Enabling Semantic Search based on Conceptual Graphs over Encrypted Outsourced Data

Zhangjie Fu, Fengxiao Huang, Xingming Sun, Athanasios V. Vasilakos, and Ching-Nung Yang,

Abstract—Currently, searchable encryption is a hot topic in the field of cloud computing. The existing achievements are mainly focused on keyword-based search schemes, and almost all of them depend on predefined keywords extracted in the phases of index construction and query. However, keyword-based search schemes ignore the semantic representation information of users' retrieval and cannot completely match users' search intention. Therefore, how to design a content-based search scheme and make semantic search more effective and context-aware is a difficult challenge. In this paper, for the first time, we define and solve the problems of semantic search based on conceptual graphs(CGs) over encrypted outsourced data in clouding computing (SSCG).We firstly employ the efficient measure of "sentence scoring" in text summarization and Tregex to extract the most important and simplified topic sentences from documents. We then convert these simplified sentences into CGs. To perform quantitative calculation of CGs, we design a new method that can map CGs to vectors. Next, we rank the returned results based on "text summarization score". Furthermore, we propose a basic idea for SSCG and give a significantly improved scheme to satisfy the security guarantee of searchable symmetric encryption (SSE). Finally, we choose a real-world dataset – ie., the CNN dataset to test our scheme. The results obtained from the experiment show the effectiveness of our proposed scheme.

Index Terms—Searchable encryption, cloud computing, ranked search, semantic search, conceptual graphs.

1 Introduction

With the development of cloud services, an increasing number of data owners are choosing to outsource their data to the cloud. Cloud computing enables users to manage their data remotely and has many benefits for users. It is convenient for users to manage their data from everywhere without fixed equipment and to ignore the storage of the data. Additionally, it can provide high quality service for users because it is a shared configurable computing resource. However, data privacy is a significant problem that concerns customers. To protect the privacy of data, users usually choose to outsource their data in encrypted form. This makes effective utilization of the data become a challenging task when dealing with a large amount of data files. Moreover, when only interested in some files, it is necessary to provide effective way to search for the data files. One of the most popular and common ways is keyword-based search. Although, this method can allow users to retrieve files of interest, the returned results are always imprecise and unable to satisfy the intention of users.

In traditional searchable encryption schemes, data users can search for encrypted data securely and effectively. Additionally, the schemes provide a ranked result for users. However, the ranked results cannot completely match users' search intentions with the loss of semantic information. In other words, in terms of keyword search, keywords cannot provide a sufficient semantic representation of users' retrieval goals. To solve this problem, we introduce the Conceptual Graph (CG) in this paper. The CG is a structure for knowledge representation based on first logic. CGs are natural, simple and fine-grained semantic representations for depicting texts. A CG is a finite, connected and bipartite graph [1]. We will provide a detailed description in section 3. When applied in a large collaborative data outsourcing cloud environment, CGs may encounter several challenges, as follows. First, for each query and source file, when showed in the CG format, we must ensure that CG is searchable in the form of encryption. Existing CG search schemes in plaintext need to compute the specific value of similarity between CGs. In the process of calculating the similarity, they always need the help of the server and a tool of semantic extension such as WordNet to learn their query further. In other words, the server needs to learn the concrete content of retrieval to facilitate the retrieval. It is impractical to apply these schemes in plaintext to encrypted data. Then, faced with large amount of data, it is important to use effective mechanisms to guarantee the accuracy of the retrieval. Finally, in the existing searchable schemes, it is imperative to give rank results according to the queries.

In this paper, we solve the problem of how to enable a searchable encryption system with the support of semantic extension. Our work is one of only a few to study ranked search over encrypted data represented by CGs in Cloud Computing. We choose CG among various modes of knowledge representation as our semantic representation tool. In our scheme, we apply a state-of-the-art technique

- *Z. Fu, F. Huang and X. Sun are with the Department of Computer and Software, Nanjing University of Information Science and Technology. E-mail: wwwfzj@126.com, hfx20101344034@163.com, sunnudt@163.com.*
- *Athanasios V. Vasilakos is Lulea University of Technology, Sweden. E-mail: th.vasilakos@gmail.com.*
- *Ching-Nung Yang is with the Department of Computer Science and Information Engineering, National Dong Hwa University. E-mail: cnyang@mail.ndhu.edu.tw.*

ie., text summarization and Tregex, a tool for simplifying sentences to summarize the document. We introduce the existing scheme of constructing CGs [19] to help generate CGs. To achieve our design goals for both system security and usability, we divide each CG into three index vectors in response to the structure, the type of concept and the value of the concept of CG. Ranked search greatly improves system usability by returning the matching files in ranked order with regard to certain relevance criteria. Thus, to indicate how the document satisfies the query and ranks the returned file, we use the – "text summarization score" (TSS), which can gauge the extent to which the documents match their summarizations according to the relevance s-core. Moreover, if the score is higher, the document is more coincident with the original document. To protect the privacy of the TSS, we then integrate order preserving symmetric encryption (OPSE). Our contributions can be summarized as follows:

1) For the first time, we define the problem of semantic search based on conceptual graphs over encrypted outsourced data, and provide an effective method to perform a secure ranked search. Additionally, we define and present a round system of semantic search based on conceptual graphs over encrypted outsourced data.
2) Rather than employing traditional keywords as our knowledge representation tool, we use conceptual graphs to realize real semantic search in encrypted form. We propose a new practical and processing method of mapping CGs to vectors that makes quantitative calculation of CGs possible.
3) Extensive experimental results demonstrate the effectiveness and efficiency of the proposed solution.

The reminder of this paper is organized as follows. Related work is discussed in Section 2, and Section 3 gives a brief introduction to the system model, the threat model, the design goals, and the preliminaries. Section 4 describes the schemes in detail. Section 5 presents security analysis. Section 6 shows the experiments and performance analysis. In Section 7, conclusions are drawn.

2 Related work

We mainly focus on semantic search based on a searchable encrypted scheme in this paper. Thus, we describe the related work of the three parts as follows:

Song *et al*. [5] was the first to propose the searchable encryption scheme in the symmetric setting. A 2-layered encryption structure is used to search over the encrypted documents with a sequential scan. The main idea is that each word is encrypted separately at first; then, a hash value with a special form is embedded into the ciphertext before searching. When users submit a search request, the server firstly extracts the hash value from the ciphertext and then determines whether the value matches the special form. It is the first practical scheme defining the problem of searching for encrypted data and had a major positive effect on later studies. However, the obvious weakness is that the scheme must adopt a fixed length of words and is not compatible with variable length words. Another weakness is that the scheme must adopt the designed two-layer encryption method, which is not suitable for other types of data, such as compressed data.

After that, single keyword searchable encryption schemes [6-8] were developed. These scheme were an initial semantic extension. [6,7] proposed to improve the security definition and search efficiency. [8] proposes an effective searchable symmetric encryption scheme to achieve keyword searches and return ranked results. The main idea is building an inverted index to store a list of mappings from keywords to the corresponding set of files that contain this keyword. When a user submits a search request, the cloud server first locates the matching list of indexes and sends back the corresponding files. However, it can only support a single keyword search.

