Advanced Science and Technology Letters
Vol.31 (MulGraB 2013), pp.271-283
http://dx.doi.org/10.14257/astl.2013.31.57

Secure keyword-based Ranked Semantic Search over Encrypted Cloud Data

Xingming Sun, Yanling Zhu, Zhihua Xia, Jin Wang, and Lihong Chen

sunnudt@163.com, zyl.112358@163.com, xia_zhihua@163.com, wangjin@nuist.edu.cn and chen_81_lihong@163.com

Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, 210044, China

School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China

Abstract. With the advent of cloud computing, many organizations and individuals are interested in outsourcing their complex data management to the public cloud for economic savings and ease of access. As sensitive information may have to be encrypted before outsourcing, the data utilization service based on plaintext keyword search is not suitable for the encrypted cloud data. In this paper, we propose a solution for ranked semantic keyword search (RSS) over encrypted cloud data. With the design of semantic extension, the proposed scheme 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, the data owner generates a piece of file metadata for each file, and uploads the encrypted metadata set and file collection to the cloud server. 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. Eventually, the result files are returned in order according to the total relevance score. 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.

1 Introduction

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 outsourcing. Unfortunately, data encryption, if not done appropriately, may reduce the effectiveness of data utilization.

In recent years, searchable encryption (SE) techniques have been developed for secure outsourced data search. But traditional SE schemes only support exact keyword search [3-8]. To enhance the search flexibility and usability, some research has been done on fuzzy keyword search [9-13]. These solutions support tolerance of

ISSN: 2287-1233 ASTL

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 considered the terms semantically related to 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 fuzzy solution that support ranked semantic search over encrypted cloud data based on symmetric key cryptography. Semantic search reinforce the system usability by returning the exactly matched files and the files including the terms semantically related to the query keyword. Note that, the “keyword” in the paper is just a single word. In the proposed scheme, the data owner generates a piece of file metadata for each file, and uploads the encrypted metadata set and file collection to the cloud. With the file metadata, the cloud builds the inverted index and constructs SRL for the keywords. The co-occurrence of terms is used to evaluate the semantic relationship between terms in SRL. Upon receiving a query request, at first the cloud server 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 extended 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.

The paper is organized as follows. The system model and threat model, design goals, notations preliminaries and basic definitions are described in Section 2. Section 3 further gives the description of our semantic search scheme. The security analysis and performance evaluation are given in Section 4 and 5 respectively. In the end, section 6 gives the conclusion of the paper.

2 Problem formulation

2.1 System Model

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

Data owner uploads a collection of n text files $F = (F_1, F_2, F_3, , F)$ in encrypted form C, together with the encrypted metadata set generated from F, to the cloud server. Note that, a piece of metadata is constructed for each file.

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

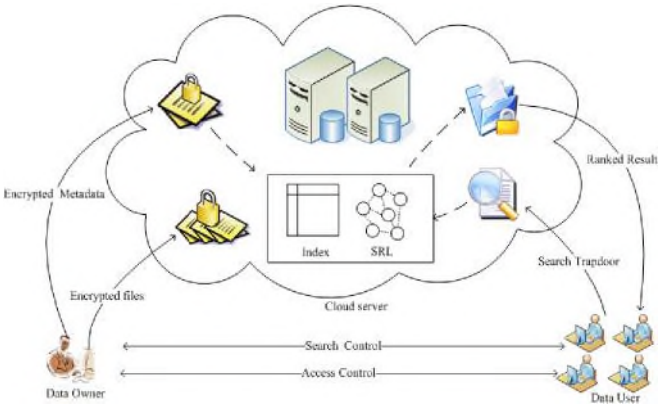


Fig. 1. Framework of the ranked semantic keyword search scheme

Cloud server first constructs the index and SRL using metadata set provided by data user, thus reduce the computing burden on owner e.g. index creating. Upon receiving the request , the cloud server is responsible to extend the query keyword upon SRL. Then search the index, and return 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.

2.2 Threat Model

In this paper, we use the same threat model described in previous searchable symmetric encryption (SSE) scheme [6, 7, 10, 11, 14, 15]. 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.

2.3 Notation

−the plaintext file collection, denoted as a set of n data files $= (_1, _2, \cdots,)$.

− the encrypted file collection, stored in the cloud server, denoted as $=$ $(_1, _2, \cdots,)$.

$()$ −the identifier of file that can help uniquely locate the actual file.
− the distinct keywords set extracted from .

−the number of distinct keywords extracted from and $= | |$.

−the encrypted metadata set, denoted as $\{_1, _2, \cdots, \}$, where is built for .

−the encrypted distinct keywords set extracted from , denoted as a set of m words $= (_1, _2, \cdots)$.

−the inverted index built from the metadata set by the server, including a set of posting lists $\{()\}$

$()$ − the set of identifiers of files that contain keyword .

−the trapdoor generated for a query keyword by a user.

−the semantically extended keywords set of , = (1, 2, 3⋯).

