

CASE-SSE: Context-Aware Semantically Extensible Searchable Symmetric Encryption for Encrypted Cloud Data

Lanxiang Chen¹, Yujie Xue, Yi Mu², Senior Member, IEEE, Lingfang Zeng¹, Member, IEEE, Fatemeh Rezaeibagha³, Member, IEEE, and Robert H. Deng¹, Fellow, IEEE

Abstract—Traditional searchable symmetric encryption (SSE) schemes rarely support context-aware semantic extension, and then lead to the searched results being incomplete or deviating from the user's query intention. To address this problem, a new context-aware semantically extensible searchable symmetric encryption based on Word2vec model (CASE-SSE) is proposed to achieve context-aware semantic extension in this article. The proposed scheme utilizes outsourced datasets as corpora to extract all keywords for training the Word2vec model, and the trained results is the ontology knowledge base that can be used to extend the semantics of query keywords directly. Further, to facilitate multi-keyword search using the extended query vector, we use the k -means clustering algorithm to classify outsourced datasets. We then construct an AVL-tree index and an inverted index based on the classified results, thereby achieving efficient context-aware semantically extensible SSE. The security analysis indicates it is secure and effective. The experimental results show that our scheme is superior in both efficiency and accuracy.

Index Terms—Searchable symmetric encryption, Context aware, Word2vec model, Semantically extensible, k -means clustering

1 INTRODUCTION

To reduce the management cost of large amount of data, more and more individuals and enterprises outsource their data to the public cloud. To ensure privacy and security of the outsourced data, they usually encrypt their data before outsourcing them to the cloud. But how to search on these encrypted data becomes an urgent problem to be solved. As there are massive data to be outsourced, they are often encrypted by symmetric

encryption algorithms. Therefore, Song *et al.* [1] first proposed the notion of searchable symmetric encryption (SSE) in 2000. Since then many SSE schemes have emerged in the past two decades. However, the majority of existing SSE schemes only return results according to the query keywords, and do not take the context or the semantic extension into account. Hence, it leads to the incomplete returned results and limit the accuracy of searchable symmetric encryption.

Different from these schemes, we propose to extend the semantics of query keywords according to document context utilizing the Word2vec model [2] trained from the original datasets, in which all keywords are represented in the form of distributed word vectors, to accurately measure the semantic distance between keywords. We summarize our brief contributions as follows.

- Lanxiang Chen is with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China, and also with Zhejiang Lab, Hangzhou 311121, China. E-mail: lxiangchen@fjnu.edu.cn.
- Yujie Xue is with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China. E-mail: xyj19970628@163.com.
- Yi Mu is with the Faculty of Data Science, City University of Macau, Macao SAR 999078, China. E-mail: ymu.ieee@gmail.com.
- Lingfang Zeng is with Zhejiang Lab, Hangzhou 311121, China. E-mail: zenglf@zhejianglab.com.
- Fatemeh Rezaeibagha is with the Discipline of Information Technology, Murdoch University, Murdoch, WA 6150, Australia. E-mail: Fatemeh.Rezaeibagha@murdoch.edu.au.
- Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: robertdeng@smu.edu.sg.

Manuscript received 26 July 2021; revised 19 Jan. 2022; accepted 22 Mar. 2022. Date of publication 25 Mar. 2022; date of current version 10 Apr. 2023.

This work was supported in part by the National Natural Science Foundation of China under Grants 62072105, 61872087 and U1805263, in part by the Natural Science Foundation of Fujian Province under Grant 2019J01274, in part by Zhejiang provincial "Ten Thousand Talents Program" under Grant 2021R52007, and in part by the Center-initiated Research Project of Zhejiang Lab under Grant 2021DA0AM01.

(Corresponding author: Lanxiang Chen.)

Digital Object Identifier no. 10.1109/TSC.2022.3162266

- A context-aware semantically extensible searchable symmetric encryption based on Word2vec model (CASE-SSE) is proposed in this paper. The proposed scheme takes the outsourced dataset as a training set to train the Word2vec model to generate a distributed word representation of the original dataset.
- The advantages of the proposed solution are two-fold. It uses the original dataset itself to construct ontology knowledge base that ensures the accuracy of returned results. It therefore will definitely return correct results if the semantics of the query keyword is similar. Moreover, it trains the context of its own dataset so that it can be semantically extended, otherwise it can only be regarded as synonyms or homonyms, but not semantics.

- To improve the search efficiency, a two-layer index structure is proposed. It first utilizes k -means [3] clustering algorithm to classify the original dataset and then constructs an AVL-tree index based on the keyword-category. After that, an inverted index is constructed based on category-document. On the one hand, the keyword-category index reduces the dimension of the index node, which improves the search efficiency. On the other hand, the two-layer index mechanism based on the AVL balanced binary tree further improves the search efficiency.
- We show that CASE-SSE is secure under the known ciphertext threat model and the known background threat model as defined in [4].
- We compare the proposed scheme with the traditional SSE schemes and experimental results demonstrate that our scheme is superior in both efficiency and accuracy.

The remainder of this paper is arranged as follows. Section 2 presents related work of SSE and word vector model. Section 3 gives notations and preliminaries. Section 4 introduces the Word2vec model. Section 5 presents the construction of CASE-SSE in detail. Sections 6 and 7 give the security analysis and performance evaluation, respectively.

2 RELATED WORK

2.1 Single Keyword SSE

In 2000, Song *et al.* [1] first proposed a practical SSE scheme in which keyword search is performed by sequential scanning, and it needs to scan all documents to complete each query. Curtmola *et al.* [5] proposed two SSE schemes based on inverted index for the first time. Since then a large number of SSE schemes based on inverted index have been proposed.

In 2007, Swaminathan *et al.* [6] introduced order preserving encryption to achieve the ranking of returned results and improve efficiency. Following this, Wang *et al.* [7] proposed an order-preserving symmetric encryption algorithm (OPSE) to support ranked keyword search. Wang *et al.* [8] then proposed a top- k SSE scheme by calculating the correlation scores between documents and keywords. Recently, some attacks [9] have shown that users' query keywords can be recovered by using access and search pattern. Therefore, how to protect access and search pattern has attracted the attention of researchers.

2.2 Multi-Keyword SSE

Due to poor search accuracy, single keyword SSE schemes often return many unrelated files, which greatly increases users' storage and computing cost. In order to improve search efficiency and accuracy of SSE, several multi-keyword SSE schemes are proposed.

In 2011, Cao *et al.* [4] proposed a multi-keyword SSE scheme based on inner product similarity. Then, Sun *et al.* [10] proposed a multi-keyword ranked search scheme using vector space model and multi-dimensional B-tree. Wang *et al.* [11] proposed a privacy-preserving multi-keyword fuzzy search SSE scheme based on binary vector and local sensitive hash (LSH) function. Later, Fu *et al.* [12] proposed a multi-keyword ranked fuzzy search scheme to improve

the efficiency of the scheme of Wang *et al.* [11]. Further, Ding *et al.* [13] proposed a tree-index based SSE scheme and designed a random traversal algorithm to realize a privacy-preserving multi-keyword SSE scheme. Even for the same keyword, it will generate a different access path. Peng *et al.* [14] proposed a multi-keyword ranked SSE scheme to support multi-user scenarios based on the tree index and TF-IDF model. However, most of the proposed SSE schemes rarely consider the context of documents and the semantics of queried keywords.

2.3 Semantic Extensible SSE

In 2014, Xia *et al.* [15] and Fu *et al.* [16] both proposed to use synonyms to extend the semantics of keywords for SSE, but they both used public thesaurus to achieve synonym extension. Different from their schemes, we train the Word2vec model using outsourced datasets as corpora and then utilize the trained results as the ontology knowledge base to directly extend the semantics of query keywords. The advantage of our scheme is that it ensures the accuracy of search results.

