IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 28, 2019, accepted January 6, 2020, date of publication January 13, 2020, date of current version April 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2966367

FSSE: An Effective Fuzzy Semantic Searchable Encryption Scheme Over Encrypted Cloud Data

GUOXIU LIU[1,2], GENG YANG[1,3] SHUANGJIE BAI[1,4], QIANG ZHOU[2], AND HUA DAI[1,3]

[1]School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[2]School of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, China
[3]Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing 210003, China
[4]Department of Electronic Engineering and Computer Science, University of Stavanger, 4036 Stavanger, Norway

Corresponding author: Geng Yang (yangg@njupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61972209, Grant 61572263, Grant 61502251, Grant 61502243, Grant 61872197, and Grant 61602263, in part by the Natural Science Foundation of Educational Commission of Anhui Province of China under Grant KJ2018A0432, in part by the Natural Science Foundation of Anhui Province under Grant 1608085MF127, in part by the China Postdoctoral Science Foundation under Grant 2019M651919, in part by the Anhui Provincial Key Laboratory of Network and Information Security under Grant AHNIS2020002, in part by the Science and Technology Major Project of Anhui Province under Grant 18030901017, in part by the Science and Technology Project of Chuzhou under Grant 2018ZN020, in part by the University Natural Science Foundation of Anhui Province under Grant KJ2017A419, in part by the Project of the Natural Science Foundation of Educational Commission of Anhui Province under Grant KJ2018B03, in part by the Natural Science Research for Colleges and Universities of Anhui Province under Grant KJ2019A0647 and Grant KJ2018B05, in part by the Natural Research Foundation of Nanjing University of Posts and Telecommunications under Grant NY217119, and in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX18_0891.

ABSTRACT Currently, searchable encryption has attracted considerable attention in the field of cloud computing. The existing research mainly focuses on keyword-based search schemes, most of which support the exact matching of keywords. However, keyword-based search schemes ignore spelling errors and semantic expansions of keywords. The significant drawback makes the existing techniques unsuitable in cloud computing as it greatly affects system usability and can not completely satisfy the users' search intentions. In this paper, we propose an effective fuzzy semantic searchable encryption scheme (FSSE) that supports multi-keyword search over encrypted data in cloud computing. In our scheme, we exploit a keyword fingerprint generation algorithm to generate a fingerprint set of the keyword dictionary and a fingerprint of the query keywords, and employ Hamming distance to quantify keywords similarity. Based on the proposed fingerprint generation algorithm and Hamming distance, we realize fuzzy search. Furthermore, we utilize the semantic expansion technique to expand query keywords and calculate the semantic similarity between the query keywords and the expanded word of the query keywords to achieve the semantic search. To improve the search efficiency, we construct an inverted index structure and use the vector intersection matching as well as short-circuit matching operations to effectively filter irrelevant documents. The theoretical analysis and experimental results demonstrate that our proposed scheme satisfies the security guarantee of searchable encryption, enhances system usability, and is more efficient in comparison with the state of the art schemes.

INDEX TERMS Searchable encryption, cloud computing, fuzzy semantic search, multi-keyword search.

I. INTRODUCTION

With the development of cloud computing, cloud storage has brought convenience to users, and many more users are increasingly willing to store their data on the cloud server. To ensure privacy, users usually encrypt the data before outsourcing them to the cloud server. However, the encrypted data lose their original characteristics. For example, directly searching over the encrypted data cannot be performed, which introduces great challenges to effective data utilization.

In order to address the above issue, some researchers have proposed fruitful searchable encryption schemes. For example, Song *et al.* [1] proposed a search scheme over the encrypted data using symmetric encryption to encrypt keywords. The scheme only supports a single keyword search. Recently, Cao *et al.* [10] created a multi-keyword ranking search scheme (MRSE) over the encrypted cloud data, which improves the accuracy of the search and reduces the communication overhead. Based on the MRSE, some

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

multi-keyword search schemes [11]–[13] were proposed. However, these schemes only consider the exact matching of keywords, and in the case of incorrect keyword input, without returning documents. Additionally, these schemes ignore keyword input and spelling errors, and several fuzzy search schemes [16]–[20] have been proposed. However, because the fuzzy set is constructed for each keyword, it will take up more storage space. These schemes [21]–[23] introduce a Bloom filter to reduce the storage space, but increase the computational overhead. Moreover, these fuzzy search schemes do not consider the semantic relationship of keywords. In practical applications, a keyword may have multiple synonyms, and the user may not be able to input all the synonyms of the keyword when performing a document search. Under this circumstance, the searched result may be inaccurate. Therefore, some semantic search schemes [25], [26] have been proposed in which the keywords are semantically expanded, and the semantic search of encrypted data is realized. However, in the existing semantic search schemes, the query keywords input by the user need to be precisely matched with the keywords in the predefined keyword set to semantically expand the query keywords, and they do not support the fuzzy search of keywords.

The objective of the paper is to design a fuzzy semantic searchable encryption scheme (FSSE) to support multi-keyword fuzzy semantic search. In the FSSE scheme, we take advantage of the keyword fingerprint generation algorithm to generate the fingerprint set of the keyword dictionary, that is, a ciphertext fuzzy set, and only one keyword needs to be processed into the corresponding fingerprint during the process. Subsequently, we utilize the Hamming distance to realize the fuzzy search of the keywords based on the ciphertext fuzzy set. Moreover, we carry out the semantic expansion of the query keywords and calculate the semantic similarity between the query keywords and the expanded words to achieve the semantic search of the keywords. Furthermore, we adopt the vector intersection matching and the short-circuit matching operations to improve the search efficiency. To ensure the privacy of the index, the inverted index is encrypted by using the modified fully homomorphic encryption over the integers (FHEI) scheme [15]. To avoid data leakage, the dual cloud server is used, and the search results are sorted according to the comprehensive relevance score on the private cloud server.

Our contributions can be summarized as follows:

- We propose a FSSE scheme, which can support multi-keyword fuzzy semantic search over encrypted data in cloud computing.
- We realize the fuzzy search of the keywords by making use the keyword fingerprint generation algorithm and Hamming distance. Moreover, the semantic expansion technique and semantic similarity are employed to achieve the semantic search of the keywords. Furthermore, we realize multi-keyword fuzzy semantic search by means of both of them in the FSSE scheme.
- To improve the search efficiency, we construct an inverted index structure and apply vector intersection matching as well as short-circuit matching operations to filter irrelevant documents.
- The experimental results performed on a real data set show that the FSSE scheme achieves a safe and efficient multi-keyword fuzzy semantic search.

The remainder of this paper is organized as follows. Section II discusses some related works. In Section III, we describe the system model, the threat model, and the notations. Section IV presents the preliminaries. Section V gives the basic idea of FSSE scheme and its construction. Section VI presents the security analysis. Section VII describes the performance evaluation. Section VIII concludes the paper.

II. RELATED WORKS

A. SINGLE KEYWORD SEARCH

Searchable encryption is a helpful technique that treats encrypted data as documents and allows a user to securely search through a single keyword and retrieve documents of interest [32]. Song *et al.* [1] first proposed a keyword search scheme based on symmetric cryptography, which caused the academic community to pay attention to searchable encryption. In the scheme, double-layer encryption is used to independently encrypt each keyword in the file, and the encrypted document is sequentially scanned during the search. Subsequently, some well-established single keyword searchable encryption schemes [2]–[4] were proposed. For example, Curtmola *et al.* [3] gave a security definition of SE formalization, and proposed an index-based single keyword searchable encryption scheme. The ranking search problem was proposed in [4], [42], it created an effective searchable symmetric encryption scheme and returned the ranked results in [4]. A model-based ranking function was proposed in [42], which improved the accuracy of the ranking. However, the differential attack scheme proposed in [5] can successfully attack the one-to-many sequence-preserving encryption scheme in [4]. Tahir *et al.* [6] proposed a ranking searchable encryption scheme based on index and probability trapdoors to improve security. Additionally, some searchable encryption schemes [38]–[40], [43] were proposed, they have improved efficiency and enhanced security. However, these schemes only support single keyword search.

