Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

Secure semantic expansion based search over encrypted cloud data supporting similarity ranking

Zhihua Xia[1,2], Yanling Zhu[1,2], Xingming Sun[1,2*] and Lihong Chen[1,2]

Abstract

With the advent of cloud computing, more and more information data are outsourced to the public cloud for economic savings and ease of access. However, the privacy information has to be encrypted to guarantee the security. To implement efficient data utilization, search over encrypted cloud data has been a great challenge. The existing solutions depended entirely on the submitted query keyword and didn't consider the semantics of keyword. Thus the search schemes are not intelligent and also omit some semantically related documents. In view of the deficiency, as an attempt, we propose a semantic expansion based similar search solution over encrypted cloud data. Our solution could return not only the exactly matched files, but also the files including the terms semantically related to the query keyword. In the proposed scheme, a corresponding file metadata is constructed for each file. Then both the encrypted metadata set and file collection are uploaded to the cloud server. With the metadata set, the cloud server builds the inverted index and constructs semantic relationship library (SRL) for the keywords set. After receiving a query request, the cloud server first finds out the keywords that are semantically related to the query keyword according to SRL. Then both the query keyword and the extensional words are used to retrieve the files. The result files are returned in order according to the total relevance score. Eventually, detailed security analysis shows that our solution is privacy-preserving and secure under the previous searchable symmetric encryption (SSE) security definition. Experimental evaluation demonstrates the efficiency and effectives of the scheme.

Keywords: Secure; Semantic expansion; Rank; Semantic relationship library; Cloud data

Introduction

Cloud Computing enables cloud customers to enjoy the on-demand high quality applications and services from a centralized pool of configurable computing resources. This new computing model can relieve the burden of storage management, allow universal data access with independent geographical locations, and avoid capital expenditure on hardware, software, and personnel maintenances, etc [1].

As cloud computing becomes mature, lots of sensitive data is considered to be centralized into the cloud servers, e.g. personal health records, secret enterprise data, government documents, etc [1,2]. The straightforward solution to protect data privacy is to encrypt sensitive data before being outsourced. Unfortunately, data encryption, if not done appropriately, may reduce the effectiveness of data utilization. Typically, a user retrieves files of interest to him/her via keyword search instead of retrieving back all the files. Such keyword-based search technique has been widely used in our daily life, e.g. Google plaintext keyword search. However, the technologies are invalid after the keywords are encrypted.

In recent years, searchable encryption (SE) techniques have been developed for secure outsourced data search [3-8]. Some further researches focus on search efficiency [9,10], multi-keyword search [11,12], and secure dynamic updating [13]. But they only support exact keyword search. To enhance the search flexibility and

* Correspondence: sunnudt@163.com
[1]Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing 210044, China
[2]School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing 210044, China

usability, some research has been done on fuzzy keyword search [14-18]. These solutions support tolerance of minor typos and format inconsistencies, such as, search for "million" by carelessly typing "milion", or "datamining" by typing "data-mining". These schemes mainly take the structure of terms into consideration and use edit distance to evaluate the similarity. They didn't consider the terms semantically related to query keyword, thus many related files are omitted. In addition, these fuzzy systems send back all relevant files solely upon presence/absence of the keyword, and result-ranking is still out of considering.

In this paper, from a new perspective, we propose a similar search solution based on semantic query expansion while supporting similarity ranking. Semantic expansion based similar search reinforces the system usability by returning the exactly matched files and the files including the terms semantically related to the query keyword. In the proposed scheme, a corresponding file metadata is constructed for each file. Then the encrypted metadata set and file collection are uploaded to the cloud server. With the metadata set, the cloud server builds the inverted index and constructs semantic relationship library (SRL) for the keywords set. The co-occurrence of terms is used to evaluate the semantic relationship between terms in SRL. Upon receiving a query request, the cloud server automatically finds out the terms which are semantically related to the query keyword according to the value of semantic relationship between terms in SRL. Then both the keyword and the semantically expanded words are used to retrieve files. Finally, the matched files are returned in order according to the total relevance score. In the process, to ensure security and final result ranking, we properly modify a crypto primitive order-preserving encryption to protect the relevance score. Detailed security analysis shows that the solution correctly realizing the goal of semantic search, while preserving the privacy. Extensive experimental evaluation demonstrates the efficiency and effectives of the scheme.

Related work

Early searchable encryption (SE) schemes provide the solution mainly for secure exact keyword search [3-8]. In the symmetric key setting, Song et al. proposed the first SE scheme, where each word in the file should be encrypted with a two-layered encryption construction independently [3]. To improve search efficiency, some researchers turn to index technique. Goh et al. and Chang et al. both proposed similar secure per-file index, where an index including trapdoors of all unique words is constructed for each file [4,6]. Curmola et al. presented a per-keyword index construction, where each entry of the whole hash table index contains the trapdoor for a keyword and an encrypted set of file identifiers [7]. To further enhance system usability, some other researchers propose ranked search. Wang et al. proposed a solution for ranked single-keyword search regarding to certain relevance score [9,10]. Cao et al. and Yang et al. proposed the scheme for multi-keyword ranked search, where "Inner product similarity" is used for result ranking [11,12]. Emil et al proposed a hierarchical index structure to achieve more secure and effective dynamic updating [13]. As a complementary approach, Boneh et al. proposed the first public key based searchable encryption scheme in the public-key setting [5].

However, all the above schemes support only exact keyword search. Namely, users' searching input should exactly match the keywords contained in the files. As an attempt to enhance search flexibility, fuzzy keyword search over encrypted cloud data has been proposed [14-16,19]. Li et al. and Wang et al. both exploited edit distance as the similarity metric of keywords to construct the fuzzy keywords set as indexes. Besides, the wildcard-based technique is used for storage-efficiency of fuzzy keywords set [15,14]. Liu proposed "dictionary-based fuzzy set construction" to further reduce the size of fuzzy keywords set [17]. Relying on an asymmetric security model, Bringer et al. proposed a fuzzy search scheme based on the embedding of edit distance into Hamming distance [19]. This scheme does not need priori define of fuzzy keywords set. Chuah proposed a fuzzy multi-keyword search scheme, where edit distance is also used to evaluate the similarity between terms [16]. Besides, an index BedTree is constructed to improve search efficiency with n-gram technique. Without the construction of fuzzy keywords set, Jin introduced new measures, e.g. n-gram bloom-filter and frequency vector, to approximately measure the similarity over encrypted string [18]. Note that, the above fuzzy search systems consider the similarity metric mainly from the structure of keywords, not from the semantic relationship. Thus, practically usable semantic search remains to be addressed in the context of encrypted data search.