In 2016, Chen *et al.* [17] proposed a hierarchical clustering method to improve search efficiency and achieve semantics to some extent. Then, Fu *et al.* [18] proposed the concept of hierarchy and used the semantic relationship between concepts to search on encrypted datasets. Their solution uses one cloud server to store the outsourced datasets and one cloud server to evaluate the similarity scores between documents and queries and sends the scores to the former server. Their solution uses a supervised learning model which may be not suitable for some applications. Recently, Dai *et al.* [19] proposed a Latent Dirichlet Allocation (LDA) topic model based SSE scheme. They used the LDA model to train the original datasets and generate a document-topic relevance matrix as the index for their scheme.

Unlike the ideas of these schemes described above, first, our proposed scheme uses the original dataset itself to construct ontology knowledge base but not the synonym or the homonym dictionary that ensures the accuracy of search results. Second, our proposed scheme extends the semantics of query keywords based on the trained Word2vec model to achieve a semantically extensible search, but not extends the semantics of index, which will greatly reduce the cost of semantic extension. Third, we propose a two-layer index structure based on k -means [3] clustering algorithm to constructs an AVL balanced binary tree index based on keyword-category and an inverted index based on category-document, which improves the search efficiency.

2.4 Word Vector Model

The most commonly used word representation is one-hot representation [20] which sets each word as a specific and unique vector index. It sets the value of corresponding position of a word to 1, and the rest of the positions to 0. As there are a large number of words, the one-hot representation will result in a dimensional disaster, more seriously, there are lots of 0 and only one 1 which cannot reveal the internal relationship between words.

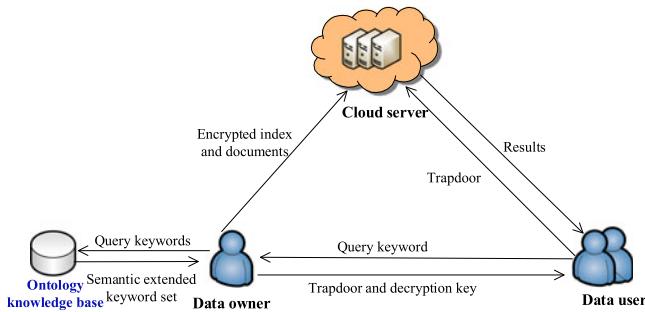


Fig. 1. System model.

Initially, in 1986, Rumelhart *et al.* [21] proposed to map words to a lower-dimensional real-number vector space according to the size of the word set. This method solves the dimensional disaster caused by high-dimensional sparse vectors and converts the similarity between words into spatial distance for correlation measurement. Thus, it meets the requirement of word vector representation for large scale documents. In 2013, Mikolov *et al.* [2] first introduced the skip-gram model to train and obtain Word2vec model as an efficient method to learn word vector representation from massive unstructured data. This model training does not involve intensive matrix multiplication that makes the model very fast. Later, Mikolov *et al.* [22] extended the original skip-gram model and proposed to resample high-frequency words during training that can significantly improve the efficiency (about 2-10 times) and the accuracy of low-frequency word representation. In addition, they proposed a simplified variant algorithm of noise contrast estimation (NCE) [23] to train the skip grammar model. In our proposed scheme, we introduce the Word2vec model to the SSE scheme for the first time to achieve context-aware semantically extensible search.

3 NOTATIONS AND PRELIMINARIES

3.1 Notation

- n : the number of documents.
- D : the plaintext collection, denoted as a set of n documents $D = \{D_1, D_2, \dots, D_n\}$.
- C : the ciphertext collection, stored in the cloud server and denoted as $C = \{C_1, C_2, \dots, C_n\}$.
- k : number of categories.
- $\vec{\mu}$: document classification result $\vec{\mu} = \{\vec{\mu}_1, \vec{\mu}_2, \dots, \vec{\mu}_k\}$.
- \vec{V} : category distribution of keywords $\vec{V} = \{\vec{V}_1, \vec{V}_2, \dots, \vec{V}_k\}$.
- \vec{I}_1 : keywords-category secure AVL-tree index.
- \vec{I}_2 : category-documents secure index.
- K : the collection of keys.
- λ : security parameter.
- Q : query keywords.
- V_{TD} : query trapdoor.
- S : a random binary vector.
- ε, ρ : two random integers greater than 0.
- W : keywords dictionary of all documents, $W = \{W_1, W_2, \dots, W_n\}$.
- m : number of keywords in W .

3.2 System Model

In our scheme, there are three entities, i.e., data owner, data user and cloud server. The system model is presented in Fig. 1.

- *Data owner*: Data owner outsources a collection of documents D to the cloud server. It first extracts keywords set W from document set D , and then it constructs two encrypted searchable indices \vec{I}_1 and \vec{I}_2 based on D . Each file in D is encrypted to output a ciphertext set C . The two-layer index and the set C are outsourced to the cloud server. Finally, the owner generates the search trapdoor V_{TD} according to the semantically extended keywords.
- *Cloud server*: Cloud server stores encrypted documents and searchable indices. It searches over the secure index and returns encrypted documents to data users. It should protect the encrypted documents and searchable indices from attackers.
- *Data user*: The data user is someone who is authorized to access the encrypted files of the data owner. In order to search on these encrypted files, each data user takes a keyword Q as input, then a query trapdoor V_{TD} is generated according to the Word2vec model and the decryption key K is sent to the data user in a secure way. Then the data user decrypts the most relevant encrypted documents returned by the cloud server.

In the system model, the ontology knowledge base is the training results of the Word2vec model.

3.3 Threat Model

In the proposed scheme, the cloud server is considered to be “semi-honest but curious”. It will perform all tasks as pre-defined instructions, but it is curious about the queries and encrypted data. According to the different levels of information held by the cloud server, there are mainly two security models [4] defined as follows.

(1) *Known Ciphertext Model*. In this model, the cloud server knows the collection of encrypted documents C , the encrypted indices \vec{I}_1, \vec{I}_2 and trapdoor V_{TD} .

(2) *Known Background Model*. In this model, the cloud server knows all related data in the known ciphertext model and it also can record search results, analyzes the relationship among different trapdoors and gets related statistical information. For example, the cloud server can use the word frequency information to obtain the frequency distribution of each keyword in the document set, and further launch some attacks based on the background knowledge.

3.4 Design Goals

Index Security. In our scheme, we propose a two-layer index structure. It first uses the k -means [3] clustering algorithm to classify the original dataset and then constructs an AVL-tree index based on keyword-category. After that, an inverted index is constructed based on category-document. To ensure the security of our two-layer index, it is encrypted before storing to cloud server.

Data Confidentiality. All data are encrypted before outsourcing to the cloud server.

Trapdoor Security. First, only the authorized users can get a trapdoor to query the cloud server. Second, all private information of trapdoors are encrypted in the scheme. It mainly needs to ensure that the trapdoor will not leak any information about index and keywords to anyone else. At the same time, the cloud server cannot identify whether two trapdoors are generated from the same keyword.

4 WORD2VEC MODEL

4.1 Vector Representation of Words

In order to facilitate natural language processing (NLP), one method is to vectorize all words. There are usually two kinds of methods for vector representation of words.

4.1.1 One-Hot Representation

One-hot representation [20] is used commonly in the traditional language representation. It first divides the original document into a collection of words, then removes punctuations and stop-words and gets a vocabulary list. It represents each word as a fixed-length vector which has the same size with the vocabulary list. Take for an example, “Today is a good day, and everyone is in a good mood.”, the vocabulary list is {"today", "good", "day", "everybody", "mood"}, and the corresponding vector subscript is “0: today, 1: good, 2: day, 3: everybody, 4: mood”. Thus the vector representation of “mood” is [0, 0, 0, 0, 1]. Based on the one-hot encoder, sentences or texts can be easily transformed into the bag-of-words model, TF-IDF model, N-gram model and other discrete representations [20] model. Combining these representation methods with the maximum entropy model, SVM, CRF and other machine learning algorithms, we can perform many natural language processing tasks.