To enrich the function of retrieval, multi-keyword schemes were also proposed [9-12]. In a general sense, [9] is the basis of [10-14]. [9] was the first to solve the problem of multi-keyword ranked search over encrypted cloud data (MRSE). In MRSE, the data owner generates a binary data vector P for each document D as the index, where each bit $P[i]$ represents whether the corresponding keyword appears in the document. Then the scheme splits the vector P into two vectors encrypted with two invertible matrices. The user constructs the query vector Q in the same way but using the inverses of invertible matrices in the process of encryption. Upon receiving a trapdoor, the cloud server computes the inner product between the index and query and returns the related, ranked files. Based on secure inner product computation, two significantly improved MRSE schemes that satisfy the various stringent privacy requirements in two different threat models were presented. [10] proposed constructing the search index based on term frequency and the vector space model and selected the cosine similarity measure to achieve higher search result accuracy. [11] took the keyword access frequencies into account when a system returns ranked results. [12] introduced parallel computing to increase the effectiveness of multi-keyword search.

Based on [9], some typical schemes were proposed and were not confined to multi-keywords. For example, the dynamic searchable encrypted scheme [13] and semantic search schemes [14] attempted to perform semantic search. [13] presented a secure multi-keyword ranked search scheme based on VSM and the widely used TF-IDF mode, which simultaneously supports dynamic update operations such as the deletion and insertion of documents. [14] used the porter algorithm to map all the semantically close words or different variants of a word to the same stem. [23-32] coverd many secure search schemes which are also important to which more attention should be paid.

Due to the limitation of keyword searches, note that all these schemes only partially support semantic search. Adding as much semantic information as possible to improve the accuracy of retrieval is our core concern in this paper.

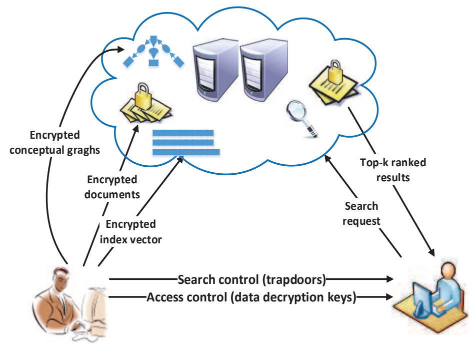

Encrypted conceptual graghs
Encrypted documents
Encrypted index vector
Top-k ranked results
Search request
Search control (trapdoors)
Access control (data decryption keys)

Fig. 1. The architecture of ranked search over encrypted cloud data

3 Problem Formulation

3.1 The System and Threat Models

A cloud data hosting service contains three different entities commonly, as illustrated in Fig. 1, data owner (O), data user (U), and cloud server (CS). Data owner owns n data files $F = \{F_1, F_2, ..., F_n\}$ and he outsources the encrypted data to the cloud server while still ensuring that the data are searchable. Then some searchable indexes are generated based on the dataset for the purpose of searching for documents of interest, the process of which is as follows. Firstly, CGs are constructed based on each document in the dataset. Secondly, three index vectors for each CG are generated based on the structure and the concepts of the CG. Finally, the searchable index is constructed with all the index vectors. Then, vectors are also encrypted and outsourced to the cloud server together with the encrypted dataset. Authorized data users can search for documents of interest from the outsourced dataset by submitting a query trapdoor to the cloud server. After receiving the trapdoor, the cloud server searches the searchable index and returns those encrypted documents that satisfy the search criterion. Note that the returned documents are ranked by their relevance scores. Once receiving the search results from the cloud server, the data user can decrypt the ciphertext and obtain the expected files in plaintext. In our model, we think that the cloud server is ”honest-but-curious”, which is the same as in the most previous works, such as [8]. We employ [22]'s classification regarding the cloud server based on the knowledge it has to assist in proving the security of our scheme as follows named level-1, level-2 and level-3 attack respectively:

- The cloud server only pays attention to the ciphertext.
- In addition to the ciphertext, the cloud server also knows some plaintext indexes.
- Moreover, the cloud server knows the corresponding encrypted values of the plaintext indexes.

3.2 Design Goal

To solve the problem of semantic search based on conceptual graphs over encrypted outsourced data effectively in the above system model, our system has the following design goals.

Ranked search: The proposed scheme is designed to provide not only semantic query based on CG but also accurate result ranking.
Search Efficiency: The scheme aims to achieve low communication and computation overhead.
Privacy: The scheme is designed to prevent the cloud server from learning additional information regarding the document collection and query.

3.3 Notations and Preliminaries

- F – the plaintext document collection, denoted as a collection of n documents $F = \{F_1, F_2, ..., F_n\}$. Each document F_i in the collection can be considered as a CG.
- n – the total number of documents in F.
- Q – the query represented by three vectors, defined as a collection of three vectors $Q = \{Q_1, Q_2, Q_3\}$.
- $F(Q_1)$ – the encrypted set of documents in F whose D_1 is similar with Q_1.
- D_1 – the source file F_i represented by three vectors, defined as a collection of three vectors $D_i = \{D_{i1}, D_{i2}, D_{i3}\}$.
- N – the total number of documents in F is similar with the query CG.

Text summarization Text Summarization (TS) always tries to determine the ”meaning” of documents. Essentially, TS techniques are classified as Extractive and Abstractive. In this paper, we focus on extractive summarization. Extractive summaries (ES) produce some of the most significant sentences extracted from the document instead of rebuilding. Through performing a rank of on sentences in the document and choosing the sentences with high scores, ES can acquire some sentences to represent the document. In the review [2], the achievements of single document summarization are shown, and a comparison between existing sentence scoring methods of the last ten years is made. We select the best algorithm in sentence scoring as our pre-processing tool.

In this paragraph, we crudely describe the advanced algorithm as follows. The modified TF/IDF algorithm can be used to compute the scores of sentences. This algorithm performs a comparison between the term frequency (tf) in a document (each sentence is treated as a document) and the document frequency (df), which denotes the number of times that a word occurs among all documents. The TF/IDF score can be calculated for the keyword w as follows:

$$TF/IDF(w) = DN \times (\frac{\log(1+tf)}{log(df)}) \tag{1}$$

where DN is the number of documents. When we attain all sentences' TF/IDF, we sort these values to select the sentence responding with the highest score.

Tregex is a powerful syntactic tree search language for identifying syntactic elements. It has been applied to many aspects in Natural Language Processing (NLP), consisting of sentence simplification which we focus on in this paper. Tregex can be used to distinguish the various relations between tree nodes. Tregex is more powerful than regular sentence classification expressions because it can capture a greater number of syntactic ways in which people write a

TABLE 1
An example of TSS

File ID	F_1	F_2	F_3	F_4	F_5
Relevance Score	21.6741596	6.2537288	23.3445375	4.8164799	23.3445375

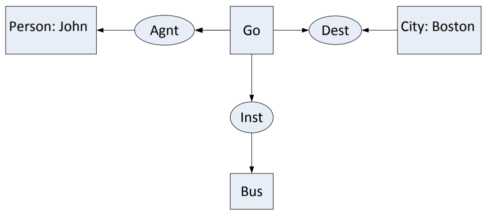


Fig. 2. CG display form for *John is going to Boston by bus*.

certain type of sentence with cue phrases [3]. After identifying the key nodes with Tregex, we use the Stanford Parser's API to analyse trees, which allows for inserting and deleting children, changing labels on tree nodes, etc [4].