B. MULTI-KEYWORD RANKING SEARCH

To enrich the search functions, some multi-keyword search schemes have been proposed. The schemes [7]–[9] support the join query of keywords, only returning documents containing all the query keywords. However, the returned search results are unordered. The ranking search allows the users to quickly find the most relevant documents, return top-k most relevant documents, and effectively reduce the cost of network transmission. Some schemes [10]–[13] supporting multi-keyword ranking search were proposed, among which the scheme proposed by Cao *et al.* [10] (MRSE) is the basis of [11]–[13]. In [10], Cao et al. solved the problem of multi-keyword ranking search over the encrypted cloud data for the first time. In the scheme MRSE, it uses the vector space model to create a document vector for each document, and uses the KNN method [14] to encrypt the document vector. The relevance score is obtained by calculating the inner product of the document vector and the query vector. Finally, the search results are sorted according to the relevance score, and the most relevant top-k documents are returned. Sun *et al.* [11] proposed a search index based on the term frequency and space vector model, and used cosine similarity, which improved the accuracy of search results. The frequency of access to keywords is taken into account when ranking search results in [12]. Fu *et al.* [13] adopted a parallel search method to improve the search efficiency of multi-keywords. The scheme proposed by Yu *et al.* [15] used the homomorphic encryption method to encrypt the index and query vector, and used the characteristics of homomorphic encryption to calculate the relevance score. Therefore, the relevance score is ciphertext, which improves the security and accuracy.

C. FUZZY SEARCH

However, the above-mentioned schemes only support the precise search of keywords, and when the keywords input by the user do not match the predefined keywords, no document is returned, and the fuzzy search for the keywords is ignored. Li *et al.* [16] first proposed the construction scheme of ciphertext fuzzy sets. Wildcards are used to construct the keyword fuzzy set, considering all keywords that may be entered incorrectly. Because the keyword fuzzy set constructed by wildcards occupies more storage space, Li *et al.* [17] improved the construction of the ciphertext fuzzy set, and adopted the gram method to construct the keyword fuzzy set, which saved storage space and improved the search efficiency through a symbolic index tree. Liu *et al.* [18] proposed a dictionary-based ciphertext fuzzy set construction. The keyword fuzzy set occupies less storage space but loses the search accuracy. Wang *et al.* [19] combined the wildcard with the index tree to construct an efficient fuzzy search scheme. Wang *et al.* [20] realized a verifiable fuzzy search scheme by extracting the path information of the index tree structure. However, in these fuzzy search schemes, it is necessary to construct a fuzzy set for each keyword, which occupies a large amount of storage space of the cloud server. Although the Bloom filter can effectively reduce the storage space in [21]–[23], since each keyword in the fuzzy set needs to be inserted into the Bloom filter with multiple hash functions, it increases the computational overhead. In [24], the uni-gram vector model was adopted in the fuzzy search scheme, which improves the accuracy and efficiency of the search. However, it cannot resist distinguishable attacks since the trapdoor is deterministic and support the semantic search for keywords.

D. SEMANTIC SEARCH

The previsouly discussed fuzzy search schemes only consider the similarity of keyword characters, and do not consider the

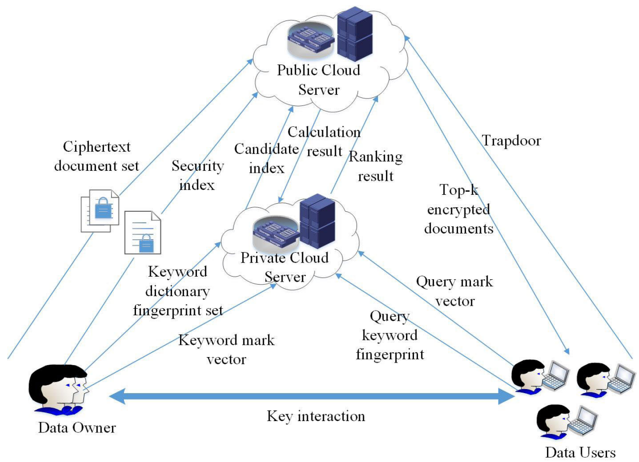

FIGURE 1. System model.

semantic similarity for keywords. Afterward, some semantic search schemes [25]–[29], [41] were proposed. The scheme proposed by Fu *et al.* [25] expands the synonym of keywords, and obtains the correlation score by calculating the inner product, which can support the multi-keyword sorting search for synonyms. Xia *et al.* [26] proposed a semantic search scheme for multi-keywords, which establishes an inverted index for the document set, and semantically expands the query keywords through the semantic library. The relevance score is encrypted by a one-to-many order-preserving encryption scheme. Dai *et al.* [41] adopted the Doc2Vec model to achieve a semantic-aware multi-keyword ranked search scheme. Fu *et al.* [27] used the concept map to implement the semantic search of encrypted data. Fu *et al.* [28], [29] proposed a content-based search scheme that implements an effective search for semantic perception on encrypted data.

The existing schemes either consider only the fuzzy search for keywords, or only consider the semantic search for keywords, and do not take into account both of them in the search scheme. In this study, we propose a fuzzy semantic scheme that supports multi-keyword search.

III. PROBLEM FORMULATION

Our system model is shown in Fig. 1, it consists of four different entities: data owner, data users, private cloud server as well as public cloud server.

1) Data Owner (DO): an entity that owns n data documents PF, the data owner extracts a keyword set $W = (w_1, w_2, \ldots, w_n)$ from the data documents PF, generates the security indexes, keyword mark vector, and keyword dictionary fingerprint set. The DO encrypts the data documents PF before outsourcing to the public cloud server.

2) Data Users (DU): The authorized DU provide t keywords of interest, and they make use of the key provided by the DO to generate the trapdoor TW corresponding to the keywords of interest. The authorized DU decrypt the received documents that meet the data user's requirements.

3) Private Cloud Server: The private cloud server stores the keyword mark vector and keyword dictionary fingerprint set,

ranks the results from the public cloud server and returns the sorted results to the public cloud server.

4) Public Cloud Server: The public cloud server stores the document mark vector, the encrypted data documents, and security index, and computes the comprehensive relevance. After computing, it sends the results to the private cloud server. The public cloud server returns top-k encrypted documents that meet the search criterion to DU.

A. THREAT MODEL

In this paper, we consider the same thread model as [33], [34], which assumes that the private cloud server is "honest and trusted" and does not reveal any information, such as mark vectors, comprehensive relevance, as well as ranking search results. The private cloud server honestly performs document matching operations and sorting of search results, and truthfully sends the sorted search results to the public cloud server.

The public cloud server is considered as "honest-but-curious", and honestly performs the storage and search operations. This server calculates the comprehensive relevance, and truthfully returns the query result according to the sorted results. Moreover, the public cloud server does not add or delete the security indexes and ciphertext document. However, the public cloud server is curious. It tries to obtain the plaintext information of the keywords and indexes, and obtains some additional information through statistical analysis. Therefore, the document collection and index need to be encrypted before uploading to the public cloud server, and should prevent the statistical analysis of the public cloud server. Additionally, we assume that there is no collusion between two cloud servers, which is adopted by most previous works [28], [35]. Our paper mainly discusses the known ciphertext model, in which the public cloud server can only obtain ciphertexts, security indexes, search results, and trapdoor.

B. NOTATIONS

For the convenience of description, Table 1 summarizes the notations employed in our paper.

TABLE 1. Notations.

