Research Article

Semantic-Based Multi-Keyword Ranked Search Schemes over Encrypted Cloud Data

Zheng Hu,1 Hua Dai,1,2 Geng Yang,1,2 Xun Yi,3 and Wenjie Sheng1

1School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
2Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing 210003, China
3Royal Melbourne Institute of Technology University, Melbourne 3001, Australia

Correspondence should be addressed to Hua Dai; daihua@njupt.edu.cn

Received 23 December 2021; Revised 22 January 2022; Accepted 17 March 2022; Published 29 April 2022

Copyright © 2022 Zheng Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Traditional searchable encryption schemes construct document vectors based on the term frequency-inverse document frequency (TF-IDF) model. Such vectors are not only high-dimensional and sparse but also ignore the semantic information of the documents. The Sentence Bidirectional Encoder Representations from Transformers (SBERT) model can be used to train vectors containing document semantic information to realize semantic-aware multi-keyword search. In this paper, we propose a privacy-preserving searchable encryption scheme based on the SBERT model. The SBERT model is used to train vectors containing the semantic information of documents, and these document vectors are then used as input to the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering algorithm. The HDBSCAN algorithm generates a soft cluster membership vector for each document. We treat each cluster as a topic, and the vector represents the probability that the document belongs to each topic. According to the clustering process in the schemes, the topic-term frequency-inverse topic frequency (TTF-ITF) model is proposed to generate keyword topic vectors. Through the SBERT model, searchable encryption scheme can achieve more precise semantic-aware keyword search. At the same time, the special index tree is used to improve search efficiency. The experimental results on real datasets prove the effectiveness of our scheme.

1. Introduction

Cloud computing is an Internet-based business computing model that provides fast and secure cloud computing services and data storage. Cloud computing uses the transmission capabilities of the Internet to transfer data storage and processing from local servers to the Internet. It is a product of the development and integration of traditional IT technologies such as distributed computing, parallel computing, network storage, and virtualization. Cloud computing distributes computing tasks on a resource pool composed of a large number of computers, enabling various application systems to obtain computing power, storage space, and various software services as needed. Cloud computing can solve the problems of enterprises’ dynamic demand for resources and storage costs, so more and more enterprises choose to outsource data to the cloud. Although cloud computing has many advantages, there are also some problems, such as data privacy issues. In order to protect the privacy of outsourced cloud data, the usual solution is to encrypt the data before outsourcing or use blockchain to achieve privacy protection [1]. However, the encrypted data hardly meet general application requirements, such as search. Therefore, research on searchable encryption schemes is necessary.

In recent years, researchers have done a lot of research on searchable encryption, such as searchable encryption for documents [2–40] and searchable encryption for images [41, 42]. The research in this paper focuses on searchable encryption of documents; many searchable encryption schemes have been proposed, such as single keyword search [2–10], multi-keyword search [11–26], fuzzy keyword search [27–31], conjunctive keyword search [32–35], and search based on semantic expansion [36–40]. Most of these
searchable encryption schemes are based on the term frequency-inverse document frequency (TF-IDF) model; they use secure KNN [11] to generate encrypted document vectors and encrypted indexes, and the data owner outsources the encrypted documents and indexes to the cloud. Once the search begins, a trapdoor for the query keyword is generated and used to perform the search using the encrypted index. In the search process, the smaller the inner product between a query vector and a document vector, the closer a query is to a document. However, the TF-IDF model is a statistical model used to evaluate the importance of keywords to a set of documents, which ignores the semantic association between keywords and documents, resulting in search results that do not meet the needs of users. At the same time, due to the large number of keywords in the dictionary, the vector generated by the TF-IDF model has an extremely high dimension and sparsity, so the existing search scheme has a large search time and space cost, but the effect is not satisfactory.

In order to enable the document vector to better express the semantic information of the document, we introduced the Bidirectional Encoder Representations from Transformers (BERT) model and the SBERT model. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right contexts in all layers. Therefore, it is only necessary to add an output layer to fine-tune the pretrained BERT model to create state-of-the-art models suitable for various tasks. A common method to solve clustering and semantic search is to map each sentence to a vector space so that semantically similar sentences are very close. Someone has tried to input the entire sentence into the BERT pretraining model to get the vector representation of the sentence. However, it is pointed out in Ref. [43] that the sentence vectors thus obtained do not have semantic information. In other words, two similar sentences may have very different sentence vectors. Aiming at the problem of the abovementioned pretraining model on the regression task of sentence pairs such as text semantic similarity, Ref. [44] proposed the SBERT model. The SBERT model uses Siamese and triplet network structures to obtain sentence embeddings with semantic information, and uses cosine similarity or Manhattan/Euclidean distance to compare and find semantically similar sentences.

In this paper, we proposed a semantic-aware searchable encryption scheme based on the SBERT model (SBERT-SRSE). The data owner first uses the SBERT model to train documents to get document embedding matrix $\Omega$. Then, $\Omega$ is used as the input of the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering algorithm to generate a soft membership vector for each document. We regard each cluster as a topic, so the document topic association matrix (DTA-matrix) can be obtained. We use Topic-Term Frequency-Inverse Topic Frequency (TTF-ITF) to get keyword topic association matrix (KTA-matrix). In order to further improve search efficiency, we construct a tree index. Finally, encrypt the documents and index tree and outsource to the cloud. When starting a multi-keyword ranked search to retrieve the $k$ most relevant documents, the data user converts the query keywords into a query topic vector, and encrypts the topic vector as a trapdoor. The cloud executes the search on the encrypted index tree and returns encrypted documents with the top-$k$ relevance scores as the search result. Our contributions are summarized as follows:

1. Based on the SBERT model, UMAP dimension reduction algorithm, and HDBSCAN clustering algorithm, we propose a document vectorization method that contains the topic semantic information of documents.