However, there are two fatal weaknesses of one-hot representation. On the one hand, it loses the semantics or context information among words. For example, “PC” and “computer” will be regarded as two different words without revealing the semantic relationship between them. In order to perform these tasks at the semantic level, it must introduce extra synonym dictionary. On the other hand, as there are a large number of words, the dimension of the vector will be very large and result in dimensional disaster. Meanwhile, the vector is high-dimensional and sparse that will cause an impractical memory overhead.

4.1.2 Distributed Representation

To conquer the shortcomings of one-hot representation described above, distributed representation is proposed to denote a word as a point in the continuous semantic space, which was first proposed by Rumelhart *et al.* [21] in 1986. It is a low-dimensional vector representation composed of real numbers. The dimension of the vector can be set freely, but it is more common to use 50-200 dimensions.

Distributed representation not only solves the dimensional disaster, but also reveals the semantic distance between words. Further, to measure the similarity among keywords, all words are mapped to a low-dimensional



Fig. 2. Illustration of negative sampling.

vector space. In this way, the semantic distance between any two words can be easily calculated by computing euclidean or cosine distance. The more similar the words in semantics, the closer the distance. For example, the distance between “PC” and “computer” in this space will be closer than that between “computer” and “bread”.

4.2 Word2vec Model

The Word2vec model was proposed by Tomas Mikolov [2], [22] in 2013 when he worked for Google. It has become a third-party open source toolkit of Python and is very popular in recent years. In this paper, the Word2vec model adopts distributed representation to represent all words. We implement the Word2vec model based on the negative sampling algorithm [22], mainly composed of CBOW (Continuous Bag-of-Words) and Skip-Gram model.

4.2.1 Negative Sampling Algorithm

In the Word2vec model, a crucial step is to predict the word w from $\text{Context}(w)$, the known context of w . For a given $\text{Context}(w)$, the word w is a positive sample and the other words are negative samples. Due to the large number of negative samples, the negative sampling algorithm is used for sampling in the Word2vec model. The sampling algorithm is essentially a weighted sampling process [2] described in detail as follows.

- First of all, we declare that each word w in the dictionary W corresponds to a line segment $l(w) = \frac{\text{counter}(w)}{\sum_{u \in W} \text{counter}(u)}$. Here $\text{counter}(\cdot)$ indicates the number of times that a word appears in corpus D (the denominator indicated by normalization), and W represents the dictionary of D .
- Second, let $l_0 = 0, l_k = \sum_{j=1}^k l(w_j)$ ($k = 1, 2, \dots, N$). Here w_j represents the j^{th} word in dictionary W .
- Third, taking $\{l_j\}_{j=0}^N$ as the split nodes that can get non-equidistant parts on the interval $[0, 1]$. $I_i = (l_{i-1}, l_i]$ denotes its N partitions where $i = 1, 2, \dots, N$ as illustrated in Fig. 2.
- Fourth, introducing equidistant divisions on the interval $[0, 1]$, the division nodes are $\{m_j\}_{j=0}^M$ where $M > > N$ as shown in Fig. 2.
- Finally, the internal division points $\{m_j\}_{j=0}^M$ are projected to the non-equidistant divisions, as shown by the dashed line in Fig. 2. In this way, the map relationship between points $\{m_j\}_{j=0}^M$ and interval $\{l_j\}_{j=0}^N$ (or $\{w_j\}_{j=0}^N$) can be established, $\text{NEG}(w_i) = w_k$, where $m_i \in I_k$ ($i = 1, 2, \dots, M-1$).

Using this map relationship for sampling, the process is simple. When a random integer r in the interval $[1, M-1]$ is generated, $\text{NEG}(w_r)$ is a negative sample.

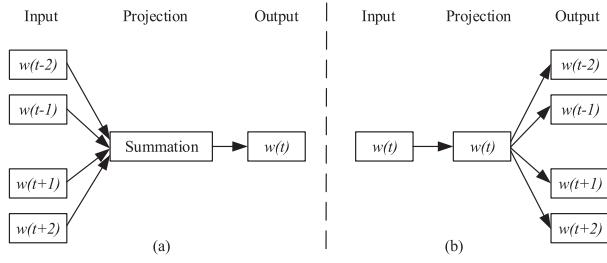


Fig. 3. (a) CBOW model; (b) Skip-gram model.

4.2.2 CBOW and Skip-Gram Model

The CBOW model and the skip-gram model contain three layers: input layer, projection layer, and output layer. The CBOW model is used to predict the current word w_t on the premise that the context $Context(w_t) = \{w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\}$ of w_t is known, as shown in Fig. 3a. On the contrary, the skip-gram is used to predict the context of w_t on the premise that the current word w_t is known, as shown in Fig. 3b.

Algorithm 1. CBOW Model

Require: The current word w and its context $Context(w)$.

Ensure: The updated parameter θ .

```

1: e = 0;
2:  $x_w = \sum_{u \in Context(w)} v(u)$ ;
3: for  $u \in \{w\} \cup NEG(w)$  do
4:    $q = \sigma(x_w^\top \theta^u)$ ;
5:    $g = \eta(L^w(u) - q)$ ;
6:    $e := e + g\theta^u$ ;
7:    $\theta^u := \theta^u + gx_w$ ;
8: end for
9: for  $u \in Context(w)$  do
10:   $v(u) := v(u) + e$ ;
11: end for
12: Return  $\theta = \theta^u$ .

```

Algorithm 2. Skip-Gram Model

Require: The current word w and its context $Context(w)$.

Ensure: The updated parameter θ .

```

1: for  $\tilde{w} \in Context(w)$  do
2:   e = 0;
3:   for  $u \in \{w\} \cup NEG(w)$  do
4:      $q = \sigma(v(\tilde{w})^\top \theta^u)$ ;
5:      $g = \eta(L^w(u) - q)$ ;
6:      $e := e + g\theta^u$ ;
7:      $\theta^u := \theta^u + gv(\tilde{w})$ ;
8:   end for
9:    $v(\tilde{w}) := v(\tilde{w}) + e$ ;
10: end for
11: Return  $\theta = \theta^u$ .

```

Both models use the stochastic gradient ascent algorithm to update each parameter, and the finally obtained the parameter θ that is a matrix with the word vector of each word in dictionary W as its row. The specific update steps for CBOW model and skip-gram model are demonstrated in Algorithms 1 and 2. Here $v(\cdot)$ denotes the vector representation of a word, $\sigma(\cdot)$ is a sigmoid activation function in machine learning, θ^u denotes the corresponding vector of word u in θ , $L^w(u)$ is 1 when $u = w$

Authorized licensed use limited to: University of Waterloo. Downloaded on June 07, 2025 at 11:04:37 UTC from IEEE Xplore. Restrictions apply.

and otherwise it is 0, and η represents learning rate and we set it to 0.2 in our scheme.

4.3 k-Means Clustering

Clustering is the process of dividing a collection of objects into several sets of objects. Similar objects are categorized into one cluster and each cluster has a central point. Objects in the same cluster are similar to each other and objects in different clusters are not similar. There are mainly four types of clustering algorithms, i.e., model-based clustering algorithm, hierarchical clustering algorithm, grid-based clustering algorithm and density-based clustering algorithm. Our proposed scheme adopts k -means clustering algorithm that is suitable for text classification.

The k -means algorithm is proposed by MacQueen [3] that is an important method for data mining and machine learning. It is a heuristic classic method based on dividing and is extensively used in several fields because of its simplicity, efficiency and flexibility. The k -means algorithm considers euclidean distance as the similarity measurement, that is, the closer between two objects in the distance, the higher the level of similarity. Assume that the aggregation points x and y have m attributes (m vectorized keywords in text clustering), the values are (x_1, x_2, \dots, x_m) and (y_1, y_2, \dots, y_m) respectively. The distance $dist(x, y)$ between x and y is calculated as $dist(x, y) = \sqrt{\sum_{k=1}^m |x_k - y_k|^2}$.

