Semantic-based Compound Keyword Search over Encrypted Cloud Data

Bo Lang, (*Member, IEEE*), Jinmiao Wang, Ming Li, and Yanxi Liu

Abstract—Keyword search over encrypted data is essential for accessing outsourced sensitive data in cloud computing. In some circumstances, the keywords that the user searches on are only semantically related to the data rather than via an exact or fuzzy match. Hence, semantic-based keyword search over encrypted cloud data becomes of paramount importance. However, existing schemes usually depend upon a global dictionary, which not only affects the accuracy of search results but also causes inefficiency in data updating. Additionally, although compound keyword search is common in practice, the existing approaches only process them as single words, which split the original semantics and achieve low accuracy. To address these limitations, we initially propose a compound concept semantic similarity (CCSS) calculation method to measure the semantic similarity between compound concepts. Next, by integrating CCSS with Locality-Sensitive Hashing function and the secure k-Nearest Neighbor scheme, a semantic-based compound keyword search (SCKS) scheme is proposed. SCKS achieves not only semantic-based search but also multi-keyword search and ranked keyword search. Additionally, SCKS also eliminates the predefined global library and can efficiently support data update. The experimental results on real-world dataset indicate that SCKS introduces low overhead on computation and the search accuracy outperforms the existing schemes.

Index Terms—searchable encryption, semantic-based keyword search, semantic similarity, compound concept

1 INTRODUCTION

In cloud computing, an increasing number of personal or enterprise users outsource their data to cloud storage to enjoy the benefits of ”pay-on-demand” services and high computation performance. To preserve privacy, users opt to encrypt data before outsourcing. Thus, the traditional keyword search cannot be directly executed on the encrypted data, which limits the utilization of data. To address this problem, Song et al. [1] proposed the idea of searchable encryption (SE) that allows users to search on encrypted data through a keyword. Subsequently, various searchable encryption schemes were proposed to meet different requirements such as fuzzy keyword search [2]–[4], multi-keyword search [5]–[8], ranked keyword search [9]–[11], and semantic-based keyword search [12]–[17].

In practice, semantic-based keyword search not only is convenient for users but also exactly expresses users' intentions. Specifically, in some circumstances, users might not be familiar with the encrypted documents stored in cloud storage or might only want the semantically related results; therefore, the search keywords are usually semantically related to the document rather than via an exact or fuzzy match. For example, the predefined keyword of a document is “*cloud-based storage*”, and the keyword that a user searches is “*distributed storage*”. Obviously, these two words are neither an exact nor a fuzzy match, but they are semantically related. Hence, the semantic-based keyword search is of practical importance and has attracted much attention. However, the existing approaches [12]–[15] must rely on a predefined global dictionary whose quality greatly influences the accuracy of the search result. Moreover, when the dataset is outsourced to the cloud, update operations that include inserting new documents and modifying and deleting existing documents are frequent. Because the predefined dictionary is constructed based on all documents in the dataset, the update of a single document can cause the reconstruction of the dictionary and even all document indexes, which is inefficient.

On the other hand, the semantic similarity between keywords in a query and keywords of documents is important because it also determines the accuracy of search results. However, in the aforementioned approaches, the semantic information used to measure the semantic similarity was mined from some knowledge bases (KBs) (such as corpus and thesaurus) containing noise data, which cause the semantic similarity to be inaccurate. Compared with other KBs, ontology has good support for logic reasoning and can structurally express the semantic information of concepts. Several ontology-based approaches [18]–[23] have been proposed to assess the similarity between concepts through mining ontology information from different aspects. However, these approaches largely aim at single concepts composed of single terms. For compound concepts composed of multiple terms, the approaches usually neglect the special constituent features of the compounds and only process them as single words. In fact, each component term has a different effect on the semantics of the compound concept. Hence, introducing the constituent features will greatly improve the accuracy of compounds' similarity. However,

- *This work was supported by the National Natural Science Foundation of China (Grant No.61170088 and 6137015) and the Foundation of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2015ZX-04)*

- *The authors are with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China. E-mail: {langbo, wangjinmiao, liming, liuyanxi}@buaa.edu.cn.*

- *This paper was presented in part at 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI).*

an efficient and practical semantic similarity measurement method for compound concepts remains an open question.

To address these problems, we propose a semantic-based compound keyword search (SCKS) scheme over encrypted data in this paper. SCKS uses a topic set in a field and Vector Space Model (VSM) to express the semantic information of keywords. Each element of the keyword vector corresponds to a field topic, and the value is the semantic similarity between the keyword and the topic. Because the keywords and field topics can be compound concepts, we initially propose an ontology-based compound concept semantic similarity (CCSS) calculation method to measure their semantic similarity [24]. In CCSS, the compound is decomposed into subject headings and auxiliary words, and the relationships between them are used to measure the similarity. Moreover, CCSS comprehensively considers the information sources of ontology, such as taxonomical features, local density, path length and depth, which efficiently improves the ultimate accuracy.

Because each document usually contains more than one keyword, the index of a document is associated with multiple keyword vectors. Locality-Sensitive Hashing (LSH) function is able to hash similar items to the same bucket with high probability. Hence, we construct the document index by using LSH to map multiple keyword vectors into only one vector. The frequency of the keyword in the document is also considered and is inserted into the index vector as the value of corresponding element. Compared with the existing schemes [2], [3], in which the vector value is only 0 and 1, SCKS can express more semantic information of the document. Another advantage of SCKS is that it can support data update efficiently because no global dictionary need be predefined and each document is individually indexed. The query vector is generated similarly, which indicates SCKS can support multi-keyword search.

To protect the privacy of documents and queries, fully homomorphic encryption (FHE) technique could be chosen to encrypt them, which allows data servers perform some flexible functions over encrypted data. However, the existing FHE schemes are far from being practical for real applications, because all of them are too complicated and inefficient [25]. Hence, we adopt a secure k-Nearest Neighbor (SkNN) [26] to encrypt the index and query. Benefiting from the features of SkNN, SCKS is able to achieve ranked keyword search and return the *Top* k most relevant results to the user. To improve the security of SCKS, we further propose a security-enhanced SCKS scheme (SE-SCKS) by introducing a pseudorandom function. The main contributions of our work are summarized as follows:

(1) An ontology-based compound concept semantic similarity (CCSS) calculation method is proposed to improve the accuracy of compound similarity measurement. By calculating the semantic similarity between keywords and field topics using CCSS, the semantic characteristics are introduced into the keyword vector.
(2) By integrating CCSS, LSH and SkNN, we propose a semantic-based compound keyword search scheme (SCKS) over encrypted data. Benefiting from these techniques, SCKS can simultaneously support semantic-based keyword search, multi-keyword search, ranked keyword search and efficient data update.
(3) We analysis the security of SCKS and further propose a security-enhanced scheme, SE-SCKS. The security analysis indicates that SE-SCKS is semantically secure under the *adaptive* model and can be used in circumstances requiring a higher security level.
(4) We implement and test CCSS and SCKS on a real-world dataset. The experimental results demonstrate that our approaches are accurate and efficient.

The remainder of this paper is organized as follows. Section 2 summarizes the related work on searchable encryption and similarity measures. Section 3 presents preliminaries used in this paper. In Section 4, we propose an ontology-based compound concept semantic similarity (CCSS) calculation method. Based on CCSS, we describe our semantic-based compound keyword search scheme in detail in Section 5. The security-enhanced scheme SE-SCKS is proposed in Section 6. Section 7 evaluates our approaches through experiments. And the final section concludes the paper.

2 Related work

A semantic-based keyword search scheme returns results according to the semantic relatedness between documents and a query. Based on the co-occurrence probability of terms, Sun et al. [12] and Xia et al. [13] respectively constructed a semantic relationship library (SRL) to record the semantic similarity between keywords. In the search phase, the query keywords are expanded based upon the SRL, and the extended query keywords are used in the search algorithm. Fu et al. [15] extended the query keywords by introducing m-best tree and term similarity tree, both of which are built based on the WordNet. Similarly, the keyword set in [14] is extended via a synonym thesaurus. In the conceptual-graphs based search scheme [16], some sentences are extracted to represent the documents and the semantic search is enforced by calculating the relevance score between the sentences in the document and the query. According to the characteristic that related keywords usually have the same root, the scheme proposed by Moataz et al. [17] extracts the keyword root by a stemming algorithm and searches with the root instead of the keywords. Obviously, this method cannot work when the semantically related keywords have different roots.

In general, the schemes aiming at multi-keyword search can also achieve ranked keyword search. Orencik et al. [7] utilized the MinHash function and Term Frequency-Inverse Document Frequency (TF-IDF) to construct a document index. However, the TF-IDF should be recalculated when documents or keywords are added or removed, which causes a data updating difficulty. Yu et al. [8] proposed a two-round search scheme employing homomorphic encryption to support multi-keyword search. This scheme also adopts TF-IDF; thus, it cannot efficiently support data updating. In the scheme proposed by Cao et al. [5], each document is associated with a binary vector of which the dimensionality is equal to the number of keywords in the global keyword set. Hence, the global keyword set should be static because its adding or deleting will cause the reconstruction of the document index. By introducing partitioned matrices into [5], Li et al. [6] improved efficiency when new keywords and documents are added, which alleviates the problem of database update.

To support data dynamic update efficiently, Liesdonk et al. [27] proposed two schemes. The first one is more efficient but requires two rounds of communication between the server and the client for each search. The second one only requires one round of communication, but the maximum times of database updates are limited. Kamara et al. [28] constructed a dynamic search scheme by employing inverted index and search table, search array, deletion table and deletion array techniques. However, this method can leak information about keywords in the updated documents. To eliminate information leakage, Kamara et al. introduced a red-black tree to the search scheme [29]. In addition to data update, this scheme can be performed in parallel mode, which improves the efficiency of search and update.

Ontology-based similarity measure methods can be divided into different categories: path-based measures, information content-based measures, featured-based measures and hybrid measures. Early path-based measures primarily assessed the similarity between concepts by counting the shortest path [23], [30] or combining the distance between concepts with the depth of the taxonomy [31], [32]. The information content (IC) of a concept is often defined as the negative log likelihood of its appearance probability. IC values can be obtained by computing the probability of concepts in a given corpus [33]–[35] or only be intrinsically derived from the ontology [22], [36]. The degree of overlapping between sets of ontological features is considered in the feature-based measures [18], [37]. Some studies also combined several information sources to improve their accuracy [19], [20]. However, all of these methods primarily aim at single concepts. For compound concepts, the methods usually neglect the special constituent features of compounds and only process them as single concepts, which decreases the accuracy of similarity measurement.

3 Preliminaries

3.1 Scheme Model

In our scheme, the index of document and the query are represented with the VSM. A document *index* is denoted by a vector generated with the keywords of the document, and a *secure index* is the encrypted index. Similarly, a *query* is a vector generated with the keywords of a search, and a *trapdoor* is an encrypted query. In general, the document can be encrypted by traditional encryption schemes such as AES. Focusing on the index and query, our scheme consists of the following algorithms.