2.4 Preliminaries and basic definitions

(1) Semantic Relationship

Church et al. had proposed a statistical description of the semantic relationship between terms through the co-occurrence of terms [16]. For two terms and , the mutual information (,) is defined as

$$(,) \equiv 2 \qquad (1)$$

Here (,) is the probability of observing and together. () and () are the probabilities of observing and independently in the collection. The higher the semantic relationship between and is, the larger the mutual information (,) is.

Then normalize the mutual information into a value of relationship. The semantic relationship library will be constructed as a weighted graph structure showed in Fig. 2.

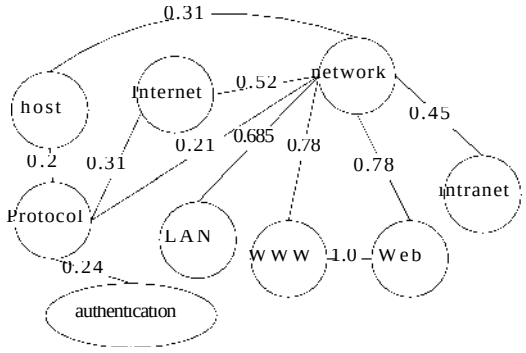


Fig. 2. An example of semantic relationship library

(2) File Metadata

A piece of file-metadata is constructed for each file. The file-metadata consists of the file ID, the keywords, and the relevance scores (refer to equation 2) of keywords in response to the file. All of the file metadata constitute metadata set, which is shown in Tab.1.

Table 1. An example of metadata set

()	11	11	12	12	13	1
()	21	21	22	22	23	2
. . .	. . .	. . .	. . .	. . .	. . .	...
()	1	1	2	2	3	3

(3) **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. Thus the relevance score of single keyword can be computed using equation 2,which is widely used in the literature[17]:

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

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

In our scheme, Fd's total relevance score will be computed for result ranking with equation 3.

$$TScore(w,F_d) = Score_w + \sum_{e_i \in S_w} Score_{e_i} \times R_i \quad (3)$$

Here Score, represents the relevance score of the input keyword; $Score_{e_i}$ represents the relevance score of extended keyword e_i, while R_i is the value of semantic relationship.

3 The ranked semantic keyword search scheme

We are now ready to proceed to the details of the effective RSS scheme. Let k, l, t, p be security parameters that will be used in $KeyGen(\cdot)$. The collision resistant hash function ir and a one-to-many order preserving encryption scheme ($OM\text{-}OPE$) are defined as:

$$ir: \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^p (p > \log m)$$

$$OM\text{-}OPE: \{0,1\}^l \times \{0,1\}^d \rightarrow \{0,1\}^r$$

Where $ir(\cdot)$ will be instantiated by existing hash function –SHA-1. m denotes the size of keywords set. OM-OPE is modified from the existing OPE. d and r respectively represent the bit length used to denote all the values in domain D and range R, the details are described later in section 3.3. The scheme can be constructed in two phases—Setup and Retrieval:

3.1 The setup phase

In this phase, the data owner mainly initializes the public and secret parameters of the system, and generates the encrypted metadata set. When receiving the data from owner, the server constructs the index and SRL based on the encrypted metadata. Details are as follows:

(1) The data owner initiates the scheme by calling $KeyGen(1^k, 1^l, 1^l, 1)$, generates random keys $x \leftarrow \{0,1\}^k$, $y \leftarrow \{0,1\}^l$, and outputs

$Key = \{x, y, 1^1, 1^1, 1^P\}$.

(2) Then the data owner builds the secure metadata for each file in file collection F by calling $BuildMD(K, F)$. Here E denotes the maximum number of keywords extracted from F_i, i.e. $E = max_{0,i,\ldots N} i$. The details are given in Algorithm 1. The 1 padding 0's indicates the valid entry.

Algorithm 1 Build metadata set M

procedure $BuildMD(K, F)$
for each file $F_i \in F$

Extract the distinct words in F_i, denoted as $W_i = tw_{i.}, w_{i2}, w_{i3} w_{iN}$~;

for 1 j | W_i |

1) calculate the score for keyword w_{ij} according to equation 2, denoted as s_{ij};

2) compute $ir_{x(wij)}$ and $0\ M\text{-}0\ PM_y(s_{ij})$;

3) store both with 1 padding 0's as $(0\ 11n_x(w_{ij})\ 110M - OP\ M_y(s_{ij}))$ in metadata M_i;

4) set remain $E - N_i$ entries to random values of the same size as the existing N_i entries of M_i;

end for

Insert M_i to M
end for

return M
end procedure

(3) When receiving the secure metadata, the server builds the inverted index by calling $BuildIndex(M)$, the details are given in Algorithm 2. The SRL is also built upon the metadata set using datamining algorithm to give the semantic relationship between keywords in the collection.

