Received October 17, 2021, accepted November 2, 2021, date of publication November 8, 2021, date of current version November 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125729

B^+-Tree Based Multi-Keyword Ranked Similarity Search Scheme Over Encrypted Cloud Data

HUANGLIN SHEN, LINLIN XUE, HAIJIANG WANG, LEI ZHANG, AND JINYING ZHANG
School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China

Corresponding author: Linlin Xue (119029@zust.edu.cn)

This work was supported in part by the Natural Science Foundation of Zhejiang Province under Grant LQ20F020010.

ABSTRACT With the sustained evolution and expeditious popularization of cloud computing, an ever-increasing number of individuals and enterprises are encouraged to outsource data to cloud servers for reducing management overhead and ease of access. Privacy requirements demand encryption of sensitive information before outsourcing, which, on the other hand, diminishes the usability of data and makes considerable efficient keyword search techniques used on plaintext inapplicable. In this paper, we propose a secure multi-keyword ranked search scheme based on document similarity to work out the problem. In order to achieve the goals of multi-keyword search and ranking search results, we adopt the vector space model and TF-IDF model to generate index and query vectors. By introducing the secure kNN computation, index and query vectors can be encrypted to prevent cloud servers from obtaining sensitive frequency information. For the need of efficiency advancement, we adopt the B^+-tree as the basic structure to build the index and construct a similar document collection for each document. Due to the use of our unique index structure, compared to linear search, the search efficiency is more exceptional. Extensive experiments on the real-world document collection are conducted to demonstrate the feasibility and efficiency of the proposed solution.

INDEX TERMS Searchable encryption, B^+-tree, cosine similarity, multi-keyword ranked search, cloud security.

I. INTRODUCTION

Cloud computing [1] has achieved extraordinary development over the past decade, both in the academic and industrial communities [2]. Moreover, it has been regarded as a brand-new model of technology infrastructure that is capable of organizing unlimited storage space and powerful computing capabilities, and enabling users to enjoy pay-as-you-go, convenient and distinguished services from a shared pool of configurable computing resources with excellent efficiency and minimal management overhead [3]–[5]. In addition, the technique is able to decrease the capital expenditure on hardware establishments, software and personnel maintenances [22]. Hence, enterprises and individuals tend to outsource data to cloud servers by occasion of these advantages [6].

Despite of the tremendous advantages of cloud services, privacy concerns brought by outsourcing data, especially sensitive data (e.g., emails, personal travel data, and company transaction records, etc.), to cloud servers restrict the promotion and popularization of the emerging model. Cloud data may be misused by cloud service providers (CSPs) in an unauthorized way, even maliciously, since data owners are no longer directly in control of their data [24]. In order to achieve more effective application and broader deployment of cloud computing [8], [14], [15], data security and privacy are indispensable considerations that must be well-addressed to avoid monetary loss or damage to reputation arise from cloud data leakage [9]. General approaches to protect data confidentiality are cryptographic approaches such as encrypting data before outsourcing [10]. However, such methods improve the difficulty of data utilization since many technologies applied on plaintext data, such as keyword-based information retrieval, are no longer suitable for ciphertext data. Furthermore, downloading and decrypting all cloud data is unrealistic and infeasible, especially in the case of large amount of data [11].

In order to decrease the impact of encryption on data availability, plenty of efforts have been put into contriving efficient mechanisms for searching over encrypted cloud data. Some general-purpose methodologies based on fully-homomorphic encryption [12] and oblivious RAMs [13] have been proposed

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

to address the above problem, while the overhead for computation and communication presented in these schemes is not acceptable for both cloud servers and users. Fortunately, many methodologies used for special purposes based on searchable encryption (SE) have been put forward to satisfy different query requirements. However, among schemes that have been proposed, the single keyword search lacks sufficient intelligence to support complex query demands, and network traffic overhead of the boolean search is excessive [16], [26]. In contrast, the multi-keyword ranked search receives increasing attention due to its better practicability. Recently, some constructive schemes based on multi-keyword ranked search have been proposed to support intelligent and economic queries over encrypted cloud data. However, in most cases, these methodologies cannot simultaneously satisfy requirements of search efficiency and data privacy protection.

Aiming at problems as are mentioned above in the field of multi-keyword ranked search, in this paper, we propose a secure and efficient multi-keyword ranked search scheme based on B^+-tree index, which has been extensively applied in database systems. For supporting multi-keyword search, we combine the vector space model and the TF-IDF model in the process of generating index and query vectors. In addition, to improve the query efficiency for the better quality of experience, we incorporate the cosine similarity measure [17], [18], [21] to the index structure. Due to the particular structure of our index, the search scheme proposed in this paper is more exceptional than linear in terms of time overhead. Moreover, on the premise of ensuring the accuracy of relevance score calculation between query vectors and index vectors, we introduce the secure kNN (k-nearest neighbour) computation [19], [20] to encrypt vectors so as to improve the ability of resisting statistical attacks from cloud servers. To defense attacks initiated by cloud servers under different threat models, we design two secure index schemes, e.g., the basic similarity-based multi-keyword ranked search (BSMRS) scheme and the enhanced similarity-based multi-keyword ranked search (ESMRS) scheme. The former can guarantee the confidentiality of index and query vectors, the latter is able to avoid sensitive frequency information being obtained by cloud servers to satisfy more stringent privacy protection requirements. Our contributions are summarized as follows:

1) We design a searchable encryption scheme that not only supports accurate multi-keyword ranked search but also ensures data privacy with little relevance score information leakage.
2) By incorporating the cosine similarity measure and constructing the keyword index tree based on B^+-tree, the search efficiency of the proposed scheme is improved significantly compared with [39] and [53].
3) Extensive experimental results demonstrate the feasibility and efficiency of the proposed scheme.

The rest of the paper is organized as follows. Section II introduces the related work. Then, we briefly introduce preliminaries, system model, threat models, and design goals in Section III, followed by Section IV, which gives the specification of our schemes. Section V presents security analysis. Experiments and performance evaluation are presented in Section VI. Section VII covers the conclusion.

II. RELATED WORK

Searchable encryption (SE) has been extensively studied with the aim of formalizing security definitions and improving efficiency. It enables clients to outsource data in encrypted form to cloud servers and conduct keyword search over ciphertext. In accordance with differences of cryptography primitives, searchable encryption can be divided into public key searchable encryption [29], [55]–[58] and symmetric searchable encryption [27], [28], [30]. On the ground of the expensive computational overhead of public key searchable encryption, this paper mainly pays attention to symmetric searchable encryption.

A. SINGLE KEYWORD SEARCH

The first symmetric searchable encryption (SSE) scheme was proposed by Song *et al*. [27]. The cloud server in their scheme needs to traverse the entire document to determine whether it contains a specific keyword. Thus time complexity of search is linearly related to the number of documents in collection. Goh [28] proposed a standardized description of the security definition of SSE and constructed a secure index architecture on the basis of pseudo-random functions and Bloom filter to resist adaptive chosen keyword attack. However, the time complexity of their scheme is $O(n)$. To further enhance security and search efficiency, SSE-1 and SSE-2 based on the inverted list were proposed by Curtmola *et al*. [30]. Such two schemes are more efficient than other works and can resist chosen-keyword attack and adaptive chosen-keyword attack respectively. However, the functionality of most of the above schemes is restricted to single keyword search.

B. MULTI-KEYWORD BOOLEAN SEARCH

To improve query experience and enrich search functionality, a great quantity of explorations [23], [31]–[38] have been carried out by research fellows to achieve multi-keyword boolean search, which enables users to query the most appropriate document by inputting several query keywords. In conjunctive keyword search schemes [23], [31], [32], [38], only documents containing all keywords are returned. Among these works, the communication overhead of the scheme proposed by Golle *et al.* [31] is linear with the number of documents, and the scheme proposed by Cash *et al*. [38] supports large databases. Unlike conjunctive keyword search, all of documents containing one or more query keywords are returned in disjunctive keyword search schemes [33], [34]. For the sake of supporting conjunctive keyword search and disjunctive keyword search simultaneously, predicate search schemes were proposed [35]–[37]. However, these schemes above are not exceptional enough since the search results are based on keywords that have existed, which are not capable of providing satisfactory results ranking functionality [39].

Consequently, some works have been proposed to handle multi-keyword ranked search with the advantage of bandwidth-saving.