In this paper, we propose a ranked semantic expansion based similar search scheme in the symmetric key setting, which take both the semantic search and result ranking into consideration.

Problem formulation

System model

We consider the system model involving three different entities: data owner, data user and cloud server, as illustrated in Figure 1.

Data owner uploads a collection of n text files $F = \{F_1, F_2, F_3, \cdots, F_n\}$ in encrypted form C, together with the encrypted metadata set, to the cloud server. Note that, a corresponding file metadata is constructed for each file. Each file in the collection is encrypted with common symmetric encryption algorithm, e.g. AES.

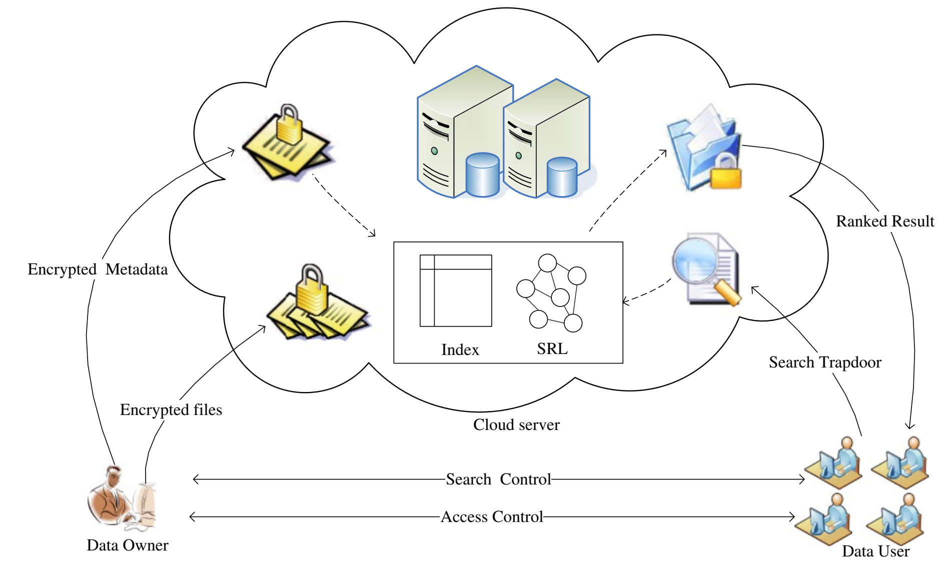

Figure 1 Framework of the semantic expansion based similar search over encrypted cloud data.

Data user provides a search trapdoor T_w for keyword w to the cloud server. In our paper, we assume the authorization between the data owner and users is appropriately done.

Cloud server first constructs the index and SRL using the metadata set provided by data owner, thus reduce the computing burden on owner, e.g. index creating. Upon receiving the request T_w, the cloud server automatically expands the query keyword based on SRL. Then the server searches the index, and returns the matching files to the user in order. Finally, the access control mechanism, which is out of the scope of this paper, is employed to manage the capability of the user to decrypt the received files.

Threat model

In this paper, we use the same threat model described in previous searchable symmetric encryption (SSE) scheme [6,7,9,11,15,16]. We consider an "honest-but-curious" server in our model. Specifically, the cloud server honestly follows the designated protocol specification, but is "curious" to infer and analyze all data information available on the server so as to learn additional information. In other words, the cloud server has no intention to actively modify the stored data or disrupt any other kind of service. Thus we consider the threat models with attack capabilities as follows.

Known background Model: In this model, except for the encrypted dataset and metadata set the owner upload, the server is assumed to have additional knowledge on the dataset, e.g. the subject and its related statistical information. For instance, the server can utilize the keyword frequency statistics to infer keywords.

Design goals

To enable effective and secure ranked semantic expansion search over outsourced cloud data under the aforementioned model, our mechanism should achieve the following design goals.

1) Ranked semantic expansion search: To design a similar search scheme that supports semantic search over encrypted cloud data by expanding the query keyword upon semantic relationship of terms, which finally returns the retrieved files in order.
2) Security guarantee: To prevent cloud server from learning the plaintext of the data files and keywords. Compared to the existing SSE schemes, the scheme should achieve the as-strong-as possible security strength.
3) Efficiency: To achieve the above goals with minimum communication and computation overhead.

Notation

F – the plaintext file collection, denoted as a set of n data files $F = \{F_1, F_2, \cdots, F_n\}$.
C – the encrypted file collection, stored in the cloud server, denoted as $C = \{c_1, c_2, \cdots, c_n\}$.
$id(F_i)$ – the identifier of file F_i that can help uniquely locate the actual file.
W – the dictionary, i.e., the keywords set extracted from F, denoted as a set of m keywords $W = \{w_1, w_2, \cdots w_m\}$.
M – the encrypted metadata set, denoted as a set of n file metadata $M = \{M(F_i)\}, i = 1, 2, \cdots n$.

I – the inverted index including a set of m posting lists $I = \{I(w_i)\},\ \ i = 1, 2, \cdots m$.

T_w – the trapdoor generated for a query keyword w by a user.

S_w – the semantically expanded keyword set of w, it is a subset of W, denoted as $S_w = \{w_1', w_2', \cdots\}$.

Preliminaries

Semantic query expansion

In the domain of plaintext retrieval, automatic query extension has been a technique to improve the recall and precision of retrieval for a long time [20]. It uses the semantically related words to expand the particular query, and makes the query request more satisfy the user's intent.

The key step of semantic query expansion is to find out the semantic relationship between the keywords. Some researchers utilized readily available corpus independent knowledge models [21], e.g. WordNet, EuroWordNet, and some others dynamically constructed the semantic relationship from the document collection by the technologies such as term clustering [22,23], and mutual information model [24-26]. Among these technologies, mutual information model is widely used [24,26-29].