2. We propose the Topic-Term Frequency-Inverse Topic Frequency (TTF-ITF) model to measure the importance of keywords to topics, and generate keyword topic vectors according to this model.

3. We construct an index tree based on the semantic-related document list; the index further improves the efficiency of searches.

2. Related Work

In recent years, searchable encryption has been widely studied, and many searchable encryption schemes have been proposed, including single keyword searchable encryption schemes [2–10], multi-keyword searchable encryption schemes [11–24, 32–35], and multi-keyword searchable encryption schemes with semantic expansion [36–40]. We discussed the work related to the performance and functionality development of searchable encryption schemes.

2.1. Single Keyword Searchable Encryption Schemes. Song et al. [7] first proposed a single keyword searchable encryption scheme in 2000, in which the search time of the scheme is linearly related to the size of data collection. Goh [8] formally defined a secure index structure and designed an index scheme based on Bloom filter. However, because Bloom filters can have false positives, the cloud server may return documents that do not contain the search keywords. Due to the lack of a ranking mechanism for the searchable encryption scheme described above, it can take a long time for users to select what they want when a large number of documents contain queried keywords. Wang et al. [5] proposed a searchable single keyword ranked search encryption scheme in 2010, which is based on the TF-IDF model and uses inverted index. During the search, the cloud server computes relevance scores between documents and queries, sorts related documents according to their relevance score, and the user can get the top-$k$ results. Some other ranked search schemes in literature [3, 9]. The above searchable encryption scheme only supports single keyword search.

2.2. Multi-Key Word Searchable Encryption Schemes. The multi-keyword searchable encryption scheme allows users to submit multiple query keywords to retrieve related documents. Among them, the conjunctive keyword search schemes [32–35] only return documents containing all
query keywords, while the disjunctive keyword search scheme [23, 24] returns all documents containing subsets of query keywords. None of the above schemes support the ranking function of search results. Cao et al. [11] first proposed a private multi-keyword ranked search scheme in 2011, and Sun et al. [12] proposed a secure multi-keyword search scheme on the basis of Ref. [11]. They used TF-IDF model and cosine distance measure to construct MDB-tree [45], which improved the search efficiency. Xia et al. [20] proposed a multi-keyword ranked search scheme based on KBB tree, which supports the dynamic updating and deletion of documents. Chen et al. [16] and Zhu et al. [21] proposed an efficient ranked search scheme, and they both used clustering algorithm to improve search efficiency.

2.3. Multi-Key-W ard Searchable Encryption Schemes with Semantic Extension. Since the traditional searchable encryption schemes based on TF-IDF model do not consider the semantic association between keywords and documents, this may result in search results that do not match the users’ search intention. In order to overcome the lack of semantics, a searchable encryption scheme based on semantic extension was proposed. Xia et al. [36] implemented a multi-keyword searchable encryption scheme using inverted index and extended queried keywords and synonyms. Fu et al. [37] used synonym extensions of dictionary keywords to implement a multi-keyword searchable encryption scheme. Xia et al. [36] and Fu et al. [37] are based on the TF-IDF model, which is not a semantic model, so it is difficult for their scheme to improve the semantic accuracy of the search. Fu et al. [38, 39] adopted different semantic models, which required supervised learning of datasets, and could not be directly applied to different datasets. Dai et al. [40] first introduced the LDA topic model into searchable encryption. Through LDA topic model training, documents are transformed into topic vectors. Compared with the document features extracted by the TF-IDF model, the topic vector has a smaller dimension and can extract the semantic features contained in the document. When the search begins, the queried keywords are transformed into topic vector to better identify the users’ search intent.

3. Notations and Preliminaries

3.1. Notations

\( D \): The set of \( n \) plaintext documents, \( D = \{d_1, d_2, \ldots, d_n\} \).
\( \tilde{D} \): The set of \( n \) encrypted documents, \( \tilde{D} = \{\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_n\} \).
\( T \): The set of \( m \) topics, \( T = \{t_1, t_2, \ldots, t_m\} \).
\( W \): The dictionary contains \( h \) keywords, \( W = \{w_1, w_2, \ldots, w_h\} \).
\( \Omega \): The document vector matrix is obtained by SBERT training.
\( \Omega' \): The document vector matrix after dimension reduction.

\( \Phi \): The \( n \times m \)-dimensional document topic association matrix (DTA-matrix).
\( \Phi' \): The \( n \times m \)-dimensional encrypted DTA-matrix which is the encrypted index.
\( \Theta \): The \( h \times m \)-dimensional keyword topic association matrix (KTA-matrix).
\( P_D \): The \( m \)-dimensional topic probability vector.
\( P_W \): The \( h \)-dimensional keyword probability vector.
\( Q \): The ranked search with multiple queried keywords, \( Q = \{w_1, w_2, \ldots, w_{q}\} \subseteq W \), where \( w_i \) is the \( i \)-th keyword in \( W \).
\( V_Q \): The \( m \)-dimensional plaintext query vector.
\( \tilde{V}_Q \): The \( m \)-dimensional encrypted query vector of \( V_Q \).
\( TD \): The trapdoor of the query.