Conceptual graph Conceptual graph as a knowledge representation model was proposed by Sowa in [15].It is defined as a graph representation of logic, that is based on the semantic networks of Artificial Intelligence (AI) and existential graphs [16]. There are usually two types of nodes: concepts (rectangles) and conceptual relations (ovals) (Fig. 2). A concept is connected with another concept by a conceptual relation. Each conceptual relation must be connected to some other concepts. For each CG, we denoted conceptual relations as semantic roles, with 30 relations approximately including 6 tenses. In this paper, we choose approximately 24 relations, regardless of tenses. *Person* and *City* in Fig. 2 in the CG are concept types and show the category of concepts. They can be null and are marked as ″*″.

Ranking Function In information retrieval, a ranking function is used to calculate the relevance scores of matching files to a given search request. In this paper, we will rank our files according to TSS. How we calculate TSS is based on Formula (1). Table 1 shows an example of TSS.

Order-Preserving Symmetric Encryption (OPSE) OPSE is a cryptographic primitive that helps sort the ranking scores in an encrypted form. However, the original OPSE always maps the plaintext into the same random-sized non-overlapping interval bucket which leads to some information leakage. In this paper, we focus on the modified OPSE mentioned in [8]. In [8], they designed a one-to-many OPSE which appends files' ID into the plaintext and makes the same plaintext no longer point to the same ciphertext deterministically. The modified OPSE effectively reduces the information leakage.

4 The Proposed Schemes

Existing schemes are [17, 18] either are only suitable for plaintext or for performing searches before encryption. They cannot realize encrypted searches based on CG. Thus, we must first construct a search scheme in plaintext that can also be easily applied to encrypted data. We will propose a complete scheme, including data pre-processing. First, we give a rough summarization of data pre-processing as follows - making summarizations of the document set, simplifying the sentences obtained in text summarization by Tregex and constructing the CGs according to the simplified sentences [19]. Then we present our scheme based on the above.

In this section, we first give a description of index construction in detail which is the foundation of our scheme. Then we will describe an effective unencrypted semantic search based on conceptual graphs (USSCG) scheme as the basis of the subsequent encrypted scheme. Finally, based on the USSCG and referring to the encrypted scheme of [8], we will present a secure search scheme–SSCG.

4.1 Index Constructions

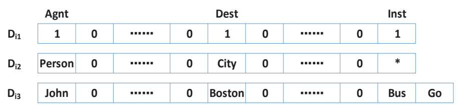
In Section 3, we gave a brief introduction to CG which assists us in index constructions. As mentioned above, the ″semantic roles″, ″concept types″ and ″concept values″ are typical parts in the CG. To generalize all information of a CG, we must take effective measures to cover all typical parts of a CG. Regarding quantitative calculation, constructing vectors is the best way. Thus, we design a new processing method for extracting vectors from typical parts. One vector is used to match the ″semantic role″ in the search request which determines whether the structure of a CG is satisfied with the search request, ie., whether query CGs' semantic roles are included in the source CG. Another two are the same as the first one to match the concept types and concept values. In the next paragraph, we will detail how to produce the three vectors from a CG.

The process of generating these two $n-dimension$ index vectors (n represents the total number of semantic roles) and one $(n+1)-dimension$ based on the extended semantic roles, concept types and concept values are as follows. For a CG G_i, we use D_{i1}, D_{i2} and D_{i3} to refer to its three index vectors. Each dimension of D_{i1}, denoted as $D_{i1}[j]$, corresponds to a node (stores semantic role sr in the CG). If G_i contains the semantic roles sr, then $D_{i1}[j]=1$, otherwise $D_{i1}[j]=0$. Similarly, each dimension of D_{i2}, D_{i3}, denoted by $D_{i2}[j]$, $D_{i3}[j]$, also corresponds to the nodes (stores concept type r_j and the value for concept c_j in the CG. If G contains sr and r_j ,c_j have values in *F*, denoted as $Val(r_j,G)$ and $Val(c_j,G)$ respectively, then $D_{i2}[j]$, $D_{i3}[j]$ is:

$$D_{i2}[j]=\begin{cases}Val(r_j,G) & \text{if } Val(r_j,G)\ is\ not\ null\\ 0 & \text{if } Val(r_j,G)\ is\ null\end{cases} \tag{2}$$

$$D_{i3}[j]=\begin{cases}Val(c_j,G) & \text{if } Val(c_j,G)\ is\ not\ null\\ 0 & \text{if } Val(c_j,G)\ is\ null\end{cases} \tag{3}$$

We take the CG in Fig.2 as an example to illustrate the process. In the CG, *John, Go, Boston, Bus* are concept nodes. *Agnt, Inst, Dest* are the semantic roles. *Person, City* are concept types. The index vectors generated for the above CG are shown in Fig.3.



	Agnt				Dest				Inst	
D_{i1}	1	0	......	0	1	0	......	0	1	
D_{i2}	Person	0	......	0	City	0	......	0	*	
D_{i3}	John	0	......	0	Boston	0	......	0	Bus	Go

Fig. 3. Index vectors display form for *John is going to Boston by bus*.

4.2 USSCG Scheme

The search process of the USSCG scheme is an algorithm named CGM (conceptual graphs match). We construct a result list denoted as *RList*, whose element is defined as ($RScore$, *FID*). Here, $RScore$ is the relevance score of the document whose id was denoted as *fFID* to the CG calculated according to Formula (1). The *RList* stores the k documents with high similarity to the query temporarily. The elements of the list are ranked in descending order according to the $RScore$ and *RList* will be revised during the search process. The following are some other notations, and the CGM algorithm is described in Algorithm 1.
$RScore(D, Q)$ –The function used to calculate the similarity between query vector Q and index vector D.
Score – The minimum value of $RScore(D_1, Q_1)$.