C. MULTI-KEYWORD RANKED SEARCH

Due to the capability of implementing more efficient and convenient search, multi-keyword ranked search is extensively utilized in the field of information retrieval, it enables the most relevant document to be retrieved in a short period of time. It estimates the relevance between query keywords and documents, and sends the top-k most relevant documents to users. Therefore, it can effectively diminish the overhead of communication. Cao *et al.* [40] proposed a privacy-preserving multi-keyword ranked search scheme and demonstrated the security of the scheme. The searchable index in their scheme is constructed on the basis of the vector space model [41] and the "coordinate matching" is selected as the scale of measurement. The scheme is capable of ranking search results in light of the number of matched keywords. However, the time complexity of search is linear to the number of documents in collection since the cloud server must traverse the whole indexes of the document collection to confirm the number of matched keywords for each query. On the other hand, the lack of consideration of the importance of different keywords results in the loss of precision. The vector space model and TF-IDF model are combined in the multi-keyword ranked search scheme with better-than-linear search time complexity proposed by Sun *et al.* [5]. Moreover, authors incorporate the cosine similarity measure to the index to provide similarity-based ranking. Although the efficiency is improved, the scheme is not accurate enough and vulnerable in protecting data privacy. The scheme proposed by Orencik *et al.* [42] clusters similar documents by utilizing LSH (local sensitive hash) functions. The algorithm is appropriate for similarity search while the ranking accuracy is not sufficient. By drawing on previous research methods and indicators, Xia *et al.* [39] proposed a "Greedy Depth-first Search" algorithm on the basis of tree-based index. The efficiency of the scheme is better than early works and the precision is excellent. However, the overhead of search and the time complexity of trapdoor generation remain high. Zhang *et al.* [43] and Zhong *et al.* [3] put forward their multi-keyword ranked search scheme respectively, while the efficiency it not ideal.

III. PROBLEM FORMULATION

A. COSINE SIMILARITY MEASURE

In this paper, we adopt the cosine similarity measure [5], [25], [44] to calculate the similarity between plaintext documents denoted as vectors. The closer the cosine value is to 1, the higher the similarity between two documents. The similarity between documents is calculated as follows:

$$\cos(P, V) = \frac{P \cdot V}{\|P\| \, \|V\|} = \frac{\sum_{i=1}^{n} P_i V_i}{\sqrt{\sum_{i=1}^{n} P_i^2} \sqrt{\sum_{i=1}^{n} V_i^2}}, \quad (1)$$

TABLE 1. Notations.

Notation	Description
$\mathcal{D}$	the plaintext document collection, which contains n documents that are unencrypted, denoted as $\mathcal{D} = \{d_1, d_2, ..., d_n\}$.
$\mathcal{W}$	the keyword dictionary composed of m keywords extracted from $\mathcal{D}$, denoted as $\mathcal{W} = \{w_1, w_2, ..., w_m\}$.
$\mathcal{W}_q$	the collection of query keywords whose elements are t keywords that contained in query vector.
$\widetilde{\mathcal{D}}$	the encrypted document collection, denoted as $\widetilde{\mathcal{D}} = \{\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_n}\}$.
$\mathcal{I}$	the document collection index tree in unencrypted form.
$\widetilde{\mathcal{I}}$	the searchable encrypted index tree that is generated from $\mathcal{I}$.
V_q	the query vector generated based on $\mathcal{W}_q$.
$\mathcal{T}$	the trapdoor generated by query vector.
V_u	the index vector stored in u, and the dimensionality is equal to m. It is worth noting that the u can be an internal node, a leaf node or a document record.
$\widetilde{V_u}$	the encrypted form of index vector V_u.
$\mathcal{R}$	the query result collection, which contains the top-k most relevant documents to the trapdoor $\mathcal{T}$.
$\mathcal{S}$	similar document collection, i.e., the collection whose members are index vectors of the $K(K \geq k)$ most similar documents of a certain document.
N	the order of the B^+-tree.

where P, V respectively represent a vector of a document and P_i,V_i denote their component.

B. VECTOR SPACE MODEL AND TF-IDF MODEL

Vector space model, in combination with TF-IDF model, is extensively employed for supporting efficient multi-keyword ranked search in the field of plaintext information retrieval [41], [45], TF (term frequency) is used to evaluate the importance of a specific term (keyword) in a document, specifically, the more times a word appears in a document, the more important it is to this document, and IDF (inverse document frequency) is used to measure the ability of a keyword to distinguish documents. If a keyword appears frequently in a document but rarely in other documents, it indicates that the discrimination coefficient of the keyword is excellent. In the vector space model, each document is represented as a vector V_u, which is composed of normalized TF values of keywords in the dictionary $\mathcal{W}$ in the corresponding document. Similarly, each query is represented as a vector and elements of the vector are normalized IDF values of query keywords. The dimensionality of index and query vectors equals to the total number of keywords in the dictionary and the relevance of query vectors and documents is quantitatively evaluated by the dot product of V_u and V_q.

The definition of relevance computation function [39] is as follows:

$$Score(V_u, V_q) = V_u \cdot V_q = \sum_{w_i \in w_q} TF_{w_i} \times IDF_{w_i}, \quad (2)$$

where TF_{w_i} is the normalized TF value of keyword w_i, and IDF_{w_i} is the normalized IDF value of keyword w_i.

If u is an internal node of the index tree $\mathcal{I}$, TF_{w_i} is computed according to index vectors in corresponding child nodes and leaf node according to index vectors in corresponding document records. If u is a document record, TF_{w_i} is calculated as:

$$TF_{w_i} = \frac{TF'_{f,w_i}}{\sqrt{\sum_{w_i \in w} (TF'_{f,w_i})^2}}, \tag{3}$$

where TF'_{f,w_i} is the TF value of w_i in specific document f, $TF'_{f,w_i} = 1 + \ln N_{f,w_i}$, N_{f,w_i} is times of keyword w_i appears in document f.

In the query vector V_q, IDF_{w_i} is computed as [46]:

$$IDF_{w_i} = \frac{IDF'_{w_i}}{\sqrt{\sum_{w_i \in w} (IDF'_{w_i})^2}}, \tag{4}$$

where IDF'_{w_i} is the normalized IDF value of w_i in $\mathcal{D}$, $IDF'_{w_i} = \ln(1 + N_d/N_{w_i})$, N_{w_i} is the number of documents that contain keyword w_i and N_d is the total number of documents.

C. KEYWORD B^+-TREE

The B^+-tree [47] is one of the most widely-used index structures for database systems and data-manipulation applications [48]. Solutions to the B^+-tree are also often applied to other tree-like index structures. The keyword B^+-tree stores data only in leaf nodes that do not have children, and internal nodes store index vectors and pointers to corresponding child nodes. The retrieval time of the index structure based on the B^+-tree is proportional to the height of the tree. Compared with the red-black tree and the binary tree, the height of the B^+-tree is lower. Therefore, we utilize the B^+-tree to construct our index structure. The formal definition of u is as follows:

$$u = \langle ID, child[N], V_u, \mathcal{S} \rangle, \tag{5}$$

If u is a document record, ID stores document identity, $\mathcal{S}$ is composed of ID and V_u of K documents most similar to the current document in the document collection $\mathcal{D}$ and $child$ is set to $null$. If the u is a leaf node or a internal node, ID and $\mathcal{S}$ are set to $null$, if the u is a leaf node, V_u denotes a vector consisting of normalized TF values which are calculated as follows:

$$V_u[i] = max\{u.record[1] \rightarrow V_u[i], \ldots, u.record[N-1] \rightarrow V_u[i]\}, \tag{6}$$

and if the u is a internal node, V_u is calculated as follows:

$$V_u[i] = max\{u.child[1] \rightarrow V_u[i], \ldots, u.child[N] \rightarrow V_u[i]\}, \tag{7}$$

where N is the order of the B^+-tree.

The construction procedure is explained detailedly in Section IV, which is denoted as $IndexGen(\mathcal{D}, \mathcal{K})$.