- **Keygen (d)**. This algorithm is executed by the data owner or a trusted authority (TA). Taking a security parameter d as input, the algorithm outputs a system symmetric key sk.
- **BuildIndex (sk, D)**. This algorithm is executed by the data owner. Based on the symmetric key sk and the document D, the algorithm generates the secure index $I(D)$.
- **Trapdoor (sk, Q)**. This algorithm is executed by the data owner or TA. With the keyword set Q that the user wants to search, the algorithm generates the corresponding trapdoor $T(Q)$.
- **Search $(I(D), T(Q))$**. This algorithm is executed by the cloud server. Based on the trapdoor $T(Q)$ and each secure index $I(D)$ stored in the server, the cloud server calculates the correlation coefficients between the query Q and each document D and returns the ranked correlation coefficients to the user.

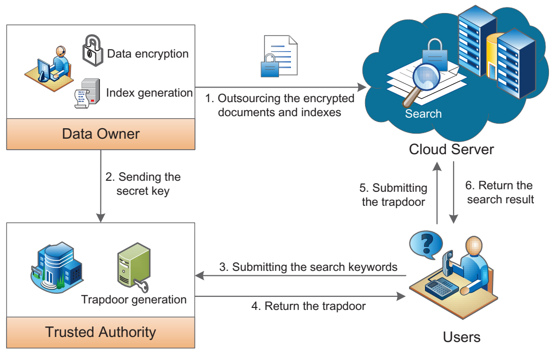
Data encryption
Index generation
Data Owner
1. Outsourcing the encrypted documents and indexes
Search
Cloud Server
2. Sending the secret key
5. Submitting the trapdoor
6. Return the search result
3. Submitting the search keywords
Trapdoor generation
4. Return the trapdoor
Trusted Authority
Users

Fig. 1. The model of keyword search

The running process of our scheme is described in Fig. 1. First, the data owner publishes the encrypted documents and secure indexes to the cloud server. To reduce the computation burden, the data owner is allowed to outsource the generation of the trapdoor to TA by giving the private key to it. In this case, when a user wants to search over encrypted documents, he/she submits the keywords to TA which generates the corresponding trapdoor and returns it to the user. Then, the user sends the trapdoor to the cloud server. Finally, the cloud server executes the search algorithm with the trapdoor on all secure indexes and returns the relevant documents to the user.

Since TA can obtain the private key, it should be entirely trustable, similar to the certificate authority (CA) of public key infrastructure (PKI). In applications, users or enterprises can choose TA according to this security requirement. For example, in an enterprise, TA is usually an internal server. If there is no such trustable server in the system, the trapdoor can be generated by the data owner. Also, some existing schemes assume that the authorization between the data owner and users is appropriately done (i.e., the private key should be sent to the authorized users in advance), then the trapdoor can be generated by each authorized user [12]–[15].

3.2 Background Knowledge

3.2.1 *Locality-Sensitive Hashing function*

The Locality-sensitive hashing (LSH) function [40] hashes input items so that similar items map to the same "bucket" with high probability. It is able to reduce the dimensionality of high-dimensional data.

We use the p-stable LSH function family [41] in our scheme. A p-stable LSH function is given by

$$h_{\vec{a},b}(\vec{v}) = \lfloor \frac{\vec{a} \cdot \vec{v} + b}{w} \rfloor \tag{1}$$

where $\vec{a}$ is a d-dimensional vector with entries chosen independently from a p-stable distribution, and b is a real number chosen uniformly from the range $[0, w]$. The hash function $h_{\vec{a},b}(\vec{v})$ maps a d-dimensional vector onto an integer. By choosing different values for $\vec{a}$ and b, different LSH functions in the family can be generated.

3.2.2 Secure k-Nearest Neighbor encryption scheme

The k-Nearest Neighbor (kNN) query is one of the fundamental query operations in some applications. The goal of SkNN is to securely identify the k-nearest points in the encrypted database to a given encrypted query, without allowing the data server to obtain the content of the database or the query. In the past few years, various methods have been proposed to address the SkNN problem. Wong et al. [26] proposed an asymmetric scalar-product-preserving encryption (ASPE) scheme which preserves scalar product between the query and tuples in database. By introducing the privacy homomorphism scheme, Hu et al. [42] proposed a novel protocols for SkNN queries on tree-based index. In the scheme proposed by Zhu et al. [43], the key of data owner is not disclosed to query user, which can be used in the applications that query users are not trustworthy. Elmehdwi et al. [44] improved the security and proposed a protocol to protect the data access pattern, including the confidentiality of database and query.

In our scheme, we employ SkNN to encrypt the document index and query vector before publishing. Because the correlation coefficient between document and query is obtained by calculating the inner product of document index and query vector, the inner product should be correctly calculated after encryption. Hence, we adopt the ASPE scheme proposed by Wong et al. [26], which preserves the scalar product between the tuples and query. The main idea of ASPE is detailed as follows.

Let M represent an invertible matrix and p, q represent vectors. The ASPE scheme [26] is to encrypt p through multiplying it by M^T, i.e., $p' = M^T p$, and encrypt q through multiplying it by M^{-1}, i.e., $q' = M^{-1}q$. According to the characteristic that the product of inverse matrices is the identity matrix, the product of the encrypted vectors equals the product of the original vectors, i.e., $p'^T q' = p^T M \cdot M^{-1} q = p^T q$. Additionally, it is not possible for one to determine the value of p or q, respectively, from p' or q' without knowing M. Benefiting from these features, SkNN can be used in searchable encryption to encrypt the document index and query. Then, the correlation coefficient between them can be calculated by multiplying the secure index and trapdoor.

4 Compound Concept Semantic Similarity Calculation Method

4.1 Features of Compound Concepts

Depending upon their semantic constituents, the compound concepts can be divided into two types: endocentric structure and exocentric structure. The endocentric structure is true when one or more constituents of a compound can play the central role and serve as a definable ***subject heading***. For example, the subject heading of "*information retrieval*" is "*retrieval*". The exocentric structure is true when there is no subject heading in a compound. For example, "*pick pocket*" can be expressed as "*a person who picks a pocket*" which means neither "*pick*" nor "*pocket*" but rather "*a person*". We found that most compounds of English have an endocentric structure, whereas compounds with exocentric structure are commonly used in informal situations [45]. Hence, in this paper, we focus on the endocentric structure compounds.

Moreover, depending upon the relationship between constituents, the noun compounds of an endocentric structure also can be divided into two types: subordination noun compound and coordination noun compound. The former indicates that each constituent usually has different status; i.e., one depends upon the others. The part of a subordinate constituent that can be called ***auxiliary word*** is used to modify the subject heading. For example, the subject heading of "*information retrieval*" is "*retrieval*", whereas the auxiliary word is "*information*". A coordination noun compound is formed by two or more constituents with same status and connected by a conjunction, such as *and* or *or*. For example, "*deaf and dumb*" is a coordination compound, and the words to the left and right of *and* share the same status.

For a subordination compound, we found that the left side constituent always depends upon the right side one, and the right side constituent not only determines the semantic category of the compound but also expresses the major meaning. Hence, the word on the far right of a subordination compound can be considered as the subject heading, and the remaining word(s) can be considered as the auxiliary word(s). Concerning the coordination compound, both the left and right side constituent can be considered as subordination compound. Moreover, the subject heading of a single noun is itself, and the auxiliary word is empty. Based on these characteristics, we propose a Subject headings and Auxiliary words Recognition (SAR) algorithm, which is described in Fig. 2, to automatically recognize the subject headings and auxiliary words (SaA) of noun compounds with the endocentric structure.

Algorithm 1: Subject headings and Auxiliary words Recognition (SAR)

```
Input: the compound word compound;
Output: the subject heading $H_{subject}$ and the auxiliary word $A_{auxiliary}$;
1: if there is a conjunction such as and and or in compound then
2:     insert the words on the left of conjunction into $C_{left}$;
3:     insert the words on the right of conjunction into $C_{right}$;
4:     GETSUBJECTAUXILIARY($C_{left}$);
5:     GETSUBJECTAUXILIARY($C_{right}$);
6: else
7:     GETSUBJECTAUXILIARY(compound);
8: end if
9: return $H_{subject}$ and $A_{auxiliary}$;

10: procedure GETSUBJECTAUXILIARY(compound)
11:     if compound is a single word then
12:         insert compound into $H_{subject}$;
13:         insert $\varnothing$ into $A_{auxiliary}$;
14:     else
15:         insert the word on the far right of compound into $H_{subject}$;
16:         insert the words not in $H_{subject}$ into $A_{auxiliary}$;
17:     end if
18:     return $H_{subject}$ and $A_{auxiliary}$;
19: end procedure
```

Fig. 2. Algorithm of subject headings and auxiliary words recognition

4.2 Semantic Similarity Calculation for Compound Concept

To measure the semantic similarity of compound concepts, we propose a novel approach that considers the concept constituent features and several other factors influencing similarity. The compound concepts are decomposed into SaA by using Algorithm 1. The taxonomical features, local density, path length and depth of ontology also influence the similarity between concepts; therefore, we respectively calculate the impact factors associated with them and then comprehensively consider all of the impact factors to improve the accuracy.

4.2.1 Compound constituent feature

For a compound concept, the subject heading generally has greater effect than the auxiliary word because the former expresses the major meaning, whereas the latter is primarily used for modifying the former. We respectively use the common subject heading sets and the overlapping of auxiliary word sets to measure the impact factors of the subject heading (i.e., f_{head}) and the auxiliary word (i.e., $f_{auxiliary}$). The relationships between f_{head} and $f_{auxiliary}$ can be used to describe the constituent features of compounds. Because the SaA recognition accuracy of Algorithm 1 is difficult to bring up to 100%, we introduce a recognition error ϵ to enhance credibility and reliability in practice.

Definition 1. Impact factor of compound constituent features can be written as follows:

$$\begin{aligned} F_{compound}(c_1, c_2) &= u \times f_{head}(c_1, c_2) \\ &+ (1-u) \times f_{auxiliary}(c_1, c_2) + \epsilon \end{aligned} \quad (2)$$

where c_1 and c_2 represent compound concepts, $u \in [0, 1]$ is a weighting parameter, $f_{head}(c_1, c_2)$ and $f_{auxiliary}(c_1, c_2)$ represent the functions of subject heading and auxiliary word between c_1 and c_2, respectively, which are described in $Eq.$ (3) and $Eq.$ (4).

$$f_{head}(c_1, c_2) = |H(c_1) \cap H(c_2)| \quad (3)$$

where $H(c)$ is the subject heading set of concept c, and the range of $f_{head}(c_1, c_2)$, which usually equals 0, 1 or 2, is discrete.

$$f_{auxiliary}(c_1, c_2) = \frac{|A(c_1) \cap A(c_2)|}{|A(c_1) \cup A(c_2)|} \quad (4)$$

where $A(c)$ is the auxiliary word set of the concept c, $f_{auxiliary}(c_1, c_2) \in [0, 1]$.