Algorithm 1 CGM

```
1:  Input: $F$, $D_1$, $D_2$, $D_3$, $Q_1$, $Q_2$, $Q_3$, $TSS$
2:  Output: $F(Q_1)$
3:  for each document $F_i$ in $F$ containing $D_1$, $D_2$, $D_3$ do
4:    if $RScore(D_1, Q_1) > Score$ then
5:      Insert $D_2$, $D_3$ into $F(Q_1)$;
        $RScore(D_2, Q_2)$;
        $RScore(D_3, Q_3)$;
        Insert a new element ($RScore(D_2, Q_2)$, $RScore(D_3, Q_3)$, $FID$) into RList;
6:    else
7:      return;
8:    end if
9:  end for
10: for each $F_j$, $F_k$ in $F(Q_1)$ do
11:   if $RScore(D_{j2}, Q_{j2}) == RScore(D_{j2}, Q_{k2})$ then
12:     if $RScore(D_{j3}, Q_{j3}) == RScore(D_{j3}, Q_{k3})$ then
13:       Compare the TSS of $F_j$, $F_k$ to sort the elements of $F(Q_1)$;
14:     else
15:       According to the $RScore(D_3, Q_3)$ to sort the elements of $F(Q_1)$;
16:     end if
17:   else
18:     According to the $RScore(D_2, Q_2)$ to sort the elements of $F(Q_1)$;
19:   end if
20: end for
```

The flow chart in Fig.4 intuitively illustrates the process of algorithm CGM. In Fig.5, we give an example of distinguishing *John is going to Boston by bus* from *John is going to Boston by bike*, when they are changed into index vectors. Observing the vectors, we can make a clear distinction between the two because of the third vector, simultaneously we also conclude that they are similar in the semantic sense because of the other ones.

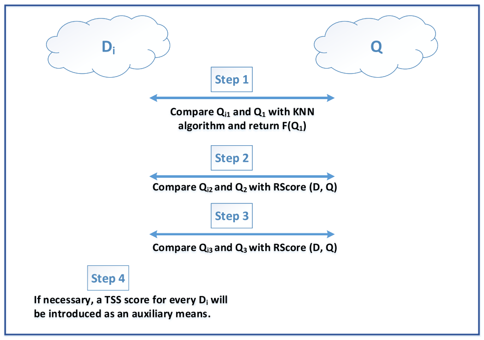
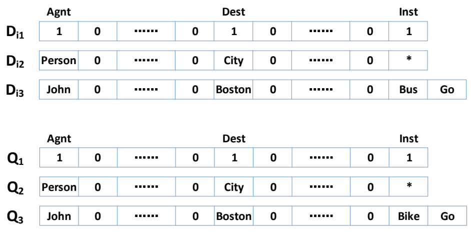

D_i
Q
Step 1
Compare Q_{i1} and Q_1 with KNN algorithm and return $F(Q_1)$
Step 2
Compare Q_{i2} and Q_2 with RScore (D, Q)
Step 3
Compare Q_{i3} and Q_3 with RScore (D, Q)
Step 4
If necessary, a TSS score for every D_i will be introduced as an auxiliary means.

Fig. 4. A flow chart of algorithm CGM



	Agnt				Dest				Inst	
D_{i1}	1	0	......	0	1	0	......	0	1	
D_{i2}	Person	0	......	0	City	0	......	0	*	
D_{i3}	John	0	......	0	Boston	0	......	0	Bus	Go

	Agnt				Dest				Inst	
Q_1	1	0	......	0	1	0	......	0	1	
Q_2	Person	0	......	0	City	0	......	0	*	
Q_3	John	0	......	0	Boston	0	......	0	Bike	Go

Fig. 5. Index vectors display form for *John is going to Boston by bus*. Index vectors display form for *John is going to Boston by bike*.

4.3 SSCG Scheme

4.3.1 Overview of the SSCG Scheme

Based on the USSCG scheme and referring to the encrypted scheme of [8, 9], we construct a semantic search scheme based on conceptual graphs over encrypted outsourced data–SSCG. Our proposed SSCG scheme tries to solve the problem of how to let a server quickly perform a ranking knowing only ciphertext, which satisfies the security guarantee of searchable symmetric encryption (SSE). To effectively support encrypted ranked search, we resort to modified OPSE [8] to achieve more practical performance.

In this paper, we employ the idea of one-to-many order-preserving mapping from [8], which mixes the unique file IDs together with the plaintext as the random seed in the final cipherext chosen process. Because the unique file ID is part of the random selection seed, the same plaintext will no longer be deterministically pointed out to the same ciphertext, but randomly scattered across a range size R. The modified OPSE is more secure and more suitable for our scheme and is presented as Algorithm 2 and 3. Specially, $Tapegen()$ is a random coin generator, m represents the plaintext, $HYGEINV()$ is a instance for the $HGD()$ sampling function.

In this paragraph, we give a rough description of SSCG based on modified OPSE: we encrypt our data documents, index vectors, and relevance score and outsource them to the cloud server. When receiving a search request, the server performs matching between index vectors and query vectors via two different processes. In the first step, the cloud server checks how similar the source and the query in the first index are which is decided by MRSE as presented in [9]. Next, we compare the second and third vectors according to their hash values. Finally, if there are still multiple index vectors in the query that cannot be distinguished well, we will rank the result according to the encrypted relevance scores by the modified OPSE. In the next paragraphs, we will present some definitions and a detailed description of SSCG.

We denote OPM as our one-to-many order-preserving mapping function with parameter:

$OPM : \{0,1\}^l \times \{0,1\}^{\log|D|} \rightarrow \{0,1\}^{\log|R|}$. The whole process is shown in Algorithm 2 and 3.

Algorithm 2 One-to-many Order-preserving Mapping-OPM

```
1: Input: $D$, $R$, $m$, $id(F)$
2: Output: $c$
3: while $|D|! = 1$ do
4:     $\{D, R\} \leftarrow BinarySearch(K, D, R, m)$;
5: end while
6: $coin \overset{R}{\leftarrow} TapeGen(K, (D, R, 1 \,||\, m, id(F)))$;
   $c \overset{coin}{\leftarrow} R$;
   Return $c$;
```

Algorithm 3 $BinarySearch$

```
1: Input: $K$, $D$, $R$, $m$
2: Output: $D$, $R$
3: $M \leftarrow |D|$; $N \leftarrow |R|$;
   $d \leftarrow min(D) - 1$; $r \leftarrow min(R) - 1$;
   $y \leftarrow r + \lceil N \rceil$;
   $coin \overset{R}{\leftarrow} TapeGen(K, (D, R, 0 \,||\, y))$;
   $x \overset{R}{\leftarrow} d + HYGEINV(coin, M, N, y - r)$;
4: if $m < x$ then
5:     $D \leftarrow \{d + 1, ..., x\}$;
       $R \leftarrow \{r + 1, ..., y\}$;
6: else
7:     $D \leftarrow \{x + 1, ..., d + M\}$;
       $R \leftarrow \{y + 1, ..., r + N\}$;
8: end if
9: Return $\{D, R\}$;
```

Here, π is a collision resistant hash function with the parameters as follows:

$\pi : \{0,1\}^k \times \{0,1\}^\star \rightarrow \{0,1\}^p$ where $p > \log m$

Generally, $\pi(\bullet)$ will be instantiated by an off-the-self hash function such as SHA-1, in which case p is 160 bits.