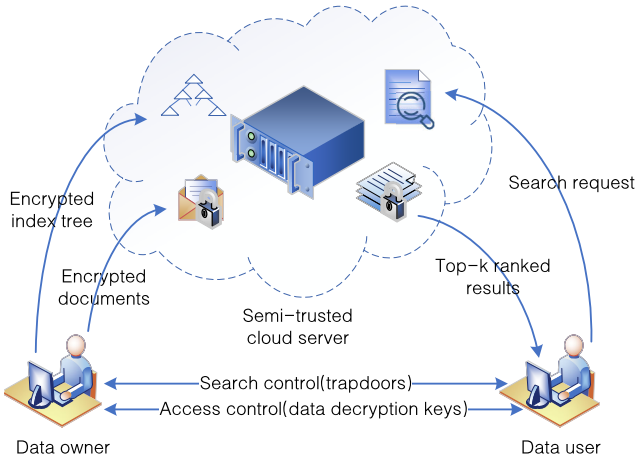


FIGURE 1. The system model of ranked search over encrypted cloud data.

D. THE SECURE kNN COMPUTATION

The secure kNN (k-nearest neighbour) computation, which is proposed by Wong *et al.* [19], is designed to calculate the Euclidean distance between a database record and a query vector and then select k nearest database records. In the secure kNN computation, the secret key $\mathcal{K}$ is composed of a randomly generated m-bit vector $\tilde{S}$ and two $(m \times m)$ invertible matrices $\{\tilde{M}_1, \tilde{M}_2\}$, where $\tilde{S}$ is regarded as a splitting indicator and $\{\tilde{M}_1, \tilde{M}_2\}$ are used to encrypt database records and query vectors, both of which are extended to m-dimension vectors. The specific encryption process is introduced in Section IV. More details of the secure kNN computation are referred to in [19].

E. THE SYSTEM MODEL

As shown in in Figure. 1, data owner, data user and cloud server are three different entities considered in this paper.

Data owner needs to construct a dictionary $\mathcal{W}$, which is composed of distinct keywords extracted from document collection $\mathcal{D}$ before outsourcing so that the data availability can be maintained while protecting data privacy. And then, with the dictionary and document collection, an unencrypted index tree can be constructed. Finally, the data owner encrypts the document collection and index tree and outsources encrypted form of them to the cloud server.

Data user is able to obtain the authorization of accessing a particular document from the data owner. In light of search control mechanisms, the data user can generate a trapdoor $\mathcal{T}$ with t query keywords and k encrypted documents will be returned after the trapdoor is uploaded to the cloud server. Finally, with the share secret key, the data user can decrypted documents.

Cloud server is responsible for storing the encrypted document collection $\widetilde{\mathcal{D}}$ and index tree $\widetilde{\mathcal{I}}$. After acquiring the trapdoor $\mathcal{T}$, search is executed by the cloud server over the encrypted index tree $\widetilde{\mathcal{I}}$. To improve the retrieval accuracy and decrease network traffic, the cloud server ranks search results

and only the top-k most relevant documents are returned to the data user.

F. THREAT MODELS

In this paper, we treat the data owner and the data user as entities that can be fully trusted, but the cloud server is regarded as ''honest-but-curious'', which reflects the view taken in most of the related works whose research direction are secure schemes of search over encrypted cloud data [49]–[51]. ''Honest'' is defined as executing instructions in the designated protocol correctly. ''Curious'' refers to inferring and analyzing data received to gain additional insight. Threat models adopted in this paper are the two suggested by Cao *et al*. [40]. They differ primarily in term of the information available to the cloud server.

Known ciphertext model. Information that is available to the cloud server in this model is restricted to encrypted document collection $\widetilde{\mathcal{D}}$, encrypted index tree $\widetilde{\mathcal{I}}$ and encrypted query vector, i.e., trapdoor $\mathcal{T}$. In other words, the attack that the cloud server can conduct is just ciphertext-only attack.

Known background model. The cloud server that utilizes this stronger model possesses a greater degree of knowledge, e.g., term frequency of a specific keyword, the correlation of trapdoors submitted by the data user and related statistical information of documents. The cloud server has the ability to deduce or even identify a keyword in a query with knowledge above [52].

G. DESIGN GOALS

Requirements that need to be satisfied include following three aspects:

Accuracy-improved multi-keyword ranked search. Accurately retrieving the document required by the data user is the most primitive requirement. The scheme is not feasible if documents returned by the cloud server are completely inconsistent with the expectation of the data user.

Search efficiency. The efficiency objective of the scheme is to diminish search time complexity to better than linear by utilizing the B^+ tree as the index structure and construct a similar document collection $\mathcal{S}$ for each document.

privacy-preserving. Document collection and trapdoor information involve privacy, so the scheme must take appropriate measures to prevent the cloud server from obtaining relevant information. The following are privacy protection requirements mainly concerned:

- *Index and query confidentiality*. The cloud server must be adequately prevented from obtaining information of plaintext of index vectors and trapdoors.
- *Trapdoor unlinkability*. The cloud server should not have the ability to identify whether two trapdoors are from the same query or not.
- *Keyword privacy*. Whether a certain keyword is included in a query should not be speculated by the cloud server. It is worth noting that protecting access pattern, i.e., the sequence of documents that be returned to the data user, is not the design objective of the scheme, for the sake of efficiency concerns.

IV. THE PROPOSED SCHEMES

In this section, we first describe the basic similarity-based multi-keyword ranked search (BSMRS) scheme, which guarantees the confidentiality of index and query. For defensing attacks under a stronger threat model, i.e., the known background model, we propose a more secure scheme, i.e., the enhanced similarity-based multi-keyword ranked search (ESMRS) scheme.

A. BSMRS SCHEME

By introducing the secure kNN computation [19], the BSMRS scheme can be configured to satisfy privacy requirements within the known ciphertext model. Following are detailed descriptions of each algorithm in the scheme.

- $\mathcal{K} \leftarrow KeyGen(m)$ The algorithm is executed by the data owner to generate the secret key $\mathcal{K}$, including a m-bit secret vector $\tilde{S}$ which is randomly generated and two $(m \times m)$ invertible matrices $\tilde{M}_1$ and $\tilde{M}_2$. Elements of $\tilde{S}$ are 0 or 1. Namely, $\mathcal{K} = \{\tilde{S}, \tilde{M}_1, \tilde{M}_2\}$. The formal process is presented in Algorithm. 1.
- $\widetilde{\mathcal{I}} \leftarrow IndexGen(\mathcal{D}, \mathcal{K})$ The algorithm is used to construct the encrypted index tree $\widetilde{\mathcal{I}}$. Figure. 3 illustrates an index tree. It is worth noting that, all data is stored in leaf nodes and ordered according to keys, thus splitting operation needs to be executed in the process of inserting to ensure the characteristic of order. The formal description of inserting is presented in Algorithm. 4 and an example is shown in Figure. 4. The encryption process is described as follows: first, the data owner splits every index vectors V_u into two random vectors $\{V'_u, V''_u\}$. Specifically, with the m-bit vector $\tilde{S}$ as the splitting indicator, if $\tilde{S}[j] = 0, j = 1, 2, \ldots, m$, $V'_u[j]$ and $V''_u[j]$ are set equal to $V_u[j]$; if $\tilde{S}[j] = 1, j = 1, 2, \ldots, m$, $V'_u[j]$ and $V''_u[j]$ are set as two random values while the summation of $V'_u[j]$ and $V''_u[j]$ equals to $V_u[j]$. Finally, the encrypted index tree $\widetilde{\mathcal{I}}$ is constructed, as shown in Figure. 2, and each u stores two encrypted index vectors $\widetilde{V}_u = \{\tilde{M}_1^T V'_u, \tilde{M}_2^T V''_u\}$. The formal process is presented in Algorithm. 2.
- $\mathcal{T} \leftarrow TrapdoorGen(\mathcal{W}_q, \mathcal{K})$ The algorithm is used for generating trapdoors and the specific description is as follows: first, generating an unencrypted query vector V_q whose dimensionality is m by query keyword set $\mathcal{W}_q$, if $w_j \in \mathcal{W}_q$, $V_q[j]$ stores the normalized IDF value of keyword w_j, else the value of $V_q[j]$ is 0. Similar to the above $IndexGen(\mathcal{D}, \mathcal{K})$ algorithm, the query vector is split into two random vectors V'_q and V''_q, on the contrary, if $\tilde{S}[j] = 0, j = 1, 2, \ldots, m$, values of $V'_q[j]$ and $V''_q[j]$ are random and the summation of $V'_q[j]$ and $V''_q[j]$ equals to $V_q[j]$, if $\tilde{S}[j] = 1, j = 1, 2, \ldots, m$, $V'_q[j]$ and $V''_q[j]$ are set equal to $V_q[j]$. Finally, the algorithm returns

