With more and more information being stored in cloud, creating indexes with independent keywords has resulted in enormous storage cost and low search accuracy, which has become an urgent problem to be solved. Thus, in this paper, we propose a new feature matching ranked search mechanism (FMRSM) for encrypted cloud data. This mechanism uses feature score algorithm (FSA) to create indexes, which allows multi-keywords which are extracted from a document as a feature to be mapped to one dimension of the index. Thus, the storage cost of indexes can be reduced and the efficiency of encryption can be improved. Moreover, FMRSM uses a matching score algorithm (MSA) in generating trapdoor process. With the help of FSA, the matching score algorithm can rank the search results according to the type of match and the number of matching keywords, and therefore it is able to return results with higher ranking accuracy. Comprehensive analysis prove that our mechanism is more feasible and effective.

INDEX TERMS Encrypted search, feature score, storage cost, matching score, ranking accuracy.

I. INTRODUCTION

In order to save local storage cost, more and more people are willing to store their private data in cloud servers. However, there are potential risks in cloud storage environments, since data leakage events have happened more and more frequently in recent years. Many people have to encrypt their data before uploading to the cloud to ensure the security of private information. For the untrustworthy cloud environment, many scholars have proposed their solutions for different cloud storage problems [1]–[4]. Their works focus on designing a searchable encrypted index that hides document information from cloud server and can only be computed through specific trapdoors. The searchable encrypted index can be a Bloom filter [5]–[7] generated by mapping keywords, or an index vector [8], [9] reflecting the importance of keywords. However, these schemes not only have large storage cost but also have low search accuracy.

In encrypted search schemes, it is particularly difficult to find a solution that can satisfy the user's accurate query. Although there are many searchable methods that support multi-keyword search [10], [11], they do not consider the relationship between extracted keywords. In addition, existing methods which focus on judging the importance of a keyword for a document are not sufficiently, and when a large number of keywords are extracted from the document, it will inevitably lead to a huge storage cost. In order to address these missing or incomplete search problems, many scholars have proposed fuzzy keyword search and ranking search [12]–[17]. These solutions can greatly enrich query results, but they often cannot meet users' actual search requirements.

As we all know, in plain document query, phrase search method is widely used and has achieved better effects. Recently, many scholars have begun to use the method of phrase or conjunctive keyword search in encrypted search system. Comparing with single keyword or multi-keyword search, the phrase search method can obtain higher query precision. However, when the number of phrases which is extracted from documents increases, the phrase search will result in a huge computational cost, and this cost tends to increase with the number of phrases [18]. Therefore, it is urgent to find a feasible and effective search solution that can not only reduce computational burden but also return results accurately.

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

In this paper, we propose a new feature matching ranked search mechanism (FMRSM) for encrypted cloud data. In the process of creating indexes, we first propose a feature score algorithm (FSA) that can map multiple keywords which are extracted from a document to one dimension of the index. Therefore, comparing with creating indexes with independent keywords, this mechanism can not only reduce index dimensions, but also improve the efficiency of encryption. During the generating trapdoor process of FMRSM, we design a novel matching score algorithm (MSA). This algorithm can not only focus on the matching relationship between query keywords and features, but also comprehensively consider each matching type to return results that are closer to user's real request. The contributions of this paper are summarized as follows.

(1) For the first time, we propose a novel feature score algorithm to create indexes, which allows multiple keywords extracted from a document as a feature to be mapped to one dimension of the index to achieve the purpose of accelerating the encryption process and reducing storage cost.
(2) In the generating trapdoor process of FMRSM, a matching score algorithm is designed. With the help of FSA, this algorithm can rank search results according to the matching types and the number of matching keywords, and therefore the returned results have a higher ranking accuracy.

The rest of this paper is arranged as follows. In section II, we introduce related work of encrypted search. Section III describes the system model, the threat model, and the design goals. Also, in this section, we describe some related terms used in this paper in detail. Then we present the proposed feature matching ranked search mechanism in section IV. In section V, we analyze the performance of our mechanism. Finally, we conclude this paper in section VI.

II. RELATED WORK

Heinlein [4] first defined the concept of encrypted search. They believe that storing encrypted data in the cloud can not only save local storage cost but also reduce security risks. However, they are also worry about the leakage of private information in uncontrolled remote storage environments at the same time. For this reason, they first proposed a full-text search scheme for encrypted data and verified its security. Goh *et al*. [19] first used a bloom filter as an index structure and each index is connected to a document. The hash function is used to map keywords to codes which are stored in corresponding positions of the bloom filter. While querying, people can determine whether each index contains query keywords by mapping again. Although this method does not result in missing search results, it may return irrelevant search information. Boneh *et al*. [20] described an untrusted routing problem and created a public-key encryption scheme that allows PIR (private information retrieval) on encrypted documents. For the first time, this scheme can solve the problem of search without leaking user query information. However, all above searchable encryption schemes are based on single keyword search or specific application situations. Thus, they can't meet users' actual search requirements.