4.3.2 Detail of the SSCG Scheme

Our proposed SSCG scheme can be described as follows:

In the **Setup** phase:

1) The data owner calls $KeyGen(1^k, 1^l, 1^l, 1^p, |D|, |R|)$,generates random keys x, y, $z \overset{R}{\leftarrow} \{0,1\}^k$, and outputs $K = \{x, y, z, 1^l, 1^l, 1^p, |D|, |R|\}$. This step is mainly performed to generate the secret key and run by **the data owner**.
2) For CG G_i in the dataset, the data owner generates three index vectors D_{i1}, D_{i2} and $D_{i3}(1 < i < n)$ using the procedure in Section 4.1 and sends them to the cloud server in encrypted form. The data owner encrypts D_{i1} by using MRSE [9] and D_{i2} and $D_{i3}(1 < i < n)$ by using $\pi(\bullet)$. Note that the dataset is also encrypted and sent to the cloud server by the data owner including encrypted relevance score $OPM(S_i)$. The data owner identifies a document by its file ID. This step mainly shows the communication between **the data owner** and **the cloud server**. Based on this, the preliminary work is ready.

In the **Retrieval** phase:

1) The data user generates three vectors Q_1, Q_2 and Q_3 according to a query. The data user encrypts Q_1 by using MRSE [9] and Q_2, Q_3 by using $\pi(\bullet)$ the same as for D, because the data user is authorized by the data owner. Then the data user sends the three encrypted vectors to the cloud server. This step mainly focuses on the interaction between **the data user** and **the data owner**.
2) Upon receiving the three vectors, the cloud server first selects K nearest neighbours of Q_1 by comparing with D_{i1}. Then the cloud server obtains a document set $F(Q_1)$ whose semantic structure is similar to that of the query. Then we choose D_{i2}, D_{i3} from $F(Q_1)$ to check whether $\pi(D_{i2})$, $\pi(D_{i3})$ are the same as $\pi(Q_2)$, $\pi(Q_3)$. This step mainly describes how **the cloud server** responds to **the data user**.
3) The cloud server now knows the encrypted file identifiers $id(F_i)$(suppose $D_{2j} = Q_{2j}$, $D_{3j} = Q_{3j}$; thus, $j \in \{1, ..., N\}$) and their associated order-preserved encrypted scores: $OPM(S_i)$). Because the vectors D_2 and D_3 are encrypted by the hash, we can directly check if $\pi(D_{i2})$ is the same as $\pi(Q_2)$ through specific corresponding hash values. If there still exist two documents that cannot be distinguished well, the cloud server will check the encrypted relevance scores $OPM(S_i)$.
4) The cloud server then returns back the files in a ranked way and the data user can provide selected k to decide to send the top-k most relevant files.

The last two steps mainly show how **the cloud server** finishes its work according to the requirement of **the data user**. The above process demonstrates how the scheme is run by the data owner, the data user and the cloud server.

4.3.3 Discussion

In this section, we focus on how our scheme achieves the design goals mentioned in Section 3.2. The concrete analysis is given as follows corresponding to these three goals:

- **Ranked search**: The goal requires the cloud server to return sorted files to the data user. In our scheme, as long as one concept in the second or the third index vector in one source document has a different value than the other source index vectors, we can draw a conclusion about which is more suitable. Moreover, if the comparison fails in this way, then the cloud server will rank their results according to TSS. Through these two steps, there will be enough to distinguish these files in an ordered way.
- **Search efficiency**: In our scheme, we compare our scheme with schemes in [8, 9] which shows the

efficiency of our scheme. For a detailed description, see Section 6.

- **Privacy**: With the help of the modified OPSE, the server can rank the files in the *RList* efficiently. In the process of retrieval, the cloud sever can only learn index vectors and the relevance score in the encrypted form instead of plaintext, thus we can further mitigate the useful information revealed to the cloud server. The above scheme clearly satisfies the security guarantee of SSE. For detailed analysis, see Section 5.

In summary, our scheme has achieved the predefined design goals.

5 Security Analysis

In this section, we will focus on the security of our scheme by analyzing the security guarantee mentioned in Part 3. In our scheme, the encrypted documents and indexes including trapdoors will be exposed to the cloud server. However, the plaintext documents are under protection. We consider that the cloud server cannot infer the encrypted information. We also use OPSE to protect our relevance scores. Thus, in the next section, we first analyse the security of one-to-many OPSE and then combine with our scheme to prove the security.

5.1 Security Analysis for One-to-Many Mapping

The one-to-many OPSE [8] which we employ from [8] is adapted from [21]. It appends the file ID as the additional seed which causes the same plaintext to no longer map to the same ciphertext. It is randomly assigned in the range R. It confuses determined mapping which also makes it more difficult for the cloud server to infer the information.

5.2 Security Analysis for Ranked Search

In our proposed scheme which is referring to the encrypted scheme RSSE of [8], data files and index vectors are outsourced in the encrypted form. Due to the security strength of the file encryption scheme, the file content is clearly well protected. Thus, we only need to focus on index vectors privacy which means we need to prove the security of our scheme based on RSSE [8] and MRSE [9].

The security of MRSE [9] depends upon secure kNN [20] which mainly contains two steps: matrix encryption, random split . We take [22] as a reference to prove the secure problem. We first describe the secure KNN which solves the problem of finding the nearer point of the two as follows.

KNN
$\Leftrightarrow ||p_1 - q|| - ||p_2 - q|| > 0$
$\Leftrightarrow (p_1 \cdot p_1 - 2p_1 \cdot q) - (p_1 \cdot p_2 - 2p_2 \cdot q) > 0$
$\Leftrightarrow ((p_2, -0.5(p_2 \cdot p_2))) \cdot (q, 1) > ((p_1, -0.5(p_1 \cdot p_1))) \cdot (q, 1)$
$\Leftrightarrow p_2^\star \cdot q^\star > p_1^\star \cdot q^\star$
$\Leftrightarrow E(p_2^\star) \cdot E(q^\star) > E(p_1^\star) \cdot E(q^\star)$

where $p^\star = (p, -0.5(p \cdot p))$, $q^\star = (q, 1)$, $||p||$ represents Euclidean norm of p, $\cdot$ is inner product, and $E(\bullet)$ is an encryption function.

Distance-preserving transformation(DPT) is the basic idea which supports KNN in theory. DPT encrypts index points so that computing the distance of two points like in the plain index database.Although DPT has been proved to be able to resist level-1 attack. Unfortunately, DPT cannot resist high level attack which is not secure for users.

Matrix Encryption: Considering the high level attack such as level-2 attack, we should protect the distance information between any two index points p_i and p_j from the attack through encryption, which is not distance-recoverable. The problem is well solved until the scheme based on matrix encryption is promoted.

- **Key**: a $n^{'} \times n^{'}$ invertible matrix M.
- **Index encryption**: The encrypted index point $\hat{p} = M^T p$.
- **Query encryption**: The encrypted query point $\hat{q} = M^{-1} rq$, where r is a random number.
- **Distance comparison**: It only needs to confirm whether $(\hat{p_2} - \hat{p_1}) \cdot \hat{q} > 0$,where $\hat{p_1}$, $\hat{p_2}$ and $\hat{q}$ are encrypted values of p_1, p_2, q respectively to judge who is closer to the q.