Symbol	Meaning
PF	the plaintext document set, denoted as $PF = (f_1, f_2, \ldots, f_m)$
CF	the ciphertext document set, denoted as $CF = (cf_1, cf_2, \ldots, cf_m)$
W	the keyword dictionary, denoted as $W = (w_1, w_2, \ldots, w_n)$
SFM	the document index vector, denoted as $SFM = (SFM_1, SFM_2, \ldots, SFM_n)$
KM	the keyword mark vector
FM	the document mark vector
FPI	the finger index list
SI	the security inverted index
QM	the query mark vector
CFS	the document candidate set

IV. PRELIMINARIES

A. SEMANTIC QUERY EXPANSION

The basic idea of query expansion is to expand some related words according to the user query keywords, forming a new query that better represents the semantic information of the user query and improving the recall and precision of the information search. The key to semantic query expansion is to mine the semantic relationship between words to determine which words should be used to implement query keyword expansion during the query process. Automatic query expansion is one of the most commonly used methods for semantic relationship establishment between words. According to the different sources of semantic relations, the automatic semantic expansion technology can be divided into semantic structure-based methods and corpus-based dynamic creation methods. The semantic structure-based approach is generally based on existing dictionaries that can express semantic relationships, such as the English semantic dictionary WordNet. In order to realize the semantic keyword search function, we utilize WordNet [36] to construct semantic database of English created by Princeton University. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets), each expresses a distinct concept. Utilizing WordNet, a keyword KW is expanded to its synonym set $\{KW, s_1, \ldots, s_n\}$, in which $s_1, \ldots, s_n$ are the synonyms of keyword KW. The synonym set are re-arranged to its lexicographical order and denoted it as γ_{KW}.

B. SEMANTIC SIMILARITY CALCULATION

Since the semantic expansion of keywords will produce more words, if all of these words are used for queries, the query range will be too large, and the query result will not be accurate enough. Therefore, the semantic similarity S between the original word and the expanded word is calculated, and the most relevant $pre - \beta$ expansion words are selected to obtain the semantic expansion set $TS = \{key_1, key_2, \ldots, key_n, tkey_1, tkey_2, \ldots, tkey_\beta\}$. At present, there are two main methods for calculating semantic similarity scores: one is based on the semantic distance and the other is based on the information content [37]. Our paper uses the Lin method based on information content [30] to calculate the similarity score between two keywords. Lin considers a general method of calculating similarity. He believes that the similarity of two concepts should be calculated by the ratio of information commonality and total information. The proposed algorithm model is as follows:

$$sim_L(c_1, c_2) = \frac{2 \times \lg p(lso(c_1, c_2))}{\lg p(c_1) + \lg p(c_2)} = \frac{2 \times IC(lso(c_1, c_2))}{IC(c_1) + IC(c_2)}, \tag{1}$$

where $lso(c_1, c_2)$ refers to the smallest common parent node in which the concepts c_1 and c_2 are located in the

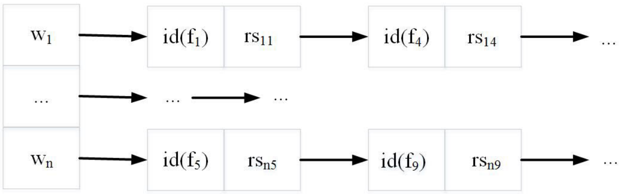

FIGURE 2. Inverted index structure.

classification tree, the IC denotes the information content, and

$$IC(c) = -\log p(c),$$

where $p(c)$ is the probability of the noun c appearing in the WordNet corpus. The calculation formula is as follows:

$$p(c) = \frac{freq(c)}{CN},$$

where CN is the number of nouns in the WordNet corpus, $freq(c)$ denotes the number of words in the corpus that contain the concept c. The $freq(c)$ is defined as

$$freq(c) = \sum_{cn \in words(c)} count(cn),$$

where $words(c)$ denotes the set of words containing the concept c.

C. INVERTED INDEX

Inverted index is one of the most commonly used data structures in search systems. By using an inverted index, the list of documents containing the word can be quickly found. The inverted index is mainly composed of two parts: the word dictionary and the inverted document. The word dictionary is a collection of strings consisting of all the words that have appeared in a collection of documents. Each index item in the word dictionary records some information about the word itself and a pointer to the "inverted list". The inverted document saves the relevant information corresponding to the word. For example, the inverted index structure constructed by the keyword set $KS = \{w_1, w_2, \ldots, w_n\}$ is shown in Fig. 2, and the index linked list of each keyword is a record in the index set, such that in each record, the $id(f_j)$ represents the identifier of the document f_j, where $f_j (f_j \in PF = \{f_1, f_2, \ldots, f_m\})$ denotes a plaintext document, and rs_{ij} represents the relevance between the keyword w_i and the document f_j.

D. RANKING CALCULATION

In the information search, we use the ranking function to measure the relevance scores of the query keywords and matching documents. The TF*IDF weight calculation rules are widely used to calculate the relevance scores of keywords and matching documents [31]. TF (term frequency) indicates the number of times the keyword w appears in the document f. IDF (inverse document frequency) is a measure of the importance of a keyword. The IDF of a given keyword w can be obtained by dividing the total number of documents by the number of documents containing the keyword. Given a keyword w, the formula for calculating the relevance score of the matching document f is as follows:

$$Sc(w, f) = TF * IDF. \tag{2}$$

The TF is calculated as follows:

$$TF = \frac{f_w}{|f|},$$

where f_w denotes the number of keyword w in the document f, $|f|$ denotes the total number of keywords contained in document f. The IDF is calculated as follows:

$$IDF = \log \frac{|PF|}{|PF_w|},$$

where $|PF|$ denotes the total number of documents in document set PF, $|PF_w|$ denotes the number of documents in the document set PF containing the keyword w.

When searching for a document, the given keyword w is semantically expanded to obtain an expanded keyword set Sw, and the comprehensive relevance between the document f and the expanded keyword set Sw needs to be calculated. The calculation formula is as follows:

$$CSc(w, f) = Sc_w + \sum\nolimits_{ew_i \in S_w} Sc_{ew_i} \times Re_i, \tag{3}$$

where Sc_w represents the relevance score of the keyword w and the document f, the ew_i denotes an expanded keyword, and Re_i represents the semantic similarity of the expanded keyword and the original query keyword w.

V. THE CONSTRUCTION OF THE PROPOSED SCHEME

To improve the flexibility and usability of search, we design a FSSE scheme, which can support multi-keyword fuzzy semantic search over encrypted data in cloud computing. In subsection **A**, we present the construction of the proposed scheme. In the FSSE scheme, it calls the fingerprint generation algorithm, fuzzy matching algorithm, and search algorithm. The fingerprint generation and fuzzy matching algorithm are described in subsection **B**. And the search algorithm is described in subsection **C**.

A. FSSE SCHEME

We now give the FSSE scheme which consists of six algorithms: Setup, Index, Enc, Trapdoor, Query, and Decrypt, it is described in **Algorithm 1**.

The six algorithms are described as follows:

1)*Setup*(PF, λ). It contains two sub-algorithms *DfInit* and *KeyGen*.

a)*DfInit*

Step-1: DO extracts n keywords from the plaintext document set $PF = \{f_1, f_2, \ldots, f_m\}$ to obtain a keyword dictionary $W = \{w_1, w_2, \ldots, w_n\}$.

Step-2: For a keyword $w_i \in W$, DO calculates its TF and IDF values.

Algorithm 1 FSSE

Input:

The plaintext document set PF;
The security parameter λ;
The query keywords QW;
The number of returned documents k;

Output:

Top-k plaintext documents PF_k;

```
(sk, SK, PK, KM, SFM) ← Setup(PF, λ);
(FPI, SI) ← Index(PK, PF);
CF ← Enc(PF, sk);
(QM, TW) ← Trapdoor(QW, PK);
RL_k ← Query(QM, KM, SI, TW, k);
PF_k ← Decrypt(CF, sk);
return Top-k plaintext documents PF_k.
```

Algorithm 2 Semantic Expansion (SEA)

Input:

Query keywords $QW = (qw_1, qw_2, \ldots, qw_t)$;

Output:

Expanded query keywords RQW;

```
for each qw_i in QW do
    for each part of speech of qw_i do
        for each meaning of qw_i do
            if qw_i has synonyms then
                RQW = RQW ∪ synonyms;
            end if
        end for
    end for
end for
return RQW.
```

Step-3: DO creates an n-dimensional keyword mark vector $KM = (KM_1, KM_2, \ldots, KM_n)$, where n is the number of keywords in the keyword dictionary, let $KM = (KM_1, KM_2, \ldots, KM_n) = (1, 1, \ldots, 1)$.