Inertia is the sum of squared distances of samples to their closest cluster center. It is generally used as the objective function for clustering and we will minimize this function as $Inertia(D, k) = \sum_{i=1}^k \sum_{p \in D_i} dist(d_i, p)$. Here k is the upper number of categories, p denotes the objects in category i , d_i denotes the cluster center of D_i and $dist$ is euclidean distance. The k -means text clustering algorithm is presented in Algorithm 3. This algorithm first divides the dataset into k categories randomly. After several iterations, the centroid of each cluster is calculated according to a specific evaluation criteria. The $\vec{\mu}_i$ represents a collection of documents belonging to category i .

Algorithm 3. k -means($\{\vec{x}_1, \dots, \vec{x}_n\}$, k) $\rightarrow \{\vec{\mu}_1, \dots, \vec{\mu}_k\}$

Require: n data samples $\{\vec{x}_1, \dots, \vec{x}_n\}$, the upper number of clusters k .

Ensure: The clustering result $\{\vec{\mu}_1, \dots, \vec{\mu}_k\}$.

```

1:  $(\vec{c}_1, \dots, \vec{c}_k) \leftarrow SelectRandomSeeds(\{\vec{x}_1, \dots, \vec{x}_n\}, k)$ ;
2: for  $i = 1 : k$  do
3:    $\vec{\mu}_i \leftarrow \vec{c}_i$ ;
4: end for
5: while stopping criterion has not been met do
6:   for  $i = 1 : n$  do
7:      $i' \leftarrow \operatorname{argmin}_{i'} ||\vec{c}_{i'} - \vec{x}_i||$ ;
8:      $\vec{\mu}_{i'} \leftarrow \vec{\mu}_{i'} \cup \{\vec{x}_i\}$ ;
9:   end for
10:  for  $i = 1 : k$  do
11:     $\vec{c}_i \leftarrow \frac{1}{|\vec{\mu}_i|} \sum_{\vec{x} \in \vec{\mu}_i} \vec{x}$ ;
12:  end for
13: end while
14: Return  $\{\vec{\mu}_1, \dots, \vec{\mu}_k\}$ .

```

There are two methods to judge whether to stop the algorithm, one is that each record of each cluster does not

change, and the other is that the optimization objective function remains stable. The time complexity of the algorithm is $O(k * N * T)$, here k is the number of central points, N is the size of the dataset and T is the number of iterations.

5 CASE-SSE CONSTRUCTION

5.1 Formal Definition

The CASE-SSE scheme is defined as a six-tuple: $CASE\text{-}SSE = (GenKey, BuildIndex, Encrypt, QueryTd, Search, Decrypt)$.

- **GenKey:** Input the security parameters and it outputs a pseudo-random stream as the key to encrypt document set.
- **BuildIndex:** It builds a two-layer secure index. Based on the frequency of keywords in the document set, the k -means clustering algorithm is used to classify each document first. Then, the category-keyword distribution vectors are generated according to the classification results and an AVL-tree index is constructed. Finally, the category-document vectors are generated according to the classification results and the inverted index is constructed. The AVL-tree index and the inverted index are encrypted before storing in the cloud server.
- **Encrypt:** The file is divided into fixed-length blocks, and then each file is encrypted by block encryption algorithm to generate ciphertext.
- **QueryTd:** The data user sends the search keyword to the data owner. Then the data owner extends the semantics of the keyword and generates a trapdoor according to the extended keyword set.
- **Search:** After receiving the trapdoor from the data user, the cloud server matches it with the two-layer security index in turn. Then it returns the ciphertext set to the user according to search results.
- **Decrypt:** The data user receives the ciphertext set from the cloud server and decrypts them using the key authorized from the data owner.

5.2 Detailed Algorithms

Among the six-tuple algorithms, GenKey, BuildIndex and Encrypt are performed by the data owner, QueryTd is performed by the data user and the data owner, Search is carried out by the cloud server and Decrypt is performed by the authorized data user. The detailed algorithms are demonstrated next.

5.2.1 Genkey(1^λ) $\rightarrow K$

The data owner initializes the system and generates the key K , as shown in Algorithm 4. Here λ is the security parameter as input, and the output is $K = (K_1, M_1, M_2, S)$. First, the data owner gets the symmetric key K_1 to encrypt documents. Then, the algorithm gets M_1 and M_2 which are $m \times m$ invertible matrices. Finally, it generates S that is a random m -dimensional vector used to encrypt the index node vector and the query vector. $P(\cdot)$ is a pseudo-random function (PRF): $P : \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^l$, where k is the length of random key and l is the length of security

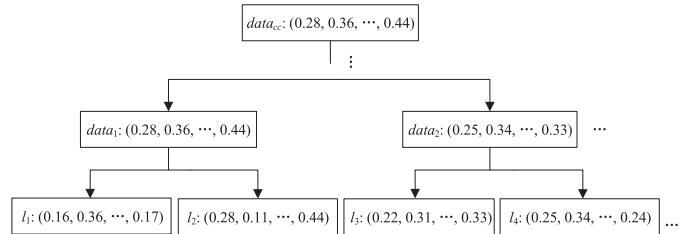


Fig. 4. The AVL-tree index.

parameter. In addition, there are three different random keys $\kappa_1, \kappa_2, \kappa_3$.

Algorithm 4. Genkey(1^λ) $\rightarrow K$

Require: Secure parameter λ .

Ensure: $K = \{K_1, M_1, M_2, S\}$.

```

1:  $K_1 \leftarrow P_{\kappa_1}(\lambda);$ 
2:  $rowNum \leftarrow m + \epsilon;$ 
3:  $colNum \leftarrow m + \epsilon;$ 
4: for  $k = 1, 2$  do
5:   for  $i = 1 : rowNum$  do
6:     for  $j = 1 : colNum$  do
7:        $M_k[i][j] \leftarrow P_{\kappa_2}(\lambda);$ 
8:     end for
9:   end for
10: end for
11: for  $i = 1 : m + \epsilon$  do
12:    $S_i \leftarrow P_{\kappa_3}(\lambda);$ 
13: end for
14: Return  $K \leftarrow \{K_1, M_1, M_2, S\}.$ 
  
```

5.2.2 BuildIndex($D, W, M_1, M_2, S, \varepsilon$) $\rightarrow I'_1, I'_2$

To improve the search efficiency, a two-layer index structure, i.e., an AVL tree index based on keyword-category and an inverted index based on category-document, is proposed. The BuildIndex algorithm consists of two sub-algorithms, i.e., BuildAVLIndex and BuildInvertedIndex. In the proposed scheme, it first utilizes k -means [3] clustering algorithm to classify the original dataset and then constructs an AVL tree index based on keyword-category. The BuildAVLIndex algorithm is demonstrated in Algorithm 5. After performing the k -means algorithm on the original dataset, we get the clustering result μ . For each category, it calculates the probability of each word in μ_i as $V[i]$. It constructs k nodes l_i with values $V[i]$ corresponding to k categories. The construction of the AVL tree is illustrated in Fig. 4.

Each leaf node stores a category id and an array V , and each non-leaf node stores an array $data_c$. The size of V and $data_c$ is m , and the value of $data_c[i]$ is the maximal value of its two children nodes, and c is the c^{th} non-leaf node in the AVL tree. After that, an inverted index is constructed based on category-document and the BuildInvertedIndex algorithm is demonstrated in Algorithm 6. It builds a list of document IDs for each category that includes these documents and the inverted index list is illustrated in Fig. 5.

The complete index construction algorithm BuildIndex is demonstrated in Algorithm 7. First, the data owner selects the optimal number of k -means clustering on lines 1 to 9. *Inertia* is the sum of squared distances of samples to their

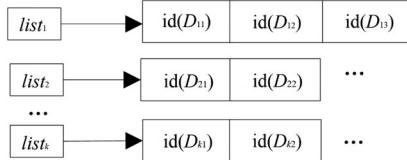


Fig. 5. The inverted index list.

closest cluster center. Even if the optimal number of classification is obtained when *Inertia* is the minimum value, sometimes the division is too fine (over-fitting) and the maximum distance from the points outside the cluster is not considered. Therefore, we add the *Silhouette* coefficient to the selection of k , which combines the two indicators, i.e., cohesion and separation. The combination of *Inertia* and *Silhouette* will minimize the distances among samples of the same cluster and maximize the distances among samples of different clusters more accurately. The calculation of $\text{Inertia}(D, k)$ is described above. While, given k and document set D , the calculation of $\text{Silhouette}(D, k)$ coefficient is as follows.