Refer to the formula used in [26], which adopted the mutual information model to implement semantic search in web. The mutual information $I(x, y)$ is defined as

$$I(x, y) \equiv log_2 \frac{P(x, y)}{p(x)p(y)} \tag{1}$$

Here $P(x, y)$ is the probability of observing x and y together. $p(x)$ and $p(y)$ are the probabilities of observing x and y independently in the collection. The higher the semantic relationship between x and y is, the larger the co-occurrence degree is, and consequently the larger the mutual information $I(x, y)$ is.

Then normalize the mutual information into a value of relationship in interval [0, 1]. The semantic relationship library will be constructed as a weighted graph structure showed in Figure 2.

Inverted index

Inverted index is a widely used indexing structure in information retrieval. It is consist of a list of mappings from keywords to the set of files that contain this keyword [30]. For the purposes of ranking, the numerical relevance score is computed for each file based on $TF \times IDF$ rule introduced later in subsection "Basic definition". An example index structure of keyword w_i is shown in Table 1. Here S_{ij} $(j = 1, \cdots, n_i)$ denotes the relevance score of file F_{ij} in response to w_i, n_i is the number of files contain keyword w_i.

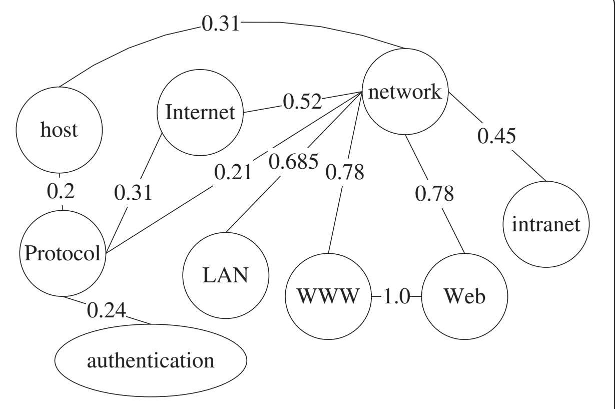

Figure 2 An example of semantic relationship library.

Order-preserving Encryption (OPE)

The OPE is a deterministic encryption scheme, whose encryption function preserves the numerical ordering in plaintext-space [31,32]. More specifically, a function f: $D = \{1, \cdots, M\} \rightarrow R = \{1, \cdots N\}$ is order-preserving, if for all $a, b \in D, f(a) > f(b)$ if $a > b$. Generally, any order-preserving function can be defined as a combination of M out of N ordered items, which can be calculated by $\begin{pmatrix} N \\ M \end{pmatrix}$. The adversary has to execute exhaustive enumeration, namely searching over all the possible combination, to break the encryption. So the number of combination, which is maximized when $M = N/2$, should be large enough to ensure the security. If the security level is chosen to be 2^{80}, since $(N/M)^M \leq \begin{pmatrix} N \\ M \end{pmatrix}$, it is suggested to choose $M = N/2 > 80$. A plaintext m in domain D is always mapped to a random-sized non-overlapping bucket in range R. Then a ciphertext c is chosen within the bucket depend on the value of some random function.

Basic definitions

Ranking function

A ranking function is used to measure relevance scores of matching files to a given query in information retrieval. The most widely used measurement for evaluating relevance score is $TF \times IDF$ rule. TF (Term frequency) is used to measure the importance of the term within the particular file, defined as the number of times a given term or keyword appears within a file. IDF is used to measure the overall importance of the term within the whole collection,

Table 1 An example of inverted index

Keyword	w_i				
File ID	$id(F_{i1})$	$id(F_{i2})$	$id(F_{i3})$	…	$id(F_{in_i})$
Relevance score	S_{i1}	S_{i2}	S_{i3}	…	S_{in_i}

defined as the total number of documents in the collection divided by the total number of documents including that word. Note that, we focus on single keyword search in our scheme. Thus without loss of generality, the relevance score of single keyword can be computed using equation 2, which is widely used in the literature [33]:

$$Score(w, F_i) = \frac{1}{|F_i|} \cdot \left(1 + \ln f_{i,w}\right) \cdot \ln\left(1 + \frac{n}{f_w}\right) \qquad (2)$$

Here w denotes the query keyword; $f_{i,w}$ is the TF of term w in file F_i; f_w denotes the number of files that contain keyword w. n is the number of files in the collection, while $|F_i|$ is the length of file F_i, obtained by counting the number of indexed terms in the file.

In our scheme, we first expand the query keyword based on SRL, and then both the keyword and its semantically related words are used to retrieve the files. So F_d's total relevance score will be computed for result ranking with equation 3.

$$TScore(w, F_d) = Score_w + \sum_{\forall w_i' \in S_w} Score_{w_i'} \times R_i \qquad (3)$$

Here $Score_w$ represents the relevance score of the input keyword; $Score_{w_i'}$ represents the relevance score of expanded keyword w_i', while R_i is the value of semantic relatedness.

File metadata

A piece of file-metadata is constructed for each file. The file-metadata consists of the file ID, keywords, and the relevance scores (refer to equation 2) of keywords in response to the file. If file F_i contains keyword w_j, a tuple w_j, s_{ji} is insert into metadata $M(F_i)$, where s_{ji} represents the relevance score of keyword w_j response to file F_i. All of the file metadata constitute metadata set, which is shown in Figure 3.

Secure Semantic Expansion based Similar Search Scheme

The scheme consists of six algorithms (*KeyGen*, *BuildMD*, *BuildIndex*, *BuildSRL*, *TrapdoorGen*, and *SearchIndex*), which can be constructed in two phases—Setup and Retrieval.