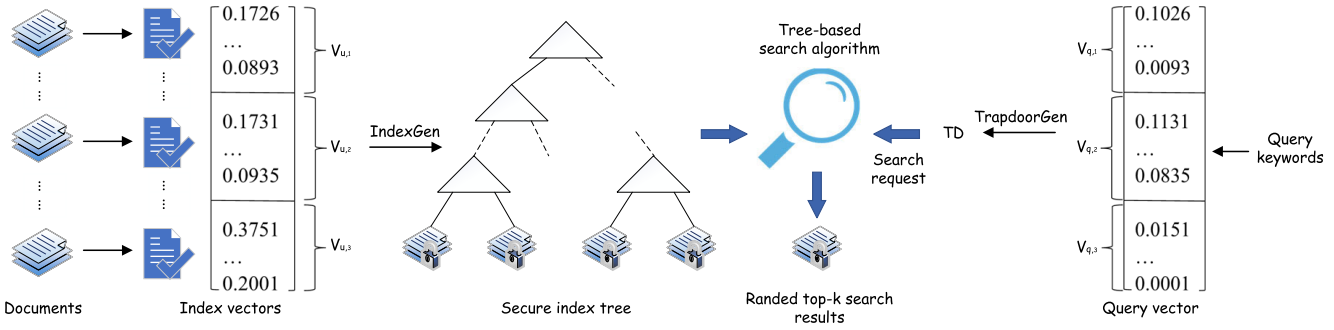


FIGURE 2. Overview of secure index scheme.

trapdoor $\mathcal{T} = \{\tilde{M}_1^{-1} V_q', \tilde{M}_2^{-1} V_q''\}$. The formal process is presented in Algorithm. 3.

- $\mathcal{R} \leftarrow Search(\mathcal{T}, k, u)$ With the trapdoor $\mathcal{T}$, the cloud server can calculate the relevance score between u of the encrypted index tree $\widetilde{\mathcal{I}}$ and the query vector V_q as in the formula (2). Therefore, upon obtaining the trapdoor $\mathcal{T}$, the cloud server performs the designated search operation (Algorithm. 5 $Search(\mathcal{T}, k, u)$) over the encrypted index tree $\widetilde{\mathcal{I}}$. During the search process, attribute to the utilization of the similar document collection, which is composed of index vectors of the K most similar documents of a certain document, after finding the document $\widetilde{d}_i$ with the largest relevance score to the trapdoor, the cloud server just need to calculate relevance scores of similar documents of $\widetilde{d}_i$, instead of continuing to access other nodes, because the similar document collection of $\widetilde{d}_i$ contains the top-k most relevant documents. Therefore, the search efficiency is improved significantly. After selecting and ranking the top-k documents, the cloud server returns the query result $\mathcal{R}$. It is worth noting that relevance scores computed from encrypted vectors are identical with that computed from unencrypted vectors, i.e., $\widetilde{V}_u \cdot \mathcal{T} = V_u \cdot V_q$, where $\widetilde{V}_u = \{\tilde{M}_1^T V_u', \tilde{M}_2^T V_u''\}$ and $\mathcal{T} = \{\tilde{M}_1^{-1} V_q', \tilde{M}_2^{-1} V_q''\}$. The detailed proof process is as follows:

$$\begin{aligned}
&\widetilde{V}_u \cdot \mathcal{T} \\
&= (\tilde{M}_1^T V_u') \cdot (\tilde{M}_1^{-1} V_q') + (\tilde{M}_2^T V_u'') \cdot (\tilde{M}_2^{-1} V_q'') \\
&= (\tilde{M}_1^T V_u')^T (\tilde{M}_1^{-1} V_q') + (\tilde{M}_2^T V_u'')^T (\tilde{M}_2^{-1} V_q'') \\
&= V_u'^T \tilde{M}_1 \tilde{M}_1^{-1} Q' + V_u''^T \tilde{M}_2 \tilde{M}_2^{-1} V_q'' \\
&= V_u' \cdot V_q' + V_u'' \cdot V_q'' \\
&= V_u \cdot V_q \\
&= Score(V_u, V_q) \qquad (8)
\end{aligned}$$

B. ESMRS SCHEME

In the BSMRS scheme, due to the introduction of the random split, non-deterministic encryption can be provided, which means that the same query vectors (e.g., identical query keywords) will be encrypted into different trapdoors. Besides, information outsourced to the cloud server is restricted to encrypted vectors and the calculation involved is only inner product operation. Accordingly, there is no information about particular keywords that can be disclosed. Therefore, the query unlinkability and the keyword privacy can be protected in the known ciphertext model. However, in the known background model, the cloud server is equipped with more knowledge. Moreover, the relevance score computed from $\widetilde{V}_u$ and $\mathcal{T}$ is identical with that from V_u and V_q, thus the cloud server is capable of identifying same query requests in light of identical access paths and relevance scores, and distinguishing keywords according to distribution differences of keywords in the term frequency distribution histogram. Consequently, the query unlinkability and the keyword privacy are in danger [7]. To enhance security and satisfy more rigorous privacy requirements, the equality must be broken. Therefore, some tunable randomness is introduced into the procedure of relevance evaluating to disturb the score. Additionally, the randomness can be calibrated for the sake of efficiency, ranked search accuracy, and keyword privacy.

The ESMRS scheme is basically consistent with the BSMRS scheme in most aspects except that:

Algorithm 1: KeyGen(m)

Input: dimension m.
Output: secret key $\mathcal{K}$.

```
Generate a m-bit vector S̃ and two (m × m) matrices M̃1 and M̃2;
for each element in S̃ do
    S̃[i] = rand()%2;
end
for each matrix do
    while |M̃i| = 0 do
        Regenerate M̃i;
    end
end
Generate a secret key K = {S̃, M̃1, M̃2};
return K;
```

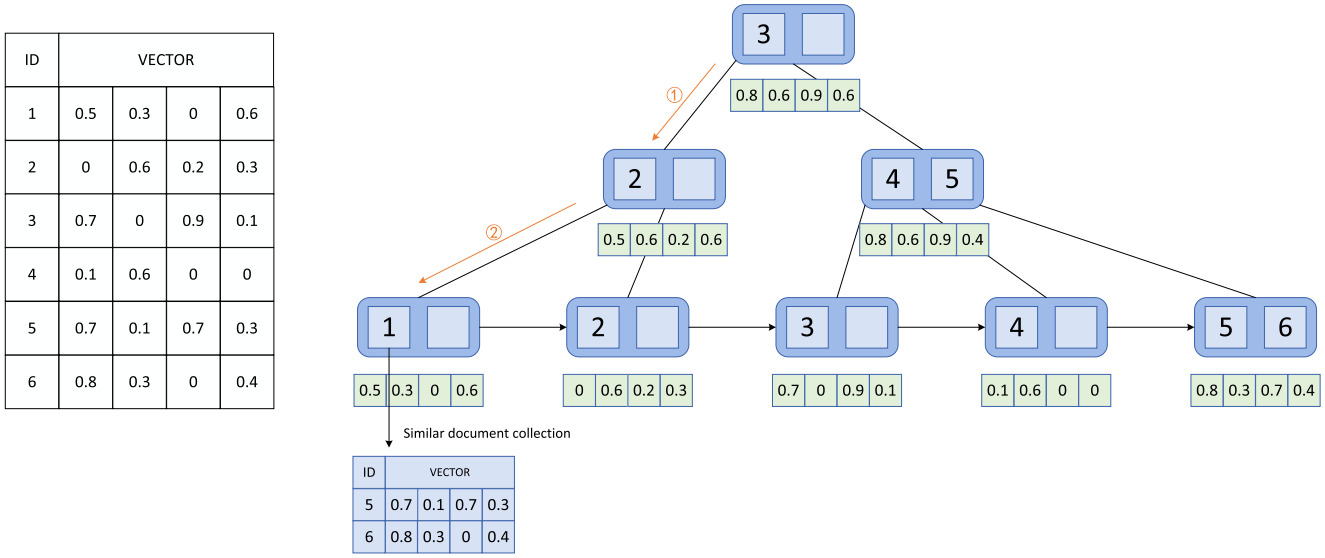


FIGURE 3. An example of index tree with document collection $\mathcal{D} = \{d_i | i = 1, 2, \ldots, 6\}$, $m = 4$, $K = 2$ and $N = 3$. In the construction procedure of the index tree, we first construct a similar document collection for each document and generate a leaf node as the root node. Then, we insert documents with splitting operation. This diagram also shows the search process using a query vector, in which the V_q is equal to (0.6, 0.2, 0.1, 0.6) and $k = 3$ (the data user will receive three documents at last). In light of the search scheme, the search begin from the root of the tree, the relevance score of (0.5, 0.6, 0.2, 0.6) to the query is 0.90, which is bigger than that of (0.8, 0.6, 0.9, 0.4), similarly, the relevance score of (0.5, 0.3, 0, 0.6) to the query is 0.98. Then, the algorithm calculates the relevance score of each similar document of d_1 and ranks by descending order. Finally, $\{d_1, d_6, d_5\}$ are returned.