Unlike single keyword search, conjunctive keywords search or multi-keyword search methods can allow users to input multiple keywords in a request to query and return results containing these keywords. Cui *et al*. [21] proposed a public key searchable encryption scheme that can perform multiple keywords search in an expression search formula. Ali and Lu [22] proposed a symmetric encryption scheme that supports connecting keywords search, and this scheme does not need to specify the location of the keyword. In addition, pseudo-random functions and bloom filters are used in the scheme to ensure index security. Yang and Ma [23] proposed a new conjunctive keywords search method for electronic health record systems. This method supports automatic delegating revocation, which is the first searchable encryption scheme that enables proxy re-encryption. Kerschbaum [24] addressed the issue of associating a keyword with a position in existing schemes, and propose a search method that does not need to specify the location of the connecting keywords. Fu *et al*. [9] first proposed a multi-keyword ranked search scheme based on synonyms for encrypted cloud data. When authorized users enter synonyms of predefined keywords, they can also perform search. Fu *et al*. later proposed central keyword-based semantic extension ranked scheme [25]. By extending the central query keyword, a good trade-off between search functionality and efficiency was made.

In recent years, some scholars have proposed phrase search schemes for encrypted data. Unlike supporting multi-keyword search or conjunctive keywords search method, phrase search requires comprehensive consideration of each keyword information, including semantics, order, and location. Poon and Miri [26] designed an encrypted search scheme that supports both conjunctive keyword search and phrase search. The scheme fully considered the statistical properties of natural language to reduce storage cost. Their next work [27] proposed a phrase search technique based on bloom filter, which for the first time supports phrase-independent query. Tang *et al*. [18] defined a symmetric searchable encryption model, and then proposed a symmetric encryption phrase search scheme that can perform secure and efficient phrase search over encrypted data. Zittrower and Zuo [28] proposed a scheme that allows phrase search and multi-keyword search for encrypted datasets. By storing the keyword location information, a comprehensive encrypted search function is achieved. Yin *et al.* [29]–[31] further consider the secure search problem based on a practical application scenario that a data owner needs to grant different keyword query permissions for different data users to achieve flexible access control on outsourced encrypted data in the cloud computing environment, several secure search schemes have been proposed to meet the challenge of secure search over encrypted cloud data. Due to the amount of encrypted files stored in cloud

is becoming very huge, which will hinder efficient query processing, Li *et al.* [32]–[35] presented a few schemes with outsourcing key-issuing and outsourcing decryption, which can implement keyword search function. Guo *et al.* [36]–[38] did valuable research in the field of data privacy in cloud computing and proposed several schemes over encrypted cloud data, which also supports dynamic update operations, such as adding or deleting files.

Although the above related work has improved the searchable encryption technology from different aspects, the following problems still need to be solved: (1) The amount of encryption calculation increases with the size of dictionary, and when a large number of keywords are extracted, it will inevitably lead to huge computational cost; (2) The search results of existing schemes depend on similarity scores or independent keyword matching relationships. They do not comprehensively consider the number of matching keywords and the overall matching relationship, which will result in low query accuracy. Therefore, it is imminent to design a new mechanism to solve these problems.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

The encrypted search system is shown in Figure 1. This system consists of three parts, that is, data owner, data user, and cloud server. The data owner first regards all kinds of data as documents, then extracts keywords from each document to create an encrypted index, and finally uploads the encrypted documents and indexes to the cloud server. While querying, the data user first generates a trapdoor through search control operation, and then submits it to the cloud server for retrieval. After receiving the trapdoor, the cloud server calculates the similarity score of the trapdoor and each document index, and then returns copies of encrypted documents to the data user according the score. In the end, the data user receives the encrypted results and decrypts them through access control operation.

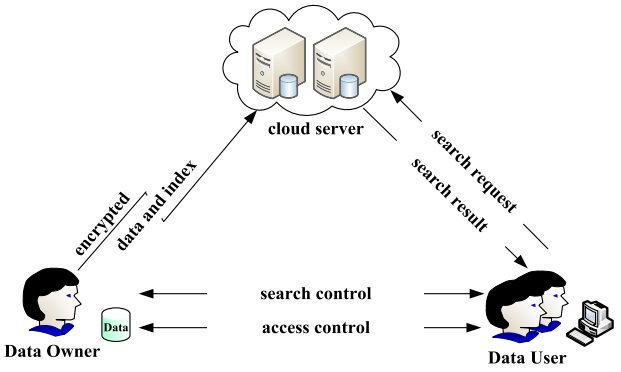

FIGURE 1. System model.

B. SYMBOL DESCRIPTION

The symbol descriptions are shown as below.

TABLE 1. Notations.

Symbol	Description
F	The plain document set, denoted as $F=\{F_1,F_2,...,F_m\}$
C	The encrypted document set, denoted as $C=\{C_1,C_2...C_m\}$
g_j	The j-th document feature formed by extracting keywords from F_j
G	The feature set consisting of document features, denoted as $G=\{g_1,g_2,...,g_m\}$
D	The feature dictionary formed by filtering repeating features in D
d_j	The j-th feature in D
n_j	The total number of keywords extracted from the j-th document
P	The plain index set, denoted as $P=\{p_1,p_2,...,p_m\}$
I	The encrypted index set, denoted as $I=\{I_1,I_2,...,I_m\}$
$\vec{p}_i$	The index vector of the i-th document
e	The query keyword set
$\vec{q}$	The query vector
T	The trapdoor generated by encrypting query vector

C. THREAT MODEL

In this paper, we believe that the cloud server is "honest but curious" [39], which means that the cloud server will honestly perform search operations according to the submitted trapdoor. However, at the same time, the cloud server may also infer the plain data from existing information. The existing information can be encrypted documents, encrypted indexes and the trapdoor which must be submitted to the cloud server, or some background knowledge about encryption. We believe that the general goal of the server's attack is to infer secret keys or plain data. To better evaluate the security of our mechanism, we consider the following two threat levels [40].