Because changes in recognition accuracy can cause the values of $f_{head}(c_1, c_2)$ and $f_{auxiliary}(c_1, c_2)$ to become larger or smaller, the recognition error ϵ can be positive or negative. Suppose the SaA recognition accuracy of any concept is P; then the recognition error rate is $1-P$. Suppose both the average errors of $f_{head}(c_1, c_2)$ and $f_{auxiliary}(c_1, c_2)$ approximately equal 1/2 according to the changes of their ranges. Combined with $Eq.$ (2), the relationship between ϵ and P is defined as follows:

$$\begin{aligned} \epsilon &= \pm[u \times \frac{1}{2} \times (1-P)^2 + (1-u) \times \frac{1}{2} \times (1-P)^2] \\ &= \pm\frac{1}{2}(1-P)^2 \end{aligned} \quad (5)$$

where $P \in [0, 1]$, and the range of ϵ is [−1/2,1/2]. $P = 1$ means that the SaA recognition is exactly correct, and $P = 0$ means that the SaA recognition is completely incorrect.

4.2.2 Taxonomical features and local density of ontology concepts

For the approaches basing on ontology paths, all possible taxonomical links (i.e., paths) between concepts are calculated, but only the shortest one is kept. To improve the accuracy, other available taxonomical features should be considered. Moreover, the local density of semantic nets affects the similarity between concepts. Although Li et al. [19] considered the local density, they computed it from a corpus that has a problem with data sparseness. Therefore, a corpus-independent local density method is needed.

Definition 2. Impact factor of taxonomical features can be written as follows:

$$F_{taxonomy}(c_1, c_2) = \frac{|\Psi(c_1) \cap \Psi(c_2)|}{|\Psi(c_1) \cup \Psi(c_2)|} \quad (6)$$

where $\Psi(c)$ is the taxonomical feature set of concept c, i.e., the set of all ancestors of c (including c itself). The range of $F_{taxonomy}(c_1, c_2)$ is [0,1].

Definition 3. Impact factor of local density can be written as follows:

$$F_{density}(c_1, c_2) = \sqrt{1 - \frac{||\Phi(c_1)| - |\Phi(c_2)||}{|\Phi(c_1)| + |\Phi(c_2)| + 1}} \quad (7)$$

where $\Phi(c)$ represents the set of concepts subsumed by concept c in the ontology (c itself is not included). The local density equals the sum of concepts in $\Phi(c)$, i.e., $|\Phi(c)|$. The range of $F_{density}(c_1, c_2)$ is [0,1].

4.2.3 Path length and depth of ontology concepts

Because all concepts in the ontology are connected by links, we also describe the effect of path length according to [32]. Concepts at different layers of the hierarchy have different similarities. Therefore, the depth of concept in the ontology should be considered.

Definition 4. Impact factor of the path length can be written as follows:

$$F_{edge}(c_1, c_2) = -\log_{10}\left(\frac{Dist(c_1, c_2)}{2 \times \mathit{\Gamma}}\right) \quad (8)$$

where the parameter $\mathit{\Gamma}$ is the depth of ontology, and $Dist(c_1, c_2)$ represents the number of nodes on the shortest path between c_1 and c_2, including themselves. $Dist(c_1, c_2) = 1$ when c_1 equals c_2.

Definition 5. Impact factor of depth can be written as follows:

$$F_{depth}(c_1, c_2) = 2^{-\frac{|depth(c_1) - depth(c_2)|}{2}} \quad (9)$$

where $depth(c)$ is the depth of concept c in the ontology, namely the levels of c up to the root of the ontology (the depth of the root is 1), and $F_{depth}(c_1, c_2) \in (0, 1]$.

4.2.4 The CCSS method

Synthesizing Definition 1 to Definition 5, we introduce a similarity computational approach to support compound concepts, i.e., CCSS.

Definition 6. CCSS similarity calculation method can be written as follows:

$$Sim_{CCSS}(c_1, c_2) = \begin{cases} [\,\alpha \times F_{edge}(c_1, c_2) + \beta \times F_{taxonomy}(c_1, c_2) \\ \;+(1-\alpha-\beta) \times F_{compound}(c_1, c_2)\,] \\ \;\times F_{depth}(c_1, c_2) \times F_{density}(c_1, c_2) & \text{if } c_1 \neq c_2 \\ \qquad 1 & \text{if } c_1 = c_2 \end{cases} \quad (10)$$

where both α and β are parameters satisfying $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ and $0 \leq \alpha + \beta \leq 1$, and $Sim_{CCSS}(c_1, c_2) \in [0, 1]$. If and only if c_1 equals c_2 or they are synonyms, $Sim_{CCSS}(c_1, c_2)$ equals the maximum value 1. When c_1 and c_2 are semantically unrelated, $Sim_{CCSS}(c_1, c_2)$ is considered approximately equal to the minimum value 0.

5 Semantic-based Compound Keyword Search Scheme

5.1 Overview

In our scheme, VSM and the topic set in a field are used to construct the semantic vector for each keyword. More specifically, in the keyword vector, each element corresponds to a field topic, and its value is the similarity between the topic and the keyword, which is obtained using CCSS. Because the topics are almost invariable, the dimensionality of the keyword vector will not change with the adding or deleting of the keywords or documents, which is helpful in supporting data update.

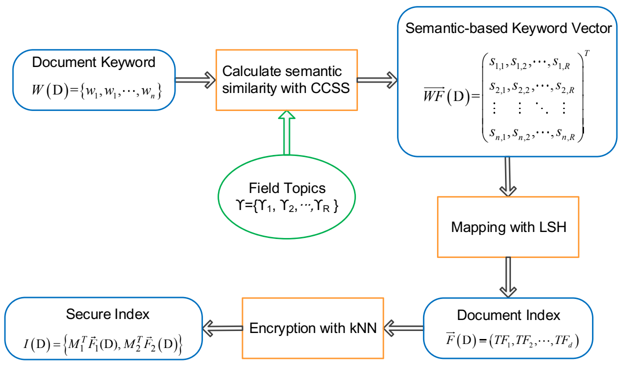
Document Keyword
$W(\mathrm{D})=\{w_1, w_1, \cdots, w_n\}$
Calculate semantic similarity with CCSS
Semantic-based Keyword Vector
$\overrightarrow{WF}(\mathrm{D})=\begin{pmatrix} s_{1,1}, s_{1,2}, \cdots, s_{1,R} \\ s_{2,1}, s_{2,2}, \cdots, s_{2,R} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ s_{n,1}, s_{n,2}, \cdots, s_{n,R} \end{pmatrix}^T$
Field Topics
$\Upsilon=\{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_R\}$
Mapping with LSH
Document Index
$\vec{F}(\mathrm{D})=(TF_1, TF_2, \cdots, TF_d)$
Encryption with kNN
Secure Index
$I(\mathrm{D})=\{M_1^T \vec{F}_1(\mathrm{D}), M_2^T \vec{F}_2(\mathrm{D})\}$

Fig. 3. Proposed secure index generation process

The main idea of the proposed semantic-based compound keyword search (SCKS) scheme is illustrated in Fig. 3. In general, each document contains more than one keyword. If the document index was generated directly using all of the keywords' vectors, it should be expressed as a matrix with high dimensions, such as $\overrightarrow{WF}(D)$ in Fig. 3. To improve efficiency, LSH is introduced in the generation of the document index to map multiple keyword vectors into only one vector, i.e., the document index. The value of each element in the index is the sum of keyword frequency appearing in the document (denoted as TF_i in $\vec{F}(D)$). To preserve privacy, the document index is encrypted with SkNN, e.g., $I(D)$, which is finally outsourced to the cloud. Benefiting from the features of SkNN, SCKS is able to achieve ranked keyword search and can return the *Top k* results.

The trapdoor is generated similarly, except that each element of the query vector is 0 or 1 rather than the frequency of the keywords. LSH is able to map multiple keyword vectors in a query to one vector; hence, SCKS also can support multi-keyword search.

5.2 Scheme Constructions

5.2.1 Generating semantic-based keyword vector

Since each element of the keyword vector corresponds to a topic in a specific field, the topics should be determined in advance by the field experts or generated by the Latent Dirichlet Allocation (LDA) [46] method. Suppose the number of topics in a field is R and the topic set is $\Upsilon = \{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_R\}$. The generation process of the keyword vector is shown in Fig. 4. For a keyword w, the algorithm *KeywordVectorGen* first initializes an R-dimension vector $\overrightarrow{wf}$ and sets the value of each element to 0. Next, it calculates the semantic similarity s_i between w and each field topic Υ_i using CCSS. Finally, the semantic-based keyword vector is output as $\overrightarrow{wf} = (s_1, s_2, \cdots, s_R)$.

Algorithm 2: KeywordVectorGen

```
Input: a keyword $w$ and a set of field topics $\Upsilon = \{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_R\}$;
Output: the semantic vector $\overrightarrow{wf}$ of keyword $w$;
1: initialize the semantic vector $\overrightarrow{wf} = (0, 0, \cdots, 0)_R$;
2: for $i = 1; i <= R; i++$ do
3:     $s_i = Sim_{CCSS}(w, \Upsilon_i)$
4: end for
5: set $\overrightarrow{wf} = (s_1, s_2, \cdots, s_R)$;
6: return $\overrightarrow{wf}$;
```

Fig. 4. Algorithm of keyword vector generation

5.2.2 Generating document index

After the semantic-based vectors for keywords in the document are generated, the LSH function is introduced to map the keyword vectors to a group of buckets. First, we initialize a group of buckets and number them from 0 to $d - 1$, where d is the total number of buckets. Then, each keyword vector is mapped using the functions in an LSH family, which produces a set of hash values. Next, a universal hash function is used to map the set of LSH values to a bucket number. Based on the characteristics of LSH, the semantically related keywords will be mapped to the same bucket with a high probability. Finally, the document index, which is also a vector, is generated by the group of buckets, and the value of each element is the frequency sum of keywords mapped to that bucket.

To improve accuracy, we choose L independent p-stable LSH function families, and each family contains l LSH functions. For example, the LSH functions in the j^{th} family are $h_{j1}(\vec{v}), h_{j2}(\vec{v}), \cdots, h_{jl}(\vec{v})$. The universal hash function that is used to map the l hash values to a bucket number is defined as $H(\vec{X}) = [a_1x_1 + a_2x_2 + \cdots + a_lx_l] \mod d$, where $\vec{X} = (x_1, x_2, \cdots, x_l)$. Suppose the keyword set of document D is $W(D) = \{w_1, w_2, \cdots, w_n\}$ and the corresponding keyword vectors constructed by Algorithm 2 are $WF(D) = \{\overrightarrow{wf_1}, \overrightarrow{wf_2}, \cdots, \overrightarrow{wf_n}\}$. The concrete algorithm for building an index vector for a document is described in Fig. 5.