The setup phase

In this phase, data owner initializes the public and secret parameters of the system by executing *KeyGen*, and preprocesses the file collection F using *BuildMD* to generate the encrypted metadata for each file. Finally the owner uploads both the encrypted file collection C and metadata set M to the cloud server. With M received from data owner, the server constructs the index using *BuildIndex* and semantic relationship library using *BuildSRL*. In addition, the necessary secret parameters, e.g. the trapdoor generation key, should be distributed to a group of authorized users by employing off-the-shelf public key cryptography or broadcast encryption. Details are as follows:

1) The data owner initiates the scheme by calling *KeyGen* $(1^k, 1^l, 1^p)$. It takes the security parameters k, l, p as inputs and generates random keys $x \overset{R}{\leftarrow} \{0,1\}^k$, $y \overset{R}{\leftarrow} \{0,1\}^l$. Finally it outputs secret keys set $K = \{x, y, 1^l, 1^p\}$ used for later encryption, such as trapdoor generation and relevance score encryption.
2) Then the data owner builds the secure metadata for each file in file collection F by calling *BuildMD*(K, F), It takes the secret K and dataset F as inputs and outputs the encrypted metadata set M. The function extracts the keywords in each file and computes the corresponding relevance score. The keyword in the metadata is encrypted with collision resistant hash function $\pi : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^p$ $(p > \log m)$, where m denotes the size of keywords set. The relevance score is encrypted with order-preserving

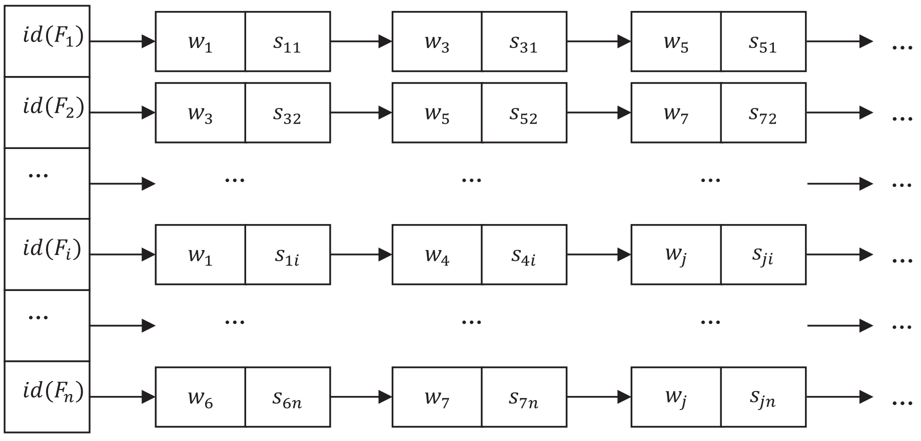
$id(F_1)$
w_1 s_{11} w_3 s_{31} w_5 s_{51} …
$id(F_2)$
w_3 s_{32} w_5 s_{52} w_7 s_{72} …
…
$id(F_i)$
w_1 s_{1i} w_4 s_{4i} w_j s_{ji} …
$id(F_n)$
w_6 s_{6n} w_7 s_{7n} w_j s_{jn} …

Figure 3 An example of metadata set.

encryption algorithm $OPE: \{0,1\}^l \times \{0,1\}^d \to \{0,1\}^r$, where d *and* r respectively represent the bit length used to denote all the values in domain D and range R. The detail is shown in Algorithm 1.

Algorithm 1 Build metadata set M

```
procedure $BuildMD(K, F)$
    for each file $F_i \in F$
        for $1 \le j \le m$
            if file $F_i$ contains keyword $w_j$
                1)calculate the relevance score for keyword $w_j$ responding to $F_i$, denoted as $s_{ji}$;
                2)encrypt the keyword as $ew_j = \pi_x(w_j)$, and encrypt relevance score as
                    $es_{ji} = OPE_y(s_{ji})$;
                3)store tuple $ew_j, es_{ji}$ in metadata $M(F_i)$;
            end if
        end for
        Insert $M(F_i)$ into $M$
    end for
    return $M$
end procedure
```

Figure 4 is an example of the encrypted metadata set.

3) When receiving the secure metadata, the server builds the inverted index by calling $BuildIndex(M)$. The function extracts the encrypted keywords and constructs a posting list for each keyword. If keyword ew_j included in file metadata $M(F_i)$, the element $\{id(F_i)||es_{ji}\}$ is inserted into posting list of keyword ew_j. The details are given in Algorithm 2. The SRL is also built upon the metadata set and uses common association rules algorithm to mining the co-occurrence relationship of keywords.

Algorithm 2 Build index I

```
procedure $BuildIndex(M)$
    for each metadata $M_i$ corresponding to file $F_i$
        if $M_i$ contains encrypted keywords $ew_j$
            1)extract the encrypted relevance score $es_{ji}$ of $ew_j$ corresponding to the file $F_i$.
            2)store $\{id(F_i) || es_{ji}\}$ as an element in the posting list $I(ew_j)$;
        end if
        Insert $I(ew_i)$ into $I$
    end for
    return $I$
end procedure
```

An example of secure inverted index constructed by cloud server is shown in Figure 5.

The retrieval phase

In this phase, the user generates a secure trapdoor of his interested keyword using *TrapdoorGen*, and submits it to the cloud server. Upon receiving the query trapdoor, the cloud server first automatically expands the query keyword. Then the server searches the index via *SearchIndex*, and eventually sends back the matched files in a ranked sequence according to the total relevance scores. During the process, beyond the order of the relevance scores, nothing or little information should be leaked. Details are as follows.

1) The user generates a trapdoor $T_w = \pi_x(w)$ for an interested keyword w, by calling $TrapdoorGen(w)$.
2) Upon receiving the trapdoor T_w, the server first expands the query keyword to obtain the extensional query trapdoor $T_w' = \{\pi_x(w), \pi_x(w_i')\}, \ \forall\ w_i' \in S_w$. By calling *SearchIndex*, the server locates the matching entries of the index via $\pi_x(w)$ and $\pi_x(w_i')$, which include the file identifiers and the associated order-preserved encrypted relevance scores.
3) The server then computes the total relevance score of each file to the query according to equation 3. In the end, the server sends back the matched files in a ranked sequence, or sends top-k most relevant files if the user provides the optional value k.

Towards one-to-many order-preserving encryption

To implement efficient result ranking, we use OPE encrypt the relevance score. Thus the server can rank the retrieved files directly according to the encrypted relevance score. However, the original OPE is a deterministic encryption scheme, if not disposed properly, it will leak as much information as any deterministic encryption scheme does [32]. In particular, the statistical information of the scores, such as the distribution slope, value range etc., can be used to identify the specific keyword in the query [9].