Step-4: DO creates the document index vector $SFM = \{SFM_1, SFM_2, \ldots, SFM_n\}$, where $SFM_i = \{KM_i, FM_i\}$, FM_i denotes the m-dimensional document mark vector, and the length of m is the same as the total number of documents.

Step-5: Initialize the document mark vector FM_i, if $w_i \in f_j (1 \leq j \leq m)$, then $FM_{ij} = 1$; otherwise, $FM_{ij} = 0$.

b)*KeyGen*

Takes a security parameter λ, generates an encryption key sk for the document, a private key SK, and a public key PK for homomorphic encryption.

2)$Index(PK, PF)$. Given a public key PK, and the plaintext document set PF, outputs the fingerprint index FPI and the security index SI.

Step-1: For each keyword $w_i \in W$, DO generates the keyword fingerprint FP_i by executing **KFPA** algorithm, and inserts the FP_i into the fingerprint index list FPI, where $FPI = (FP_{w_1}, FP_{w_2}, \ldots, FP_{w_n})$. The detail of the **KFPA** algorithm is described in **Algorithm 3**.

Step-2: For each document $f_j (1 \leq j \leq m) \in PF$, DO creates a unique document identifier $FID_j (1 \leq j \leq m)$, let $FID_j = j$.

Step-3: For each document $f_j (1 \leq j \leq m) \in PF$, if $w_i \in f_j$, DO calculates the relevance score $Score_{w_i, FID_j} = TF * IDF$, and encrypts the relevance score $Score_{w_i, FID_j}$ by executing the modified FHEI [15].

Step-4: For each keyword $w_i \in W$, DO generates the subindex $SI_{w_i} = \{FP_{w_i}, FS\}$ by running **Step-3**, where $FS_i = \{FID_j, Encrypt(PK, Score_{w_i, FID_j})\} (1 \leq j \leq L)$, L represents the total number of documents containing the keyword w_i, and the elements in the FS are arranged in ascending order according to the document identifier FID_j. Then, DO inserts the the subindex SI_{w_i} into the security inverted index SI.

Step-5: Return a security inverted index SI, where $SI = (SI_{w_1}, SI_{w_2}, \ldots, SI_{w_n})$.

3)$Enc(PF, sk)$. Given the plaintext document set PF and an encryption key sk, produces a ciphertext document set CF.

Step-1: For each document $f_j (1 \leq j \leq m) \in PF$, DO adopts a secure symmetric encryption algorithm (e.g. AES) to encrypt the f_j.

Step-2: Return a ciphertext document set $CF = \{cf_1, cf_2, \ldots, cf_m\}$.

4) $Trapdoor(QW, PK)$. It contains four sub-algorithms *QwFinger*, *SemExp*, *SemSim*, and *EncSim*.

a)*QwFinger*

Step-1: For the given query keywords $QW = (qw_1, qw_2, \ldots, qw_t)$, DU initializes an n-dimensional query mark vector QM, let $QM = \{QM_1, QM_2, \ldots, QM_n\} = \{0, 0, \ldots, 0\}$.

Step-2: For each query keyword $qw_j \in QW$, if $qw_j = w_i \in W$, set $QM_i = 1$; otherwise, set $QM_i = 0$.

Step-3: For each query keyword $qw_j \in QW$, if $qw_j \notin W$, generate the fingerprint QFP_i of the qw_j by calling **KFPA** algorithm.

Step-4: Return the query keywords fingerprint $QFP = (QFP_1, QFP_2, \ldots, QFP_o)$ and query mark vector QM.

b)*SemExp*

Step-1: The private cloud server receives the query keywords fingerprint QFP, for each fingerprint $QFP_i \in QFP$, uses **FYMA** algorithm to get the keyword w_i that the user wants to input, and sets $QM_i = 1$. The detail of **FYMA** algorithm is described in **Algorithm 4**.

Step-2: The private cloud server utilizes the SEA algorithm to semantically expand the β query keywords $RQW = (rqw_1, rqw_2, \ldots, rqw_o)$. The detail of SEA algorithm is described in **Algorithm 2**.

c)*SemSim*

Step-1: Calculate the semantic similarity between the original word rqw_i and the expanded word based on the Lin method of the information content [30], and sort the semantic similarity.

Step-2: Select the most relevant α semantic similarity $RE = (Re_1, Re_2, \ldots, Re_o, Re_{o+1}, \ldots, Re_{o+\sigma})$.

Algorithm 3 The Generation of Keyword Fingerprint (KFPA)

Input: Keyword W;
Output: Keyword fingerprint;

```
Initialize the α-dimensional vector M and vector Y;
set M = [0, 0, ..., 0, 0];
set Y = [0, 0, ..., 0, 0];
Use the 2-gram to process each keyword w_i ∈ W to get the string array src;
for i = 0 to i = src.length − 1 do
    Use the hash algorithm HMacMD5 to encrypt src[i] to get the hash value macmd5;
    for the first bit of macmd5 to the last bit of macmd5 do
        if the i-th bit of the hash value macmd5 is 1 then
            M[i] = M[i] + 1;
        else
            M[i] = M[i] − 1;
        end if
    end for
end for
for M[0] to M[α − 1] do
    if M[i] > 0 || M[i] = 0 then
        Y[i] = 1;
    else
        Y[i] = 0;
    end if
end for
return Y.
```

Step-3: If the expanded word is in the keyword dictionary, the value of the corresponding position of the query vector is set to 1.

d)*EncSim*

Step-1: For each semantic similarity $Re_i \in RE$, convert the semantic similarity Re_i into an integer(The semantic similarity needs to satisfy $Re_i * 10a < 2b$, and the expanded value is represented by b bits.), and encrypt the Re_i by executing the modified FHEI [15].

Step-2: Return the trapdoor $TW = \{Encrypt(PK, Re_1), Encrypt(PK, Re_2), \ldots, Encrypt(PK, Re_{o+\sigma})\}$.

5)$Query(QM, KM, SI, TW, k)$. Given the query mark vector QM, the keyword mark vector KM, the security inverted index SI, the trapdoor TW, and the number of returned documents k, outputs the sorted result RL_k. The $Query(QM, KM, SI, TW, k)$ is described in **Algorithm 5**.

6)$Decrypt(CF, sk)$. Given a ciphertext document set CF and an encryption key sk, outputs the top-k plaintext documents PF_k.

Step-1: The authorized user utilizes the secret key sk to decrypt the returned top-k ciphertext document.

Step-2: Return the top-k plaintext documents PF_k.

The detail of the generation of keyword fingerprint and the fuzzy matching algorithm are described in **Algorithm 3** and **Algorithm 4**, respectively.

Algorithm 4 Fuzzy Matching (FYMA)

Input:
Fingerprint QFP_i of the query keyword qw_j;
Fingerprint index list FPI;
Output:
Keyword QW;

```
Initialize the linked list Hamlist;
for FP_{w_i}  in  FP do
    Call Hamming distance calculation method HammingDistance to get the fingerprint QFP_i of the query keyword qw_j and FP_{w_i} Hamming distance L_{w_i};
    Insert L_{w_i} into the linked list Hamlist;
end for
Sort the values in the linked list Hamlist to get the minimum;
According to the minimum Hamming distance, the keyword QW in the keyword dictionary closest to the query keyword fingerprint is obtained;
return QW.
```

B. FINGERPRINT GENERATION AND FUZZY MATCHING ALGORITHM

In this subsection, we describe keyword fingerprint generation and fuzzy matching, as shown below:

- The generation of keyword fingerprint.