Algorithm 5. BuildAVLIndex($\mu, W, k \rightarrow I_1$)

Require: The clustering result μ , a keywords collection W , the number of categories k .

Ensure: An AVL-tree index I_1 .

```

1:  $V \leftarrow \{\}$ ;
2: for  $i \in [1 : k]$  do
3:   Calculate the probability of each word in  $\mu_i$  as  $V[i]$ ;
4:   Construct a node  $l_i$  with values  $V[i]$ ;
5: end for
6: while  $k \geq 2$  do
7:    $count = 0$ ;
8:   for  $i \in [1 : \lfloor k/2 \rfloor]$  do
9:     Construct a new node  $data_{count}$ ;
10:     $data_{count}.lchild \leftarrow l_i$ ,  $data_{count}.rchild \leftarrow l_{i+1}$ ;
11:    for  $j \in [1 : m]$  do
12:       $data_{count}[j] \leftarrow \max\{V[i][j], V[i+1][j]\}$ ;
13:    end for
14:     $count = count + 1$ ;
15:  end for
16:  if  $k \% 2 == 1$  then
17:     $count = count + 1$ ;
18:  end if
19: end while
20: Return  $I_1 := data_{count}$ .

```

(1) For the i^{th} object of each cluster, the average distance to all other objects of the same cluster is calculated as a_i (indicating the degree of cohesion).

(2) For the i^{th} object of each cluster, the average distance between this object and all objects in the other clusters is calculated as b_i (indicating the degree of separation).

(3) The *Silhouette* coefficient of the i^{th} object is calculated as $s_i = (b_i - a_i) / \max(a_i, b_i)$.

Thus, the range of the *Silhouette* coefficient is $[-1, 1]$ and it is better to be as large as possible. When the value is negative ($b_i > a_i$), it means that the sample is assigned to the wrong cluster, so the clustering result is not acceptable. If the value is close to 0, then it means that the clustering results is overlapping. In the proposed scheme, the two-

layer indices, i.e., an AVL tree index based on keyword-category and an inverted index based on category-document, are encrypted using the secure k -nearest neighbor (kNN) technique.

Algorithm 6. BuildInvertedIndex($\mu, k \rightarrow I_2$)

Require: The clustering result μ , the number of categories k .

Ensure: An inverted index I_2 .

```

1: for  $i \in [1 : k]$  do
2:   Store  $\vec{ID}_i := [\text{id}(D_{i1}), \text{id}(D_{i2}), \dots, \text{id}(D_{i|\mu_i|})]$  to a linked list
    $list_i$ ;
3: end for
4: Return  $I_2 \leftarrow \{\vec{ID}_1, \vec{ID}_2, \dots, \vec{ID}_k\}$ .

```

Algorithm 7. BuildIndex($D, W, M_1, M_2, S, \varepsilon \rightarrow I'_1, I'_2$)

Require: The document set D , a keyword collection W , two $(m + \varepsilon) \times (m + \varepsilon)$ invertible matrices M_1, M_2 , a random binary $(m + \varepsilon)$ -dimensional vector S , an integer ε greater than 0.

Ensure: Two-layer secure index I'_1, I'_2 .

```

1:  $\vec{\mu} \leftarrow []$ ;
2:  $Inertia\_eval \leftarrow []$ ; // Initialize Inertia.
3:  $Silhouette\_score \leftarrow []$ ; // Initialize Silhouette.
4: for  $k \in [2 : t]$  do
5:    $\vec{\mu}[k-2] \leftarrow k\text{-means}(W, k)$ ;
6:    $Inertia\_eval[k-2] \leftarrow \text{Inertia}(D, k)$ ;
7:    $Silhouette\_score[k-2] \leftarrow \text{Silhouette}(D, k)$ ;
8: end for
9: Choose the best  $k$  from Inertia_eval and Silhouette_score
   and get  $\vec{\mu}_k$ ;
10:  $I_1 \leftarrow \text{BuildAVLIndex}(\vec{\mu}_k, W, k)$ ;
11:  $I_2 \leftarrow \text{BuildInvertedIndex}(\vec{\mu}_k, k)$ ;
12: Extend all the vectors in  $I_1$  and  $I_2$  to  $(m + \varepsilon)$ -dimensional
   and set the last  $\varepsilon$  members of each vector to random;
13: for  $i = 1 : 2$  do
14:   for  $j = 1 : m + \varepsilon$  do
15:     if  $S[j] = 1$  then
16:        $(I'_i[j], I''_i[j]) \leftarrow I_i[j]$ ;
17:     end if
18:     if  $S[j] = 0$  then
19:        $I'_i[j] = I''_i[j] = I_i[j]$ ;
20:     end if
21:   end for
22: end for
23: Calculate  $\vec{I}'_1 \leftarrow (M_1^T I'_1, M_2^T I''_1)$ ,  $\vec{I}'_2 \leftarrow (M_1^T I'_2, M_2^T I''_2)$ ;
24: Return  $\vec{I}'_1, \vec{I}'_2$ .

```

On the one hand, the keyword-category index reduces the dimension of index node, improving search efficiency. On the other hand, the two-layer index mechanism based on the AVL-tree further improves the search efficiency.

5.2.3 Encrypt($K_1, D \rightarrow C$)

The data owner encrypts the plaintext document set $D = \{D_1, D_2, \dots, D_n\}$ to return the ciphertext document set $C = \{C_1, C_2, \dots, C_n\}$ using the secret key K_1 .

5.2.4 QueryTd($\theta, Q, M_1, M_2, S, \varepsilon \rightarrow V_{TD}$)

When the data user requests to search the encrypted documents, he first needs to be authorized by the data owner to issue a token to the data user. This process is demonstrated

in Algorithm 8. It consists of semantic extension of the query keyword and query vector construction. Once the data owner received the query keyword Q from the data user, the data owner calculates the word vector of Q from the trained parameter θ of Word2vec model, denoted as θ^Q . Then, the data owner calculates the distances between θ^Q and the vectors of other words in Word2vec based on the cosine distance. For two vectors \vec{u} and \vec{v} , the cosine distance is calculated as Equation:

$$\text{cosine_distance}(\vec{u}, \vec{v}) = \frac{\vec{u} \bullet \vec{v}}{\|\vec{u}\| \|\vec{v}\|}$$

In the proposed scheme, we set the number of extended keywords to 5. According to the cosine distances, the first five keywords are sorted in descending order to J . After that, the construction of query vectors are as follows.

- First, the data owner constructs a query vector $V(J)$ to represent which words need to be queried. The dimension is m and the corresponding position of each word in $V(J)$ is the probability of the occurrence of this word in the document set. Other positions are set to 0.
- Second, the data owner confuses the query vector $V(J)$ with the $(m + \varepsilon)$ -dimension vector S , and $V(J)$ is split into two vectors $V(J)'$ and $V(J)''$ according to S . Using the secure-kNN algorithm, the two split vectors are encrypted by multiplying two invertible matrix M_1 and M_2 .

Algorithm 8. QueryTd($\theta, Q, M_1, M_2, S, \varepsilon$) $\rightarrow V_{TD}$

Require: The trained parameter θ of Word2vec model, a query keyword Q , two $(m + \varepsilon) \times (m + \varepsilon)$ invertible matrices M_1, M_2 , a random binary $(m + \varepsilon)$ -dimensional vector S , an integer ε greater than 0.

Ensure: Query trapdoor V_{TD} .