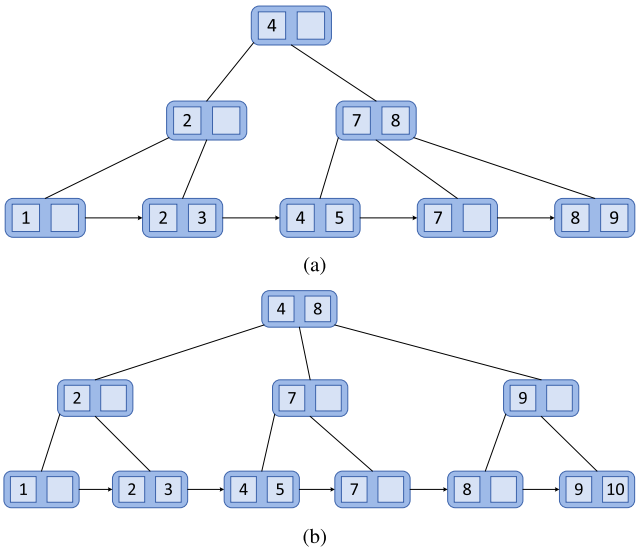


FIGURE 4. An example of inserting operation. Before inserting, the B^+-tree whose order is 3 is shown as (a). Now we try to insert a document with *ID* 10. Firstly, we find the leaf node that meets the condition is [8, 9]. However, because the node is full, it is unable to continue to insert, so it is necessary to split the node into [8] and [9]. Then the document is inserted into [9], and the *ID* 9 is inserted into the parent node [7, 8]. At this time, the parent node is full, and it is also unable to continue to insert and needs to be reorganize globally. The tree after inserting is shown in (b).

- $\mathcal{K} \leftarrow KeyGen(m+\varepsilon)$ In this algorithm, the secret vector $\tilde{S}$ is a $(m+\varepsilon)$-dimension vector, and $\tilde{M}_1$ and $\tilde{M}_1$ are $(m+\varepsilon) \times (m+\varepsilon)$ invertible matrices, where ε is the number of phantom terms.
- $\widetilde{\mathcal{I}} \leftarrow IndexGen(\mathcal{D}, \mathcal{K})$ In this algorithm, the index vector V_u is a $(m+\varepsilon)$-dimension vector, and $V_u[j], j = m+1, \ldots, m+\varepsilon$ is set as a random value η_j.
- $\mathcal{T} \leftarrow TrapdoorGen(\mathcal{W}_q, \mathcal{K})$ Similar to the index vector V_u, the dimensionality of the query vector is increased to $(m+\varepsilon)$ before encryption as well. The difference is that values of a random number of extended elements are 1, and others are 0.
- $\mathcal{R} \leftarrow Search(\mathcal{T}, k, u)$ After introducing some phantom terms, the final relevance score of index vector $\widetilde{V}_u$ and $\mathcal{T}$ equals to $V_u \cdot V_q + \sum \eta_v$, where $v \in \{j | V_q[j] = 1\}$.

V. SECURITY ANALYSIS

In this section, we analyze the security of the ESMRS scheme. The security depends on the secure kNN computation.

A. SECURITY PROOF

Theorem: Due to the introduction of the random split, the scheme is capable of preventing the cloud server from decrypting ciphertext if it does not get the secret key $\mathcal{K}$.

Proof: For each index vector V_u, the cloud server knows the encrypted value $\widetilde{V}_u = \{\widetilde{V_{ua}}, \widetilde{V_{ub}}\} = \{\tilde{M}_1^T V'_u, \tilde{M}_2^T V''_u\}$. Without the splitting indicator $\tilde{S}$, the cloud server has to set V'_u and V''_u as two random m-dimension vectors, and set the following equations: $\widetilde{V_{ua}} = \tilde{M}_1^T V'_u$ and $\widetilde{V_{ub}} = \tilde{M}_2^T V''_u$. The number of unknown variables in V'_u and V''_u is $2m$ and that in $\tilde{M}_1$ and $\tilde{M}_2$ is $2m^2$, but the number of equations is $2m$. Therefore, the information known by the cloud server is not enough to crack matrices $\tilde{M}_1$ and $\tilde{M}_2$. Basically, the cloud

Algorithm 2: IndexGen($\mathcal{D}$, $\mathcal{K}$)

Input: the secret key $\mathcal{K}$ and the document collection $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ with identities $\mathcal{ID} = \{ID|ID = 1, 2, 3 \ldots n\}$.
Output: the encrypted index tree $\widetilde{\mathcal{I}}$.

Generate a leaf node U;
Initialize a new list *CurrentRecordSet*;
for *each document* d_{ID} *in* $\mathcal{D}$ **do**
  Generate a record u for d_{ID}, with $u.child = null$, $u.ID = ID$, and $V_u[i] = TF_{d_{ID},w_i}$;
  Find the K most similar documents and insert $\{ID, V_u\}$ of them into $u.\mathcal{S}$ by using formula (1);
  insert u into *CurrentRecordSet*;
end
while *the number of records in CurrentRecordSet is more than* 0 **do**
  Insert(u, U);
end
Clear *CurrentRecordSet*;
for *each index vector* V_u *in the index tree* **do**
  Generate two index vectors V'_u and V''_u;
  for *each element in* V'_u *and* V''_u **do**
    if $\tilde{S}[i] = 0$ **then**
      $V'_u[i] = V''_u[i] = V_u[i]$;
    else
      $V'_u[i] = \eta', V''_u[i] = V_u[i] - \eta'$;
    end
  end
  Generate an encrypted index vector $\widetilde{V}_u = \{\tilde{M}_1^T V'_u, \tilde{M}_2^T V''_u\}$ and clear V_u, V'_u and V''_u;
end
return the root of index tree $\widetilde{\mathcal{I}}$;

Algorithm 3: TrapdoorGen($\mathcal{W}_q$, $\mathcal{K}$)

Input: the query keyword collection $\mathcal{W}_q$ and the secret key $\mathcal{K}$.
Output: the trapdoor $\mathcal{T}$.

Generate a m-bit query vector V_q;
for *each keyword in* $\mathcal{W}$ **do**
  if $w[j] \in \mathcal{W}_q$ **then**
    $V_q[j] = IDF_{w_j}$;
  else
    $V_q[j] = 0$;
  end
end
Generate two query vectors V'_q and V''_q;
for *each element in* V'_q *and* V''_q **do**
  if $\tilde{S}[j] = 1$ **then**
    $V'_q[j] = V''_q[j] = V_q[j]$;
  else
    $V'_q[j] = \eta', V''_q[j] = V_q[j] - \eta'$;
  end
end
Generate a trapdoor $\mathcal{T} = \{\tilde{M}_1^{-1} V'_q, \tilde{M}_2^{-1} V''_q\}$ and clear V_q, V'_q and V''_q;
return $\mathcal{T}$;

server is obliged to try out all configurations of splitting so as to solve the matrices. Since there are 2^m possible splitting configurations, the introduction of random split makes the scheme 2^m more costly to attack. Accordingly, if m is large enough, the cloud server is not able to decrypt the ciphertext without the secret key.

B. PRIVACY ANALYSIS

1) INDEX AND QUERY CONFIDENTIALITY

With the introduction of the random split, index vectors are encrypted by invertible matrices. Therefore, the cloud server is not able to deduce initial vectors without the secret key, which has been proved above. Moreover, the degree of difficulty of figuring out matrices is increased by introducing phantom terms. Consequently, index confidentiality can be protected. Based on the same principle, the query keywords are invisible to the cloud server as well.