On line 7 of Algorithm 3, the keyword vector is mapped with the LSH functions in a family, which produces a set of hash values $g_j(\overrightarrow{wf}_i)$. Next, we utilize $H(\vec{X})$ to reduce the dimensionality of $g_j(\overrightarrow{wf}_i)$ and map it to bucket b_{ji} on line 8. Obviously, the range of b_{ji} is $[0, d-1]$ and b_{ji} corresponds to a bucket number. Line 9 means that the sum of the keywords' frequency is reassigned to the value

Algorithm 3: IndexGen

```
Input: a set of document keyword $W(D) = \{w_1, w_2, \cdots, w_n\}$;
Output: the index $\overrightarrow{F}(D)$ of document $D$;
1: initialize a group of buckets and number them from 0 to $d-1$;
2: initialize the index $\overrightarrow{F}(D) = \{TF_0, TF_1, \cdots, TF_{d-1}\} = \{0, 0, \cdots, 0\}$;
3: for $i = 1; i <= n; i++$ do
4:     $\overrightarrow{wf_i} = KeywordVectorGen(w_i, \Upsilon)$;
5:     get the frequency $tf_i$ of $w_i$ in the document;
6:     for $j = 1; j <= L; j++$ do
7:         $g_j(\overrightarrow{wf_i}) = \{h_{j1}(\overrightarrow{wf_i}), h_{j2}(\overrightarrow{wf_i}), \cdots, h_{jl}(\overrightarrow{wf_i})\}$;
8:         $b_{ji} = H(g_j(\overrightarrow{wf_i})) = [a_1 \cdot h_{j1}(\overrightarrow{wf_i}) + \cdots + a_l \cdot h_{jl}(\overrightarrow{wf_i})] \bmod d$
9:         $TF_{b_{ji}} = TF_{b_{ji}} + tf_i$;
10:    end for
11: end for
12: return $\overrightarrow{F}(D)$;
```

Fig. 5. Algorithm of index generation

of the mapped bucket. Finally, the algorithm outputs the document index as $\overrightarrow{F}(D) = (TF_0, TF_1, \cdots, TF_{d-1})$.

The generation of document index indicates that the process does not rely on any predefined global library and that each document is individually indexed. Hence, any data update, including insert, delete and modify, only involves the document to be updated and will not affect any other documents, which means that our scheme can support data updating efficiently.

5.2.3 Generating query vector

The generation of the query vector is similar to the process for the document index. Suppose the search keyword set is $Q = \{qw_1, qw_2, \cdots, qw_m\}$, where m is the number of keywords in the query. Q can be considered the keyword set of a virtual document QW. Then, the query can be generated by Algorithm 3, except the value assignment for each bucket (i.e., Line 9); i.e., in the generation of a query vector, if the value of bucket b_{ji} is 0, it is set to 1; otherwise, no substitution is made. Hence, the values in a query vector are only 0 or 1. Finally, we obtain the query vector $\overrightarrow{Q} = (qTF_0, qTF_1, \cdots, qTF_{d-1})$.

5.2.4 SCKS scheme

To preserve the privacy of data and users, both the index and query are encrypted with SkNN. The SCKS scheme includes the following 4 algorithms.

- **KeyGen $(\boldsymbol{d})$.** Given a security parameter d that equals the number of buckets in Fig. 5, generate two invertible matrices $M_1, M_2 \in \mathbb{R}^{d\times d}$ and a bit vector $S \in \{0,1\}^d$ of which S_j denotes the j^{th} bit in S. Note that the numbers of 0s and 1s in S should be almost equal. The secret key is $sk = (M_1, M_2, S)$.
- **BuildIndex $(\boldsymbol{sk}, \boldsymbol{D})$.** Generate the index $\overrightarrow{F}(D) = (TF_0, TF_1, \cdots, TF_{d-1})$ for document D. Then, the secure index can be generated as follows.
  1) Index splitting. Split the index $\overrightarrow{F}(D)$ into two d-dimension vectors $\overrightarrow{F_1}(D) = (t_{1,0}, t_{1,1}, \cdots, t_{1,d-1})$ and $\overrightarrow{F_2}(D) = (t_{2,0}, t_{2,1}, \cdots, t_{2,d-1})$ as follows. If $S_j = 1$, set $t_{1,j} = t_{2,j} = TF_j$; otherwise if $S_j = 0$, set $t_{1,j} = TF_j + \tilde{r}_1$ and $t_{2,j} = TF_j - \tilde{r}_1$ where $\tilde{r}_1$ is a random number. The index satisfies $\overrightarrow{F}(D) = \{\overrightarrow{F_1}(D), \overrightarrow{F_2}(D)\} = 1/2(\overrightarrow{F_1}(D) + \overrightarrow{F_2}(D))$.
  2) Index encryption. Encrypt $\{\overrightarrow{F_1}(D), \overrightarrow{F_2}(D)\}$ with (M_1, M_2) and get the secure index $I(D) = \{M_1^T\overrightarrow{F_1}(D), M_2^T\overrightarrow{F_2}(D)\}$.
- **Trapdoor $(\boldsymbol{sk}, \boldsymbol{Q})$.** Generate the query $\overrightarrow{Q} = (qTF_0, qTF_1, \cdots, qTF_{d-1})$ as detailed in Section 5.2.3. Then the trapdoor can be generated as follows.
  1) Query splitting. Split the query $\overrightarrow{Q}$ into two d-dimension vectors $\overrightarrow{Q_1} = (qt_{1,0}, qt_{1,1}, \cdots, qt_{1,d-1})$ and $\overrightarrow{Q_2} = (qt_{2,0}, qt_{2,1}, \cdots, qt_{2,d-1})$. The splitting method of $\overrightarrow{Q}$ is contrary to the method of index $\overrightarrow{F}(D)$; i.e., if $S_j = 0$, set $qt_{1,j} = qt_{2,j} = qTF_j$; if $S_j = 1$, set $qt_{1,j} = qTF_j + \tilde{r}_2$ and $qt_{2,j} = qTF_j - \tilde{r}_2$ where $\tilde{r}_2$ is a random number. Similarly, the query satisfies $\overrightarrow{Q} = \{\overrightarrow{Q_1}, \overrightarrow{Q_2}\} = 1/2(\overrightarrow{Q_1} + \overrightarrow{Q_2})$.
  2) Query encryption. Encrypt $\{\overrightarrow{Q_1}, \overrightarrow{Q_2}\}$ with (M_1^{-1}, M_2^{-1}) and get the trapdoor $T(Q) = \{M_1^{-1}\overrightarrow{Q_1}, M_2^{-1}\overrightarrow{Q_2}\}$.
- **Search $(\boldsymbol{I(D)}, \boldsymbol{T(Q)})$.** To search documents that are semantically related to trapdoor $T(Q)$, the cloud server calculates the inner products of $T(Q)$ with each secure index $I(D)$ as follows.

$$\begin{aligned}
&C(D,Q)\\
&= I(D)^T \cdot T(Q)\\
&= \{M_1^T\overrightarrow{F_1}(D), M_2^T\overrightarrow{F_2}(D)\}^T \cdot \{M_1^{-1}\overrightarrow{Q_1}, M_2^{-1}\overrightarrow{Q_2}\}\\
&= \overrightarrow{F_1}(D)^T M_1 \cdot M_1^{-1}\overrightarrow{Q_1} + \overrightarrow{F_2}(D)^T M_2 \cdot M_2^{-1}\overrightarrow{Q_2}\\
&= \overrightarrow{F_1}(D)^T \cdot \overrightarrow{Q_1} + \overrightarrow{F_2}(D)^T \cdot \overrightarrow{Q_2}\\
&= \{\overrightarrow{F_1}(D), \overrightarrow{F_2}(D)\}^T \cdot \{\overrightarrow{Q_1}, \overrightarrow{Q_2}\}\\
&= \overrightarrow{F}(D) \cdot \overrightarrow{Q}
\end{aligned} \tag{11}$$

Therefore, the inner product of the secure index $I(D)$ and the trapdoor $T(Q)$ equals the inner product of index $\overrightarrow{F}(D)$ and the query $\overrightarrow{Q}$. Based on the characteristics of LSH, if the document keyword and the query keyword are equal or related, they will be mapped to the same bucket with a high probability. Hence, the non-zero elements of the index and query are roughly the same, and their inner product indicates their correlation coefficients. Finally, the cloud server returns the *Top k* most relevant results to users based on the correlation coefficients.

To protect the privacy of documents, each can be encrypted by a traditional encryption scheme such as AES. Then, the encrypted documents are outsourced to the cloud servers with the corresponding secure index. When the user receives the encrypted documents from the cloud server, he can decrypt them with the AES key to obtain the plaintext. In the query process, the users need do nothing except decrypting the relevant documents, as shown in Fig. 1.

5.3 Security Analysis

To prove the security of a searchable symmetric encryption (SSE) scheme, Curtmola et al. [38], [39] defined semantic security under *non-adaptive* and *adaptive* models, respectively. Both models have been widely used in some existing schemes [2]–[4]. In this section, we will proof that SCKS is secure under the *non-adaptive* model. Before stating the definition of *non-adaptive* model, we introduce some notions based on similar ones in [38], [39].

Definition 7 (History)**.** Let Δ be a document collection. An m-query *history* over Δ is a tuple $\mathcal{H}_m = (\Delta, Q_m)$ that includes the document collection Δ and m queries $Q_m = (q_1, \cdots, q_m)$.

Definition 8 (View)**.** Let Δ be a document collection and $\mathcal{H}_m$ be a history over m queries. A view of $\mathcal{H}_m$ under secret key sk is defined as $V_{sk}(\mathcal{H}_m) = \{Enc(\Delta), I(\Delta), T(Q_m)\}$, where $Enc(\Delta)$ is the encrypted document collection, $I(\Delta)$ is the secure indexes of Δ and $T(Q_m)$ is the trapdoors of m queries.

Definition 9 (Access pattern)**.** Let Δ be a document collection and $\mathcal{H}_m$ be a history over m queries. The access pattern of $\mathcal{H}_m$ contains the search results and is denoted as a tuple $\Omega(\mathcal{H}_m) = (\varphi(q_1), \cdots, \varphi(q_m))$ where $\varphi(q_i) = \{(D_j, \mu_{i,j})_{q_i \subset D_j}, 1 \leq j \leq |\Delta|\}$ and $\mu_{i,j}$ denotes the correlation coefficients between query q_i and document $D_j \in \Delta$.

Definition 10 (Search pattern)**.** Let Δ be a document collection and $\mathcal{H}_m$ be a history over m queries. The search pattern of $\mathcal{H}_m$ is a symmetric binary matrix $\Pi(\mathcal{H}_m)$ such that for $1 \leq i, j \leq m$, the element in the i^{th} row and j^{th} column is 1 if $q_i = q_j$, and 0, otherwise.