1)Enter the keyword $w_i \in W$ to be processed, and create an α-dimensional vector M and an α-dimensional vector Y, respectively. The value of each of the vectors M and Y is 0. The value of α is the same as the number of bits of the hash value generated by the hash function H.

2)Each keyword $w_i \in W$ is processed using the n-gram method. In order to make the search result more accurate, $n = 2$ is taken to obtain multiple features of the keyword w_i. For example, $w_i = english$, after 2-gram processing, we get $english = \{en, ng, gl, li, is, sh\}$, where each element in $english$ is a feature of the keyword w_i.

3)Using the one-way hash function Hmac-MD5 to encrypt each element in *english*, calculate the hash value of each element, and protect the privacy of keywords and indexes.

4)The hash value calculated in 3) is sequentially mapped into the vector M. If the i-th bit of the hash value is 1, the i-th bit of the vector M is incremented by 1, otherwise the i-th bit of the vector M is decremented by 1.

5)The obtained vector M is mapped to the vector Y. If the i-th bit of the vector M is greater than or equal to 0, the i-th bit of the vector Y is set to 1, otherwise the i-th bit of the vector Y is set to 0.

6)Finally, the output vector Y is used as the fingerprint of the keyword w_i.

- Fuzzy matching: the closest keyword to the keyword dictionary fingerprint set is matched by the Hamming distance.

1)Enter the misspelled query keyword fingerprint QFP_i and the extracted fingerprint set of the keyword dictionary.

2)Compare query keyword fingerprint QFP_i and each element in the fingerprint collection of the keyword dictionary to find the keyword with the smallest Hamming distance, which is the query keyword that the user wants to search.

3)Output eligible keyword.

Algorithm 4 describes the fuzzy matching algorithm in detail.

C. SEARCH ALGORITHM

In this subsection, we introduce the search process of the document and give a description of the search algorithm in **Algorithm 5**.

Algorithm 5 Search (SEAM).

Input:
- Query vector QM;
- Keyword mark vector KM;
- Security index SI;
- Trapdoor TW;
- The number of returned documents k;

Output:
- Top-k ciphertext documents;

Initialize the result list $Rlist$;
$CKI = QKMatch(QM, KM)$;
$CFS = DMatch(CKI, SI)$;
for CFS_i in CFS **do**
  $CSc(w_i, f_i) = Cscore(TW, CFS, CKI)$;
  Insert $CSc(w_i, f_i)$ into the result list $Rlist$;
end for
$Rank(Rlist)$;
return Top-k ciphertext documents based on the sorted query results.

In **Algorithm 5**, we call some functions to implement the search. These functions are described as follows:

$QKMatch(QM, KM)$: It is the vector intersection matching operation. The **Function 1** gives its description.

Function 1 QKMatch

Input:
- Query vector QM;
- Keyword mark vector KM;

Output:
- Keyword candidate index CKI;

for $i = 0$ to $i = KM.length - 1$ **do**
  if $QM_i = KM_i = 1$ **then**
    Insert i into the CKI;
  end if
end for
return CKI.

$DMatch(CKI, SI)$: It is a short-circuit matching operation, and its description is given in **Function 2**.

$Cscore(TW, CFS, CKI)$: This operation calculates the comprehensive relevance between the document f and the trapdoor TW. The **Function 3** gives its description.

Function 2 DMatch

Input:
- Keyword candidate index CKI;
- Security inverted index SI;

Output:
- Document candidate set CFS;

$flag = 1$;
for $i = 0$ to $i = FS.length - 1$ **do**
  for $j = 1$ to $j = CKI.length - 1$ **do**
    if $Fid_{w_{CKI[0]},i} == Fid_{w_{CKI[j]},i} == 1$ **then**
      ;
    else
      $flag = 0$;
      break;
    end if
  end for
  if $flag == 1$ **then**
    Insert i into the CFS;
  end if
end for
return CFS.

Function 3 Cscore

Input:
- Trapdoor TW;
- Document candidate set CFS;
- Keyword candidate index CKI;

Output:
- Result list $Rlist$;

Initialize the result list $Rlist$;
for $i = 0$ to $i = CFS.length - 1$ **do**
  $sum = 0$;
  for $j = 0$ to $j = CKI.length - 1$ **do**
    $sum = sum + CFS_{score}[j] * TW_{Re}[j]$;
  end for
  Insert sum into the $Rlist$;
end for
return $Rlist$.

$Rank(Rlist)$: The operation sorts the comprehensive relevance in $Rlist$ in descending order, and its description is given in **Function 4**.

Fig. 3 describes the flow of search process. The data user inputs the query keywords of interest, generates the query keywords fingerprint, finds the keyword in the keyword dictionary closest to the query keywords fingerprint in the private cloud server, and performs semantic expansion on the found query keywords. The private cloud server sends the generated keyword candidate index to the public cloud server, calculates a document candidate set including all the query keywords in the public cloud server, and calculates a comprehensive relevance between the candidate document and the trapdoor according to the document candidate set. The calculation result is inserted into the result linked list, and the result linked list is returned to the private cloud server. The private

Function 4 Rank

Input:

Result list *Rlist*;

Output:

Result list *Rlist*;

for $i = 0$ to $i = Rlist.length - 1$ **do**
  for $j = i + 1$ to $j = Rlist.length - 1$ **do**
    if $Rlist[i] < Rlist[j]$ **then**
      $tmp = Rlist[i]$;
      $Rlist[i] = Rlist[j]$;
      $Rlist[j] = tmp$;
    end if
  end for
end for
return *Rlist*.

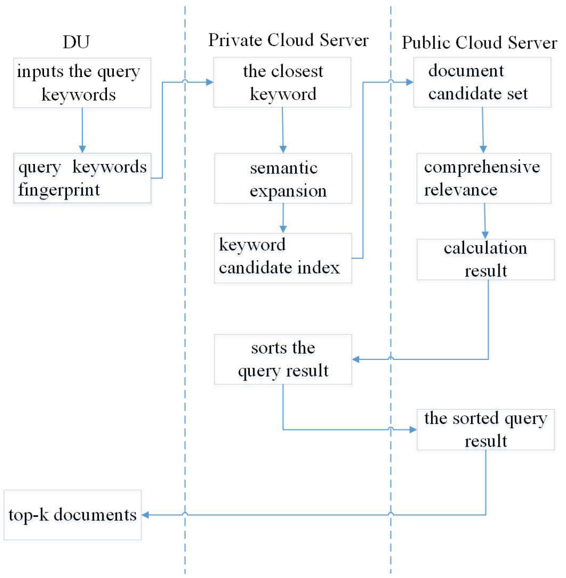

FIGURE 3. The flow of search process.

cloud server sorts the query result and sends the sorted query result to the public cloud server, and the public cloud server returns the top-k documents to the user.

VI. SECURITY ANALYSIS

We analyze the FSSE scheme in terms of security of data privacy, keyword privacy, relevance score privacy, index privacy and query privacy.

1) Privacy protection of data. The data owner encrypts data using the symmetric encryption algorithm AES before outsourcing the data to the cloud server. If the cloud server or the attacker does not obtain the key, the obtained ciphertext document cannot be decrypted. The data owner sends the key to the authorized data user through the secure channel, thus ensuring the privacy of the data.

2) Privacy protection of keywords. To protect the privacy of keywords, it is necessary to prove that the keyword fingerprint generation algorithm is safe under the chosen plaintext attack. Theorem 1 is given below with the proof.

Theorem 1: The keyword fingerprint generation algorithm is safe under the CPA (chosen plaintext attack).

In the FSSE scheme, as described in the KFPA algorithm, the n-gram is first used to process keywords and generate a series of strings. Then, the one-way hash function is used with the key to calculate the generated string. Finally, the fingerprint of the keyword is generated.