Level 1: The cloud server knows the encrypted information, that is, the cloud server can know the encrypted document set C, the encrypted index set I, and the query trapdoor T, which must be uploaded to the cloud server.

Level 2: The cloud server knows more information than that in Level1, for example, the cloud server may judge whether some query keywords are in a trapdoor by combining existing trapdoors and query results, or it may use encrypted background information to infer secret keys.

D. DESIGN GOAL

In order to reduce storage cost and improve search accuracy, our mechanism tries to achieve the following goals.

1) FEATURE INDEX

Our goal is to map multiple keywords extracted from a document to one dimension of the index by using a feature score algorithm. That is, each dimension of the index corresponds to a feature.

2) MATCHING SEARCH AND PRECISION

We calculate matching scores based on the matching relationship between query keyword set and feature dictionaries to generate a query vector. While querying, the results are ranked according to the inner product of the index and the query vector and returned to the authorized user.

E. PRELIMINARIES

Four important techniques should be explained in our design, which are as follows.

(1) *Normalized term frequency and anti-term frequency*: The query score reflects the matching relationship between query keywords and features. Term frequency and anti-term frequency (TF-IDF) [40] are often used to calculate a document score. Term frequency reflects the importance of a keyword for a document. In general, we use the occurrence of the term appearing in the document represents term frequency. In this paper we use the normalized term frequency [41] to calculate the composite score, and the normalized term frequency of each feature keyword can be calculated as:

$$TF(v_{i,b}) = \frac{1 + \ln f_{i,b}}{\sqrt{\sum_{w_{i,k} \in g_i} (1 + \ln f_{i,k})^2}} \tag{1}$$

where $g_i$is the feature of the i-th document, $w_{i,k}$ represents the k-th keyword in g_i, $f_{i,k}$ and $f_{i,b}$ represent the number of occurrences of the keywords $w_{i,k}$ and $v_{i,b}$ in the i-th document respectively, where $v_{i,b} \in g_i \cap d_j$, and d_j indicates the j-th feature in D. Anti-term frequency indicates the degree of discrimination of a keyword for a document, and we calculate anti-term frequency by dividing the total number of features by the number of features containing the term. In this paper, we use the normalized anti-term frequency [41] to calculate the matching score, and the normalized anti-term frequency of each query keyword can be calculated as:

$$IDF(v_{j,b}) = \frac{\ln(1 + n/f_b)}{\sqrt{\sum_{w_{j,k} \in d_j} \ln(1 + n/f_k)^2}} \tag{2}$$

where n is the total number of dictionary, $w_{j,k}$ represents the k-th keyword in d_j, f_k and f_b represent the number of feature containing the keywords $w_{j,k}$ and $v_{j,b}$ respectively, where $v_{j,b} \in d_j \cap e$, e represents the query keyword set.

(2) "*Top-k*" *query*: When a user enters keywords to search, it is often not necessary to return all the query results. In this paper, we used the "*top-k*" method to query [40]. When the user submits a query request, they also need to submit parameter k, which indicates the number of results to be returned. During the search process, the cloud server ranks search results and returns the "*top-k*" document to the query user.

(3) *Exact match and partial match*: According to the matching relationship between document feature $g_i (i = 1, 2, \ldots, m)$ or query keyword set e and feature $d_j (j = 1, 2, \ldots, n)$, we divide the matching relationship into two different types, which are described as follows.
Exact match: It indicates that the keywords in g_i or e are the same as the keywords in d_j.
Partial match: It means that the keywords in g_i or e are not the same as the keywords in d_j.

(4) *Ranking accuracy*: In this paper, we define ranking accuracy δ to measure the accuracy of returned results. In partial match, we assume that the feature g_i of document F_i matches n keywords with the query keyword set e. If F_i is returned before all the documents whose number of matching keywords is less than n, we say that the document F_i is returned in order. And if the exact matching document is returned before all partial matching documents, we specify that it is returned in order. In this paper, we use $\delta = \mathrm{r}/k$ to calculate the ranking accuracy, where r indicates the number of documents returned in order when performing "*top-k*" query, and k indicates the total number of returned results.

IV. FMRSM

In this section, the specific implementations of our proposed FMRSM are as follows.

A. GENERATION OF SECRET KEYS

The data owner first generates encrypting keys M_1 and M_2, and an indicator S. Here, M_1 and M_2 are $(n + u + 1) \times (n + u + 1)$ invertible matrixes, and $S \in \{0, 1\}^{n+u+1}$, where n is the total number of features in D, and $u+1$ is the extended dimension. Hence, the secret key can be expressed as a 3-tuple $\{S, M_1, M_2\}$.

B. CREATION OF INDEX

Step1: The data owner first creates an n-dimension index vector $\vec{p_i}$. Each dimension of the index vector $\vec{p_i}$ is a feature score of a feature $d_j (j = 1, 2, \ldots, n)$ in the corresponding document F_i. The feature score algorithm is shown in Algorithm1.