- 1: Use the Word2vec model to calculate the word vector of Q from the trained parameter θ to get θ^Q ;
 - 2: $\text{word_cos_dis} \leftarrow []$;
 - 3: $i \leftarrow 0$;
 - 4: **for** $\theta^u \in \theta$ **do**
 - 5: **if** $u \neq Q$ **then**
 - 6: $\text{word_cos_dis}[i] \leftarrow \text{cosine_distance}(\theta^Q, \theta^u)$;
 - 7: $i \leftarrow i + 1$;
 - 8: **end if**
 - 9: **end for**
 - 10: According to the cosine distances, the first five extended keywords are sorted in descending order to J ;
 - 11: Construct the query vector $V(J)$ according to J ;
 - 12: Extend $V(J)$ to $(m + \varepsilon)$ -dimension and set the last ε members of it to random;
 - 13: **for** $i = 1 : m + \varepsilon$ **do**
 - 14: **if** $S[i] = 1$ **then**
 - 15: $(V(J)'[i], V(J)''[i]) \leftarrow V(J)[i]$;
 - 16: **end if**
 - 17: **if** $S[i] = 0$ **then**
 - 18: $V(J)'[i] = V(J)''[i] = V(J)[i]$;
 - 19: **end if**
 - 20: **end for**
 - 21: $V_{TD} \leftarrow (M_1^{-1}V(J)', M_2^{-1}V(J)'')$;
 - 22: Return V_{TD} .
-

5.2.5 Search($C, \vec{I}_1, \vec{I}_2, V_{TD}, topK$) $\rightarrow C'$

The cloud server searches on the encrypted two-layer index with the query trapdoor that is illustrated in Algorithm 9. First, the cloud server takes the id of the category from \vec{I}_1 . Then, it takes the document IDs vector \vec{ID} according to id of the inverted index. At last, the cloud server obtains the top- k most relevant encrypted documents and returns them to the user. Actually each node of the AVL-tree is a vector that denotes the keywords' probability distribution of a certain category and the element in the parent node is the maximal value of the two children. Therefore, the secure kNN product of the query trapdoor and the node vector indicates the relevance between them, so it can return the top- k most relevant results.

Algorithm 9. Search($C, \vec{I}_1, \vec{I}_2, V_{TD}, topK$) $\rightarrow C'$

Require: Encrypted document C , secure index \vec{I}_1, \vec{I}_2 , query trapdoor V_{TD} , the number of returned documents $topK$.

Ensure: Search results C' .

- 1: Initialize the root node of \vec{I}_1 as $Node$;
 - 2: **while** $Node.lchild$ is not NULL **do**
 - 3: $leftScore \leftarrow Node.lchild \times V_{TD}$;
 - 4: $rightScore \leftarrow Node.rchild \times V_{TD}$;
 - 5: **if** $leftScore \geq rightScore$ **then**
 - 6: $Node \leftarrow Node.lchild$;
 - 7: **end if**
 - 8: **end while**
 - 9: $id \leftarrow Node.id$;
 - 10: $\vec{ID} \leftarrow \vec{I}_2[id]$;
 - 11: **for** $i = 1 : topK$ **do**
 - 12: $C'[i] \leftarrow C[\vec{ID}[i]]$;
 - 13: **end for**
 - 14: Return C' .
-

5.2.6 Decrypt(C', K_1) $\rightarrow D'$

Taking as input the search result C' from the cloud server, the user decrypts C' with the secret key K_1 to output the plaintext documents D' .

6 SECURITY ANALYSIS

In our scheme, the cloud server is considered to be “honest-but-curious”. Specifically, the cloud server honestly and correctly performs the operations according to the designated protocol and will not distort users' data or search results, but the cloud server is curious about the sensitive data that can be used for inference and statistical analysis. According to the information available to the cloud server, the security of the proposed scheme follows the two threat models proposed by Cao *et al.* [4].

6.1 CASE-SSE in Known Ciphertext Model

Adversaries cannot infer plaintext information from ciphertext documents, encrypted indices, and trapdoors in the known ciphertext model. In this scheme, the cloud server only knows the encrypted document set C , encrypted indices \vec{I}_1, \vec{I}_2 and the trapdoor V_{TD} . Therefore, the cloud server can carry out ciphertext-only attack (COA) [24] in this model.

In CASE-SSE, the secret key K_1 is used to encrypt the document set D , and the key can only be shared between the data owner and the authorized data users. In a chosen plaintext attack secure (CPA-secure) symmetric encryption algorithm, such as AES for our scheme, a probabilistic polynomial time (PPT) adversary cannot distinguish whether the ciphertext C of a document is encrypted from the document A or B . Therefore, the encrypted documents of the proposed scheme satisfy the indistinguishable security under the known ciphertext model.

6.2 CASE-SSE in Known Background Model

In the known background model, it is assumed that the cloud server can gain more knowledge about whether any two queries contain the same keywords and the results of each query. Through the statistics of this information, the cloud server can get the distribution of keywords and the term frequency (TF) of some specific keywords. Then, it analyzes the value range or histogram of the frequency distribution to infer (or even directly identify) some keywords [8], [25].

In CASE-SSE, a PPT adversary can make statistical analysis attacks on queried trapdoors and obtain the importance of query keywords for some documents from the statistical information. However, because the dimension of the query vector in our scheme is extended to $(m + \varepsilon)$ and any ρ positions in $V_{TD}[m + j]$, $0 \leq j < \varepsilon$ is set to 0, it will make the query trapdoor and the search results of the same keyword to be different. Meanwhile, the relevance score calculated from the inner product of the query vector V_{TD} and the node vector V of the AVL-tree is also different. The calculation of relevance scores using the secure kNN algorithm is demonstrated as follows. As $\sum \eta_v$ of each query is random and unpredictable, the adversary cannot carry out statistical attacks on the *Search* operation.

$$\begin{aligned} \text{Score}(V_{TD}, V) &= V_{TD} \cdot V \\ &= \{M_1^T I', M_2^T I''\} \cdot \{M_1^{-1} Q', M_2^{-1} Q''\} \\ &= M_1^T I' \cdot M_1^{-1} Q' + M_2^T I'' \cdot M_2^{-1} Q'' \\ &= I'^T \cdot Q' + I''^T \cdot Q'' = I^T \cdot Q \\ &= \sum_{i=1}^m V_{TD}[i] \times V[i] + \sum \eta_v, \\ v &\in \{j | Q[m + j] = 1\}, 0 \leq j < \varepsilon. \end{aligned}$$

Suppose S is a stateful simulator, A is a stateful adversary. The procedure represented by $\text{Real}_A(k)$ is as follows.

- (1) Inputs a security parameter k , a challenger generates the key K with the key generation algorithm.
- (2) A inputs (I, D) and obtains (I, C) for the challenger. A selects $q \in \{w_i, ID\}$.
- (3) For each query q , A first generates some auxiliary information and sends it to the challenger. Then the challenger returns search results. Finally, A outputs a bit b .

The process of $\text{Ideal}_{A,S}(k)$ represents that S simulates indices and ciphertext generation and then sends them to A as follows.

- (1) When asking for a query for $q = w_i$, S acquires keywords and query results.

- (2) When performing the query $q = ID$, S provides the output of all keywords that appear in the adaptive query. A will send some auxiliary information to S , then S returns the trapdoor, and finally the adversary outputs a bit b' .

For any A , if there is a simulator that satisfies $|\Pr[\text{Real}_A(k) = 1] - \Pr[\text{Ideal}_{A,S}(k) = 1]| \leq negl(k)$, it can be considered that the scheme is secure and can resist adaptive chosen keyword attacks. Here k is the security parameter and $negl(k)$ indicates that the probability is negligible.

Given any adversary A and state simulator S , in order to simulate $\text{Ideal}_{A,S}(k)$, S will construct the security index and ciphertext set according to the algorithm described in Section 5. As the CPA-secure symmetric encryption algorithm is adopted, the symmetric key is generated by a pseudo-random function, and it cannot be distinguished from the real key. Therefore, the simulated process of document encryption is indistinguishable from the real encryption procedure. Meanwhile the result returned by S is indistinguishable from that returned by A through the random oracle, so for arbitrary adversaries, the probabilities of $\text{Real}_A(k)$ and $\text{Ideal}_{A,S}(k)$ can be ignored. Thus, the proposed scheme is secure under the known background model.