3.2. BERT Model and Sentence-BERT Model. Bidirectional Encoder Representations from Transformers (BERT) [43] has set a new state-of-the-art performance on sentence-pair regression tasks, such as semantic textual similarity (STS). BERT uses a cross-encoder, two sentences are passed to the transformer network and the target value is predicted. However, this setup is unsuitable for various pair regression tasks due to huge computational overhead. A common method to solve clustering and semantic search is to map each sentence to a vector space so that semantically similar sentences are close. Researchers have begun to input a single sentence into BERT and to derive fixed-size sentence embeddings. This approach produces quite bad sentence embeddings. Sentence-BERT (SBERT) adds a pooling operation to the output of BERT to generate a fixed-size sentence embedding vector. In Ref. [44], three pooling strategies are adopted for comparison: (1) Using the output of the CLS-token; (2) Computing the mean of all output vectors (MEAN-strategy); (3) Computing a max-over-time of the output vectors (MAX-strategy). Among of them, MEAN-strategy has the best effect. So in Ref. [44], MEAN-strategy is used as the default configuration.

In order to be able to fine-tune BERT/RoBERTa, SBERT uses the Siamese and triplet networks to update the weight parameters, so that the generated sentence vector has semantic information. The embedding vectors’ distance of sentences with similar semantics is closer, so it can be used for similarity calculation (Cosine similarity, Manhattan Distance, Euclidean Distance). When classification is used as the objective function, the structure of SBERT is shown in Figure 1.

3.3. Uniform Manifold Approximation and Projection. Uniform Manifold Approximation and Projection (UMAP) [46] is a novel manifold learning technique for dimension reduction. UMAP is constructed based on the theoretical framework of Riemannian geometry and algebraic topology, which makes it a scalable algorithm used for real-world data. The UMAP algorithm is competitive with the t-SNE algorithm in terms of visualization quality and retains more global structure with excellent runtime performance. In addition, UMAP has no computational limitations on the embedding dimension.
3.4. Hierarchical Density-Based Spatial Clustering of Applications with Noise. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based spatial clustering algorithm. The algorithm divides the regions with sufficient density into clusters, and finds clusters of arbitrary shapes in the noisy spatial database. It defines the clusters as the largest collection of densely connected points. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [47] extends DBSCAN to a hierarchical clustering algorithm. The difference from traditional DBSCAN is that HDBSCAN can handle clustering problems with different densities. HDBSCAN defines the clusters as the largest collection of densely connected regions with sufficient density into clusters, and finds clusters of arbitrary shapes in the noisy spatial database. It defines the clusters as the largest collection of densely connected points. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [47] extends DBSCAN to a hierarchical clustering algorithm. The difference from traditional DBSCAN is that HDBSCAN can handle clustering problems with different densities.

The mutual reach distance of two sample points is defined as follows:

\[ d_{mr}(a, b) = \max\{\text{core}_k(a), \text{core}_k(b), d(a, b)\}. \]  

HDBSCAN provides a method to generate soft cluster membership vector for each sample point. The method not only considers the distance from the point to the cluster but also considers the outlier value of the point relative to each other. The generated vector can be interpretable as a probability of being a member of that cluster.

3.5. Hadamard Product. Assuming that two matrices \( A, B \in \mathbb{R}^{m \times n} \) and \( A = [a_{ij}], B = [b_{ij}] \). The Hadamard Product between \( A \) and \( B \) is denoted as \( A \cdot B \), which is obtained by equation (3).

\[
A \cdot B = \begin{bmatrix}
a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\
a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn}
\end{bmatrix}, \tag{3}
\]

where \( \odot \) is the Hadamard operator. The result of \( A \cdot B \) is also a \( m \times n \)-dimensional matrix.

3.6. Relevance Score Measurement. The relevance score between a pair of documents or between a document and a ranked search is calculated by performing the inner product of the corresponding vectors. Assuming that \( p \) and \( q \) are two \( n \)-dimensional vectors, the relevance score between \( p \) and \( q \) is shown in equation (4),

\[
\text{Score}(p, q) = p \cdot q = \sum_{i=1}^{n} p[i] \times q[i]. \tag{4}
\]

In this paper, both the document vectors and the query vectors contain topic semantic information, thus the inner product of two vectors is a semantic relevance score between a document and a ranked search.

3.7. Secure Inner Product Operation. The secure inner product is implemented by a homomorphic matrix encryption scheme proposed in Ref. [11]. We assume that \( p \) and \( q \) are two \( n \)-dimensional vectors, and \( M \) is a random \( n \times n \) invertible matrix used as a secure key. The encrypted vectors of \( p \) and \( q \) are denoted as \( \overline{p} \) and \( \overline{q} \), respectively, where \( \overline{p} = M^T \cdot p \) and \( \overline{q} = M^{-1} \cdot q \). The inner product of \( p \) and \( q \) is equal to the inner product of \( \overline{p} \) and \( \overline{q} \), i.e., \( \overline{p} \cdot \overline{q} = p \cdot q \). The proof is shown in equation (5). It indicates that we can get the inner product of two vectors without knowing their plaintext.

\[
\overline{p} \cdot \overline{q} = (M^T p)^T \cdot (M^{-1} q) = p^T M M^{-1} q = p \cdot q. \tag{5}
\]

4. Model and Problem Statement

4.1. System Model. The system considered in this paper is the same as Refs. [14, 20]. As shown in Figure 2, the system model in this paper involves three different entities: data owner (DO), data user (DU), and cloud server (CS). They collaborate as follows:

1. DO owns a collection of documents \( D \). In our scheme, DO first applies the SBERT training [44] on \( D \) to generate the document embedding vectors \( \{\Omega[1], \Omega[2], \ldots, \Omega[n]\} \). DO uses the UMAP algorithm to reduce the dimensionality of the document embedding vectors to generate \( \{\Omega'[1], \Omega'[2], \ldots, \Omega'[n]\} \). According to the document embedding vectors \( \{\Omega'[1], \Omega'[2], \ldots, \Omega'[n]\} \), DO performs the HDBSCAN algorithm on \( D \) to generate a soft
mem

for each document, which is treated as the topic vector. These topic vectors form the document topic association matrix (DTA-matrix) \( \Phi \). Then, DO encrypts topic associations \( D \) and \( \Phi \) into ciphertext \( D \) and \( \Phi \). When DO receives a search result from CS, he/she decrypts the result with the secret key shared with DO to obtain the plaintext search result.