2) TRAPDOOR UNLINKABILITY

The introduction of random value η_j enables the ESMRS scheme to generate different query vectors and obtain different relevance score distributions when search requests are identical. That is to say, the trapdoor unlinkability is enhanced. However, since the access pattern protection is not the design objective of the proposed scheme from the efficiency point of view, similarities contained in query results from identical search requests can be taken advantage of by the cloud server. In the proposed ESMRS scheme, the value of $\sum \eta_v$ can be adjusted to keep the balance of efficiency and privacy. The data user is able to make a trade-off between the two options.

3) KEYWORD PRIVACY

By introducing the random value η_j and setting a random number of extended elements of query vector as 1, the $\sum \eta_j$ as a part of the final relevance score will not be identical even search requests are the same. In consideration of ranked search accuracy, η_j follows the identical uniform distribution $U(\mu' - \xi, \mu' + \xi)$, where the mean is μ', and the variance as σ'^2 is $\xi^2/3$. In light of the central limit theorem, the summation of ω independent η_j, i.e., $\sum \eta_j$ follows the normal distribution $N(\mu, \sigma^2)$, where the expectation μ and the standard deviation σ can be calculated as:

$$\begin{cases} \mu = \omega\mu' \\ \sigma^2 = \omega\xi^2/3 \end{cases} \tag{9}$$

Algorithm 4: Insert(u, *root*)

Input: $u = \langle ID, child[N], V_u, \mathcal{S} \rangle$ and root.

Find the corresponding leaf node L whose range of ID includes $u.ID$ by binary search.
if *the number of records of* $L < N - 1$ **then**
  Insert u into L;
  Update the index vector V_u of L and its ancestor nodes;
  return;
else
  if $L.parent$ *is not full* **then**
    Split L into two nodes L_1 and L_2;
    $L_1 \leftarrow$ first half of records in the L;
    $L_2 \leftarrow$ the rest records in the L;
    Insert u into one of them;
    Insert the middle ID of L_1 and L_2 into $L.parent$;
    Link L_2 to L_2 with a live link;
    Update index vector V_u of L_1 and L_2;
    Update index vector V_u and *child* of ancestor nodes of L_1;
  else
    Execute global reorganization;
  end
end

Algorithm 5: Search($\mathcal{T}$, k, u)

Input: the trapdoor $\mathcal{T}$, the number of documents to be returned k and u
Output: the result collection $\mathcal{R}$

Initialize a new list $\mathcal{R}$;
if u *is not a leaf node* **then**
  MAX_SCORE = 0;
  MAX_CHILD = u;
  for *each child in* $u.child[N]$ **do**
    score = u.child[i].$\widetilde{V}_u \cdot \mathcal{T}$;
    if *score* > *MAX_SCORE* **then**
      MAX_SCORE = score;
      MAX_CHILD = u.child[i];
    end
  end
  Search($\mathcal{T}$, k, MAX_CHILD);
else
  Find the record whose relevance score with $\mathcal{T}$ is the largest;
  for *each similar document in* $u.\mathcal{S}$ **do**
    Calculate the relevance score;
  end
  Rank u and its similar documents in descending order according to relevance scores;
  Insert the top-k $\{ID, Score\}$ into $\mathcal{R}$;
end
return $\mathcal{R}$;

Thus, we can generate the random value η_j according to the value of $\mu' = \mu/\omega$ and $\xi = \sqrt{3/\omega}\sigma$. The standard deviation σ can be considered as a trade-off parameter between security and ranked search accuracy. It is worth noting that σ needs to be set small enough out of the concern of effectiveness, but it will increase the risk that the cloud server gets more statistical information of original scores. Therefore, σ can be adjusted to keep the balance of accuracy and privacy.

VI. PERFORMANCE EVALUATION

The purpose of this section is to evaluate the performance of our proposed schemes by performing extensive experiments on the real-world document collection: the 20 Newsgroups data set [54]. We implement all algorithms mentioned above using Python language on a 1.80GHz Intel(R) Core(TM) processor, Windows 10 operation system with a RAM of 8.00GB. The tests include 1) the precision and rank privacy of search, and 2) the efficiency of index construction, trapdoor generation and search.

A. PRECISION AND PRIVACY

As presented in Section IV. Phantom terms are introduced to prevent the cloud server from linking identical search requests for better data security. Therefore, the relevance scores between index vectors and trapdoors will not be exactly accurate. In the ESMRS scheme, there are two accessible factors (i.e., the number of phantom terms and the level of random value) that can influence the precision and rank privacy. Similar to related works, the ''precision'' P_k is defined as [40]:

$$P_k = k'/k \tag{10}$$

where k' is the number of the real top-k documents that the data user receives. Figure. 5(a) shows that the fluctuation of precision of the ESMRS scheme attributes to the number of phantom terms and the level of the random value, and with small level of random value and number of phantom terms, the capability of search is not influenced much. The definition of ''rank privacy'' is obtained from [40] as well:

$$R_k = \sum |l_i - l_i'|/k^2 \tag{11}$$

where l_i is the rank number of document in the search results, and l_i' is that in the real ranked documents. The larger rank privacy means that the security is better, Figure. 5(b) shows the rank privacy with various numbers of phantom terms and levels of random value.

B. EFFICIENCY

1) INDEX TREE CONSTRUCTION

The procedure of index construction for document collection $\mathcal{D}$ can be divided into three main steps, i.e., 1) finding the K most similar documents for each document in light of formula(1), 2) constructing an unencrypted B^+-tree based on the document collection $\mathcal{D}$, and 3) encrypting all vectors in the index tree. When constructing the index tree,

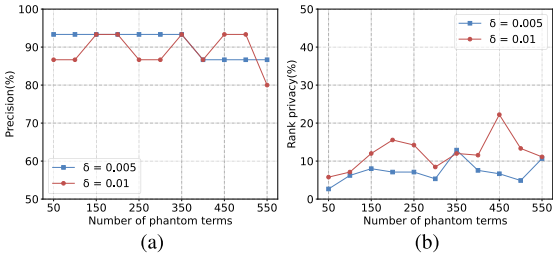


FIGURE 5. The search precision (a) and rank privacy (b) with different numbers of phantom terms and random value upper limit δ.

the number of documents in the collection $\mathcal{D}$ and the size of keyword dictionary $\mathcal{W}$ are principal factors that influence the time overhead. Figure. 6(a) shows that the time consumed to construct the index tree is basically linear with the number of documents. Figure. 6(b) shows that with the fixed document collection, the time overhead is proportional to the number of keywords in the dictionary when constructing the index tree. Due to the expansion of vector dimensionality, the ESMRS scheme consumes slightly more time than the BSMRS scheme in constructing encrypted index tree. It is worth noting that the index construction is a one-time operation. In this paper, we compare our schemes with the EDMRS scheme [39] and the DVMRS scheme [53]. The results show that the time overhead of our schemes is less than EDMRS and is approximate to DVMRS with increased size of document collection, and is less than both of them with increased size of keyword dictionary. Note that, in the process of encrypting leaf node, we store the encrypted index vector of each plaintext index vector temporarily, which can be used in the subsequent encryption process, so each index vector is only encrypted once, and the number of similar documents has little impact on the time overhead of index construction, as shown in Figure. 6(c). Moreover, the order of the index tree can influence the time overhead to a certain extent, as shown in Figure. 6(d).

2) TRAPDOOR GENERATION

The trapdoor generation process includes two multiplications of a matrix and a vector splitting operation. Therefore, the time complexity is $O(\alpha^2)$, where $\alpha = m + \varepsilon$. Figure. 7(a) shows that the time overhead of generating trapdoors primarily contingents on the number of keywords in the dictionary since most of the time is used to encrypt the query vector, and the dimensionality of the vector contingents on the size of the dictionary. Thus the time overhead increases as the size of the keyword dictionary is enlarged. Moreover, the ESMRS scheme consumes more time because the dimensionality have been extended compared to the BSMRS scheme. Figure. 7(b) indicates that the generation time of trapdoor is almost unaffected by the number of query keywords.

3) SEARCH EFFICIENCY

We improve the search efficiency in two ways: 1) introducing B^+-tree as the basic structure to build the index tree, 2) constructing a similar document collection for each document.