7 PERFORMANCE EVALUATION

To evaluate the performance of our scheme, we compare the proposed scheme CASE-SSE with the TFIDF-SSE [25] scheme under different settings. The dataset used in our experiment is a subset of Wikipedia articles and the language is English. Texts from 10 fields and 6500 texts in each field are selected as experimental data. There are about 50,000 texts in training dataset for the Word2vec model, 5,000 texts in validation set and 10,000 texts in testing set. The Word2vec model in our experiment is implemented by Python, and all tasks are completed in Jupyter Notebook development environment, using pytorch neural network framework. We implemented the schemes in 64-bits Windows 10 with an Intel(R) Core(TM) i5-4590 3.30 GHz CPU and 32 GB RAM.

7.1 Selection k for k -Means Clustering

Before training the Word2vec model and building the two-layer index, it is necessary to classify the texts by k -means algorithm. The selection of k determines whether the number of clusters is accurate and directly determines our scheme's efficiency and accuracy. As described above, to select better k for k -means clustering, two parameters *Inertia* and *Silhouette* are chosen to more accurately minimize the distances among samples of the same cluster and maximize the distances among samples of different clusters.

To minimize the distances among samples of the same cluster, it needs to minimize *Inertia*. While to maximize the distances among samples of different clusters, it needs to maximize the *Silhouette* coefficient. The number of clusters k from 2 to 25 is selected for the experiment. The parameters *Inertia* and *Silhouette* coefficient with different k are illustrated in Fig. 6. The *Inertia* is decreasing and then intersects with the *Silhouette* coefficient. It shows that the *Silhouette* coefficient reaches the first highest point when $k = 10$, so

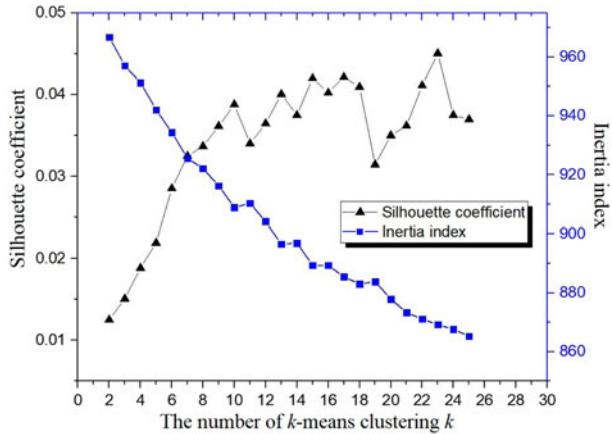


Fig. 6. Inertia and Silhouette coefficient with different k .

the best number of k -means clustering is set to 10 in this paper.

7.2 Performance Analysis

To evaluate the search efficiency of CASE-SSE, it is assumed that a user only requests a single keyword each time, and different categories of documents are not relevant. In our scheme, the semantics of this keyword is extended to several keywords. We use the precision P to measure the accuracy of the search results returned by the cloud server that is presented as: $P = TP / (TP + FP) \times 100\%$.

Here TP denotes the number of returned documents that meet the user's retrieval intention and FP denotes the number of returned documents that do not match. We compare the search accuracy and time consumption of CASE-SSE with TFIDF-SSE [25].

7.2.1 Search Accuracy and Time Consumption With Different n

The experimental results of search accuracy and time consumption are presented in Fig. 7.

Fig. 7a shows the search accuracy with different number of original documents when we set the number of search keywords to $t = 5$ and the security parameter of secure-kNN to $\varepsilon = 2$. Both schemes decreases with the growth of the number of original documents, but the search accuracy of our scheme is always significantly better than that of TFIDF-SSE. The TFIDF-SSE scheme is more sensitive to the size of the dataset and its accuracy decreases rapidly with the increase in n . Specifically, when $n = 10000$, the average

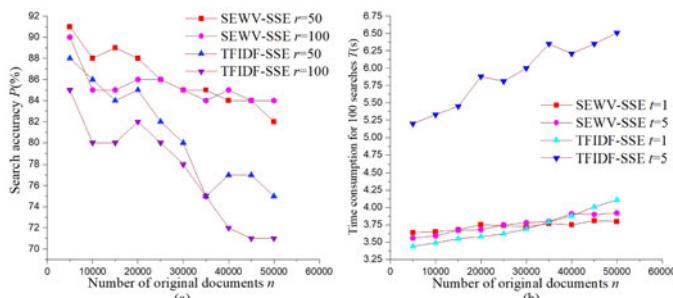


Fig. 7. Comparisons of search accuracy and time consumption with different n .

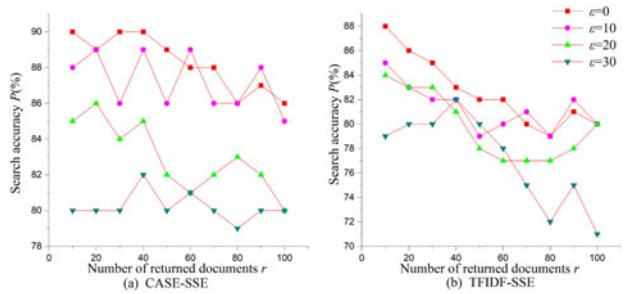


Fig. 8. Comparison of search accuracy with different ε .

precision of CASE-SSE is 90.5%, but that of TFIDF-SSE is 86.5%. When $n = 40000 \sim 50000$, the average precision of CASE-SSE is still 84% but that of TFIDF-SSE is only 73.5%. Overall, when the number of returned documents is 50 and 100 respectively, the search accuracy of CASE-SSE always remains at more than 80%. When the dataset is larger, the search accuracy is relatively stable that benefits from the k -means clustering.

Fig. 7b shows the time consumption with different number of original documents when we set the number of returned documents to $r = 10$ and $\varepsilon = 2$. The TFIDF-SSE scheme is more sensitive to the number of search keywords t than our scheme. In detail, when $t = 1$ and $n = 10000$, the search time of CASE-SSE is 3.6 seconds, slightly higher than that of TFIDF-SSE. When $t = 1$ and $n = 40000 \sim 50000$, the time assumption of CASE-SSE is 3.75 seconds, slightly better than that of TFIDF-SSE which is 4.1 seconds. Although it is more efficient when t and n are small, the time assumption of interactions between the three parties will be substantially increasing when t is set to 5. The experimental results show that TFIDF-SSE takes 2/3 more search time than that of CASE-SSE. Actually, the trapdoor generation of our scheme is hardly affected by t because it supports multi-keyword search. In the case of semantic extension of keywords, our scheme will not significantly reduce the search accuracy and efficiency.

7.2.2 Search Accuracy With Different ε

Since the dimension of query vector and that of the probability distribution vector of all nodes of the AVL-tree is the same, and they both are the size of keywords set m . Therefore, the cloud server can record multiple queries and analyze the frequency of each keyword to carry out statistical attacks. For this reason, we extend the m dimensional vectors to $m + \varepsilon$. Specifically, we add ε random numbers to the end of the query vector to confuse the search keywords and ensure the non-relevance of multiple queries.

The comparison of search accuracy with different ε is illustrated in Fig. 8. It shows that the search accuracy P changes with the number of returned documents r under four different values of ε . When $\varepsilon = 10$, the average precision of CASE-SSE is the best, reaching to 89%. While, when $\varepsilon = 10$, the average precision of TFIDF-SSE is only 80% and it decreases significantly. At the same time, the larger ε makes the TF-IDF value of each word larger, and that will change its true meaning, indirectly leading to the volatility of search accuracy. With the increase in ε , the search accuracy will decrease, but the security of

TABLE 1
Storage Cost of Index

Number of documents	10000	20000	30000	40000	50000
TFIDF-SSE (MB)	85.11	156.31	245	328.8	446.51
Our scheme (MB)	15.9	21.41	29.9	41.21	59.4

the SSE scheme will be improved. It is necessary to make a trade-off between security and search accuracy to determine ε . With the increasing of r , the search accuracy of CASE-SSE is more stable than that of TFIDF-SSE under different ε settings. By comparison, the latter is greatly affected by r because it does not cluster the documents before constructing indices, that further indicates the superiority of the two-layer index structure of our scheme.