4.3. Security definition. The security definition of our scheme is realized through simulation games. For simplicity, \( \Pi \) is used to represent the scheme. \( \mathcal{A} \) and \( \mathcal{S} \) are the adversary and the simulator, respectively. Specifically, we designed two games involving PPT (Probabilistic Polynomial-Time) \( \mathcal{A} \), namely \( \text{Real}_\mathcal{A}(\lambda) \) and \( \text{Ideal}_\mathcal{A}(\lambda) \). The first game is the execution of the scheme, and the second game is the \( \mathcal{S} \) simulation using the leakage function \( \mathcal{F}_\Pi \) in the plan. If \( \mathcal{A} \) cannot distinguish between the output of \( \text{Real}_\mathcal{A}(\lambda) \) and \( \text{Ideal}_\mathcal{A}(\lambda) \), the scheme is \( \mathcal{F} \)-adaptively secure. The detailed description of the simulated games can be found in Ref. [48]. \( \mathcal{F}_\Pi \) is a set of leak functions whose formal definition is described in the Security Analysis section. The security definition is shown in Definition 1.

**Definition 1.** A scheme \( \Pi \) is \( \mathcal{F} \)-adaptively secure if for an adversary \( \mathcal{A} \), there exists a simulator \( \mathcal{S} \) such that

\[
| \text{Pr}[\text{Real}_\mathcal{A}(\lambda) = 1] - \text{Pr}[\text{Ideal}_\mathcal{A}(\lambda) = 1] | \leq \text{negl}(\lambda),
\]

where \( \text{negl}(\lambda) \) is a negligible function.

4.4. Design Goals. Given a search request \( Q \), the proposed ranked search scheme has the following design goals:

**Multi-keywords ranked search.** In this scheme, CS can return the \( k \) document that is the most relevant to the query and has the highest ranking in the form of ciphertext.

**Figure 2: System model.**
Search efficiency and effectiveness. The scheme realizes efficient and accurate search by using special tree-based index and efficient search algorithm.

Privacy-preserving. The scheme is designed to prevent CS from learning additional information about the document set, index tree, and query. Specific requirements are as follows:

- **Document confidentiality.** The plaintext of the documents should be protected.
- **Index confidentiality and query confidentiality.** The index and queried keywords should be protected.
- **Trapdoor unlinkability** [20]. CS should not determine whether two trapdoors are generated from the same query.
- **Keyword privacy.** CS should not identify specific keywords in documents or trapdoors.

5. Secure Search Scheme Based on the Sentence-BERT Model

In this section, we propose a semantic-aware multi-keyword ranked search scheme based on the SBERT model (SBERT-SRSE). We first present the framework of our scheme shown in Figure 3. Two modules are contained in the framework of SBERT-SRSE, the setup module and the search module, respectively. In the setup module, DO performs four algorithms to build the system, which are GenKey, GenMatrix, GenPB, and EncData. DO outsources the generated encrypted index and documents to CS, and shares the security keys and the necessary information for generating trapdoor with DU. When a ranked search is started, the search module will be activated between DU and CS. In the search module, GenTrapdoor is executed to generate trapdoor, SASearch is executed to return search results. We give the detailed description of the algorithms in the two modules in the following two subsections.

5.1. Algorithms in Setup Module. $SK \leftarrow \text{GenKey}(1^{\lambda(n)})$. DO generates the secured key $S K = [S, M_1, M_2, g]$, where $S$ is an $n$-bit random vector, $M_1$ and $M_2$ are two random $m \times m$ invertible matrices, and $g$ is the key for symmetric encryption. $S K$ is only shared between DO and DU while CS does not know the information.

$(\Phi, \Theta) \leftarrow \text{GenMatrix}(D)$. In order to vectorize the documents and reduce the semantic loss caused by the documents conversion process, DO uses the SBERT model to generate the document vector matrix $\Omega$. Then, DO generates the document topic association matrix (DTA-matrix) $\Phi$ and keyword topic association matrix (KTA-matrix) $\Theta$. The detailed steps are given as follows:

1. DO takes the document set $D$ as the input of the SBERT model, and the SBERT model returns the document vector matrix $\Omega$, where $\Omega[i]$ represents the document vector of the document $d_i$.

2. Before generating the DTA-matrix, DO uses the UMAP to reduce the dimension of document vectors, because the HDBSCAN clustering algorithm is prone to suffer the dimensional curse. At the same time, using the UMAP to reduce dimension can also bring some other benefits. On the one hand, it can reduce the computational complexity, reduce the amount of calculation, and reduce the amount of memory used. On the other hand, in the process of manifold dimension reduction, more local semantic features can be found.

3. DO uses the document vectors after dimension reduction $[\Omega' [1], \Omega' [2], \ldots, \Omega' [n]]$ as the input of the clustering algorithm HDBSCAN. A soft membership vector is generated for each document during the clustering process, we treat each cluster as a topic and finally DTA-matrix $\Phi$ is output. The matrix describes the relevance between documents and topics. Assuming that there are $n$ documents and $m$ topics in $D$ and $T$, respectively. $\Phi$ is a $n \times m$-dimensional matrix. The row vector $\Phi[i]$ in $\Phi$ is the document topic vector of $d_i$ and $\Phi[i][j]$ represents the probability that document $d_i$ belongs to topic $t_j$.