Step2: We first extend $\vec{p_i}$ from n dimensions to $n + u$ dimensions and set each of them to a random number ε, which follows the same uniform distribution $U(\mu - c, \mu + c)$. Then, we extend $\vec{p_i}$ to $n + u + 1$ dimensions and set it to 1. The extended index vector can be expressed as $\vec{p_{i\sim}}$.

Step3: According to the value of the indicator S, we randomly split the extended index $\vec{p_{i\sim}}$ into $\vec{p'_i}$ and $\vec{p''_i}$. Namely, if $S[j]$ is 0, $\vec{p'_i}[j]$ and $\vec{p''_i}[j]$ can be set as two different random values while the sum of them should be equal to $\vec{p_{i\sim}}[j]$; if $S[j]$ is 1, $\vec{p'_i}[j]$ and $\vec{p''_i}[j]$ are set as the same value as $\vec{p_{i\sim}}[j]$.

Step4: The split index vectors $\vec{p'_i}$ and $\vec{p''_i}$ are encrypted using the secret keys M_1 and M_2 to generate an encrypted index I_i, where $I_i = \{\vec{p'^T_i} M_1, \vec{p''^T_i} M_2\}$.

Algorithm 1 FeatureScore(g_i, d_j)

Require:
g_i : The feature of document F_i.
d_j : The *j-th* feature in D.
Ensure:
$v_{i,b}$: The *b-th* keyword in $g_i \cap d_j$.
FS : The feature score of g_i.
IS : The highest value of feature score.
$TF(v_{i,b})$: The normalized term frequency calculated by formula (1).
α : Initial distinguish parameter.
q_α : Common ratio of geometric progression $\{\alpha_b\}$.
if g_i exactly match d_j **then**
  $FS = IS$;
else if g_i partly match d_j **then**
  $FS = 0$;
  for every keyword $v_{i,b}$ in $g_i \cap d_j$**do**
    if b is equal to 1 **then**
      $FS = TF(v_{i,b})$;
      $\alpha_1 = \alpha$;
    else
      $FS = FS + (TF(v_{i,b}) + \alpha_{b-1})$;
      $\alpha_b = \alpha_{b-1} q_\alpha$;
    end if
  end for
end if
return FS;

Algorithm 2 MatchingScore(e, d_j)

Require:
e : The query keywords set.
d_j : The *j-th* feature in D.
Ensure:
$v_{j,b}$: The *b-th* keyword in $d_j \cap e$.
MS : The matching score of query keywords.
QS : The highest value of matching score.
$IDF(v_{j,b})$: The normalized anti-term frequency calculated by formula (2).
β : Initial distinguish parameter.
q_β : Common ratio of geometric progression $\{\beta_b\}$.
if e exactly match d_j **then**
  $MS = QS$;
else if e partly match d_j **then**
  $MS = 0$;
  for every keyword $v_{j,b}$ in $d_j \cap e$**do**
    if b is equal to 1 **then**
      $MS = IDF(v_{j,b})$;
      $\beta_1 = \beta$;
    else
      $MS = MS + (IDF(v_{j,b}) + \beta_{b-1})$;
      $\beta_b = \beta_{b-1} q_\beta$;
    end if
  end for
end if
return MS;

C. CREATION OF TRAPDOOR

Step1: When authorized users enter keywords to query, we first create an n-dimension query request $\vec{q}$. Each dimension of $\vec{q}$ is the matching score of a query keyword set e and a feature $d_j (j = 1, 2, .., n)$. The method of calculating the matching score is shown in Algorithm2.

Step2: We first extend $\vec{q}$ from n dimensions to $n + u$ dimensions and randomly select v positions from them to be set as 1, and the rest positions are set as 0. Then the value of the $n + u + 1$ dimensions is set as a random value $t (t \neq 0)$ and all the other positions are multiplied by another random value $r (r \neq 0)$.The extended query request can be expressed as $\vec{q_\sim}$.

Step3: According to the value of the indicator S, we randomly split the extended query request $\vec{q_\sim}$ into $\vec{q'}$ and $\vec{q''}$. Namely, if $S[j]$ is 1, $\vec{q'}[j]$ and $\vec{q''}[j]$ can be set as two different random values while the sum of them should be equal to $\vec{q_\sim}[j]$; if $S[j]$ is 0, $\vec{q'}[j]$ and $\vec{q''}[j]$ are set as the same value as $\vec{q_\sim}[j]$.

Step4: The split query requests $\vec{q'}$ and $\vec{q''}$ are encrypted to generate a trapdoor T, where $T = \{M_1^{-1}\vec{q'}, M_2^{-1}\vec{q''}\}$.

D. SEARCH

The authorized user enters query keywords to generate a trapdoor and submits it to the cloud server. The cloud server calculates the score of the trapdoor and each document index. Then, the search results are ranked according to the value of the score, and the "*top-k*" documents with the highest scores are returned to the authorized user. The search process are as follows.

$$\begin{aligned} I_i \cdot T &= \{\vec{p_i'^T} M_1, \vec{p_i''^T} M_2\} \cdot \{M_1^{-1}\vec{q'}, M_2^{-1}\vec{q''}\} \\ &= \vec{p_i'^T}\vec{q'} + \vec{p_i''^T}\vec{q''} \\ &= r(\vec{p_i}\vec{q} + \sum \varepsilon^{(v)}) + t \end{aligned}$$

After receiving the encrypted documents, the authorized user decrypts them into plain documents through access control operation.