Therefore we need to modify the OPE to suit our requirement. The original OPE first maps the plaintext m in domain D to an interval bucket in range R. Then the ciphertext c is chosen in the bucket using m as the random seed for the random selection function. The modified OPE should map the same plaintext score to different ciphertext, and still globally preserve the order of relevance score. Thus a one-to-many OPE scheme is desired to reduce the amount of information leakage. More specifically, in the final ciphertext selection process, together with the plaintext m, the unique file ID is introduced as an additional random seed. Thus the same plaintext will not be deterministically mapped to the same ciphertext, but a random value within the randomly assigned bucket in range R. Algorithm 3 shows the whole process, where $GetCoins(\cdot)$ is a random coin generator, $HYGEINV(\cdot)$ is the $HGD(\cdot)$ sampling function instance in MATLAB. In the process, a plaintext m in domain $D = \{1, \cdots, M\}$ is mapped into ciphertext c selected in range $R = \{1, \cdots N\}$, $id(F)$ denotes the corresponding file ID. In the paper, the one-to-many OPE is denoted as $OM - OPE$.

Algorithm 3 one-to-many Order-preserving encryption

```
1:  procedure $OM-OPE_k(D,R,m,id(F))$
2:      while $|D| != 1$ do
3:          $\{D,R\} \leftarrow BinarySearch(K,D,R,m)$;
4:      end while
5:      $coin \overset{R}{\leftarrow} GetCoins(K,(D,R,1||m,id(F)))$;
6:      $c \overset{coin}{\leftarrow} R$;
7:      return $c$
8:  end procedure
9:  procedure $BinarySearch(K,D,R,m)$
10:     $M \leftarrow |D|$; $N \leftarrow |R|$;
11:     $d = min(D)-1$; $r = min(R)-1$;
12:     $y \leftarrow r + N/2$;
13:     $coin \overset{R}{\leftarrow} GetCoins(K,(D,R,0||y))$;
14:     $x \overset{R}{\leftarrow} d + HYGEINV(coin,M,N,y-r)$;
15:     if $m \leq x$ then
16:         $D \leftarrow \{d+1,\cdots,x\}$; $R \leftarrow \{r+1,\cdots,y\}$;
17:     else
18:         $D \leftarrow \{x+1,\cdots,d+M\}$; $R \leftarrow \{y+1,\cdots,r+N\}$;
19:     end if
20:     return $\{D,R\}$;
21: end procedure
```

The mapping scheme should be as random as possible to eliminate the predictability of the keyword specific score distribution. Obviously, the larger the size of range R is, the less specific characteristics will be preserved. However, considering the efficiency of *HGD* function, the size of range R cannot be unboundedly large. So the range size $|R|$ should be properly tradeoff between randomness and efficiency.

To guarantee the security of keywords in the metadata set, the relevance score should be encrypted with $OM-OPE_y(\cdot)$ instead of $OPE_y(\cdot)$ in Algorithm 1.

Security analysis

We estimate the security of the proposed scheme by proving the security guarantee stated above (refer to Design goals). That is, both the data files and the keywords are not leaked to the server.

Security analysis for the ranked semantic expansion Search

We analyze the solution with respect to the aforementioned search privacy requirement, e.g. keyword privacy and file confidentiality.

- File confidentiality: the file confidentiality depends on the inherently security strength of the symmetric encryption scheme, so the file content is obviously protected well.
- Keyword privacy:
  1. The query trapdoor is generated using the symmetric encryption scheme, so the privacy of query keyword depends on the inherently security strength of the symmetric encryption scheme.
  2. The proposed scheme introduces some additional information in the index compared to the original SSE, such as the encrypted relevance scores and the values of relationship between terms. Thus the privacy of keyword in the index depends on not only the symmetric encryption scheme. We discuss the security from two aspects.

On one hand, as defined in the thread model, the server may predict the plaintext of keyword depends on the score distribution. Thus the $OM-OPE$ is used to encrypt the score, which could flatten the distribution of relevance score. So the keyword privacy mainly depends on the security of $OM-OPE$. In the next part, we analyze the security of $OM-OPE$ in detail. As discussed, if the data owner properly enlarges the range R, the relevance score will be randomly mapped to a sequence of order-preserved numeric values with very low duplicates. So $OM-OPE$ makes it difficult for the adversary to predict the plaintext score distribution, let alone predict the keywords.

On the other hand, as shown in Table 2, the semantic relationship values between terms do not have their peculiarities, which cannot be effectively used for statistical analysis. Note that, in the previous literature with inverted index [9], the server can also get the co-occurrence degree of terms by recording and analyzing the search result. Thus the leaking of relationship information shouldn't be a main secure problem we have to solve in current work.

Security analysis for one-to-many OPE

The one-to-many OPE scheme introduces the file ID as the additional seed in the ciphertext chosen process. So the same plaintext will not be deterministically mapped

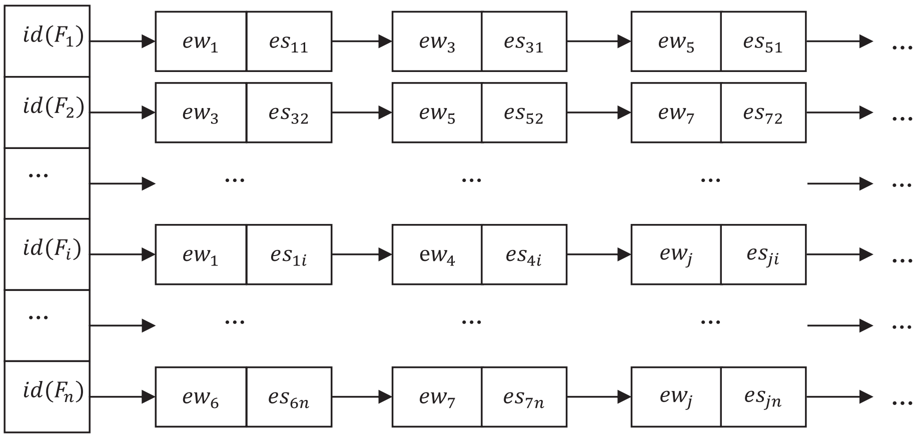

Figure 4 **An example of encrypted metadata set.**

to the same ciphertext, but a random value in the assigned bucket in range R. This helps flatten the score distribution of keyword, and protect the keyword privacy from statistical attack.

However, if there are many duplicates of plaintext m, the ciphertext distribution may not be flattened effectively for the small size of assigned bucket in range R. So we should expand the range R properly to ensure the low duplicates on the ciphertext range, it will be difficult for the adversary to analyze which points in R belong to the same plaintext score.