Definition 11 (Trace)**.** Let Δ be a document collection and $\mathcal{H}_m$ be a history over m queries. The trace of $\mathcal{H}_m$ is defined as $Tr(\mathcal{H}_m) = \{|D_1|, \cdots, |D_{|\Delta|}|, \Omega(\mathcal{H}_m), \Pi(\mathcal{H}_m)\}$ which consists of the lengths of documents in Δ, the access pattern and the search pattern of $\mathcal{H}_m$.

Specifically, the *trace* contains all of the information that we are willing to leak to the adversary. In other words, nothing can be leaked from the secure index and trapdoor beyond the trace. Next, we state the security definition for SSE in [39]. Under the *non-adaptive* model, the adversary is required to generate the history at once, i.e., it is not allowed to see the secure indexes of the document collection or the trapdoors of any queries it chooses before it finishes generating the history.

Definition 12 (Non-adaptive semantic security [39])**.** Let $d \in \mathbb{Z}_{\mathbb{N}}$ be the security parameter of an SSE scheme, $\mathcal{A}$ be an adversary, and $\mathcal{S}$ be a simulator. Consider the following probabilistic experiments:

```
$\mathbf{Real}_{\mathcal{A}}(d)$
  $sk \leftarrow KeyGen(d)$
  $\mathcal{H} = (\Delta, Q) \leftarrow \mathcal{A}(d)$
  $Enc(\Delta) \leftarrow$ Encrypt $\Delta$
  for $1 \leq i \leq |\Delta|$
    $I(D_i) \leftarrow BuildIndex(sk, D_i)$                (12)
  for $1 \leq j \leq m$
    $T(q_j) \leftarrow Trapdoor(sk, q_j)$
  Let $I(\Delta) = \{I(D_1), I(D_2), ..., I(D_{|\Delta|})\}$
  Let $T(Q) = \{T(q_1), T(q_2), ..., T(q_m)\}$
  Output $V = \{Enc(\Delta), I(\Delta), T(Q)\}$

$\mathbf{Sim}_{\mathcal{A},\mathcal{S}}(d)$
  $\mathcal{H} = (\Delta, Q) \leftarrow \mathcal{A}(d)$
  $V \leftarrow S(Tr(\mathcal{H}))$
  Output $V$
```

An SSE scheme is non-adaptively semantically secure if for all polynomial-size adversaries $\mathcal{A}$, there exists a polynomial-size simulator $\mathcal{S}$ such that for all polynomial-size distinguishers $\mathcal{D}$,

$$|\Pr[\mathcal{D}(V \leftarrow \mathbf{Real}_{\mathcal{A}}(d)) = 1] - \Pr[\mathcal{D}(V \leftarrow \mathbf{Sim}_{\mathcal{A},\mathcal{S}}(d)) = 1]| \leq \text{negl}(d) \tag{13}$$

where negl(d) denotes a negligible function in d.

Theorem 1. *SCKS is a non-adaptively secure SSE scheme.*

Proof. We adopt a simulation-based proof process similar to that used in [39] to prove Theorem 1. According to Definition 12, we will show that there exists a polynomial-size simulator $\mathcal{S}$ such that for all polynomial-size adversaries $\mathcal{A}$, the outputs of $\mathbf{Real}_{\mathcal{A}}(d)$ and $\mathbf{Sim}_{\mathcal{A},\mathcal{S}}(d)$ are indistinguishable. Let Enc be a semantically secure symmetric encryption scheme. Given the trace $Tr(\mathcal{H}_m) = \{|D_1|, \cdots, |D_{|\Delta|}|, \Omega(\mathcal{H}_m), \Pi(\mathcal{H}_m)\}$ of a history $\mathcal{H}_m$, $\mathcal{S}$ first chooses two random invertible matrices $M_1', M_2' \in \mathbb{R}^{d \times d}$ and a splitting vector $S' \in \{0,1\}^d$ with the restriction that the numbers of 0s and 1s in S' should be almost equal, and sets the secret key as $sk' = (M_1', M_2', S')$. Then, $\mathcal{S}$ generates $V' = \{Enc(\Delta'), I(\Delta'), T(Q_m')\}$ as follows.

- **Generating $\boldsymbol{Enc(\Delta')}$.** $\mathcal{S}$ selects a random string $Enc(D_j') = \{0,1\}^{|D_j|}$ where $|D_j|$ is included in the trace and $1 \leq j \leq |\Delta|$. Then, $\mathcal{S}$ outputs $Enc(\Delta') = \{Enc(D_1'), \cdots, Enc(D_{|\Delta|}')\}$.
- **Generating $\boldsymbol{T(Q_m')}$.** For each $q_i' \in Q_m', 1 \leq i \leq m$, $\mathcal{S}$ generates a query vector $\overrightarrow{q_i}' \in \{0,1\}^d$. Then, $\mathcal{S}$ performs the **Trapdoor** algorithm to split $\overrightarrow{q_i}'$ into two d-dimension vectors with S' and encrypt them with matrices $(M_1^{'-1}, M_2^{'-1})$. Finally, $\mathcal{S}$ outputs $T(Q_m') = \{T(q_1'), \cdots, T(q_m')\}$.
- **Generating $\boldsymbol{I(\Delta')}$.** For each document $D_j' \in \Delta', 1 \leq j \leq |\Delta|$, $\mathcal{S}$ generates a d-dimension vector with each element assigned 0 as the index, denoted as $\overrightarrow{F}(D_j')$. Next, the value of $\overrightarrow{F}(D_j')$ is assigned as follows.
  a) For each $q_i \in Q_m$, if $q_i \subset D_j$ (i.e., the correlation coefficient $\mu_{i,j} \neq 0$, which is contained in the trace), select a random $c_{i,j}$ and set $\overrightarrow{F}(D_j') = \overrightarrow{F}(D_j') + c_{i,j}\overrightarrow{q_i}'$.
  b) $\mathcal{S}$ tunes each $c_{i,j}$ to satisfy $\overrightarrow{F}(D_j') \cdot \overrightarrow{q_i}' = \mu_{i,j}, 1 \leq i \leq m, 1 \leq j \leq |\Delta'|$.
  c) $\mathcal{S}$ performs the **BuildIndex** algorithm to generate $I(D_j')$ by splitting $\overrightarrow{F}(D_j')$ into two d-dimension vectors with S' and encrypting them with matrices (M_1', M_2'). Finally, $\mathcal{S}$ outputs $I(\Delta') = \{I(D_1'), \cdots, I(D_{|\Delta|}')\}$.

Let $V_{sk}(\mathcal{H}_m) = \{Enc(\Delta), I(\Delta), T(Q_m)\}$ be the outcome of a $\mathbf{Real}_{\mathcal{A}}(d)$ experiment. Through the above construction, we can claim that no polynomial-size distinguisher $\mathcal{D}$ can distinguish between V' and $V_{sk}(\mathcal{H}_m)$ for any $\mathcal{H}_m$; otherwise, $\mathcal{D}$ could distinguish between at least one of the elements of V' and its corresponding element in $V_{sk}(\mathcal{H}_m)$. However, $\mathcal{D}$ cannot do so because each element of V' is indistinguishable from its corresponding element in $V_{sk}(\mathcal{H}_m)$. The reason is as follows.

- Encrypted documents. Because the encryption key of $Enc(D_j)$ is private, the semantical security of Enc will guarantee that $Enc(D_j')$ is indistinguishable from $Enc(D_j)$. Hence, $Enc(\Delta')$ and $Enc(\Delta)$ are indistinguishable.

- Index and trapdoor. The index and trapdoor are generated by encrypting a vector with SkNN encryption, respectively. Because the SkNN encryption has achieved indistinguishability by introducing the splitting vector and two random invertible matrices, the elements in $I(\Delta')$ and $T(Q'_m)$ are indistinguishable from the elements in $I(\Delta)$ and $T(Q_m)$, respectively.

Therefore, no polynomial-size distinguisher $\mathcal{D}$ can distinguish view V' from $V_{sk}(\mathcal{H}_m)$ for any $\mathcal{H}_m$, which indicates that SCKS is a *non-adaptively* secure SSE scheme. □

6 Security-Enhanced SCKS Scheme

In the *adaptive* model, the adversary is allowed to submit queries adaptively, i.e., submitting the next query after receiving the outcomes of previous queries. Thus, the adversary can decide the next query depending upon the previous outcomes. However, the SkNN encryption has been proved vulnerable under linear analysis [47]. When the server obtains d query vectors and the corresponding trapdoors, it can enforce linear analysis to recover the index of documents. Hence, SCKS is vulnerable under the adaptive model. In this section, we propose a security-enhanced SCKS (SE-SCKS) scheme that is secure under the adaptive model.

6.1 Constructions of SE-SCKS

Inspired by the method mentioned in [2], we introduce a pseudo-random function in the generation of document indexes and trapdoors to improve the security of SCKS. In SE-SCKS, most processes are the same as SCKS except for the following steps.

(1) Generating a hash key pool. In the **KeyGen** algorithm, in addition to the secret key $sk = \{M_1, M_2, S\}$, generate a hash key pool $HK = \{key_i | key_i \leftarrow \{0,1\}^\theta, 1 \le i \le l \times L\}$ with another given parameter θ, where L is the number of LSH families, and l is the number of LSH functions in a family.

(2) Introducing the pseudo-random function. In Algorithm 3, choose a pseudo-random function $f : \{0,1\}^* \times \{0,1\}^\theta \to \{0,1\}^*$ in addition to the L independent p-stable LSH function families. Then, the LSH functions used in the algorithm are replaced by new functions $G = \{hf_i | hf_i = f_{key_i} \circ h_i, h_i \in LSH, 1 \le i \le l \times L\}$.

Because the pseudo-random function is strongly collision-resistant, using this function will not affect the search results. In practice, HMAC-SHA1 [48] can be used as the pseudo-random function because its collision rate is extremely low.

6.2 Security Analysis of SE-SCKS

The security definition in the *adaptive* model is similar to the one in the non-adaptive model except that the adversary is allowed to choose the history adaptively. Specifically, the adversary initially submits the document collection and receives the corresponding index before he chooses the first query. Then, he will receive the query's trapdoor before he chooses the next query, and so on. Intuitively, the adversary in the adaptive model is able to perform more sophisticated attacks than in the previous case.