4. In order to be able to compare the importance of keywords between topics, we propose a model called Topic-Term Frequency-Inverse Topic Frequency. Definition 2. Topic-Term Frequency. The frequency of keyword $w_i$ in topic $t_j$ is denoted by $\text{ttf}(w_i, t_j)$.

$$\text{ttf}(w_i, t_j) = \frac{n_{i,j}}{\sum n_{k,j}}$$

where $n_{i,j}$ is the number of occurrences of keyword $w_i$ in topic $t_j$ and $\sum n_{k,j}$ is the total number of occurrences of all keywords in topic $t_j$.

Definition 2. Inverse Topic Frequency. Inverse topic frequency of $w_i$ is denoted by $\text{itf}(w_i)$.

$$\text{itf}(w_i) = \log \frac{|T|}{1 + \sum |\{j: w_i \in t_j\}|},$$

where $|\{j: w_i \in t_j\}|$ represents the number of topics containing the keyword $w_i$.

Definition 3. Topic-Term Frequency-Inverse Topic Frequency. The topic-term frequency-inverse topic frequency (TTF-ITF) score between $w_i$ and $t_j$ is denoted by $\text{ttf-itf}(w_i, t_j)$. It measures the importance of $w_i$ in $t_j$.

$$\text{ttf-itf}(w_i, t_j) = \text{ttf}(w_i, t_j) \cdot \text{itf}(w_i).$$

The higher the TTF-ITF value of $w_i$ in $t_j$ is, the higher the importance of $w_i$ in $t_j$. According to equation (11), the KTA-matrix is generated. The KTA-matrix $\Theta$ describes the relevance between keywords and topics. The row vector $\Theta[i]$ in $\Theta$ is the keyword topic vector of $w_i$ and $\Theta[i][j]$ represents the importance of the keyword $w_i$ to the topic $t_j$. 
\((P_D, P_W) \leftarrow \text{GenPB}(\Phi, \Theta)\). DO generates the topic probability vector \(P_D\) and the keyword probability vector \(P_W\) from \(\Phi\) and \(\Theta\).

(1) Assuming that each document \(d_i \in D\) is equally important, we can further derive the following probability:

\[
P_D = \frac{1}{n} \sum_{j=1}^{n} \Phi[j],
\]

where \(\Phi[j]\) is the \(j\)th row vector in \(\Phi\). \(P_D\) is the topic probability vector; each dimension of \(P_D\) stores the prior probability of the document \(d\) belonging to topic \(t \in T\).

(2) DO generates the keyword probability vector \(P_W\), each dimension of which records the prior belief of each keyword in \(W\) appearing in \(D\). The calculation of \(P_W\) is shown in equation (13).

\[
P_W = P_D \cdot \Theta^T,
\]

\((\tilde{D}, \Phi, \tilde{\Phi}, \text{SK}) \leftarrow \text{EncData}(D, \Phi, \text{SK})\). DO encrypts \(\Phi\) and \(D\) into \(\tilde{D}\) and \(\Phi\). DO first splits the vectors in \(\Phi\) with the \(n\)-bit random vector \(S\). Given a vector \(V\), \(\tilde{V}\) is split into two \(n\)-dimensional vectors \(\{V', V''\}\) following formula (14).

\[
\begin{align*}
V'[i] &= V''[i] = V[i], \quad S[i] = 0, \\
V'[i] + V''[i] &= V[i], \quad S[i] = 1.
\end{align*}
\]

Then, DO uses the secure matrices \(M_1\) and \(M_2\) in \(S\) to encrypt them into the ciphertext \(\tilde{V}\),

\[
\tilde{V} = [V', V''] = [V' \cdot M_1^T, V'' \cdot M_2^T].
\]

When all vectors in \(\Phi\) have been encrypted, the encrypted index \(\tilde{\Phi}\) is generated.

Finally, DO generates the encrypted documents set \(\tilde{D}\) with the key \(g\) in \(\text{SK}\).

After the setup module is finished, \(\tilde{\Phi}\) and \(\tilde{D}\) are outsourced to CS, where \(\tilde{\Phi}\) is used as the secure index for multi-keyword ranked searches. Meanwhile, DO shares \(\text{SK}, \Theta, P_D, P_W\) with DU.

5.2. Algorithms in Search Module. \(\tilde{V}_Q \leftarrow \text{GenTrapdoor}(Q, \Theta, P_D, P_W)\). When starting a ranked search \(Q = \{w_1, w_2, \ldots, w_p\}\) where \(w_p\) is the \(p\)th keyword in \(W\), DU generates the corresponding query topic vector \(V_Q\) and uses the security key to encrypt \(V_Q\) to get the trapdoor \(TD = (\tilde{V}_Q, k)\). We describe in detail the generation process of trapdoor as follows:

(1) DU generates an \(m\)-dimensional query topic vector from equation (16),

\[
V_Q = \sum_{w_i \in Q} \frac{P_D \cdot \Theta[i]}{|P_D| \cdot P_W[i]}.
\]

where \(V_Q[i]\) represents the probability that the topic intention of \(Q\) belongs to the topic \(t_i\).

(2) DU divides \(V_Q\) into two random vectors \(\{V_Q', V_Q''\}\) with the \(n\)-bit random vector \(S\) in \(S\). The split rule is shown in formula (17).

\[
\begin{align*}
V_Q'[i] + V_Q''[i] &= V_Q[i], \quad S[i] = 0, \\
V_Q'[i] &= V_Q''[i] = V_Q[i], \quad S[i] = 1.
\end{align*}
\]

(3) DU encrypts the divided \(V_Q = \{V_Q', V_Q''\}\) into \(\tilde{V}_Q = [\tilde{V}_Q', \tilde{V}_Q'']\) using secure matrices \(M_1\) and \(M_2\) in \(S\) to get the trapdoor \(TD = (\tilde{V}_Q, k)\), generated, it is submitted to CS as a search instruction.