V. PERFORMANCE ANALYSIS

A. THEORETICAL ANALYSIS

1) INDEX AND KEY SECURITY

Under Level 2 model, the cloud server is able to establish $2(n+u+1)m$ equations, the number of which is less than the sum of unknown variables in the index and the key, which are $2(n + u + 1)m$ and $(n + u + 1)^2$ respectively. Thus, our mechanism is resilient against the key attack, and the index information and the query confidentiality can be well protected.

2) KEYWORDS PRIVACY

The information leakage in level 2 model is mainly incurred by the random number ε in each index vector [40]. To meet

the preset security, the number of $\sum \varepsilon^{(v)}$ should be larger than 2^{ω} [25] when giving system parameter ω. Besides, the number of different $\sum \varepsilon^{(v)}$ is C_u^v which is maximized when $u/v = 2$ and $C_u^v \geq (u/v)^v$. Therefore, when $u = 2\omega$ and $v = \omega$, our scheme can achieve the security goal. In addition, each $\varepsilon^{(j)}$ follows the same uniform distribution $U(\mu' - c, \mu' + c)$, and their mean is μ' and variance is $\sigma' = c^2/3$. According to the central limit theorem, the $\sum \varepsilon^{(v)}$ follows the normal distribution $N(\mu, \sigma^2)$, where their mean μ is $\omega\mu'$ and variance σ^2 is $\omega c^2/3$. In the normal distribution $N(\mu, \sigma^2)$, the larger the σ value, the more difficult it is for the cloud server to analyze the similarity information between initial index vectors, but the search accuracy will be reduced. Therefore, it is necessary to choose the value of σ reasonably between the privacy protection and the search accuracy.

3) EFFICIENCY

FMRSM uses feature score algorithm to create indexes to reduce dimensions. The specific analysis are as follows. When we create indexes with independent keywords, the number of dimensions of each index is related to the total number of terms and is greater than the total number of documents. When a large number of keywords are extracted from a document, it will inevitably result in huge storage cost and computational burden. Our feature score algorithm allows a plurality of keywords which are extracted from a document as a feature to be mapped to one dimension of the index. Thus, the index dimension n is not greater than the total number of documents m. Therefore, comparing with creating indexes with independent keywords, this algorithm can reduce storage cost and accelerate the encryption and search process.

In existing encrypted ranked search schemes, multiple keywords in the dictionary are independently mapped to different dimensions of the index. When we use ''TF-IDF'' rule to calculate a document score, it is very likely that an ''error order'' will occur. For example, when we extract keywords from documents F_i and F_j to form features $g_i = \{$''cloud computing'', ''ranked search''$\}$ and $g_j = \{$''cloud computing''$\}$, it may happen that the sum of the weights of ''cloud computing'' and ''ranked search'' in F_i is less than the weight of ''cloud computing'' in F_j. This situation is very common, because ''cloud computing'' and ''ranked search'' appear less in F_i. However, ''cloud computing'' appears more in F_j. If users enter ''cloud computing'' and ''ranked search'' to query, F_j will be returned preferentially according to its score, which obviously does not meet the user's real query requirement. Algorithm1 and Algorithm 2 can improve the accuracy of query results. They comprehensively consider the matching relationship between features and query terms, and then give each query keyword different score according to the matching types and the number of matching keywords.

In the process of creating indexes, Algorithm1 comprehensively calculates the value of each dimension in the index, and the processes are as follows. When a feature d_j is exactly the same as the document feature g_i formed by extracting keywords from document F_i, we specify that g_i is exactly match with d_j. Exact match indicates that the feature d_j can fully reflect the real content of F_i. Therefore, the j-th dimension of index p_i should have the highest score, and in Algorithm1, we set the highest feature score to IS. When g_i is different from d_j, we think that they are partially matched. Partial match includes two cases, where g_i and d_j have no intersections or they have common elements. The first case indicates that the feature d_j can not reflect the information of F_i, Thus, the j dimensional feature score of the index p_i will be 0. The second case indicates that the feature d_j can reflect some of the information of F_i, which depends on the number of matching keywords in g_i and d_j. We use the sum of the normalized term frequency of these matching keywords in F_i to represent their feature score. However, using this method directly will result in the ''error order''. In order to improve ranking accuracy, distinguishing parameters are introduced while calculating the normalized term frequency of each matching keyword in Algorithm1. That is, we use $TF(v_{i,b}) + \alpha_b$ to represent the score of the b-th matching keyword $v_{i,b}$, where $\alpha_0 = 0$, $\{\alpha_b\}(b > 1)$ is a geometric progression whose first item is α and public ratio is q_α. For example, when the feature $d_j = \{$''cloud computing'', ''multi-keyword'', ''ranked search''$\}$ and the feature $g_i = \{$''cloud computing'', ''ranked search''$\}$, we use $\sum_{v_{i,b} \in \widehat{W}} (TF(v_{i,b}) + \alpha_b)$ to represent the value of the j-th dimension in the index p_i, where $\widehat{W} = \{$''cloud computing'', ''ranked search''$\}$. The introduction of distinguishing parameters will make the feature score of $(n+1)$-th matching keywords much larger than that of n-th matching keywords. Therefore, in partial match, the more matching keywords there are in g_i and d_j, the higher the feature score will be.