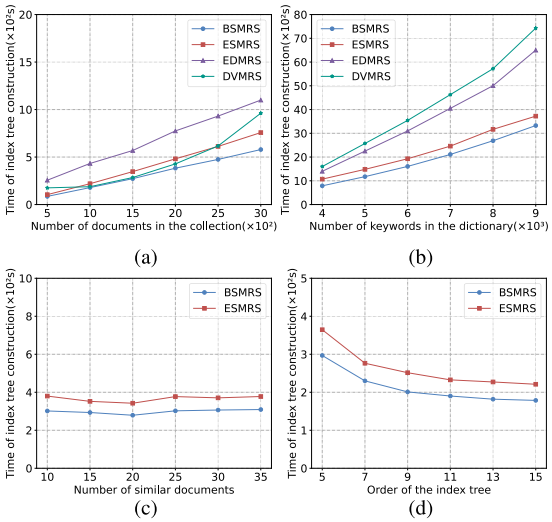


FIGURE 6. The time overhead of constructing the index tree: (a) for different sizes of document collection with fixed size of dictionary, $m = 4000$, (b) for different sizes of dictionary with fixed size of document collection, $n = 4000$, (c) for different numbers of similar documents, (d) for different levels of order of B^+-tree.

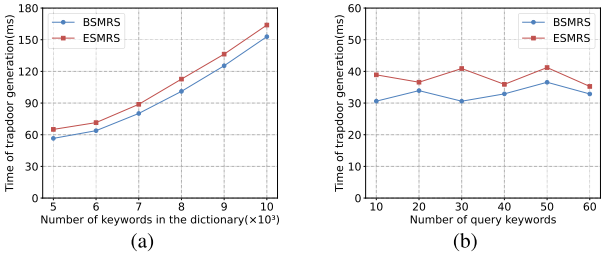


FIGURE 7. The time overhead for generating trapdoor: (a) for different sizes of dictionary with fixed number of query keywords, $t = 10$, and (b) for different numbers of query keywords with fixed size of dictionary, $m = 4000$.

The search process performed by the cloud server mainly includes searching for the document which is most relevant to the trapdoor and ranking the document and its K most similar documents in descending order according to relevance scores with the trapdoor. The search algorithm terminates after the top-k documents are selected. We evaluate the search efficiency of our proposed schemes and compare with the EDMRS scheme and the DVMRS scheme under different parameter settings. In particular, we study the effect of the size of document collection and the cardinality of keyword dictionary. In our schemes, B^+-tree is the basic structure of the index tree, the height of the tree is $O(\log_N n)$, and the computation times is N on each layer of the index tree, so the time complexity of search is $O(N \log_N n)$, it is better than linear. In addition, benefit from the use of similar document collection, the number of nodes that need to be visited is less than other schemes, it contributes to the improvement of the search efficiency as well. Results in Figure. 8 demonstrate that our search scheme is significantly more efficient in terms of time overhead. In particular, the efficiency of search in EDMRS and DVMRS drops obviously with the increased

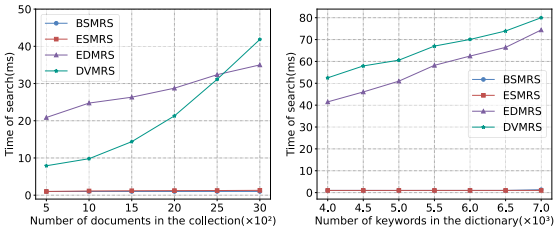


FIGURE 8. The time overhead of search using 10 words as query keywords: (a) for different sizes of document collection with fixed size of dictionary, $m = 4000$, and (b) for different sizes of dictionary with fixed size of document collection, $n = 4000$.

number of documents and cardinality of keyword dictionary, while ours maintain high efficiency. Note that, for the purpose of keeping the balance of accuracy and privacy, the number of phantom terms that added to disturb the relevance score is 400 (10% of the number of keywords). Thus, the search efficiency is not influenced apparently, so curves of the BSMRS scheme and the ESMRS scheme in Figure. 8 are adjacent.

In conclusion, without losing the efficiency of index tree construction, we effectively improve the efficiency of search, which indicates that our scheme is feasible and efficient.

VII. CONCLUSION

In this paper, we conduct thorough research on the efficiency and security issues of multi-keyword ranked search over encrypted cloud data and propose a secure and efficient search scheme. The scheme can not only achieve accurate multi-keyword ranked search but also make the search time better than linear. In terms of accuracy, the vector space model and TF-IDF model are exploited to effectively acquire accurate ranked search results. The secure kNN computation is combined to protect the scheme against two threat models. To improve the search efficiency, we construct the index tree based on the B^+-tree structure and construct a similar document collection for each document before encryption. Through thorough security analysis, our proposed scheme is proved that it is secure and privacy-preserving while maintaining the precision of multi-keyword ranked search. Extensive experimental results on the real-world document collection demonstrate the feasibility and efficiency of the scheme.

In the proposed scheme, the similar document collection increases the storage overhead to a certain extent. Therefore, in our future work, we will explore schemes that support better space efficiency.

REFERENCES

[1] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015, doi: 10.1016/j.ins.2015.01.025.

[2] M. Babar, M. S. Khan, F. Ali, M. Imran, and M. Shoaib, "Cloudlet computing: Recent advances, taxonomy, and challenges," *IEEE Access*, vol. 9, pp. 29609–29622, 2021, doi: 10.1109/ACCESS.2021.3059072.

[3] H. Zhong, Z. Li, J. Cui, Y. Sun, and L. Liu, "Efficient dynamic multi-keyword fuzzy search over encrypted cloud data," in *J. Netw. Comput. Appl.*, vol. 149, pp. 102–469, Jan. 2020, doi: 10.1016/j.jnca.2019.102469.

[4] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012, doi: 10.1109/MIC.2012.14.

[5] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, Nov. 2014, doi: 10.1109/TPDS.2013.282.

[6] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–36, Jan. 2020, doi: 10.1145/3362031.

[7] 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, doi: 10.1109/TPDS.2011.282.

[8] H. Xiong, Y. Zhao, L. Peng, H. Zhang, and K.-H. Yeh, "Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 453–461, Aug. 2019, doi: 10.1016/j.future.2019.03.008.

[9] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu, and J. Shen, "Privacy-preserving multi-keyword searchable encryption for distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 561–574, Mar. 2021, doi: 10.1109/TPDS.2020.3027003.

[10] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Financ. Cryptogr. Data Secur.*, 2010, pp. 136–149.

[11] T. Xiao, D. Han, J. He, K.-C. Li, and R. F. de Mello, "Multi-keyword ranked search based on mapping set matching in cloud ciphertext storage system," *Connection Sci.*, vol. 33, no. 1, pp. 95–112, Apr. 2020, doi: 10.1080/09540091.2020.1753175.

[12] C. Gentry, *A Fully Homomorphic Encryption Scheme*. Ann Arbor, MI, USA: Stanford Univ., 2009.

[13] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, 1996.

[14] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, May 2010, doi: 10.1007/s13174-010-0007-6.

[15] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 356–367, Apr. 2020, doi: 10.1109/TSC.2019.2947914.

[16] X. Ding, P. Liu, and H. Jin, "Privacy-preserving multi-keyword top-k similarity search over encrypted data," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 2, pp. 344–357, Mar. 2019, doi: 10.1109/TDSC.2017.2693969.

[17] H. Liao and Z. Xu, "Approaches to manage hesitant fuzzy linguistic information based on the cosine distance and similarity measures for HFLTSs and their application in qualitative decision making," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5328–5336, Jul. 2015, doi: 10.1016/j.eswa.2015.02.017.

[18] D. Liu, X. Chen, and D. Peng, "Some cosine similarity measures and distance measures between q-rung orthopair fuzzy sets," *Int. J. Intell. Syst.*, vol. 34, no. 7, pp. 1572–1587, 2019, doi: 10.1002/int.22108.

[19] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2009, pp. 139–152, doi: 10.1145/1559845.1559862.

[20] Z. Xia, T. Shi, N. N. Xiong, X. Sun, and B. Jeon, "A privacy-preserving handwritten signature verification method using combinational features and secure kNN," *IEEE Access*, vol. 6, pp. 46695–46705, 2018, doi: 10.1109/ACCESS.2018.2866411.

[21] G. W. Wei, "Some similarity measures for picture fuzzy sets and their applications," *Iranian J. Fuzzy Syst.*, vol. 15, no. 1, pp. 77–89, 2018, doi: 10.22111/IJFS.2018.3579.

[22] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020, doi: 10.1109/TNET.2020.2979807.