\(R \leftarrow \text{SASearch}(\tilde{\Phi}, TD)\). CS receives the trapdoor \(TD\) and performs the secured ranked search by conducting the secure inner products between \(\tilde{V}_Q\) and the encrypted document topic vectors \(\{\Phi[1], \Phi[2], \ldots, \Phi[n]\}\). The topic semantic relevance scores between the query and the encrypted documents are calculated by equation (19),

\[
\begin{align*}
\text{Score} & \left[\{V_Q' \cdot M_1^{-1}, V_Q'' \cdot M_2^{-1}\}, \{\Phi'[i] \cdot M_1^{-1}, \Phi''[i] \cdot M_2^{-1}\}\right] \\
& = V_Q' \cdot M_1^{-1} \cdot (\Phi'[i] \cdot M_1^{-1})^T + V_Q'' \cdot M_2^{-1} \cdot (\Phi''[i] \cdot M_2^{-1})^T \\
& = V_Q' \cdot \Phi'[i] + V_Q'' \cdot \Phi''[i] \\
& = V_Q \cdot \Phi[i].
\end{align*}
\]

The encrypted documents having the highest \(k\) scores are obtained and returned to DU. After receiving the search result, DU uses the secure key \(g\) which is shared with DO to decrypt the encrypted documents to obtain the plaintext ranked search result.
6. Enhanced Search Scheme

The SASearch algorithm in SBERT-SRSE needs to perform the inner products in turn between the encrypted document topic vectors and the trapdoor, which has a linear time complexity. To improve the efficiency of the ranked searches, we first construct a semantic-related document list, and then use the method in Ref. [20] to construct the index tree TS-tree. We propose a TS-tree-based search algorithm at the same time. The enhanced scheme is denoted as SBERT-ESRSE.

6.1. Semantic-Related Document List. To construct the semantic-related document list, we classify each document into a certain topic. The classification is based on document topic vectors, where each dimension in the document topic vector represents the probability that the document is related to the topic. We classify the documents into topics that are most relevant to them. We traverse the topic \{t_1, t_2, \ldots, t_m\} and add the documents belonging to it to the list. When all topics are traversed, a semantic-related document topic list \(L = \{d_1, d_2, \ldots, d_n\}\) is generated. Usually in \(L\), the closer the distance between the two documents, the more similar their topic semantics.

6.2. TS-Tree-Based Index Construction Algorithm. In order to improve search efficiency, we use a binary tree to replace the original index. DO builds index tree based on semantic-related document list \(L\).

Definition 4. A node \(u\) of TS-tree is defined as a four-element tuple:

\[
u = \langle \text{id}, \text{pl}, \text{pr}, \text{d}_\text{vec} \rangle,
\]

where \(\text{id}\) is the identifier of the document, \(\text{pl}\) points to the left child of the node, \(\text{pr}\) points to the right child of the node, and \(\text{d}_\text{vec}\) is the vector stored in the node.

We give a detailed description of the different types of nodes in the TS-tree.

- If \(u\) is a leaf node, \(u.pl = u.pr = \emptyset\), \(u.id\) represents the corresponding document \(d_i\)'s id, \(u.d_{\text{vec}}\) represents \(d_i\)'s topic vector.
- If \(u\) is an internal node, \(u.pl\) and \(u.pr\) correspond to the left and right child nodes of \(u\), respectively. \(u.id = \emptyset\), \(u.d_{\text{vec}}\) represents the filtering vector of node \(u\), \(u.d_{\text{vec}}[i] = \max[u.pl.d_{\text{vec}}[i], u.pr.d_{\text{vec}}[i]], i = 1, 2, \ldots, m]\).

After using TS-tree as the index, the search efficiency is increased in proportion to the height of the tree in the tree index. The detailed algorithm of TS-tree construction is described in Algorithm 1. An example of TS-tree construction in the plaintext state is given in Figure 4.

6.3. TS-Tree-Based Search Algorithm. When CS receives a search request, CS performs the algorithm on the TS-tree to obtain the search result list \(R\) with the largest relevance scores. In addition, we need to keep \(R\) ranked in descending order and sorted when a new document is added. The relevance score calculation between trapdoor and encrypted document vectors is described in SBERT-SRSE. In the search process, CS uses the filtering threshold to reduce the number of inner product operations. Assuming that the filtering threshold is \(\zeta\) and the search request is \(V_Q\), when the search reaches node \(u\),

\[\begin{align*}
&\text{if } \zeta < \text{Score}(u.d_{\text{vec}}, V_Q), \text{ searching the node } u \text{ and the child nodes of } u. \\
&\text{if } \zeta > \text{Score}(u.d_{\text{vec}}, V_Q), \text{ filtering the node } u \text{ and the child nodes of } u.
\end{align*}\]

In addition, \(\zeta\) changes dynamically during searches, when searches reach the leaf node \(u\) of TS-tree.

The details of the search algorithm are described in Algorithm 2. An example of searching TS-tree in plaintext state is given in Figure 4.