In generating trapdoor process, Algorithm2 gives different matching scores according to different matching relationships between query keyword set e and feature d_j, which are respectively described as follows. When e is the same as d_j, we specify that they are exact match. Exact match indicates that the j-th dimension of query request q is the user's real query requirements, so it should have the highest matching score, which is set to QS in Algorithm2. If e and d_j are not the same, we define them as partial match. In this case, we use the sum of the normalized anti-term frequency of their matching keywords to represent the matching score. In order to avoid the ''error order'', different parameters are introduced while calculating normalized anti-term frequency. That is, we use $IDF(v_{j,b}) + \beta_b$ to represent the score of the b-th matching keyword $v_{j,b}$, where $\beta_0 = 0$, and $\{\beta_b\}(b > 1)$ is a geometric progression whose first item is β and public ratio is q_β. For example, when the feature $d_j = \{$''cloud computing'', ''multi-keyword '', '' ranked search''$\}$ and the query keyword set $e = \{$''cloud computing'', ''feature matching'', ''ranked search''$\}$, we use $\sum_{v_{j,b} \in \widehat{W}} (\mathrm{IDF}(v_{j,b}) + \beta_b)$ to represent the value of the j-th dimension in query request q, where

$\widehat{W}$ = {"cloud computing", "ranked search"}. In partial match, with the help of FSA, this matching score algorithm will give priority to returning documents with more matching keywords.

In both algorithms, the values of IS and QS should be larger than the highest score of partial match to ensure the ranking accuracy. Generally, the larger the values of IS, QS, q_α and q_β are, the more accurate the query results. However, their values should not be too large, because in this case, there is a high risk of privacy leakage under Level2 model. In general, when $\alpha \in (TF, 1)$, $\beta \in (IF, 1)$, $q_\alpha, q_\beta \in (1.2, 1.5)$, $IS \in (\alpha \cdot (q_\alpha)^M, \alpha \cdot (q_\alpha)^M + 2TF)$ and $QS \in (\beta \cdot (q_\beta)^M, \beta \cdot (q_\beta)^M + 2IF)$, the query results will achieve a good trade-off between ranking accuracy and privacy protection. Where TF is the maximum normalized term frequency in $g_i (i = 1, 2, \ldots, m)$, IF is the maximum normalized anti-term frequency in $d_j (j = 1, 2, \ldots, n)$, M is the maximum number of keywords in feature $g_i (i = 1, 2, \ldots, \text{m})$ or $d_j (j = 1, 2, \ldots, \text{n})$.

B. EXPERIMENTAL ANALYSIS

In this part, we evaluate the performance of our FMRSM system using Java language on Windows 7 server with Intel Core3 CPU 2.5GHz. We use Request for Comments database (RFC) [42] as our dataset, and compare our mechanism with MRSE (multi-keyword ranked search over encrypted cloud data) [40] and CKSER (central keyword semantic extension ranked scheme) [25].

The experimental process is as follows. In FMRSM, we extract the keywords from each document to form the document features, and filter duplicate features to construct the feature dictionary. Then for MRSE and CKSER, we extract the keywords from each document to form the keyword dictionary. Finally, we perform five group of experiments, including creating index, generating trapdoor, search, and testing storage cost and accuracy. The experimental results are shown in Figures 2, 3, 4, 5 and Table 2.

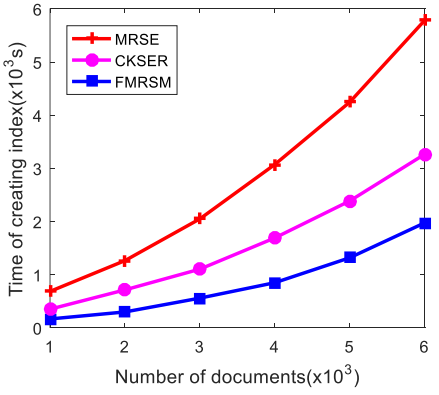

FIGURE 2. Time of creating index. For different number of documents with system parameter $\omega = 2000$.

The time of creating index, generating trapdoor, and search, and the size of the index and trapdoor are all related to the total number of features in FMRSM and the total number of keywords in MRSE and CKSER. In MRSE and CKSER scheme, each dimension of the index or query request corresponds to a keyword in the keyword dictionary. Thus, when the total number of keywords is N, the dimensions of the index and query request will be $(N + u + 1)$. In FMRSM, all the keywords extracted from a document are mapped to one dimension of the index by feature score algorithm.

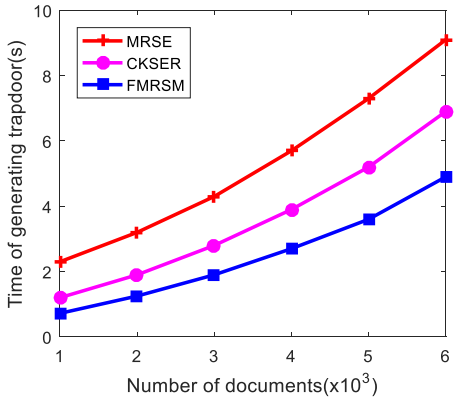

FIGURE 3. Time of generating trapdoor. For different number of documents with the same query keywords $t = 10$.