Definition 13 (Adaptive semantic security [39])**.** Let $d \in \mathbb{Z}_\mathbb{N}$ be the security parameter of an SSE scheme, $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_m)$ be an adversary such that $m \in \mathbb{Z}_\mathbb{N}$, and $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_m)$ be a simulator. Consider the following probabilistic experiments:

```
$\mathbf{Real}^*_{\mathcal{A}}(d)$
  $sk \leftarrow KeyGen(d)$
  $\Delta \leftarrow \mathcal{A}_0(d)$
  $Enc(\Delta) \leftarrow$ Encrypt $\Delta$
  for $1 \le i \le |\Delta|$
    $I(D_i) \leftarrow BuildIndex(sk, D_i)$
  Let $I(\Delta) = \{I(D_1), ..., I(D_{|\Delta|})\}$

  $q_1 = \mathcal{A}_1(Enc(\Delta), I(\Delta))$
  $T(q_1) \leftarrow Trapdoor(sk, q_1)$
  for $2 \le j \le m$
    $q_j \leftarrow \mathcal{A}_j(Enc(\Delta), I(\Delta), T(q_1), ..., T(q_{j-1}))$
    $T(q_j) \leftarrow Trapdoor(sk, q_j)$
    Let $T(Q) = \{T(q_1), ..., T(q_j)\}$
    Output $V = \{Enc(\Delta), I(\Delta), T(Q)\}$

$\mathbf{Sim}^*_{\mathcal{A},\mathcal{S}}(d)$
  $\Delta \leftarrow \mathcal{A}_0(d)$
  $(Enc(\Delta), I(\Delta)) \leftarrow \mathcal{S}_0(Tr(\Delta))$
  $q_1 \leftarrow \mathcal{A}_1(Enc(\Delta), I(\Delta))$
  $T(q_1) \leftarrow \mathcal{S}_1(Tr(\Delta, q_1))$
  for $2 \le j \le m$
    $q_j = \mathcal{A}_j(Enc(\Delta), I(\Delta), T(q_1), ..., T(q_{j-1}))$
    $T(q_j) \leftarrow \mathcal{S}_j(Tr(\Delta, q_1, ..., q_j))$
    Let $T(Q) = \{T(q_1), ..., T(q_j)\}$
    Output $V = \{Enc(\Delta), I(\Delta), T(Q)\}$
```
(14)

An SSE scheme is adaptively semantically secure if for all polynomial-size adversaries $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_m)$, there exists a polynomial-size simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_m)$ such that for all polynomial-size distinguishers $\mathcal{D}$,

$$|\Pr[\mathcal{D}(V \leftarrow \mathbf{Real}^*_{\mathcal{A}}(d)) = 1] - \Pr[\mathcal{D}(V \leftarrow \mathbf{Sim}^*_{\mathcal{A},\mathcal{S}}(d)) = 1]| \le \text{negl}(d) \tag{15}$$

Theorem 2. *SE-SCKS is an adaptively secure SSE scheme.*

Proof. According to Definition 13, we will show that there exists a polynomial-size simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_m)$ such that for all polynomial-size adversaries $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_m)$, the outputs of $\mathbf{Real}^*_{\mathcal{A}}(d)$ and $\mathbf{Sim}^*_{\mathcal{A},\mathcal{S}}(d)$ are indistinguishable. Let Enc be a semantically secure symmetric encryption scheme. Consider simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_m)$, which adaptively generates a view $V' = \{Enc(\Delta'), I(\Delta'), T(Q'_m)\}$ as follows.

- $\mathcal{S}_0(Tr(\Delta))$. $\mathcal{S}_0$ initially randomly chooses two invertible matrices $M'_1, M'_2 \in \mathbb{R}^{d \times d}$, a splitting vector $S' \in \{0,1\}^d$ and a pseudo-random permutation G', and sets $sk' = (M'_1, M'_2, S', G')$. Then, $\mathcal{S}_0$ selects a random string $Enc(D'_j) = \{0,1\}^{|D_j|}$ for $1 \le j \le |\Delta|$, where $|D_j|$ is included in $Tr(\Delta)$, and sets $Enc(\Delta') = \{Enc(D'_1), \cdots, Enc(D'_{|\Delta|})\}$. Next, $\mathcal{S}_0$ generates a d-dimension random vector $\overrightarrow{F}(D'_j)$ as the index and performs the **BuildIndex** algorithm to compute $G'(\overrightarrow{F}(D'_j))$ and generate $I(D'_j)$ by splitting $G'(\overrightarrow{F}(D'_j))$ into two d-dimension vectors with S' and encrypting them with matrices (M'_1, M'_2). Finally, $\mathcal{S}_0$ sets $I(\Delta') = \{I(D'_1), \cdots, I(D'_{|\Delta|})\}$ and outputs $\{Enc(\Delta'), I(\Delta')\}$.
  Analysis: Because Enc is semantically secure and the encryption key is kept private, $Enc(D'_j)$ is indistinguishable from a real ciphertext $Enc(D_j)$ for $1 \le j \le |\Delta|$. Hence, $Enc(\Delta')$ and $Enc(\Delta)$ are indistinguishable. On the other hand, even though the

attacker can recover $G'(\overrightarrow{F}(D'_j))$ through a successful linear analysis, no polynomial-size distinguisher can distinguish the output of the linear analysis, i.e. $G'(\overrightarrow{F}(D'_j))$, from a random string, due to the indistinguishability of pseudo-random function G. Additionally, because SkNN has achieved indistinguishability by introducing the splitting vector and two random invertible matrices, the elements in $I(\Delta')$ are indistinguishable from the elements in $I(\Delta)$.

- $\mathcal{S}_1(Tr(\Delta, q_1))$. $\mathcal{S}_1$ generates a d-dimension query vector $\overrightarrow{q_1}'$ with each element set to 0. Next, the value of $\overrightarrow{q_1}'$ is assigned as follows.
  a) For each $D_j \in \Delta$, if $q_1 \subset D_j$ (i.e., the correlation coefficient $\mu_{1,j} \neq 0$, which is contained in $Tr(\Delta, q_1)$), select a random $c_{1,j}$ and set $\overrightarrow{q_1}' = \overrightarrow{q_1}' + c_{1,j}\overrightarrow{F}(D'_j)$.
  b) $\mathcal{S}_1$ tunes each $c_{1,j}$ to satisfy $\overrightarrow{F}(D'_j) \cdot \overrightarrow{q_1}' = \mu_{1,j}$.
  c) $\mathcal{S}_1$ performs the **Trapdoor** algorithm to compute $G'(\overrightarrow{q_1}')$ and generate $T(q'_1)$ by splitting $G'(\overrightarrow{q_1}')$ into two d-dimension vectors with S' and encrypting them with matrices $(M_1'^{-1}, M_2'^{-1})$. Finally, $\mathcal{S}_1$ outputs $T(q'_1)$.

  Analysis: Because SkNN has achieved indistinguishability by introducing the splitting vector and two random invertible matrices, $T(q'_1)$ is indistinguishable from $T(q_1)$.
- $\mathcal{S}_i(Tr(\Delta, q_1, \cdots, q_i))$ for $2 \leq i \leq m$. $\mathcal{S}_i$ first checks whether q_i has appeared before by checking whether there exists a $1 \leq j \leq i-1$ such that $\Pi(\mathcal{H}_m)[i,j] = 1$. If q_i appeared previously, $\mathcal{S}_i$ retrieves the trapdoor previously generated for q_i and uses it as $T(q'_i)$. Otherwise, if q_i has not appeared previously, $\mathcal{S}_i$ generates a trapdoor the same way $\mathcal{S}_1$ does. Finally, $\mathcal{S}_i$ outputs $T(q'_i)$.

  Analysis: Similarly, $T(q'_i)$ is indistinguishable from $T(q_i)$ due to the indistinguishability of the SkNN encryption.

Therefore, there is no polynomial-size distinguisher $\mathcal{D}$ that can distinguish view V' from $V_{sk}(\mathcal{H}_m)$ which is the outcome of a $\mathbf{Real}^*_{\mathcal{A}}(d)$ experiment for any $\mathcal{H}_m$, which indicates SE-SCKS is an adaptively secure SSE scheme. □

7 Performance Evaluation

7.1 Experiments and Analysis of CCSS

7.1.1 Dataset

We utilize the 2012 ACM Computing Classification System (CSS) [1] as our experiment data source and employ the tool Protégé [2] and the reasoning tool Jena [3] to construct a domain ontology of Information System (named IS ontology) that contains 318 noun concepts, with a subset of 295 compound concepts. The following experiments are performed based on this dataset.

First, we recognize the SaA of all concepts in this IS ontology by using algorithm SAR. The recognition results are summarized in Table 1, where P_s and P_c indicate the recognition accuracy of subordination concepts and coordination concepts, respectively.

TABLE 1
The recognition results of the concepts in IS ontology by SAR

Num of concepts	Subordination concepts			Coordination concepts			Average accuracy P
	Total	Correct	P_s	Total	Correct	P_c	
318	302	289	95.7%	16	14	87.5%	95.28%

To have an objective evaluation dataset, we select 30 concept pairs from the *Information Storage System* branch of the IS ontology and assess their similarity respective by 18 computer experts and 47 ordinary computer personnel. The assessment rating is on a scale from 0 (semantically unrelated) to 4 (highly synonymous). Because the computer experts and ordinary computer personnel have different degrees of knowledge on the selected concepts, the results can be divided into three categories: 1) results assessed by computer experts (Expert for short); 2) results assessed by ordinary computer personnel (Ordinary for short) and 3) the average results of Expert and Ordinary (Average for short).

In addition, to compare with the corpus-based IC approaches, the corpus we used is the abstract set of 38,469 papers that are primarily classified as information systems in the ACM Digital Library and have publication dates from 2002 to 2011.

7.1.2 Experimental results and analysis

We evaluate the performance of our method through two experiments. The first one compares CCSS with some other approaches against three categories of human assessments; the second one tests the performance of CCSS versus different SaA recognition accuracies P.

There are 15 approaches in the first experiment, including Rada [30], L&CH [32], Resnik [35], J&C [34], Lin [33], Seco [36], Sánchez [22], Sánchez [18], Li [19], Pirró [20] and our approach CCSS. To make our scheme more believable, we set the recognition accuracy P of SaA to be the worst accuracy in Table 1; i.e., we set $P = P_c = 87.5\%$. Other parameters of CCSS are $\alpha = 0.5$, $\beta = 0.15$ and $u = 0.75$. The parameters for the other approaches are chosen according to the *best* correlation values reported in the authors' experiments. Then we compute the Pearson correlation coefficient for each approach. As shown in Fig. 6(a), (b) and (c), compared with other approaches, CCSS provides the highest accuracy because it synthesizes the concept constituent features, taxonomical features, local density, path length and depth. Concerning efficiency, our approach retains low computation complexity by only requiring the information sources of ontology and the constituent features of the concepts. Moreover, the SaA of concepts need only be recognized and extracted once, without re-recognizing in future computations.

In the second experiment, we test the performance of our CCSS approach versus different SaA recognition accuracies P, as shown in Fig. 6(d). $P = 100\%$ means that the SaA recognition of compounds is exactly correct; i.e., $\epsilon = 0$. In this case, the correlations of the three categories are 0.8882 for Expert, 0.8309 for Ordinary and 0.8416 for Average. When P equals the minimum value 30% and the recognition

1. ACM CSS. http://www.acm.org/about/class/2012/.
2. Protégé. http://protege.stanford.edu/.
3. Jena. http://jena.apache.org/.