$Theorem$ 1: Suppose $\hat{p_1}$, $\hat{p_2}$, $\hat{q}$ are the encrypted values of the index points p_1, p_2 and the query point q, respectively, matrix encryption scheme correctly determines whether p_2 is closer to q than p_1 by checking $(\hat{p_2} - \hat{p_1}) \cdot \hat{q} > 0$.

$Proof$: Note that,
$(\hat{p_2} - \hat{p_1}) \cdot \hat{q}$
$= (\hat{p_2} - \hat{p_1})^T \hat{q}$
$= (p_2 - p_1)^T rq$

That means
$(p_2 - p_1)^T rq > 0$
$\Leftrightarrow p_2^T > p_1^T$
$\Leftrightarrow ||p_1 - q|| - ||p_2 - q|| > 0$

This matrix encryption scheme reveals no information about distance between any two index points and thus it can resist level-2 attack. However, this scheme is not secure against level-3 attack.

Random Split: To deal with the limitation of matrix encryption scheme above, it just needs to make the matrix M harder to crack. Introducing some randomness is one popular method to solve the above problem which makes it difficult to set up the equations. We describe the advanced scheme integrating random spilt into matrix encryption.

- **Key**: two $n \times n$ invertible matrix M_1,M_2; a n-bit randomly generated vector as S.
- **Index encryption**: The index point p is firstly split into two parts $\{p_{00}, p_{01}\}$. If $S[j]$ is 0, $p_{00}[j]$ and $p_{01}[j]$ are both set as $p[j]$; otherwise, $p_{00}[j]$ and $p_{01}[j]$ can be set as random values while the sum of them should be equal to $p[j]$. The encrypted value of p is the pair $\hat{p} = \{M_1^T p_{00}, M_2^T p_{01}\}$.
- **Query encryption**: The query point q is also firstly split into two parts $\{q_0, q_1\}$. If $S[j]$ is 0, $q_0[j]$ and $q_1[j]$ can be set as random values so that the sum of them should be equal to $q[j]$; otherwise, $q_0[j]$ and $q_1[j]$ are both set as $q[j]$. The encrypted value of q is the pair $\hat{q} = \{M_1^{-1} q_0, M_2^{-1} q_1\}$.
- **Distance comparison**: To determine whether p_2 is closer to q than p_1, it only needs to check whether $(\hat{p_2} - \hat{p_1}) \cdot \hat{q} > 0 \Leftrightarrow (p_2 - p_1) \cdot q > 0$, where $\hat{p_1}$, $\hat{p_2}$ and $\hat{q}$ are encrypted values of p_1, p_2, q respectively.

$Theorem$ 2: With combined of random split,the scheme can resist the level-3 attack if the cloud server cannot recover the splitting configuration (e.g., the vector S in key).

$Proof$: According to the definition, for any index point $p_i \in P_A$, a level-3 cloud server knows the encrypted values $I_i = \{\hat{p_{ia}}, \hat{p_{ib}}\} = \{M_1^T p_{00}, M_2^T p_{01}\}$. If the cloud server doesn't know splitting configuration, he has to set p_{00i} and p_{01i} as two random n-dimension vectors. And then he sets the following equations: $M_1^T p_{00i} = \hat{p_{ia}}$ and $M_2^T p_{01i} = \hat{p_{ib}}$, where M_1 and M_2 are two $n \times n$ matrix. There are $2n\,|P_A|$ unknown variables in p_{00i} and p_{01i} and $2n^2$ unknown variables in M_1 and M_2. Since there are only $2n\,|P_A|$ equations, the cloud server doesn't have sufficient information to crack the matrix. Hence, random split based on matrix encryption can resist the level-3 attack.

Next, we analyse the security of our scheme SSCG based on the above proof.

The index privacy of the first vector in the CG is well protected if the secret key is kept confidential because such a vector encryption method has been proven to be secure in the known ciphertext model [20]. With the randomness introduced by the splitting process and the random numbers r, and t which is adapted from [9]. Our index vectors generate two totally different trapdoors for the same query Q_1. This nondeterministic trapdoor generation can guarantee the security of the trapdoor and protect the trapdoor from an adversary. Moreover, with a properly selected parameter for the random factor, even the final score results can be obfuscated very well, preventing the cloud server from learning the relationships of given trapdoors and the corresponding CG.

The remaining vectors in the CG are encrypted by using $\pi(\bullet)$; the relevance scores are encrypted by using OPM. $\pi(\bullet)$ is a function of hash and is an irreversible process. We consider that the encryption and the encrypted "keywords" are secure enough which includes the adversary not being able to learn the plaintext. As for relevance scores, under the premise of balancing efficiency and security, we choose a reasonable range size R based on [8]. Although we can make the range size R large enough and ensure the encrypted scores will have few copies - this is not practical. Furthermore, the score is still an order-preserving numeric sequence. The adversary is also unable to calculate the specific scores.

Certainly, we can add a certain number of dummies into the vectors with mixture order. This means that will be harder for the adversary to restore the original sentences according to the index vectors, which further reduces the information leakage from an over point of view.

6 Performance Analysis

In this section, we realize the search system using CSharp language to estimate the overall performance of the proposed scheme. The implementation platform is a Windows 7 server with Core2 CPU 2.93GHz. In the experiment, we use a real-world dataset: the CNN dataset [20] (www.cnn.com) which is publicly available to build the outsourced dataset. We use approximately 1,000 files from the CNN dataset as the outsourced dataset.

6.1 Precision

In this section, we mainly focus on the precision of our scheme. The precision here indicates means whether users attain the desired documents according to queries. In our scheme, we also need to balance precision and privacy. To reduce the loss of precision, the balance strategy of precision and privacy involving introducing dummy keywords in [9] is used in our scheme. In theory, It is highly possible to return all the query results related to the users' query because of the construction of the CG. Our index vectors have covered all the information of the CG and the CG is a perfect semantic representation of knowledge. It is reasonable that our retrieval has high precision.

6.2 Efficiency

6.2.1 Index Vectors Construction

The process of index vectors construction for document collection F includes two main steps:

1) generating vectors based on the document collection F;
2) encrypting the index vectors using MRSE and $\pi(\bullet)$.

The index structure is constructed by following post-order traversal of the CGs based on the document collection F. For each CG, generation of index vectors takes $O(3n+1)$ time, and n is constant (the dimensions of vectors are fixed and n, n, $n+1$ respectively). As for the first vector of the three, the vector splitting process takes $O(m)$ time, and two multiplications of a $(m \times m)$ matrix takes $O(m^2)$ time. Here, m is also a constant because it is related to n. When we encrypt the left two vectors by using $\pi(\bullet)$, it is easy to know that their cost is a total $O(2n+1)$. As a whole, the time complexity for index vectors construction is $O(N \times (5n+2+m^2))$, where N represents the total number of documents in the document collection F. Apparently, the time cost for building index vectors mainly depends on the cardinality of document collection F. Although the index vectors construction consumes relatively much time at the data owner side, it is noteworthy that this is a one-time operation.