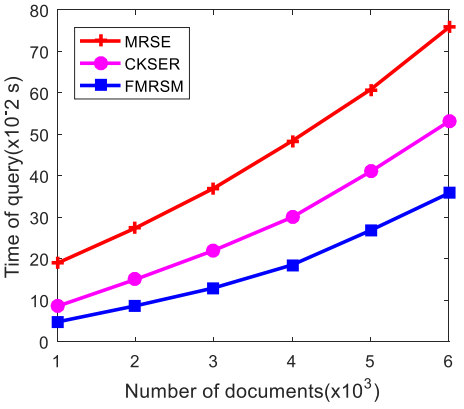

FIGURE 4. Time of query. For different number of documents with the same query keywords $t = 10$.

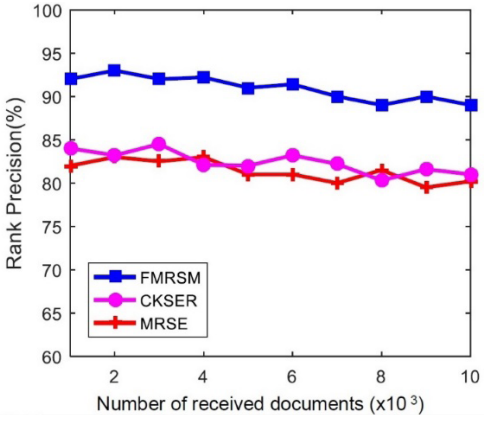

FIGURE 5. Ranking accuracy. For different number of retrieved documents with standard deviation $\sigma = 0.3$.

TABLE 2. Storage of subindex/trapdoor.

Number of documents	2000	3000	4000	5000	6000
MRSE(KB)	32.4	48.1	66.4	80.5	94.2
CKSER(KB)	32.1	47.8	65.7	79.3	92.8
FMRSM(KB)	14.9	22.5	30.8	38.3	45.6

When there are m documents, the largest dimension of the index and query request will be $(m+u+1)$. Since the number of keywords N extracted from the document is much greater than the number of documents m, the time of creating index, generating trapdoor, and search for FMRSM is less than that of MRSE and CKSER, as shown in figure 2, 3, and 4. It should be noted that since CKSER uses sub-matrices technology, the time of creating index, generating trapdoor, and search will be less than MRSE. At the same time, comparing with MRSE and CKSER, FMRSM saves the storage cost of the index and trapdoor, as shown in Table 2.

Another highlight of FMRSM is that it can improve the ranking accuracy. In the experiment, we calculate the ranking accuracy according to the definition mentioned in E. Preliminaries, Section III. Figure 5 shows the relationship between the ranking accuracy and the number of documents. It can be seen from the figure that FMRSM has better ranking accuracy than MRSE and CKSER. Therefore, for users' query requests, FMRSM can return the search results that best match user's query requirements.

VI. CONCLUSION

In this paper, we propose a novel feature matching ranked search mechanism for encrypted cloud data. In this mechanism, a feature score algorithm is used to create indexes so that a plurality of keywords extracted from a document are only mapped to one dimension of the index. Comparing with creating indexes with independent keywords, this mechanism can reduce the index dimension. In addition, a matching score algorithm is designed in the generating trapdoor process of FMRSM. This algorithm can give the query request an accurate score based on the type of match and the number of matching keywords, so that the query results are more in line with users' actual search requests. It can be seen from the experiment results that our mechanism can speed up the creation of index, the generation of trapdoor, and the search process. Moreover, our mechanism can save storage cost and improve the ranking accuracy.

REFERENCES

[1] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Ottawa, ON, Canada, Jun. 2012, pp. 917–922.

[2] X. Zhang, C. Xu, R. Xie, and C. Jin, "Designated cloud server public key encryption with keyword search from lattice in the standard model," *Chin. J. Electron.*, vol. 27, no. 2, pp. 304–309, Mar. 2018.

[3] Y. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, vol. 5. Berlin, Germany: Springer, 2005, pp. 442–455.

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

[5] A. J. Aviv, M. E. Locasto, S. Potter, and A. D. Keromytis, "SSARES: Secure searchable automated remote email storage," in *Proc. 23rd Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Miami Beach, FL, USA, Dec. 2007, pp. 129–139.

[6] M. Raykova, B. Vo, S. M. Bellovin, and T. Malkin, "Secure anonymous database search," in *Proc. ACM Workshop Cloud Comput. Secur. (CCSW)*, New York, NY, USA, 2009, pp. 115–126.

[7] S. M. Bellovin and W. R. Cheswick, "Privacy-enhanced searches using encrypted Bloom filters," IACR Cryptol. ePrint Arch., Tech. Rep., 2004, vol. 22.

[8] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[9] Z. Fu, X. Sun, Z. Xia, L. Zhou, and J. Shu, "Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing," in *Proc. IEEE 32nd Int. Perform. Comput. Commun. Conf. (IPCCC)*, San Diego, CA, USA, Dec. 2013, pp. 1–8.

[10] H. Yin, Z. Qin, J. Zhang, W. Li, L. Ou, Y. Hu, and K. Li, "Secure conjunctive multi-keyword search for multiple data owners in cloud computing," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Wuhan, China, Dec. 2016, pp. 276–286.

[11] 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, Dec. 2016.