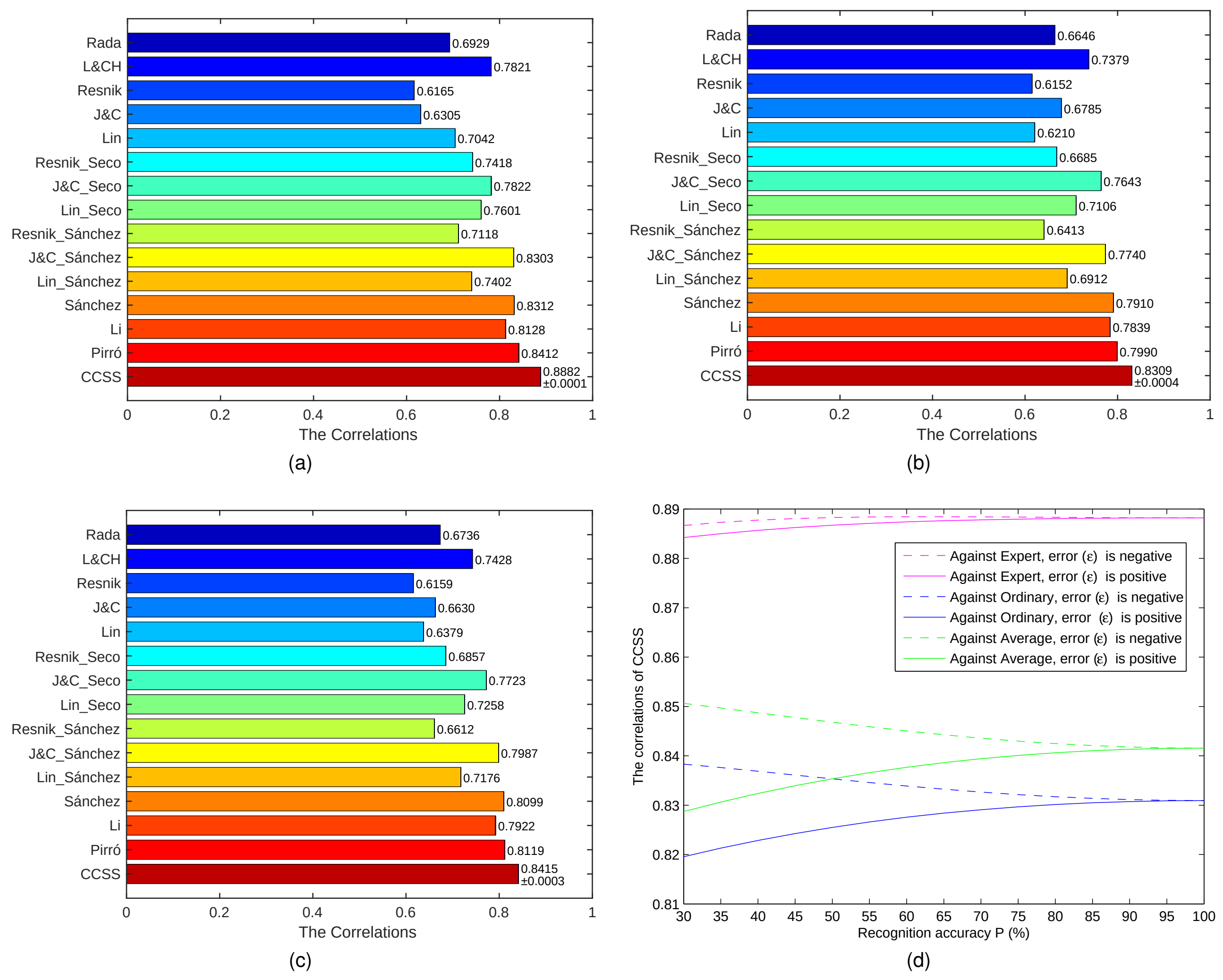

Fig. 6. Pearson correlations of each approach against (a) Expert, (b) Ordinary, (c) Average, and (d) Pearson correlations of CCSS versus different SaA recognition accuracy

error ϵ is positive, i.e., with $P = 30\%$ and $\epsilon > 0$, the correlations of the three categories are 0.8842 for Expert, 0.8196 for Ordinary and 0.8287 for Average. In such a worst-case situation, the accuracy of our approach still remains the highest. In practice, recognition accuracy ϵ is always greater than 30%.

Discussion. To summarize, CCSS can provide high accuracy without depending upon corpus. The weights of each impact factor in CCSS are adjusted by the three parameters (see $Eq.$ (10)). In practice, one can achieve the best performance by adjusting the optimal value for each parameter individually, such as using the variable-controlling method [20].

7.2 Experiments and Analysis of SCKS

7.2.1 Indicators and dataset

In this experiment, we compare SCKS with the search scheme WYL+ [2], which achieves fuzzy search over encrypted data based on LSH and Bloom Filter. The comparison focuses on the following three factors.

- Accuracy, denoted as $\mathcal{P}$. In this factor, the order of the outputs is discarded and the weight of each result is the same.
- Recall rate, denoted as $\mathcal{R}$.
- Average accuracy, denoted as $\mathcal{AP}$, which indicates the average accuracy at many recall rates. The order of the outputs is considered in this factor.

In the following experiments, the parameters of WYL+ are set as $l = 10, L = 30, d = 8000$ and $n = 767$, and the parameters of SCKS are set as $l = 5, L = 30, d = 400$ and $n = 318$, where l and L are the parameters in LSH functions, d is the dimensionality of document index, and n is the dimensionality of keyword vector. From the parameters, we can determine that the index dimensionality of WYL+ is 20 times that of SCKS, which causes the index generation and keyword search of SCKS to be more efficient than those of WYL+.

The dataset in the experiments is the abstract set of 38,469 papers, which are primarily classified as information systems in the ACM Digital Library and have publication dates from 2002 to 2011.

7.2.2 Accuracy and recall rate comparison

(1) Exact keyword search

For exact search, as shown in Table 2, the accuracy $\mathcal{P}$, average accuracy $\mathcal{AP}$ and recall rate $\mathcal{R}$ of SCKS are all greater than those of WYL+ because SCKS uses CCSS to measure the similarity between keywords, which greatly improves the search accuracy of the compound concepts.

TABLE 2
Experimental results of exact search

Indicators	SCKS	WYL+
$\mathcal{P}$	**0.7682**	0.6949
$\mathcal{AP}$	**0.5344**	0.5163
$\mathcal{R}$	**0.8827**	0.7412

(2) Semantic-based keyword search

The semantic correlation is determined by a correlation threshold; i.e., a document is considered semantically related to a query when the correlation coefficient between them equals or is greater than the threshold. The experimental results of semantic search are shown in Table 3 and indicate that the accuracy $\mathcal{P}$, average accuracy $\mathcal{AP}$ and recall rate $\mathcal{R}$ of SCKS are all much greater than those of WYL+. There are two reasons:

(1) SCKS generates the keyword vector by measuring the semantic correlation between the keyword and each field topic. It introduces semantic information to the keyword vector, whereas WYL+ only introduces the characteristic of keyword spelling.

(2) In the generation of document index, SCKS set each element of the index to be the frequency of keywords mapped to that element. Thus, SCKS introduces the weight of the keyword in the document into the index. However, the scheme of WYL+ assigns each element of the index to 0 or 1 and never considers the weight of keywords.

TABLE 3
Experimental results of semantic search

threshold	$\mathcal{P}$		$\mathcal{AP}$		$\mathcal{R}$	
	SCKS	WYL+	SCKS	WYL+	SCKS	WYL+
0.5	**0.7861**	0.5647	**0.5450**	0.2968	**0.9824**	0.6865
0.6	**0.7818**	0.6707	**0.5388**	0.3955	**0.9032**	0.7159
0.7	**0.7852**	0.5717	**0.5434**	0.3022	**0.9085**	0.6890

Discussion. In both of the exact keyword search and semantic-based search, the accuracy $\mathcal{P}$, average accuracy $\mathcal{AP}$ and recall rate $\mathcal{R}$ of SCKS are all much greater than those of WYL+. On the other hand, since most of the existing semantic-based search schemes need to predefine a global library [12]–[15], the accuracy, average accuracy and recall rate all depend on the quality of the library. However, the libraries, datasets and implementations used in these schemes are unavailable and the accuracy, average accuracy and recall rate are not reported in the papers . Hence, we cannot compare our scheme with them. Compared with the schemes focusing on exact keyword search [25] [49], SCKS is much more flexible because it can return the documents semantically related with the query keywords, beside exactly match.

7.2.3 Efficiency analysis

In WYL+, the dimensionality of invertible matrixes used for encryption is 8000 (i.e., $d = 8000$). When we calculate the inverse matrix with UJMP (Universal Java Matrix Package), the Eclipse platform (64-bit) outputs an error of "Available Memory Is Low", that confirms WYL+ is rather costly in the generation of secure index and trapdoor. However, invertible matrixes used in SCKS only need 400 dimensions; therefore, we only test the efficiency of SCKS in the following experiments. The experiments are performed on a PC with Intel (R) Quad-Core (TM) CPU@3.4GHz processor, 8GB RAM, Microsoft Windows 7 and Eclipse 4.5.0 (64-bit).

(1) Secure index and trapdoor generation

For a single document, the time of secure index generation increases linearly with the number of keywords in the document, as shown in Fig. 7(a), because more keyword vectors should be generated with CCSS and processed with LSH functions when the number of keywords increases. Compared with SkNN and LSH, the time costed by CCSS is little and can be ignored. Similarly, the time of trapdoor generation also increases linearly with the number of keywords in a query, as shown in Fig. 7(c).

Because SCKS generates one independent index item for each document, the total time for secure index generation increases linearly with the number of documents in the dataset, as shown in Fig. 7(b). Note that the secure index item for each document is generated only once, except when the document is modified. The update operations, including insert, delete and modify, only involve the updated document and will not affect any others. Hence, the dataset can be expanded freely without affecting the previous inserted documents.

(2) Keywords Search

Fig. 7(d) shows that the search time remains stable when the number of keywords in the query increases because SCKS maps multiple query keywords to only one trapdoor and only searches for one round no matter how many keywords are queried.

Because there are more inner products should be calculated when the documents increase, the search time increases linearly with the number of documents in the dataset, as shown in Fig. 7(e).