Proof: To protect the privacy of keywords, the keyword fingerprint should be indistinguishable; that is, the keyword fingerprint generation algorithm is a random algorithm, not a pseudorandom algorithm. In the game below, the attacker is A and the challenger is C, and the game is divided into initialization, challenge and guess phases.

- Initialization: Create a keyword fingerprint generation algorithm KFPA, where challenger C enters keywords, and KFPA generates keywords fingerprints for the input keywords.
- Challenge: Attacker A outputs two keywords w_0 and w_1 of the same length. Challenger C randomly selects $b \in \{0, 1\}$, enters the keyword w_b, generates the fingerprint FP_{w_b} of w_b, and sends FP_{w_b} to the attacker A.
- Guess: Attacker A outputs b', if $b' = b$, which means that FP_{w_b} is not randomly generated. KFPA is not a random algorithm, but a pseudorandom algorithm.

Since the attacker A can correctly guess with a probability of approximately 1/2, A cannot obtain information of the keyword from the fingerprint of the keyword, which indicates that the keyword fingerprint generation algorithm is safe under the CPA and thereby also proving that the privacy of keywords is protected.

3) Privacy protection for the relevance score. The FSSE scheme encrypts the relevance score using the modified FHEI. The security of the modified FHEI is demonstrated in [15]. First, if the two relevance scores are the same, two different ciphertexts will be obtained after the modified FHEI encryption. Second, since the public key is randomly selected when the modified FHEI encrypts the plaintext, the ciphertext does not have an order-preserving feature. Furthermore, the relevance score using the modified FHEI encryption is not continuous, and the interval between the ciphertext values is random. Because they are randomly distributed, the original order of the plaintext is completely disrupted. The encrypted relevance score does not preserve similarity and relevance. Therefore, the attacker cannot guess the plaintext and match the correct keyword, which effectively protects the privacy of the relevance score.

4) Index privacy protection. The keyword fingerprint, the document mark vector and the encrypted relevance score are included in the inverted index structure of the scheme. The security of the keyword is analysed in 1) because the randomness of the keyword fingerprint algorithm protects the privacy of the keyword fingerprint. The privacy protection of the relevance scores is analysed in 3). The document mark

vector is mainly used to filter ciphertext documents without revealing the information of the ciphertext documents. Therefore, an attacker cannot obtain the information of keywords, documents, and relevance scores from the inverted index.

5) Query privacy protection. The authorized user inputs the query keyword to generate a trapdoor. The trapdoor contains the query keyword fingerprint and encrypted semantic similarity. The security of the keyword fingerprint is analysed in 1). The semantic similarity is encrypted by the modified FHEI in [15]. The relevance score is also encrypted by the modified FHEI. In 3), the security of the relevance score is also analysed. In addition, the security analysis of semantic similarity is similar to 3) and will not be repeated here. An attacker cannot generate a valid trapdoor if he or she does not obtain the key. Therefore, the security of query privacy is also protected.

VII. PERFORMANCE EVALUATION

The paper mainly tests the space overhead of keyword ciphertext fuzzy set, the time overhead of keyword fingerprint generation, the time cost of inverted index construction, and the efficiency of trapdoor generation and search. In the experiment, our FSSE scheme is compared with the MRSE scheme proposed in the literature [1]. When testing the space overhead of the ciphertext fuzzy set of keywords, the FSSE scheme is compared with the wildcard and gram scheme introduced in [10], [11]. The test results show that the space cost of the FSSE scheme is lower.

In the paper, the simulation experiment uses Java language to realize the test of FSSE scheme performance. The data set used is the NSF research award summary data set provided by UCI. This data set includes 129, 000 abstracts. This paper randomly extracts 10, 000 abstracts as test data sets and extracts 10, 000 keywords from the test data set. The simulation was performed on a 3.3 GHz Intel Xeon E3-1226 processor with a 16.0 GB memory using CentOS.

A. KEYWORD FINGERPRINT AND CIPHERTEXT FUZZY SET

1) SPACE OVERHEAD

In the FSSE scheme, the KFPA is used to generate the fingerprint of the keyword, and the keyword fingerprint constitutes the ciphertext fuzzy set. In the KFPA, a one-way hash function Hmac-MD5 is used to encrypt each element in the keyword, which protects the privacy of the keyword. As seen from Fig. 4, compared with the keyword ciphertext fuzzy set constructed by the traditional method, the space overhead of the keyword ciphertext fuzzy set generated by our scheme is the smallest. In the traditional method using wildcards and grams, when the editing distances are $d = 1$ and $d = 2$, the ciphertext storage space of the keywords increases with the number of keywords. The larger the edit distance, the faster the space overhead grows. In the FSSE scheme, each keyword only generates one keyword fingerprint, so in the case of having the same number of keywords, the storage space occupied by the keyword fingerprint set is

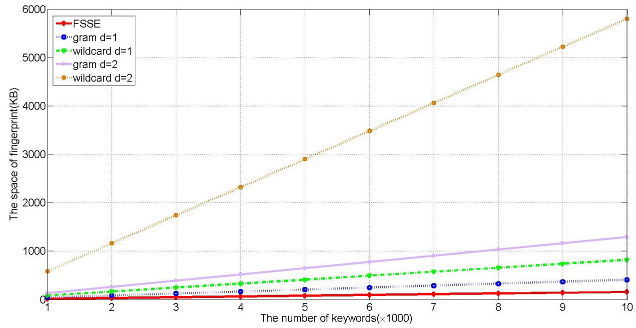

FIGURE 4. Space overhead of ciphertext fuzzy set.

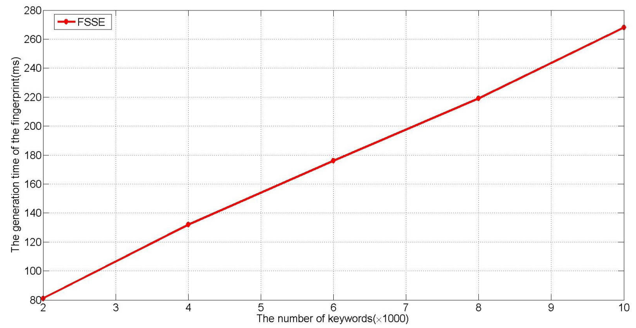

FIGURE 5. Time cost of keyword fingerprint set.

much smaller than that of the fuzzy ciphertext set constructed by using the wildcard and the gram method. As the number of keywords increases, the space advantage of our scheme will become increasingly more obvious.

2) TIME OVERHEAD

Fig. 5 shows the time overhead of the FSSE scheme to generate keyword fingerprint. In the keyword fingerprint generation stage, the n-gram is first used to process keywords, a series of strings are generated. In addition, the one-way hash function is used with the key to calculate the generated string. Finally, the fingerprint of the keyword is generated. The test starts with 2, 000 keywords, increasing by 2, 000 in each iteration, and the maximum number of test keywords is 10, 000. It can be seen from the Fig. 5 that as the number of keywords increases, the time cost increases linearly and slowly. In Fig. 5, the X-axis represents the number of keywords, and the Y-axis represents the time cost of generating the keywords fingerprint. For 10, 000 keywords, the time spent generating the keyword fingerprint is approximately 268 ms, and the time overhead is not very high. Since the fingerprint set of the keywords only needs to be generated once, in an actual application, the time cost can be tolerated.

B. INDEX

1) To facilitate the search, an inverted index is created in the scheme, and the inverted index structure includes the keyword fingerprint, document mark vector and relevance score. First, we extract the keyword set W from the document

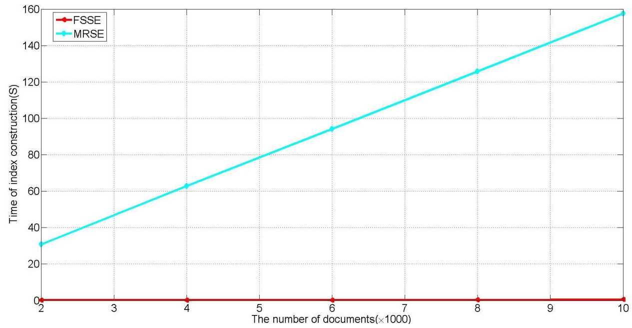

FIGURE 6. The time overhead of index creation.