In this paper, we use the min-entropy to choose the size of R. It is defined as: $H(\sigma) = -\log\left(\max_{\alpha} Pr[\sigma = \alpha]\right)$, where σ is a discrete random variable, α denotes a state of σ with the max probability. In general, the higher $H(\sigma)$ is, the more difficult the σ can be predicted. If $H(\sigma) \in w(\log k)$, the min-entropy of variable σ will be high, where k is the bit length needed to denote all the states of σ [8]. We could choose $H(\sigma)$ as $(\log k)^c$ where $c > 1$ [9]. Then the least size $|R|$ should satisfy the equation 4:

$$(\log(\log|R|))^C \leq -\log\left(\frac{max/\left(|R|\cdot\frac{1}{2}^{5\log M+12}\right)}{\delta}\right) \quad (4)$$

Here max denotes the maximum number of score duplicates within the metadata set. δ denotes the totoal number of scores to be mapped within metadata set. Wit $D = \{1, \cdots, M\}, M = |D|$, the total recursive calls of $BinarySearch(\cdot)$ function (line 9 in Algorithm 3) is at most $5\log M + 12$ on average. If the range size $|R|$ is denoted in bits, namely $k = \log|R|$, we will get equation 5. With the established file metadata set, it is easy to determine the proper rage size $|R|$.

$$\frac{max \cdot 2^{5\log M+12}}{2^k \cdot \delta} = \frac{max \cdot M^5}{2^{k-12} \cdot \delta} \leq 2^{-(\log k)^c} \quad (5)$$

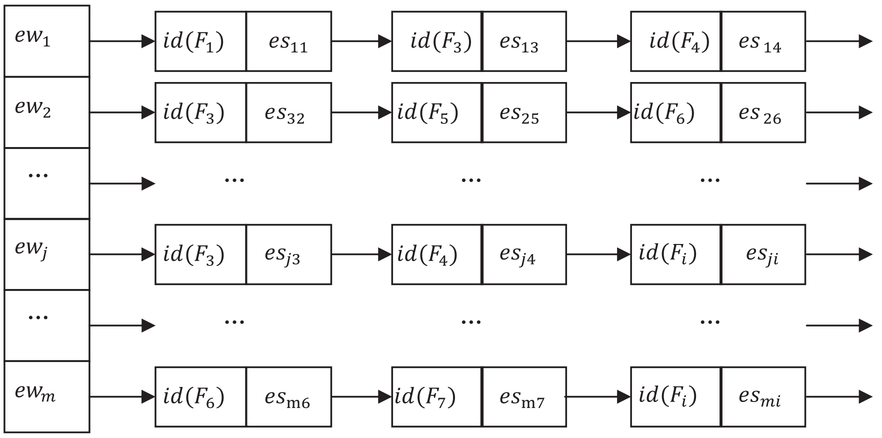

Figure 5 **An example of secure inverted index.**

Table 2 An example of semantic relationship between terms

Keyword	Keyword	Similarity
host	network	0.31
lan	ethernet	0.31
protocol	internet	0.31

As discussed above, if we properly choose the range R, the randomness in the ciphertext selection process will effectively mitigate the useful information revealed to the cloud server.

Performance analysis

To evaluate the performance of our proposed scheme, we implemented the secure search system using C++ on a windows machine with Intel Core 2 Duo CPU Processor running at 2.93GHZ, 2.94GHZ. The experimental evaluation was conducted on a real data set: Request for comments database(RFC) [34], this file set contains a large number of technical keywords. The overall performance evaluation of our scheme includes the cost of metadata construction, the time necessary for index and SRL construction as well as the efficiency of search.

Metadata construction

The main overheads for data owner are time cost and storage cost of metadata construction. To build a metadata for each document F_i in the dataset F, we should extract the keywords and compute the associated relevance score, then encrypt the keywords and scores. The time cost of each entry directly depends on the number of keywords in the file, while the overall efficiency is also related to the number of the files in the collection. So Table 3 lists the metadata construction performance for a dataset of RFC files. Both the metadata size and construction time listed are the average value, for the reason that it eliminates the difference of various file set construction choices.

Index and SRL construction

In our construction we should scan the whole metadata set to extract the keywords and build the inverted index with corresponding scores. Figure 6 shows that the whole index is nearly linear with the size of M, namely the number of documents in the collection. The SRL is also built by scanning the metadata set, with the certain support threshold, the number of entries is the main factor to the efficiency. Figure 7 shows the time cost of building SRL against the increasing size of M or dataset. In addition, taking into account the abundant computing resources on server, the performance of building index and SRL is practically efficient.

Table 3 File metadata construction overhead

Number of files	Per file metadata size	Per file metadata build time
1000	0.18 KB	0.28 s
2000	0.20 KB	0.30 s
3000	0.21 KB	0.32 s

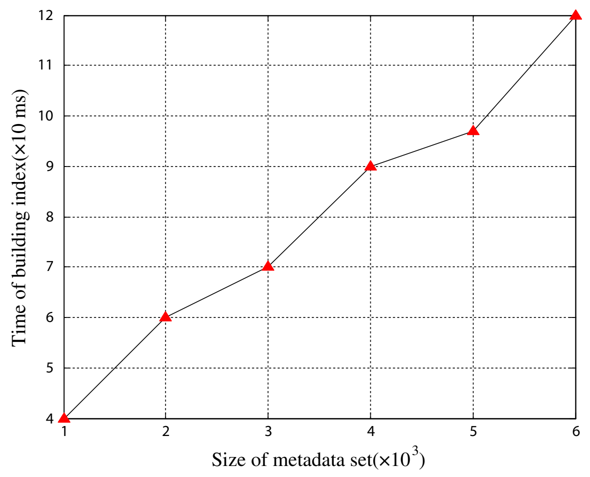

Figure 6 The time cost for building index.