6.4. Algorithms in SBERT-ESRSE. In SBERT-ESRSE, the algorithm BuildTSTree is added and the original algorithms, EncData and SASearch, are improved. These algorithms are introduced as follows:

\(I \longrightarrow \text{BuildTSTree}(\mathcal{L}, \Phi)\). After we obtain the DTA-matrix \(\Phi\) and the semantic-related document list \(\mathcal{L}\), we build the TS-tree as plaintext \(I\) on the basis of them; the detailed procedures are shown in Algorithm 1.

\[(\widetilde{D}, \widetilde{T}) \longrightarrow \text{EncData}(D, I, SK)\]. We use TS-tree \(I\) as the index to replace the original index \(\Phi\) and encrypt \(I\) with the similar encryption algorithm given as EncData\((D, \Phi, SK)\). The encrypted documents \(\widetilde{D}\) and the encrypted tree index \(\widetilde{T}\) are outsourced to CS.

\(R \longrightarrow \text{SASearch}(\widetilde{I}, T D)\). When CS receives a search trapdoor, CS performs a search on the encrypted index \(\widetilde{I}\) through Algorithm 2. The encrypted TS-tree is used as a search index to improve search efficiency. In the search process, the inner product operation between the trapdoor and the encrypted vectors is the same as \(\text{SASearch}(\Phi, T D)\), which has been given in SBERT-SRSE. The search result \(R\) is returned to DU. Then, DU decrypts documents with secure key \(g\) and gets the plaintext ranked search result.

7. Security Analysis

Before proving \(\mathcal{I}\) -adaptively secure of our schemes, we provide a formal description of the leakage functions for our schemes:

Definition 5. \(\mathcal{F}_\mathcal{I} = \{\mathcal{F}_{\text{GK}}, \mathcal{F}_{\text{Set}}, \mathcal{F}_{\text{BI}}, \mathcal{F}_{\text{GT}}, \mathcal{F}_{\text{SA}}\}\). The leakage functions in \(\mathcal{F}_\mathcal{I}\) corresponds to the secure key generation stage, initial setting stage, index generation stage, and trapdoor generation stage in the schemes of this paper, respectively. Specifically, the leakage are described as follows:

Leak\(_{\text{SP}}\) is the search pattern leakage, which reflects the same results obtained by the same queries.
**Theorem 1.** The proposed schemes, SBERT-SRSE and SBERT-ESRSE, are $\Phi$-adaptively secure with the leakage functions set $\Phi_{II}$.  

**Proof.** We adopt a method based on simulation to prove the theorem. In the ideal game, $\text{Ideal}_{\Phi_{II}}^{\Pi}(\lambda)$, the simulator $\delta$ simulates the steps in SBERT-SRSE and SBERT-ESRSE according to the given leak functions in $\Phi_{II}$. If adversary $A$ cannot distinguish between the output of the real game and the ideal game, we can conclude that for $A$ given $\Phi_{II}$, the proposed schemes are $\Phi$-adaptively secure.
According to the leakages Leak SP, Leak AP, and Leak SIP, the security of $\mathcal{Q}$ stores the pair $(\Phi, \Sigma)$ simulator topic vectors and encrypted document topic vectors that support secure inner product operations, that is, the random invertible matrices $M_1$ and $M_2$ in key $SK$ are used to generate encryption vectors according to (15) and (18). Therefore, the advantage that $A$ can distinguish $G_0$ and $G_1$ is equivalent to the advantage that $A$ can crack the encryption process shown in formula (23), that is, in the known $V^\prime$ and $V^\prime\prime$, by solving formula (23), the advantages of the exact solutions of $M_1, M_2, V^\prime$, and $V^\prime\prime$ in the corresponding equations are obtained.

$$\begin{align*}
&\left\{ V^\prime \times M_1 = V^\prime, \\
&V^\prime\prime \times M_2 = V^\prime\prime, 
\right.
\end{align*}$$

In the equations corresponding to formula (23), $M_1$ and $M_2$ are $m \times m$-dimensional random invertible matrices, $V^\prime$ and $V^\prime\prime$ are $m$-dimensional random vectors obtained from random vector $S$, $V^\prime$ and $V^\prime\prime$ are $m$-dimensional encryption vectors, so the equations have $2m \times (m+1)$ unknowns and $2m$ known numbers. Obviously, in polynomial time, the exact solutions of $M_1, M_2, V^\prime$, and $V^\prime\prime$ cannot be obtained in these equations. This means that the advantage of being able to distinguish between random numbers and secure inner product operations is negligible, thus

$$\left| \Pr[G_0^d = 1] - \Pr[G_1^d = 1] \right| \leq \text{negl}(\lambda).$$

According to formulas (23), (21), (22), and (24), combined with Definition 1, it can be derived:

$$\left| \Pr[\text{Real}^d_G(\lambda) = 1] - \Pr[\text{Ideal}^d_{G,\delta}(\lambda) = 1] \right| \leq \text{negl}(\lambda).$$

According to the security definition of the scheme in this paper (Definition 1), based on the leaked function set $\mathcal{F}$ in Definition 5, the schemes proposed in this paper are $\mathcal{F}$-adaptively secure. □
8. Performance Analysis

In this section, we evaluate the proposed schemes SBERT-SRSE and SBERT-ESRSE and compare them with the schemes presented in Refs. [20, 40], which are denoted as TFIDF-MRSE and LDA-EMRSE. We implement these schemes by using Python 3.6 in Windows 10 with Intel Core(TM) i5-9300H. The assessment is based on two indicators. One is the semantic precision of search results; the other is the cost of search time. The data set used in the evaluation was 20 newsgroups [49], including 20 different categories of news, including a total of 11315 articles.

Default parameters are summarized in Table 1, where \( n,|Q|,\) and \( k \) are the number of documents, the number of queried keywords, and required documents, respectively.

At the same time, in order to make the number of topics trained by the LDA topic model closer to the number of real topics, we set the number of topics to 20.