Table 2 shows when the size of documents is approximately 1000, the time cost and space cost of building the every index. Fig.6 shows the cost of overall index construction in our scheme; it is nearly linear with the size of data documents. Table 3 shows the time cost of RSSE[8] of index construction. Although our scheme greatly differs from the traditional keyword search, the time cost of index construction in [8] can partly prove that our scheme is acceptable when contrary to the existing published statistics in Table 2 and Table 3. We also list the total time cost of index construction for our scheme and MRSE[9] in Fig.6 and Fig.7. As a one-time operation, the results also demonstrate the receivability through the comparison between our scheme and MRSE.

6.2.2 Query

For each search request Q, the search procedure is implemented as follows:

1) determine whether the index vector D_1 for document F satisfies the search vector Q_1;

TABLE 2
Index Construction overhead for approximate 1000 files.

Scheme	Number of files	index vectors size	Construction of index vectors for per CG
SSCG	1036	8802 KB	1.72s

TABLE 3
Index Construction overhead for approximate 1000 files.

Scheme	Number of files	Per keyword list size	Per keyword list build time
RSSE	1000	12414 KB	5.44s

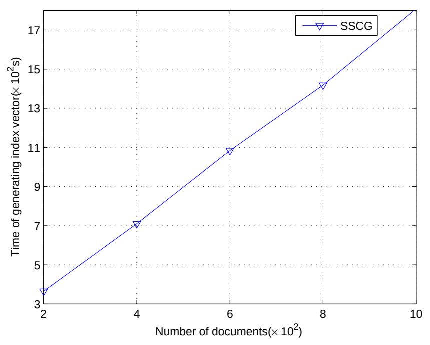


Fig. 6. The time cost for index vectors construction

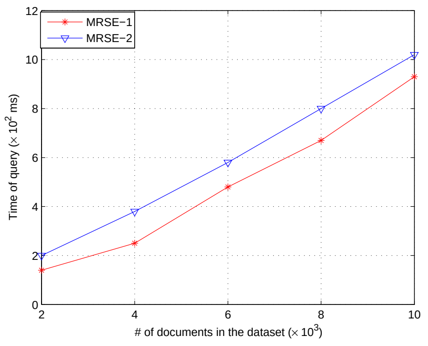


Fig. 9. Time cost of query in MRSE[9]

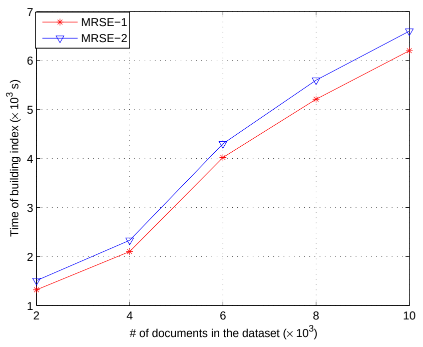


Fig. 7. The time cost for index vectors construction in MRSE[9]

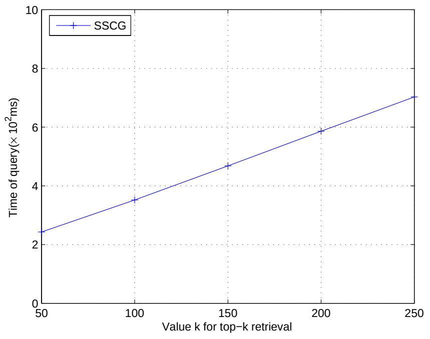


Fig. 10. The time cost for top-k retrieval

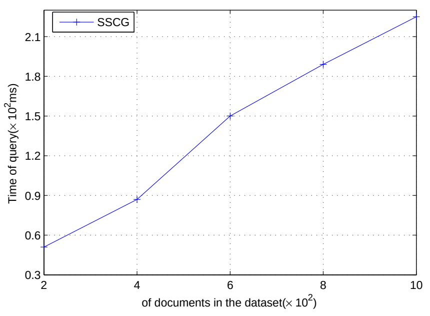


Fig. 8. Time cost of query

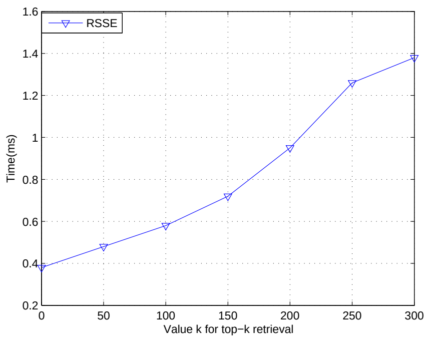


Fig. 11. The time cost for top-k retrieval in RSSE[8]

2) compute the similarity score between D_2 ,Q_2 and D_3, Q_3 if D_1 satisfies Q_1;
3) compare the TSS mentioned in section 1 with each other if D_2 satisfies Q_2 and D_3 satisfies Q_3 ;
4) rank all the similarity scores of relevant documents.

Next, we then present the result of time cost for top-k retrieval and query to prove the efficiency of our scheme.

Fig.8 and Fig.9 show the time cost for query of our scheme and MRSE[9]. We can still compare with MRSE to prove that our scheme is acceptable instead of efficient, although our scheme is different from MRSE essentially. From the two figures, we can conclude that our scheme is not as effective as MRSE. However due to the additional information, we return more accurate results to make up for the loss of efficiency. Fig.10 and Fig.11 show the time for top-k retrieval of our scheme and RSSE [8]. The reason to do so is the same as the query; we simplify our interpretation here. Although the result is worse than that of RSSE, we still consider 0.2s as acceptable for data users for more precise results.

7 Conclusion

In this paper, we define the problem of semantic search based on conceptual graphs over encrypted outsourced data for the first time. We choose CGs among various methods of knowledge representation to represent the documents. To generate the CGs, we apply a state-of-the-art technique, ie.,text summarization and Tregex a tool for simplifying sentences in our method. And to achieve our design goals of both system security and usability, we divide each CG into three index vectors based on the structure of the CG, the type of concept and the value of concept. We apply order preserving symmetric encryption (OPSE) to our scheme to enhance security. Experimental results demonstrate the efficiency of our proposed scheme.

In further work, we will continue to study the semantic search over encrypted cloud data with the support of natural language processing technology. Specifically, we are considering introducing other semantic representations in encrypted form or modifying the process of changing a conceptual graph into a numerical vector which can help improve accuracy and efficiency.

Acknowledgments

This work is supported by the NSFC (U1536206, 61232016, U1405254, 61373133, 61502242), BK20150925, and PAPD fund.

References

[1] S.Miranda-Jimnez, A.Gelbukh, and G.Sidorov, "Summarizing conceptual graphs for automatic summarization task," *Conceptual Structures for STEM Research and Education*,Springer Berlin Heidelberg,pp. 245-253,2013.

[2] R.Ferreira, L.de Souza Cabral, and R.D.Lins, "Assessing sentence scoring techniques for extractive text summarization," *Expert systems with applications*,vol.40,no.14,pp.5755-5764,2013.

[3] M.Liu, R.Calvo, and A.Aditomo, "Using wikipedia and conceptual graph structures to generate questions for academic writing support," *Learning Technologies,IEEE Transactions on*,vol.5,no.3,pp.251-263,2012.