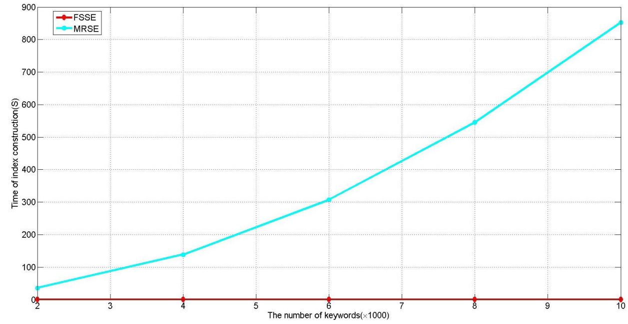

FIGURE 7. The time overhead of index creation.

set PF, input the keyword set W, call the keyword fingerprint generation algorithm, and output the keyword fingerprint set. As shown in Fig. 6, when the number of keywords is fixed to $n = 3,000$, the number of documents increases from 2,000 to 10,000. As the number of documents increases, the index creation time overhead of the FSSE scheme increases slowly and is less affected by the number of documents. With the increase of the number of documents, the index creation time of the MRSE scheme increases linearly and grows faster. Because the time complexity of index creation in the FSSE scheme is $O(n)$, and the time complexity of index creation in the MRSE scheme is $O(mn^2)$. So the time cost of the FSSE scheme for index creation is significantly lower than that of the MRSE scheme. Moreover, the FSSE scheme uses the modified FHEI [15] to encrypt the relevance score when creating the index, and applies the vector intersection matching as well as short-circuit matching operations to filter the documents without keywords, which improves the efficiency. And the literature [15] also proves the efficiency of the modified FHEI. The time spent by the MRSE scheme to create the entire index is linear with the size of the document set because the time spent creating each sub-index is fixed. Therefore, the greater the number of documents, the more obvious the advantages of the FSSE scheme are.

2) As shown in Fig. 7, when the number of documents is $m = 5,000$, the time cost of index creation increases with the number of keywords. The time cost of the MRSE scheme is significantly higher than that of the FSSE scheme, and the FSSE scheme grows slowly. The number of keywords increases from 2,000 to 10,000, and each iteration increases by 2,000. Although the sub-index increased with the increase of keywords, the FSSE scheme filters the documents without keywords, which improves the efficiency of index creation. The modified FHEI is used to encrypt the relevance score, and the time complexity of the modified FHEI is $O(n)$. In the MRSE scheme, since it uses KNN to encrypt the index, the calculation of creating sub-indexes includes splitting and multiplication, and the time complexity is $O(mn^2)$. Therefore, the MRSE scheme creates an index with low efficiency.

C. TRAPDOOR

As shown in Fig. 8, if the number of query keywords entered does not change, the generation time of the trapdoor of the FSSE does not change much with the increase of the number of keywords, and the growth rate of the MRSE scheme is obvious. As seen from Fig. 9, when the number of keywords is $n = 3,000$, the number of query keywords increases from 10 to 35, and the time of trapdoor generation of the FSSE and the MRSE scheme is very small. However, the time of trapdoor generation of the MRSE scheme is higher than that of the FSSE scheme, which is approximately 20 times that of the FSSE scheme. In the FSSE scheme, the generation of the trapdoor includes the generation of the query keyword fingerprint and the encryption semantic similarity. The time cost of query keyword fingerprint is shown in Fig. 5. It can be seen from Fig. 5 that the efficiency of generating the keyword fingerprint is higher. The semantic similarity is encrypted using the modified FHEI with complexity $O(n)$. The MRSE scheme uses the KNN with complexity $O(mn^2)$ for encryption. Therefore, the efficiency of trapdoor generation of the FSSE scheme is significantly higher than that of the MRSE scheme. It can also be seen from Fig. 9 that the query keyword has little effect on the time overhead of generating the trapdoor, which is an important advantage for multi-keyword searchable encryption scheme.

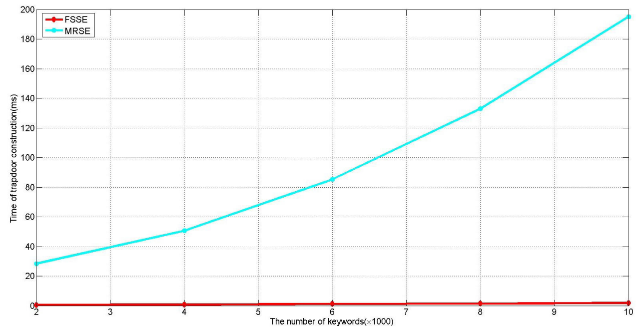

FIGURE 8. The generation time of the trapdoor.

D. SEARCH

As shown in Fig. 10, when the query keyword $t = 15$, as the number of documents increases, the time spent searching increases, and the search cost of the MRSE scheme is higher than that of the FSSE scheme. Because the search operation of the MRSE scheme includes the calculation and sorting of the

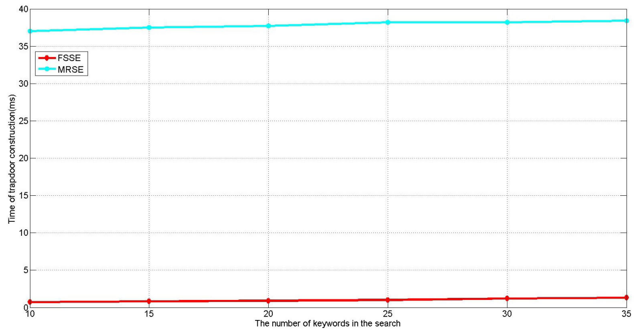

FIGURE 9. The generation time of the trapdoor.

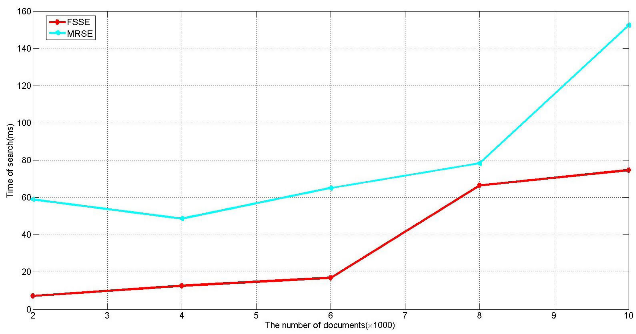

FIGURE 10. Time cost of search.

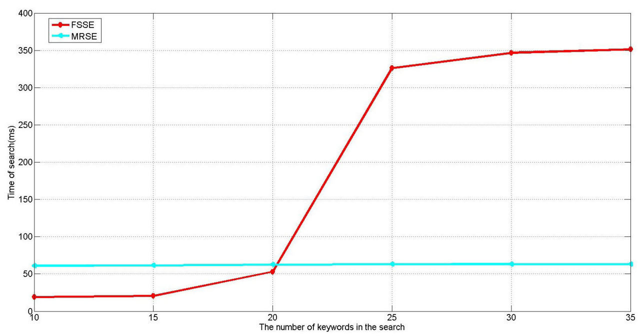

FIGURE 11. Time cost of search.

relevance scores of all the documents in the document collection, however, the vector intersection matching and the short-circuit matching operation are used in the FSSE scheme, the irrelevant documents are effectively filtered, so the search time is reduced. As seen from Fig. 11, the number of query keywords is gradually increased from 10 to 35, and the search time of MRSE scheme is slowly increased. With the increase of the number of query keywords, the search time of the FSSE scheme gradually increases, and the increase rate is also faster. When the number of query keywords is less than 20, the search time of the MRSE scheme is higher than that of the FSSE scheme, and when the number of query keywords is greater than 20, the search time of the FSSE scheme is significantly higher than that of the MRSE scheme. The main reason is that the FSSE scheme supports fuzzy semantic search. After the user inputs the keyword of interest, the KFPA is used to generate a corresponding fingerprint for the query keyword input by the user, the closest keyword to the keyword dictionary fingerprint set is matched by the Hamming distance, and the matched query keyword is semantically expanded. The MRSE scheme does not support the fuzzy semantic search. There is no such operation, so there is no corresponding time overhead. Therefore, as the number of query keywords increases, the time cost of the above operations also gradually increases. However, in practical applications, the query keywords input by the user generally do not exceed 20, so the FSSE scheme is more practical and can also support fuzzy semantic search.