7.3 Evaluation of the Two-Layer Index

7.3.1 Comparison With Other Tree Index

In SSE schemes, the cloud server needs to store encrypted documents and indices to achieve the secure search on encrypted data. Thus we evaluate the storage cost and search time overhead from the perspective of secure index.

The comparison of storage cost between CASE-SSE and TFIDF-SSE is presented in Table 1. It shows that the storage cost of the secure index in CASE-SSE is far less than that of TFIDF-SSE. It is because of that CASE-SSE only needs to store n documents' IDs and k additional encrypted document category vectors, each of that is m -dimensional, so the storage complexity of CASE-SSE is $O(km)$. However, TFIDF-SSE uses a balanced binary tree as the secure index, which needs to store m -dimensional encrypted document vectors and search vectors to the leaf nodes and intermediate nodes respectively. As there are n encrypted document vectors, the balanced binary tree has n leaf nodes. Therefore, the storage complexity of TFIDF-SSE is $O(mn)$. Since the number of categories k is much less than n , the storage cost of the secure index in CASE-SSE is significantly smaller than that of TFIDF-SSE. The storage cost of both schemes increases with the growth of the number of documents, because the tree-based index needs more storage to store the nodes. But when dealing with the large scale dataset, it is an acceptable trade-off between the search efficiency and storage cost.

The comparison of search time overhead between CASE-SSE and TFIDF-SSE is presented in Table 2. It shows that the search time overhead of the two-layer index of CASE-SSE is less than that of TFIDF-SSE. The reason is that CASE-SSE greatly reduces the scale of the tree index by using the k -means clustering method, thus leading to less search time overhead.

TABLE 2
Search Time Overhead of Index

Number of documents	10000	20000	30000	40000	50000
TFIDF-SSE (ms)	40.8	45.6	48.6	50.25	52.1
Our scheme (ms)	36	37.1	37.5	38.3	38.51

TABLE 3
Comparisons of CASE-SSE and Other SSE Schemes

Index type	Scheme	Storage	Time	Function
Linear scan	[1]	$O(Mn)$	$O(n)$	Single
Forward index	[26]	$O(n)$	$O(n)$	Single
	[27]	$O(Mn)$	$O(t)$	Single
Inverted index	[5]	$O(n + m)$	$O(t)$	Single
	[28]	$O(n)$	$O(t)$	Single
Tree index	[29]	$O(mn)$	$O(tlogn)$	Multiple
Two-layer index	Our's	$O(mk + n)$	$O(logk)$	Semantic

7.3.2 Comparison With Other SSE Schemes

The comparison of storage overhead, time complexity and function among CASE-SSE and other SSE schemes are presented in Table 3. Here M is the number of words in vocabulary, m is the total number of keywords in the dataset, n is the total number of documents, t is the number of keywords matched by the trapdoor, k is the number of categories in the dataset and it is generally assumed to be $k < n$ and $t < m$.

The efficiency of the forward index is poor, so the binary tree based index is proposed to improve the efficiency. In the tree based index, document identifiers are used as the leaf nodes to be retrieved, and the non-leaf nodes store the keywords information of the documents. However, when the number of documents increases to one hundred thousand or millions of magnitude, the tree index will be too large. Actually, as the time overhead of one disk I/O is usually much longer than that of one search, if many disk I/O operations are required, it will severely reduce the efficiency of SSE schemes. In our scheme, we use the k -means clustering algorithm to cut down the scale of the AVL-tree and use the two-layer index to reduce the dimension of the index vectors. For example, if the number of AVL-tree layers reaches to twenty, there are only about five hundred thousand documents at most. After k -means clustering, because the number of categories is much less than the number of documents, the scale of the AVL-tree will be greatly reduced.

REFERENCES

- [1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.
- [2] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Representations*, 2013.
- [3] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
- [4] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2011, pp. 829–837.
- [5] 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.*, 2006, pp. 79–88.
- [6] A. Swaminathan *et al.*, "Confidentiality-preserving rank-ordered search," in *Proc. ACM Workshop Storage Secur. Survivability*, 2007, pp. 7–12.
- [7] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, 2010, pp. 253–262.

- [8] 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.
- [9] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2012, Art. no. 12.
- [10] W. Sun *et al.*, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inf. Comput. Commun. Secur.*, 2013, pp. 71–82.
- [11] W. Bing, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 2112–2120.
- [12] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2706–2716, Dec. 2017.
- [13] X. Ding, P. Liu, and H. Jin, "Privacy-preserving multi-keyword top-k similarity search over encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 2, pp. 344–357, Mar./Apr. 2019.
- [14] T. Peng, Y. Lin, X. Yao, and W. Zhang, "An efficient ranked multi-keyword search for multiple data owners over encrypted cloud data," *IEEE Access*, vol. 6, pp. 21924–21933, 2018.
- [15] Z. Xia, Y. Zhu, X. Sun, and L. Chen, "Secure semantic expansion based search over encrypted cloud data supporting similarity ranking," *J. Cloud Comput.*, vol. 3, no. 1, pp. 1–11, 2014.
- [16] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Trans. Consum. Electron.*, vol. 60, no. 1, pp. 164–172, Feb. 2014.
- [17] C. Chen, P. S. Shen, S. Guo, X. J. Zhu, and J. K. Tari, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.
- [18] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017.
- [19] H. Dai, X. Dai, X. Yi, G. Yang, and H. Huang, "Semantic-aware multi-keyword ranked search scheme over encrypted cloud data," *J. Netw. Comput. Appl.*, vol. 147, 2019, Art. no. 102442.
- [20] D. Harris and S. L. Harris, *Digital Design and Computer Architecture*. Burlington, MA, USA: Morgan Kaufmann, 2014.
- [21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
- [22] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013, *arXiv:1310.4546*.
- [23] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 307–361, 2012.
- [24] B. Englert, "Introduction to cryptography: Principles and applications (3rd ed.)," *Comput. Rev.*, vol. 57, no. 6, pp. 341–342, 2016.
- [25] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.
- [26] T. Moataz and A. Shikfa, "Boolean symmetric searchable encryption," in *Proc. 8th ACM SIGSAC Symp. Inf. Comput. Commun. Secur.*, 2013, pp. 265–276.
- [27] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.
- [28] D. Cash and S. Tessaro, "The locality of searchable symmetric encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2014, pp. 351–368.
- [29] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2013, pp. 258–274.



Lanxiang Chen received the MS and PhD degrees in computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2009, respectively. She is currently a professor with Fujian Normal University, Fuzhou, China. Her research interests include cryptography, cloud storage security, privacy protection, and structured encryption. She is a member of the Computer Society of China.



Yujie Xue is currently working toward the MS degree with Fujian Normal University, Fuzhou, China. His research interests include natural language processing and structured encryption.



Yi Mu (Senior Member, IEEE) received the PhD degree from Australian National University, Australia, in 1994. In 2021, he joined the City University of Macau, China, as a professor. He was a professor and the head of the School of Computer Science and Software Engineering, University of Wollongong, Australia. His research interests include cryptography and cybersecurity.



Lingfang Zeng (Member, IEEE) received the PhD degree in computer architecture from the Huazhong University of Science and Technology, China, in 2006. He was a research fellow with the National University of Singapore. He is currently a PI with Zhejiang Lab, Hangzhou, China.



Fatemeh Rezaeibagha (Member, IEEE) received the MS degree in information security from the Luleå University of Technology, Sweden, in 2013, and the PhD degree in computer science from the University of Wollongong, Australia, in 2017. She is currently a lecturer of cyber security with Murdoch University, Perth, Australia. Her main research interests include cryptography, IoT, blockchain, and cybersecurity.



Robert H. Deng (Fellow, IEEE) is currently the AXA chair professor of cybersecurity and the director of Secure Mobile Centre, Singapore Management University, Singapore. His research interests include data security and privacy, network security, and system security. He is a fellow of the Academy of Engineering Singapore.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csl.