8 Conclusion

Focusing on the keyword search over encrypted cloud data, we propose a semantic-based compound keyword search (SCKS) scheme in this paper. To accurately extract the semantic information of keywords, we first propose an ontology-based compound concept semantic similarity calculation method (CCSS), which greatly improves the accuracy of similarity measurement between compound concepts by comprehensively considering the compound features and a variety of information sources in ontology. Then, the SCKS scheme is constructed by integrating CCSS with LSH and SkNN. In addition to a semantic-based keyword search, SCKS can achieve multi-keyword search and ranked keyword search at the same time. Because each document is indexed individually, the update of one document will not affect other documents, which means that SCKS can support dynamic data efficiently. To improve the security of SCKS, we propose a security-enhanced SCKS (SE-SCKS) by introducing a pseudo-random function. Thorough security

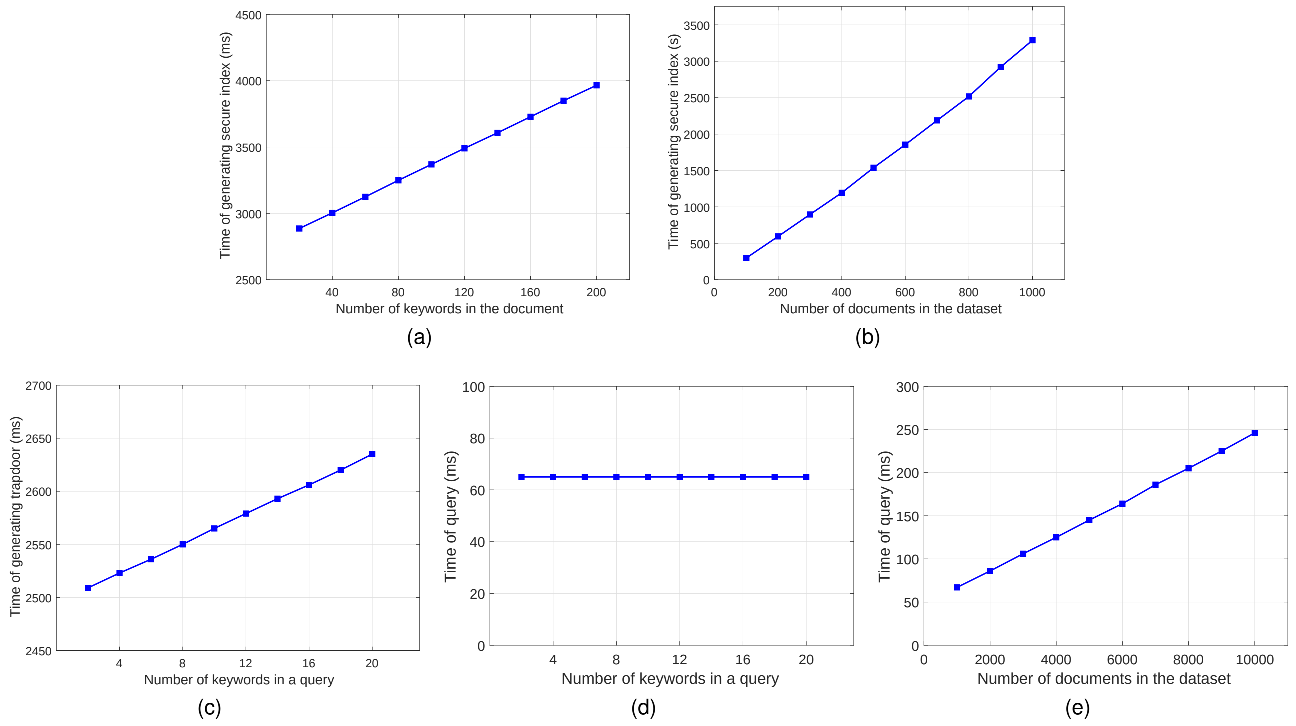

Fig. 7. Time cost of SCKS. (a) Secure index generation for a single document with different number of keywords. (b) Secure index generation for different-sized datasets. (c) Trapdoor generation with different number of keywords. (d) Search for different keywords in a dataset containing 1000 documents. (e) Search for 5 keywords in different-sized datasets

analysis of both SCKS and SE-SCKS is given, and the experiments on real-world dataset demonstrate that the proposed approaches introduce low overhead on computation and that the search accuracy outperforms the existing schemes.

References

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy*, 2000, pp. 44–55.

[2] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *IEEE International Conference on Computer Communications*, 2014, pp. 2112–2120.

[3] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *IEEE 28th International Conference on Data Engineering (ICDE)*, 2012, pp. 1156–1167.

[4] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *2012 Proceedings of IEEE INFOCOM*, 2012, pp. 451–459.

[5] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[6] R. Li, Z. Xu, W. Kang, K. C. Yow, and C. Z. Xu, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Generation Computer Systems*, vol. 30, no. 1, pp. 179–190, 2014.

[7] C. Orencik, M. Kantarcioglu, and E. Savas, "A practical and secure multi-keyword search method over encrypted cloud data," in *IEEE Sixth International Conference on Cloud Computing (CLOUD)*, 2013, pp. 390–397.

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

[9] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, 2010, pp. 253–262.

[10] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *ACM SIGMOD International Conference on Management of Data*, 2004, pp. 563–574.

[11] C. Chen, X. Zhu, P. Shen, J. Hu, S. Guo, Z. Tari, and A. Y. Zomaya, "An efficient privacy-preserving ranked keyword search method," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 951–963, 2016.

[12] X. Sun, Y. Zhu, Z. Xia, and L. Chen, "Privacy- preserving keyword-based semantic search over encrypted cloud data," *International Journal of Security & Its Applications*, vol. 8, no. 3, pp. 9–20, 2014.

[13] 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. 8:1–8:11, 2014.

[14] 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 Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 164–172, 2014.

[15] Z. Fu, J. Shu, X. Sun, and N. Linge, "Smart cloud search services: verifiable keyword-based semantic search over encrypted cloud data," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 4, pp. 762–770, Nov 2014.

[16] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C. N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[17] T. Moataz, A. Shikfa, N. Cuppens-Boulahia, and F. Cuppens, "Semantic search over encrypted data," in *20th International Conference on Telecommunications (ICT)*, 2013, pp. 1–5.

[18] M. Batet, D. Sánchez, and A. Valls, "An ontology-based measure to compute semantic similarity in biomedicine." *Journal of Biomedical Informatics*, vol. 44, no. 44, pp. 118–25, 2011.

[19] Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Transactions on Knowledge & Data Engineering*, vol. 15, no. 4, pp. 871–882, 2003.

[20] G. Pirró, "A semantic similarity metric combining features and intrinsic information content," *IEEE Transactions on Data & Knowledge Engineering*, vol. 68, no. 11, pp. 1289–1308, 2009.

[21] M. A. Rodrguez and M. J. Egenhofer, "Determining semantic similarity among entity classes from different ontologies," *IEEE Transactions on Knowledge & Data Engineering*, vol. 15, no. 2, pp. 442–456, 2003.

[22] D. Sánchez and M. Batet, "A semantic similarity method based on information content exploiting multiple ontologies," *Expert Systems with Applications*, vol. 40, no. 4, pp. 1393–1399, 2013.

[23] Z. Wu and M. Palmer, "Verb semantics and lexical selection," in *32nd Annual Meeting on Association for Computational Linguistics*, 1994, pp. 133–138.

[24] M. Li, B. Lang, and J. Wang, "Compound concept semantic similarity calculation based on ontology and concept constitution features," in *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, 2015, pp. 226–233.

[25] F. Baldimtsi and O. Ohrimenko, "Sorting and searching behind the curtain," in *19th International Conference on Financial Cryptography and Data Security*, R. Böhme and T. Okamoto, Eds. Springer Berlin Heidelberg, 2015.

[26] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *ACM SIGMOD International Conference on Management of Data*, 2009, pp. 139–152.

[27] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, "Computationally efficient searchable symmetric encryption," in *Secure Data Management: 7th VLDB Workshop (SDM)*, 2010, pp. 87–100.

[28] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *ACM Conference on Computer and Communications Security*, 2012, pp. 965–976.

[29] S. Kamara and P. Charalampos, "Parallel and dynamic searchable symmetric encryption," in *Financial Cryptography and Data Security*, 2013, pp. 258–274.

[30] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 1, pp. 17–30, 1989.

[31] D. Bollegala, Y. Matsuo, and M. Ishizuka, "A relational model of semantic similarity between words using automatically extracted lexical pattern clusters from the web." in *Conference on Empirical Methods in Natural Language Processing*, 2009, pp. 803–812.

[32] C. Leacock and M. Chodorow, "Combining local context and wordnet similarity for word sense identification," *Wordnet: An Electronic Lexical Database*, vol. 49, no. 2, pp. 265–283, 1998.

[33] D. Lin, "An information-theoretic definition of similarity," in *Fifteenth International Conference on Machine Learning*, 1998, pp. 296–304.

[34] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *10th International Conference on Research in Computational Linguistics (ROCLING' 97)*, 1997.

[35] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *14th International Joint Conference on Artificial Intelligence*, 1995, pp. 448–453.

[36] N. Seco, T. Veale, and J. Hayes, "An intrinsic information content metric for semantic similarity in wordnet." in *16th European Conference on Artificial Intelligence (ECAI)*, 2004, pp. 1089–1090.

[37] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7718–7728, 2012.

[38] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 79–88.

[39] ——, "Searchable symmetric encryption: Improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[40] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *25th International Conference on Very Large Data Bases*, 2000, pp. 518–529.

[41] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Twentieth Annual Symposium on Computational Geometry (SCG '04)*, 2004, pp. 253–262.

[42] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *2011 IEEE 27th International Conference on Data Engineering*, 2011, pp. 601–612.

[43] Y. Zhu, R. Xu, and T. Takagi, "Secure k-nn computation on encrypted cloud data without sharing key with query users," in *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, 2013, pp. 55–60.

[44] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *2014 IEEE 30th International Conference on Data Engineering*, 2014, pp. 664–675.

[45] B. J. Lyons, *Introduction to theoretical linguistics*. Cambridge University Press, 1977.

[46] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[47] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *IEEE 29th International Conference on Data Engineering (ICDE)*, 2013, pp. 733–744.

[48] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," in *Advances in Cryptology - EUROCRYPT*, 1997.

[49] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds., 2004, pp. 506–522.

Bo Lang received the Ph.D. degree from Beihang University (BUAA), Beijing, China, 2004. She has been a visiting scholar at Argonne National Lab / University of Chicago for one year. She is a Professor in School of Computer Science and Engineering, Beihang University. Her current research interests include data management, information security and distributed computing. As a principle investigator or core member, she has engaged in many national research projects founded by the National Natural Science Foundation of China, National High Technology Research and Development (863) Program, etc.

Jinmiao Wang received the B.S. degree from Shangdong University of Technology, China, 2012. She is now a Ph.D. candidate in Beihang University (BUAA), China. Her current research interests include cloud security, information security and cryptography.

Ming Li received the B.S. degree from Beihang University (BUAA), Beijing, China, 2016. His research interests include semantic content retrieval and information security.

Yanxi Liu is currently pursuing his bachelor degree with the School of Computer Science and Engineering, Beihang University (BUAA). His research interests include distributed computing and applied cryptography.