[12] Z. Guo, H. Zhang, C. Sun, Q. Wen, and W. Li, "Secure multi-keyword ranked search over encrypted cloud data for multiple data owners," *J. Syst. Softw.*, vol. 137, pp. 380–395, Mar. 2018.

[13] X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Inf. Sci.*, vols. 403–404, pp. 22–41, Sep. 2017.

[14] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–5.

[15] M. A. M. Ahsan, F. Z. Chowdhury, M. Sabilah, A. W. B. A. Wahab, and M. Y. I. B. Idris, "An efficient fuzzy keyword matching technique for searching through encrypted cloud data," in *Proc. Int. Conf. Res. Innov. Inf. Syst. (ICRIIS)*, Langkawi, Malaysia, Jul. 2017, pp. 1–5.

[16] X. Zhu, Q. Liu, and G. Wang, "A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, Aug. 2016, pp. 845–851.

[17] H. Zhu, Z. Mei, B. Wu, H. Li, and Z. Cui, "Fuzzy keyword search and access control over ciphertexts in cloud computing," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, 2017, pp. 248–265.

[18] Y. Tang, D. Gu, N. Ding, and H. Lu, "Phrase search over encrypted data with symmetric encryption scheme," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst. Workshops*, Macau, China, 2012, pp. 471–480.

[19] E.-J. Goh, "es," IACR Cryptol. ePrint Arch., Tech. Rep., 2003, p. 216.

[20] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. Skeith, III, "Public key encryption that allows PIR queries," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2007, pp. 50–67.

[21] H. Cui, Z. Wan, R. Deng, G. Wang, and Y. Li, "Efficient and expressive keyword search over encrypted data in cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 409–422, May/Jun.2016.

[22] F. S. Ali and S. Lu, "Searchable encryption with conjunctive field free keyword search scheme," in *Proc. Int. Conf. Netw. Inf. Syst. Comput. (ICNISC)*, Wuhan, China, Apr. 2016, pp. 260–264.

[23] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 746–759, Apr. 2017.

[24] F. Kerschbaum, "Secure conjunctive keyword searches for unstructured text," in *Proc. 5th Int. Conf. Netw. Syst. Secur.*, Milan, Italy, Sep. 2011, pp. 285–289.

[25] Z. Fu, X. Wu, Q. Wang, and K. Ren, "Enabling central keyword-based semantic extension search over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 2986–2997, Dec. 2017.

[26] H. T. Poon and A. Miri, "An efficient conjunctive keyword and phase search scheme for encrypted cloud storage systems," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, New York, NY, USA, Jun. 2015, pp. 508–515.

[27] H. T. Poon and A. Miri, "Fast phrase search for encrypted cloud storage," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1002–1012, Oct. 2019.

[28] S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Anaheim, CA, USA, Dec. 2012, pp. 764–770.
[29] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, "CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme," *IEEE Access*, vol. 7, pp. 5682–5694, 2019.
[30] H. Yin, Z. Qin, J. Zhang, H. Deng, F. Li, and K. Li, "A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing," *J. Parallel Distrib. Comput.*, vol. 135, pp. 56–69, Jan. 2020.
[31] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, and K. Li, "Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners," *Future Gener. Comput. Syst.*, vol. 100, pp. 689–700, Nov. 2019.
[32] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *Int. J. Commun. Syst.*, vol. 30, no. 1, p. e2942, Jan. 2017.
[33] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep. 2017.
[34] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Services Comput.*, early access, Apr. 11, 2019, doi: 10.1109/TSC.2019.2910113.
[35] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multirecipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2553–2562, Apr. 2020.
[36] C. Guo, X. Chen, Y. Jie, F. Zhangjie, M. Li, and B. Feng, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Trans. Services Comput.*, early access, Oct. 30, 2017, doi: 10.1109/TSC.2017.2768045.
[37] C. Guo, S. Su, K.-K.-R. Choo, and X. Tang, "A fast nearest neighbor search scheme over outsourced encrypted medical images," *IEEE Trans. Ind. Informat.*, early access, Nov. 27, 2018, doi: 10.1109/TII.2018.2883680.
[38] C. Guo, J. Jia, Y. Jie, C. Z. Liu, and K.-K.-R. Choo, "Enabling secure cross-modal retrieval over encrypted heterogeneous IoT databases with collective matrix factorization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3104–3113, Apr. 2020.
[39] Z. A. Kissel and J. Wang, "Verifiable phrase search over encrypted data secure against a semi-honest-but-curious adversary," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. Workshops*, Philadelphia, PA, USA, Jul. 2013, pp. 126–131.
[40] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[41] 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, Feb. 2016.
[42] *Request for Comments Database*. Accessed: Nov. 25, 2019. [Online]. Available: http://www.ietf.org/rfc.html

LIANGGUI LIU received the Ph.D. degree in communications and information system from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2007. He was a Visiting Professor with Cornell University, Ithaca, NY, USA, and the University of Houston, Houston, TX, USA. He is currently a Professor with the College of Information Science and Technology, Zhejiang Shuren University, Hangzhou, China. His current research interests include social computing, network security, and natural computation.

QIUXIA CHEN received the Ph.D. degree from the Department of Automation, Zhejiang University of Technology, Hangzhou, China, in 2011. She is currently an Associate Professor with the College of Information Science and Technology, Zhejiang Shuren University, Hangzhou, China. Her current research interests include model predictive control and network security.