8.1. Semantic Precision Evaluation. In order to reasonably describe the semantic precision of the scheme, we assume that the query keywords come from the same category and the documents in different categories are semantically irrelevant. We evaluate the search precision of SBERT-SRSE, SBERT-ESRSE, LDA-EMRSE, and TFIDF-MRSE using the method adopted by Ref. [50], which is shown as follows:

\[
P = \frac{TP}{TP + FP} \times 100\%,
\]

where TP and FP indicate the numbers of documents in the search result that belong or do not belong to the category of the search intent, respectively.

(1) Semantic Precision versus \( n \). As shown in Figure 5, with the increase of the number of documents \( n \), the semantic precision of SBERT-SRSE and LDA-EMRSE remained stable, while that of TFIDF-MRSE gradually decreased. The reason is that in SBERT-SRSE and LDA-EMRSE, semantic models are used to extract the semantic features of the documents. As the number of documents increases, the topics extracted by the semantic models better reflect the semantic features of the documents, and have no significant impact on the semantic accuracy of the search. At the same time, because the document vectors trained by the SBERT model have more semantic features than those trained by the LDA model, the semantic precision of SBERT-SRSE is higher than that of the semantic precision of LDA-EMRSE.

(2) Semantic Precision versus \( k \). Figure 5(b) shows that the semantic precision of the three schemes gradually decreases as the number of documents required \( k \) increases. The reason for this is that as \( k \) increases, documents that are less relevant to the search intent are added to the search results, which may result in a slight decrease in the semantic precisions.

(3) Semantic Precision versus \( |Q| \). Figure 5(c) shows that as the number of queried keywords \( |Q| \) increases, semantic precisions of SBERT-SRSE, LDA-EMRSE, and TFIDF-MRSE are increased gradually, and then tend to be stable. The reason is that in a multi-keyword search, the more keywords that represent the semantic features of the query, the more the returned document reflects the latent semantics of the query. However, when the queried keywords can well represent the semantic features of the query, increasing the number of queried keywords has little impact on the search results. At the same time, the semantic precision of SBERT-SRSE is the highest, LDA-EMRSE is the second, and TFIDF-MRSE is the lowest.

8.2. Search Time Cost Evaluation. In this section, we evaluate the time cost of the search algorithm in SBERT-SRSE, SBERT-ESRSE, LDA-EMRSE, and TFIDF-MRSE.

(1) Time Cost versus \( n \). Figures 6(a) and 6(b) show that the search time costs of the four schemes increase with the increase of \( n \). Among them, the reason for SBERT-SRSE is that the search process of SBERT-SRSE requires traversing all documents and obtaining the search result. With the increase of \( n \), the number of documents grows, so the search time increases. The reason why the other three schemes increase in time is that they all need to selectively traverse document nodes. The larger the \( n \) is, the more document nodes are traversed, which leads to the increase of search time cost.

(2) Time Cost versus \( k \). Figures 7(a) and 7(b) show that as the number of required documents \( k \) grows, the search time cost of SBERT-SRSE remains stable, while the other three schemes increases. Since SBERT-SRSE needs to traverse all documents to obtain the search result, no matter how \( k \) changes, the search time cost of SBERT-SRSE changes little. The reason for the increase in the search time cost of the other three schemes is that as \( k \) grows, more document nodes need to be traversed to find these incremental required documents.

(3) Time Cost versus \( |Q| \). Figures 8(a) and 8(b) show that with the growth of \( |Q| \), the search time costs of four schemes all remain stable. This means that the number of queried keywords has little effect on the search time overhead. The reason for this is that no matter how many keywords are queried, in each of the four schemes, they are always transformed into vectors of fixed dimension, respectively.

In Figures 6(a), 7(a), and 8(a), the evaluation results all indicate that the search time cost of TFIDF-MRSE is significantly higher than that of the other schemes. This is because the dimension of the vector in TFIDF-MRSE is equivalent to the size of the keyword dictionary, which is much larger than the scale of the other three schemes. Figures 6(b), 7(b), and 8(b) all show that in the other three
Figure 5: Semantic precision versus $n$, $k$, and $|Q|$. 

Figure 6: Time cost versus $n$, (b) is the detail of (a).
schemes, SBERT-SRSE has the highest search time cost, LDA-EMRSE is the second, and SBERT-ESRSE is the lowest. The reason is that SBERT-SRSE needs to traverse all the documents in the search process, while SBERT-ESRSE and LDA-EMRSE both use binary-tree-based index to search. Meanwhile, SBERT-ESRSE arranges the documents into leaf nodes according to the results of document clustering, in which the documents of adjacent leaf nodes have a high semantic correlation. So SBERT-ESRSE traverses fewer document nodes than LDA-EMRSE, requiring less search time.

9. Conclusion
The searchable encryption scheme over cloud data has been a popular and challenging research issue. This paper introduces the SBERT model into searchable encryption, and proposes a semantic-based multi-keyword ranked search scheme for encrypted cloud data. Documents are trained as vectors containing semantic information, and then the documents are represented as vectors with topic relationships through HDBSCAN soft clustering. At the same time, specific keywords in the search are replaced by topic relevance. This solves the problem of semantic ignorance or unclear semantics in traditional searchable encryption schemes. The realization of the tree-based index further improves the search time. The results show that the scheme is an efficient and privacy-protected semantic-based multi-keyword ranked search scheme over encrypted cloud data.

Data Availability
The data that support the findings of this study are available from the corresponding author, Hua Dai, upon reasonable request.
Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under the Grant Nos. 61782197, 61972209, 61902199, 31671006, and 61771251; the Post-doctoral Science Foundation of China under the Grant No. 2019M651919; and the Natural Research Foundation of NJUPT under the Grant Nos. NY217119 and NY219142.

References