[23] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing-Based Cryptography—Pairing* (Lecture Notes in Computer Science), vol. 4575. Berlin, Germany: Springer, 2007, pp. 2–22, doi: 10.1007/978-3-540-73489-5_2.

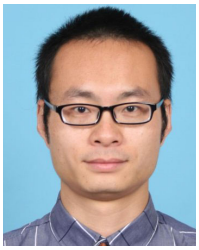
[24] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016, doi: 10.1109/TPDS.2015.2506573.

[25] T. Mahmood, M. Ilyas, Z. Ali, and A. Gumaei, "Spherical fuzzy sets-based cosine similarity and information measures for pattern recognition and medical diagnosis," *IEEE Access*, vol. 9, pp. 25835–25842, 2021, doi: 10.1109/ACCESS.2021.3056427.
[26] Y. Cui, F. Gao, Y. Shi, W. Yin, E. Panaousis, and K. Liang, "An efficient attribute-based multi-keyword search scheme in encrypted keyword generation," *IEEE Access*, vol. 8, pp. 99024–99036, 2020, doi: 10.1109/ACCESS.2020.2996940.
[27] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy. (S&P)*, May 2000, pp. 44–55, doi: 10.1109/SECPRI.2000.848445.
[28] E.-J. Goh, "Secure indexes," *Cryptol. ePrint Arch.*, vol. 2003, p. 216, Oct. 2003.
[29] H. Wang, J. Ning, X. Huang, G. Wei, G. Sen Poh, and X. Liu, "Secure fine-grained encrypted keyword search for E-healthcare cloud," *IEEE Trans. Depend. Sec. Comput.*, vol. 18, no. 3, pp. 1307–1319, Jun. 2021, doi: 10.1109/TDSC.2019.2916569.
[30] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, Oct. 2006, pp. 79–88, doi: 10.1145/1180405.1180417.
[31] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 3089. Berlin, Germany: Springer, 2004, pp. 31–45, doi: 10.1007/978-3-540-24852-1_3.
[32] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. 7th Int. Conf. Inf. Commun. Secur.*, vol. 3783, 2005, pp. 414–426, doi: 10.1007/11602897_35.
[33] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. 4th TCC*, vol. 4392. 2007, pp. 535–554, doi: 10.1007/978-3-540-70936-7_29.
[34] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 262–267, 2011, doi: 10.1016/j.jnca.2010.07.007.
[35] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 4965, N. Smart, Ed. Berlin, Germany: Springer, 2008, pp. 146–162, doi: 10.1007/978-3-540-78967-3_9.
[36] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Theory of Cryptography—TCC* (Lecture Notes in Computer Science), vol. 5444, O. Reingold, Ed. Berlin, Germany: Springer, 2009, pp. 457–473, doi: 10.1007/978-3-642-00457-5_27.
[37] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology—EUROCRYPT* (LNCS), vol. 6110, H. Gilbert, Ed. Berlin, Germany: Springer, 2010, pp. 62–91, doi: 10.1007/978-3-642-13190-5_4.
[38] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8042. Berlin, Germany: Springer, Aug. 2013, pp. 353–373, doi: 10.1007/978-3-642-40041-4_20.
[39] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Jan. 2016, doi: 10.1109/TPDS.2015.2401003.
[40] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837, doi: 10.1109/INFCOM.2011.5935306.
[41] H. Pang, J. Shen, and R. Krishnan, "Privacy-preserving similarity-based text retrieval," *ACM Trans. Internet Technol.*, vol. 10, no. 1, pp. 1–39, Feb. 2010, doi: 10.1145/1667067.1667071.
[42] C. Orencik, M. Kantarcioglu, and E. Savas, "A practical and secure multi-keyword search method over encrypted cloud data," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun. 2013, pp. 390–397, doi: 10.1109/CLOUD.2013.18.
[43] Q. Zhang, S. Fu, N. Jia, and M. Xu, "A verifiable and dynamic multi-keyword ranked search scheme over encrypted cloud data with accuracy improvement," in *Secur. Privacy Commun. Netw.*, vol. 254, pp. 588–604, Dec. 2018, doi: 10.1007/978-3-030-01701-9_32.
[44] I. H. Witten, A. Moffat, and T. C. Bell, "Managing gigabytes: Compressing and indexing documents and images," *IEEE Trans. Inf. Theory*, vol. 41, no. 6, p. 2101, Nov. 1999, doi: 10.1109/TIT.1995.476344.
[45] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 985–998, Nov./Dec. 2017, doi: 10.1109/TSC.2017.2757467.
[46] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Financial Cryptography and Data Security*, vol. 7859. Berlin, Germany: Springer, 2013, pp. 258–274, doi: 10.1007/978-3-642-39884-1_22.
[47] D. Comer, "Ubiquitous B-tree," *ACM Comput. Surv.*, vol. 11, no. 2, pp. 121–137, 1979, doi: 10.1145/356770.356776.
[48] H.-W. Fang, M.-Y. Yeh, P.-L. Suei, and T.-W. Kuo, "An adaptive endurance-aware B^+-tree for flash memory storage systems," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2661–2673, Nov. 2014, doi: 10.1109/TC.2013.158.
[49] J. Chen, K. He, L. Deng, Q. Yuan, R. Du, Y. Xiang, and J. Wu, "EliMFS: Achieving efficient, leakage-resilient, and multi-keyword fuzzy search on encrypted cloud data," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1072–1085, Nov. 2020, doi: 10.1109/TSC.2017.2765323.
[50] J. Sun, S. Hu, X. Nie, and J. Walker, "Efficient ranked multi-keyword retrieval with privacy protection for multiple data owners in cloud computing," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1728–1739, Jun. 2020, doi: 10.1109/JSYST.2019.2933346.
[51] Y. Yang, X. Liu, and R. H. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 2, pp. 320–334, Mar. 2020, doi: 10.1109/TDSC.2017.2787588.
[52] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 235–247, Feb. 2021, doi: 10.1109/TSC.2018.2791601.
[53] H. Wang, K. Fan, H. Li, and Y. Yang, "A dynamic and verifiable multi-keyword ranked search scheme in the P2P networking environment," *Peer Peer Netw. Appl.*, vol. 13, no. 6, pp. 2342–2355, May 2020, doi: 10.1007/s12083-020-00912-7.
[54] *20 Newsgroups*. Accessed: Aug. 2021. [Online]. Available: http://qwone.com/~jason/20Newsgroups/
[55] Y. Lu, J. Li, and F. Wang, "Pairing-free certificate-based searchable encryption supporting privacy-preserving keyword search function for IIoTs," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2696–2706, Apr. 2021, doi: 10.1109/TII.2020.3006474.
[56] Y. Miao, X. Liu, K.-K.-R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Depend. Sec. Comput.*, vol. 18, no. 3, pp. 1080–1094, May 2021, doi: 10.1109/TDSC.2019.2897675.
[57] 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./Oct. 2017, doi: 10.1109/TSC.2016.2542813.
[58] 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, doi: 10.1109/JIOT.2019.2943379.

HUANGLIN SHEN is currently pursuing the bachelor's degree with the Zhejiang University of Science and Technology. His research interests include information security and searchable encryption.

LINLIN XUE received the B.E. degree in electronic information engineering and the Ph.D. degree in electromagnetic field and microwave technology from the University of Science and Technology of China, Anhui, China, in 2008 and 2013, respectively.

She was a Lecturer with the Zhejiang University of Technology, from 2013 to 2019. Since 2019, she has been a Lecturer with the Zhejiang University of Science and Technology. She has been authored or coauthored over 20 journal articles and conference papers in her areas of expertise. Her current research interests include the areas of modeling and simulation of photonics devices and subsystems.



HAIJIANG WANG received the M.S. degree from Zhengzhou University, in 2013, and the Ph.D. degree from Shanghai Jiao Tong University, in 2018. He is currently a Teacher with the School of Information and Electronic Engineering, Zhejiang University of Science and Technology. His research interests include cryptography and information security, in particular public-key encryption, attribute-based encryption, and searchable encryption.

LEI ZHANG received the M.S. degree from Tsinghua University, in 2006. He is currently a Teacher with the School of Information and Electronic Engineering, Zhejiang University of Science and Technology. His research interests include the communication and its security in the Internet of Things.

JINYING ZHANG is currently pursuing the degree with the Zhejiang University of Science and Technology. Her research interest includes the application of blockchain in the medical field.