VIII. CONCLUSION

Input or spelling errors that may occur when a user enters a query keyword may cause the query keyword input by the user to not match the predefined keyword, and no document is returned, thereby ignoring the fuzzy search of the keyword. In practical applications, a keyword may have multiple synonyms. Since the user does not consider the semantic expansion of the keyword, the search result will be inaccurate. To solve the fuzzy semantic search problem, we propose an effective fuzzy semantic searchable encryption scheme that supports multi-keyword search over encrypted data in cloud computing. In the paper, we use a keyword fingerprint generation algorithm to generate the fingerprint set of the keyword dictionary and the fingerprint of the query keywords, and perform the fuzzy search by combining the Hamming distance. Furthermore, we expand the query keyword and calculate the semantic similarity between the query keyword and the expanded word of the query keyword to realize the semantic search. To improve the search efficiency, we create an inverted index and use the vector intersection matching and short-circuit matching operations to quickly filter a large number of irrelevant documents. From the theoretical analysis and experimental results, we show that our proposed scheme is secure and privacy-preserving, while correctly realizing the goal of multi-keyword fuzzy semantic search.

However, there are still some challenges in FSSE scheme. The collusion attack between two cloud servers is an important problem which has not been solved. In the future work, we will try to handle the challenge.

REFERENCES

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

[2] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. ACNS*, 2005, pp. 391–421.

[3] 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, Nov. 2011.

[4] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[5] K. Li, W. Zhang, C. Yang, and N. Yu, "Security analysis on one-to-many order preserving encryption-based cloud data search," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1918–1926, Sep. 2015.

[6] S. Tahir, S. Ruj, Y. Rahulamathavan, M. Rajarajan, and C. Glackin, "A new secure and lightweight searchable encryption scheme over encrypted cloud data," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 4, pp. 530–544, Oct. 2019.

[7] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, vol. 4. Berlin, Germany: Springer, 2004, pp. 31–45.

[8] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptogr. Conf.*, 2007, pp. 535–554.

[9] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. Int. Conf. Inf. Commun. Secur.* Berlin, Germany: Springer, 2005, pp. 414–426.

[10] 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.

[11] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.-ASIACCS*, 2013.

[12] 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, pp. 179–190, Jan. 2014.

[13] 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. 98, no. 1, pp. 190–200, 2015.

[14] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. 35th SIGMOD Int. Conf. Manage. Data-SIGMOD*, New York, NY, USA, 2009, pp. 139–152.

[15] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Toward secure multikeyword top-k retrieval over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 4, pp. 239–250, Jul./Aug. 2013.

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

[17] J. Li, Q. Wang, N. Cao, K. Ren, W. Lou, and C. Wang, "Enabling efficient fuzzy keyword search over encrypted data in cloud computing," in *Proc. Cryptol. ePrint Arch.*, 2009.

[18] C. Liu, L. Zhu, L. Li, and Y. Tan, "Fuzzy keyword search on encrypted cloud storage data with small index," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, Beijing, China, Sep. 2011, pp. 269–273.

[19] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 451–459.

[20] J. Wang, H. Ma, J. Li, H. Zhu, S. Ma, X. Chen, and Q. Tang, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Comput. Sci. Inf. Syst.*, vol. 10, no. 2, pp. 667–684, 2013.

[21] L. I. Jin and X. Chen, "Efficient multi-user keyword search over encrypted data in cloud computing," *Comput. Inform.*, vol. 32, no. 4, pp. 723–738, 2013.

[22] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 2112–2120.

[23] C.-M. Yu, C.-Y. Chen, and H.-C. Chao, "Privacy-preserving multikeyword similarity search over outsourced cloud data," *IEEE Syst. J.*, vol. 11, no. 2, pp. 385–394, Jun. 2017.

[24] 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.

[25] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Trans. Consum. Electron.*, vol. 60, no. 1, pp. 164–172, Feb. 2014.

[26] Z. Xia, Y. Zhu, X. Sun, and L. Chen, "Secure semantic expansion based search over encrypted cloud data supporting similarity ranking," *J. Cloud Comput.*, vol. 3, no. 1, pp. 1–11, 2014.

[27] Z. Fu, F. Huang, X. Sun, A. V. Vasilakos, and C.-N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 813–823, Sep. 2019.

[28] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, "Semantic-aware searching over encrypted data for cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2359–2371, Sep. 2018.

[29] 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.

[30] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th Int. Conf. Mach. Learn.* San Francisco, CA, USA: Morgan Kaufmann, 1998, pp. 296–304.

[31] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[32] J. Yao, Y. Zheng, C. Wang, and X. Gui, "Enabling search over encrypted cloud data with concealed search pattern," *IEEE Access*, vol. 6, pp. 11112–11122, 2018.

[33] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneide, "Twin clouds: An architecture for secure cloud computing," in *Proc. Workshop Cryptogr. Secur. Clouds*, 2011.

[34] Y. Yang, J. Liu, S. Cai, and S. Yang, "Fast multi-keyword semantic ranked search in cloud computing," *Chin. J. Comput.*, vol. 41, no. 6, pp. 1126–1139, 2018.

[35] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related SQL range queries with privacy preserving," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 7, pp. 1596–1608, Jul. 2017.

[36] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[37] C. Wang, C. Wang, Z. Wang, X. Ye, J. X. Yu, and B. Wang, "DeepDirect: Learning directions of social ties with edge-based network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2277–2291, Dec. 2019.

[38] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 715–725, Sep. 2017.

[39] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Serv. Comput.*, early access, Apr. 11, 2019, doi: 10.1109/TSC.2019.2910113.

[40] 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, Jan. 2017, Art. no. e2942.

[41] X. Dai, H. Dai, G. Yang, X. Yi, and H. Huang, "An efficient and dynamic semantic-aware multikeyword ranked search scheme over encrypted cloud data," *IEEE Access*, vol. 7, pp. 142855–142865, 2019.

[42] A. Ghanbarpour and H. Naderi, "An attribute-specific ranking method based on language models for keyword search over graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 1, pp. 12–25, Jan. 2020.

[43] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Inf. Sci.*, vol. 494, pp. 193–207, Aug. 2019.

GUOXIU LIU was born in 1982. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications. Her current research interests include data security and privacy protection, and cloud computing security.

GENG YANG was born in 1961. He is currently a Professor and a Ph.D. Supervisor with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications. His current research interests include computer communication and networks, parallel and distributed computing, cloud computing security, and information security.

SHUANGJIE BAI was born in 1992. He is currently pursuing the Ph.D. degree with the College of Computer Science and Software, Nanjing University of Posts and Telecommunications, Nanjing, China. He is also a Joint Ph.D. Student with the University of Stavanger, Stavanger, Norway. His research interests include privacy-preserving and blockchain.

QIANG ZHOU was born in 1978. He received the Ph.D. degree from the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, in 2014. He is currently a Professor with Chuzhou University. His research interests include wireless sensor networks, parallel and distributed computing, and information security.

HUA DAI was born in 1982. He is currently an Associate Professor with the Nanjing University of Posts and Telecommunications. His research interests include data management and security, and database security. He is a member of CCF.