[4] M.Heilman, and N.A.Smith, "Extracting simplified statements for factual question generation ," *Proceedings of QG2010: The Third Workshop on Question Generation*,pp.11-20,2010.

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

[6] Y.-C.Chang and M.Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," *Proceedings of ACNS*,pp.391-421,2005.

[7] R.Curtmola, J.A.Garay, S.Kamara, and R.Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Proceedings of ACM CCS*,pp.79-88,2006.

[8] C.Wang, N.Cao, and J.Li, "Secure ranked keyword search over encrypted cloud data," *Proceedings of Distributed Computing Systems (ICDCS),2010 IEEE 30th International Conference on*,pp.253-262,2010.

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

[10] W.Sun, B.Wang, and N.Cao, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *Proceedings of the 8th ACM SIGSAC symposium on Information,computer and communications security*,pp.71-82,2013.

[11] R.Li, Z.Xu, and W.Kang, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Generation Computer Systems*,vol.30,pp.179-190,2014.

[12] Z.Fu, X.Sun, and Q.Liu, " Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing," *IEICE Transactions on Communications*, vol.98,no.1,pp.190-200,2015.

[13] Z.Xia, X.Wang, and X.Sun, "A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data,"*Parallel and Distributed Systems,IEEE Transactions on*, vo.27,no.2,pp.340-352,2015.

[14] Z.Fu, J.Shu, and X.Sun, "Semantic keyword search based on trie over encrypted cloud data," *Proceedings of the 2nd international workshop on Security in cloud computing*,ACM, pp.59-62,2014.

[15] J.F.Sowa, "*Conceptual structures: information processing in mind and machine*," 1983.

[16] J.F.Sowa, *Conceptual Graphs*. In: Handbook of Knowledge Representation,pp.213-237,2008.

[17] J.Zhong, H.Zhu, and J.Li, Conceptual graph matching for semantic search. *Conceptual structures: Integration and interfaces*,Springer Berlin Heidelberg,pp.92-106,2002.

[18] G.S.Poh, M.S.Mohamad, and M.R.Zaba, Structured encryption for conceptual graphs. *Advances in Information and Computer Security*.Springer Berlin Heidelberg,pp.105-122,2012.

[19] S.Hensman, and J.Dunnion, "Automatically building conceptual graphs using VerbNet and WordNet," *Proceedings of the 2004 international symposium on Information and communication technologies*,Trinity College Dublin,pp.115-120,2004.

[20] W.K.Wong, D.W.Cheung, and B.Kao, "Secure KNN computation on encrypted databases," *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*,pp.139-152,2009.

[21] A.Boldyreva, N.Chenette, and Y. Lee, Order-preserving symmetric encryption.*Advances in Cryptology-EUROCRYPT 2009*, Springer Berlin Heidelberg, pp. 224-241,2009.

[22] Z.Fu, K.Ren, and J.Shu, "Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement," *Parallel and Distributed Systems,IEEE Transactions on*, 2015.

[23] J. Wang, X. Chen,and X. Huang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*,vol.64,no.11,pp.3293-3303,2015.

[24] F. Cheng, Q. Wang,and Q. Zhang, "Highly Efficient Indexing for Privacy-Preserving Multi-keyword Query over Encrypted Cloud Data," *International Conference on Web-Age Information Management*,pp.348-359,2014.

[25] M. Li, S. Yu, and N. Cao, "Authorized private keyword search over encrypted data in cloud computing," *Distributed Computing Systems (ICDCS), 2011 31st International Conference on, IEEE*,pp.383-392,2011.

[26] R. Li, A. X. Liu,and A. L. Wang, "Fast range query processing with strong privacy protection for cloud computing," *Proceedings of the VLDB Endowment*,vol.7,no.14,pp.1953-1964,2014.

[27] X. Chen, J. Li,and J. Ma , "New algorithms for secure outsourcing of modular exponentiations," *Parallel and Distributed Systems,IEEE Transactions on*,vol.25,no.9,pp.2386-2396,2014.

[28] C. Wang, N. Cao, and K. Ren, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *Parallel*

and Distributed Systems,IEEE Transactions on, vol.23,no.8,pp.1467-1479,2012.

[29] J. Li, J. Li, and X. Chen, ”Identity-based encryption with outsourced revocation in cloud computing,” *Computers,IEEE Transactions on*, vol.64,no.2,pp.425-437,2015.

[30] Q. Wang, C. Wang, and K. Ren, ”Enabling public auditability and data dynamics for storage security in cloud computing,” *Computers,IEEE Transactions on*, vol.22,no.5,pp.847-859,2011.

[31] W. Zhang, Y. Lin, and S. Xiao, ”Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing,” *Computers,IEEE Transactions on*, vol.65,no.5,pp.1566-1577,2016.

[32] X. Yi, E. Bertino, and J. Vaidya, ”Private searching on streaming data based on keyword frequency,” *IEEE Transactions on Dependable and Secure Computing*, vol.11,no.2,pp.155-167,2014.

Zhangjie Fu is a visiting scholar of Department of Computer Science and Engineering, Michigan State University. He received his PhD in computer science from the College of Computer, Hunan University, China, in 2012. He works as an assistant professor from 2012 in Department of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include Cloud and Outsourcing Security, Digital Forensics, Network and Information Security. His research has been supported by NSFC, PAPD, and GYHY. Zhangjie is a member of IEEE, and a member of ACM.

Fengxiao Huang received his BS in software engineering from Nanjing University of Information Science and Technology (NUIST), Nanjing, China, in 2014. He is currently pursuing his M-S in computer science and technology at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include cloud security and information security.

Xingming Sun received his BS in mathematics from Hunan Normal University, China, in 1984; his MS in computing science from Dalian University of Science and Technology, China, in 1988; and his PhD in computing science from Fudan University, China, in 2001. He is currently a professor at the Department of Computer and Software, Nanjing University of Information Science and Technology, China. In 2006, he visited the University College London, UK; he was a visiting professor in University of Warwick, UK, between 2008 and 2010. His research interests include network and information security, database security, and natural language processing.

Athanasios V. Vasilakos is recently Professor with the Lulea University of Technology, Sweden. He served or is serving as an Editor for many technical journals, such as the IEEE Transactions on Network and Service Management; IEEE Transactions on Cloud Computing; IEEE Transactions on Information Forensics and Security; IEEE Transactions on Cybernetics; IEEE Transactions on NanoBioscience; IEEE Transactions on Information Technology in Biomedicine; ACM Transactions on Autonomous and Adaptive Systems; the IEEE Journal on selected Areas in Communications. He is also General Chair of the European Alliances for Innovation.

Ching-Nung Yang received the B.S. degree and the M.S. degree, both from Department of Telecommunication Engineering at National Chiao Tung University. He received Ph.D. degree in Electrical Engineering from National Cheng Kung University. He is presently a professor in the Department of Computer Science and Information Engineering at National Dong Hwa University, and is also an IEEE senior member. His current research interests include coding theory, information security, and cryptography.