Search efficiency

The search process includes query extension, fetching the posting list in the index, calculating the total relevance score and ranking the result in descending order. Compared to the original ranked search, our approach introduces the keyword extension cost, and the calculation cost of final relevance score. So the size of semantically expanded keywords set is a factor to the query efficiency. Figure 8 shows the average time cost of query against the size of S_w. With result ranking, top-k search could return the most satisfied files more efficiently. In addition, as the evaluation of overall search performance,

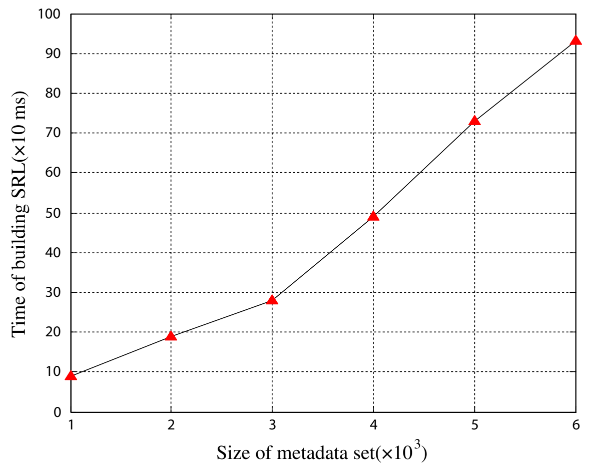

Figure 7 The time cost for building SRL.

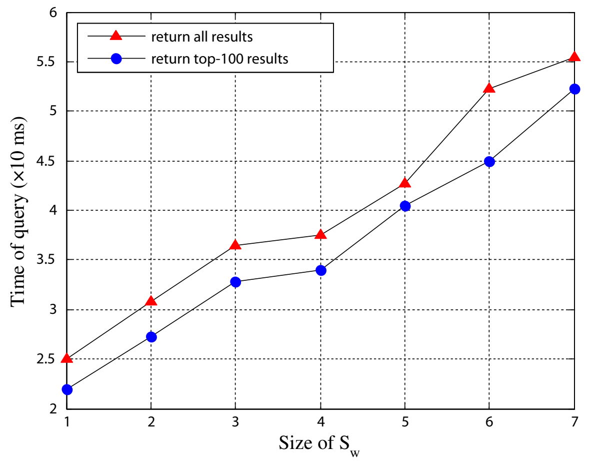

Figure 8 Time cost of query. For query keyword with different size of semantically expanded keywords set, n=1000.

Figure 9 shows the average time cost of query against the number of files. Besides, the index and SRL could be stored with a tree based data structure, so that the server does not need to traverse all the keywords entries.

Recall factor of the search

By analyzing the search result, the overall recall rate is improved, and the query results are more in line with the user's actual intentions. E.g. a user inputs a keyword 'protocol', the files which contain related words like 'internet', 'network', 'authentication' will also be returned, in addition, the files which include most of the words will also be ranked forward.

Conclusion

In this paper, as an initial attempt, we propose a secure semantic expansion based similar search scheme over encrypted cloud data. The proposed scheme could return not only the exactly matched files, but also the files including the terms semantically related to the query keyword. The encrypted files and metadata set are outsourced to the server by the owner. With the file metadata, the cloud builds the inverted index and constructs semantic relationship library (SRL) for the keywords. The co-occurrence of terms is used to capture the semantic relationship of keywords in the dictionary, which offers appropriate semantic distance between terms to accomplish the query keyword extension. Then we derive a one-to-many OPE scheme to protect the term frequency, while ensure the computing of total relevance score. Experimental evaluation demonstrates the efficiency and effectives of the scheme.

As our future work, the most practical one is to further improve the security of our solution. Thus new crypto techniques still need to be designed to protect the semantic information while keep the ability to calculate the relevance score. In addition, we intend to research on multi-keyword semantic search scheme which further introduces the semantic relationship between terms, e.g. the position of terms.

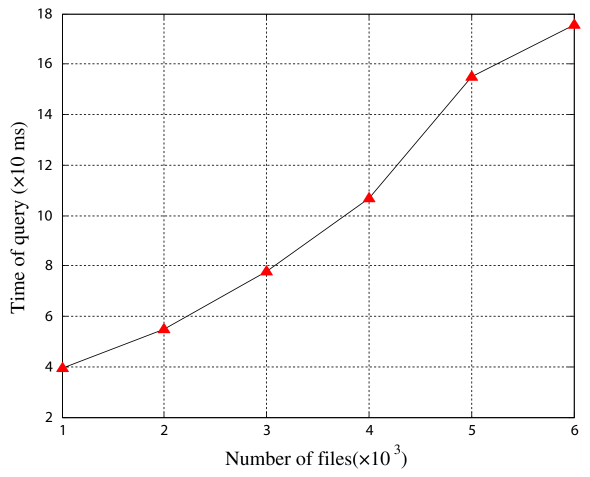

Figure 9 The overall query performance.

Abbreviations

SRL: Semantic relationship library; SE: Searchable encryption; SSE: Searchable symmetric encryption; OPE: Order-preserving encryption; OM-OPE: One-to-many order-preserving encryption.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

ZX, YZ and XS proposed the solution of secure ranked semantic key word search over encrypted cloud data and conducted experiments. ZX and YZ designed the secure semantic keyword search model using semantic extension. XS and YZ developed the encryption scheme for security. YZ completed the main programming work, and analyzed the experiment result and security with ZX and LC. ZX and LC also helped revise the manuscript. The submission was approved by all co-author. All authors read and approved the final manuscript.

Authors' information

Zhihua Xia received his PhD in computer science and technology from Hunan University, China, in 2011. He works as a lecture in School of Computer& Software, Nanjing University of Information Science & Technology. His research interests include Steganography and Steganalysis, digital forensic, image processing, pattern recognition, and cloud security.
Yanling Zhu is currently studying for her master's degree in School of Computer & Software, Nanjing University of Information Science & Technology, China. Her research interests include information security and cloud security.
Xingming Sun received his BS in mathematics from Hunan Normal University, China, in 1984, MS in computing science from Dalian University of Science and Technology, China, in 1988, and PhD in computing science from Fudan University, China, in 2001. He is currently a professor in School of Computer & Software, Nanjing University of Information Science & Technology, China. His research interesting include network and information security, digital watermarking, digital forensic, database security, natural language processing, and cloud security.
Lihong Chen is currently studying for the master's degree in School of Computer & Software, Nanjing University of Information Science & Technology, China. Her research interests include information security and cloud security.

Acknowledgements
This work is supported by the NSFC (61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61373133), GYHY201206033, 201301030, 2013DFG12860, BC2013012, Open fund of Jiangsu Engineering Center of Network Monitoring (KJR1308) and PAPD fund.

Received: 30 August 2013 Accepted: 19 May 2014
Published online: 02 July 2014

References

1. Ren K, Wang C, Wang Q (2012) Security challenges for the public cloud. IEEE Internet Comput 16(1):69–73
2. Kamara S, Lauter K (2010) Cryptographic cloud storage. In: Financial Cryptography and Data Security. Springer, Berlin/Heidelberg, pp 136–149
3. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy. IEEE, Berkeley, California, pp 44–55
4. Goh E-J (2003) Secure indexes. Cryptology ePrint Archive, Report 2003/216
5. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Advances in Cryptology-Eurocrypt 2004. Springer, Berlin/Heidelberg, pp 506–522
6. Chang Y-C, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. In: Applied Cryptography and Network Security. Springer, Berlin/Heidelberg, pp 442–455
7. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on Computer and communications security. ACM, Alexandria, VA, USA, pp 79–88
8. Bellare M, Boldyreva A, O'Neill A (2007) Deterministic and efficiently searchable encryption. In: Advances in Cryptology-CRYPTO 2007. Springer, Berlin/Heidelberg, pp 535–552
9. Wang C, Cao N, Li J, Ren K, Lou W (2010) Secure ranked keyword search over encrypted cloud data. In: 30th IEEE International Conference on Distributed Computing Systems (ICDCS). IEEE, Genoa, Italy, pp 253–262
10. Wang C, Cao N, Ren K, Lou W (2012) Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans Parallel Distrib Syst 23(8):1467–1479
11. Cao N, Wang C, Li M, Ren K, Lou W (2011) Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of IEEE INFOCOM. IEEE, Shanghai, China, pp 829–837
12. Yang C, Zhang W, Xu J, Xu J, Yu N (2012) A Fast Privacy-Preserving Multi-keyword Search Scheme on Cloud Data. In: International Conference on Cloud and Service Computing (CSC). IEEE, Shanghai, China, pp 104–110
13. Stefanov E, Papamanthou C, Shi E (2014) Practical Dynamic Searchable Encryption with Small Leakage. NDSS '14, San Diego, CA, USA
14. Wang C, Ren K, Yu S (2012) Urs KMR Achieving usable and privacy-assured similarity search over outsourced cloud data. In: Proceedings of IEEE INFOCOM. IEEE, Orlando, Florida, USA, pp 451–459
15. Li J, Wang Q, Wang C, Cao N, Ren K, Lou W (2010) Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of IEEE INFOCOM. IEEE, San Diego, CA, USA, pp 1–5
16. Chuah M, Hu W (2011) Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In: 31st International Conference on Distributed Computing Systems Workshops (ICDCSW). IEEE, Minneapolis, Minnesota, USA, pp 273–281
17. Liu C, Zhu L, Li L, Tan Y (2011) Fuzzy keyword search on encrypted cloud storage data with small index. In: IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). IEEE, Beijing, China, pp 269–273
18. Ibrahim A, Jin H, Yassin AA, Zou D (2012) Approximate Keyword-based Search over Encrypted Cloud Data. In: IEEE Ninth International Conference on e-Business Engineering (ICEBE). IEEE, Hangzhou, China, pp 238–245
19. Bringer J, Chabanne H (2012) Embedding edit distance to enable private keyword search. Human-centric Comput Inf Sci 2(1):1–12
20. Xu J, Croft WB (1996) Query expansion using local and global document analysis. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, USA, pp 4–11
21. Fu G, Jones CB, Abdelmoty AI (2005) Ontology-based spatial query expansion in information retrieval. In: On the move to meaningful internet systems 2005: CoopIS, DOA, and ODBASE. Springer, Berlin/Heidelberg, pp 1466–1482
22. Lesk ME (1969) Word-word associations in document retrieval systems. Am Doc 20(1):27–38
23. Minker J, Wilson GA, Zimmerman BH (1972) An evaluation of query expansion by the addition of clustered terms for a document retrieval system. Information Storage and Retrieval 8(6):329–348
24. Wei J, Bressan S, Ooi BC (2000) Mining term association rules for automatic global query expansion: methodology and preliminary results. In: Proceedings of the First International Conference on Web Information Systems Engineering. IEEE, Hong Kong, China, pp 366–373
25. Pal D, Mitra M, Datta K (2013) Query expansion using term distribution and term association., arXiv preprint arXiv:13030667
26. Lai L-F, Wu C-C, Lin P-Y, Huang L-T (2011) Developing a fuzzy search engine based on fuzzy ontology and semantic search. In: IEEE International Conference on Fuzzy Systems (FUZZ). IEEE, Taipei, Taiwan, pp 2684–2689
27. Fonseca BM, Golgher PB, De Moura ES, Pôssas B, Ziviani N (2003) Discovering search engine related queries using association rules. J Web Eng 2(4):215–227
28. Song M, Song I-Y, Hu X, Allen R (2005) Semantic query expansion combining association rules with ontologies and information retrieval techniques. In: Data Warehousing and Knowledge Discovery. Springer, Berlin/Heidelberg, pp 326–335
29. Song M, Song I-Y, Hu X, Allen RB (2007) Integration of association rules and ontologies for semantic query expansion. Data Knowl Eng 63(1):63–75
30. Singhal A (2001) Modern information retrieval: a brief overview. IEEE Data Eng Bull 24(4):35–43
31. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of data. ACM, Paris, France, pp 563–574
32. Boldyreva A, Chenette N, Lee Y, O'neill A (2009) Order-preserving symmetric encryption. In: Advances in Cryptology-EUROCRYPT 2009. Springer, Berlin/Heidelberg, pp 224–241
33. MOFFAT AA, Bell TC (1999) Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann, San Francisco, California, USA
34. Request For Comments Database. http://www.ietf.org/rfc.html. Accessed 27 Oct 2013

doi:10.1186/s13677-014-0008-2
Cite this article as: Xia *et al.*: Secure semantic expansion based search over encrypted cloud data supporting similarity ranking. *Journal of Cloud Computing: Advances, Systems and Applications* 2014 **3**:8.