Algorithm 2 Build index I

procedure $BuildIndex(M)$
Initialization: extract the encrypted distinct words from M as $EW = \{ew_1, ew_2, ew_m\}$;
for each $ew_i \in EW$

Build a set of identifiers of files that contain ew_i, denoted as $F(ew_i)$;

for 1 j | $F(ew_i)$ |

1) extract the encrypted relevance score of ew_i corresponding to the j-th file $F_j(ew_i)$, denoted as es_{ij};

2) store the $\{id(F_j(ew_i))||es_{ij}\}$ as an element in the posting list of $I(ew_i)$;

end for

Insert $I(ew_i)$ into I
end for

return I
end procedure

3.2 The retrieval phase

In this phase, the user submits a secure trapdoor of his interested keyword to the cloud server. Then cloud server performs query keyword extension and returns the ranked search results. Details are as follows.

(1) The user generates a trapdoor = () for an interested keyword , by calling (). Upon receiving the trapdoor , the server first extends the query keyword to obtain the extensional query keywords set

′ = {(),()}, ∈ .

(2) Then the server locates the matching entries of the index via $\pi_x(w)$ and $\pi_x(e_i)$, which include the file identifiers and the associated order-preserved encrypted 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.

3.3 One-to-many order-preserving encryption

The original OPE is a deterministic encryption scheme, if not disposed properly, it will leak as much information as any deterministic encryption scheme does[18]. 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[15].

Therefore we need to modify the OPE to suit our requirement. Specially together with the plaintext m, the unique file ID is introduced as an additional random seed in the final ciphertext selection process. 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(•) is a random coin generator, HYGEINV(•) is the HGD(•) sampling function instance in MATLAB. In the process, a plaintext m in domain D = {1, ⋯, M} is mapped into ciphertext c selected in range R = {1, ⋯ 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

procedure - (, , , ())

while | |! = 1 do

{, } ← h(, ,,);

end while

```
 ← (, (,,1||, ()));

~□~ ;
return
end procedure


procedure h( ,, ,)
    ← ||; ← ||;
```

$d = min(D) - 1$; $r = min(R) - 1$;
$y <- r + [N/21$;

$coin <- GetCoins(K_R, (D, R, 0||y))$;

$x <- d + HYGEINV(coin, M, N, y - r)$;
if $m < x$ **then**

$D <- \{d + 1, \cdots, x\}$; $R <- \{r + 1, \cdots, y\}$;
else

$D <- \{x + 1, \cdots, d + M\}$; $R <- \{y + 1, \cdots, r + N\}$;
end if

return $\{D, R\}$;
end procedure

4 Security analysis

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

4.1 Security analysis for the ranked semantic keyword Search

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

(1) File confidentiality: the file confidentiality depends on the inherently security strength of the symmetric encryption scheme, so the file content is obviously protected very well.

(2) Keyword privacy:

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.

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 depend 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 Tab.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[15], 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.

Table 2. An example of semantic relationship between terms

Keyword	Keyword	similarity
host	network	0.31
LAN	Ethernet	0.31
protocol	internet	0.31

4.2 Security analysis for the ranked semantic keyword Search

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 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: $H(a) = -\log(\max_a Pr[a = a])$, where a is a discrete random variable, a denotes a state of a with the max probability. In general, the higher $H(a)$ is, the more difficult the a can be predicted. If $H(a) \ E\ w(\log k)$, the min-entropy of variable a will be high, where k is the bit length needed to denote all the states of a[8]. We could choose $H(a)$ as $(\log k)^{`}$ where $c > 1$ [15]. Then the least size $|R|$ should satisfy the equation 4:

$$(\log(\log |R|))^{c} \leq -\log \frac{\max/(|R|.6_{12}51011M+12}{} \sim \quad (4)$$

Here *max* denotes the maximum number of score duplicates within the metadata set. S denotes the totoal number of scores to be mapped within metadata set. With $D = \{1, \bullet\bullet\bullet, M\}$, $M = |D|$, the total recursive calls of *BinarySearch(.)* function is at most $5 \log M + 12$. 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 $|R|$.

$$\frac{\cdot 2^{5+12}}{2\cdot} = 2^{-12} \cdot \leq_{2} -() \qquad (5)$$

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.

5 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)[19], 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.

(1) Metadata construction

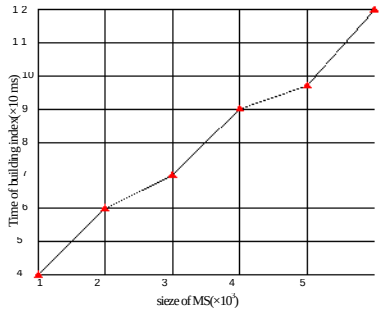
The main overheads for data owner are time cost and storage cost of metadata construction. To build a metadata for each document F_i in the dataset F, we should extract the distinct words and compute the associated relevance score, then encrypt the keywords and scores. The time cost of each entry directly 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 Tab. 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.

Table 3. File metadata construction overhead

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

(2) Index and SRL construction

In our construction, the index and SRL are built on cloud server based on the received MS. We should scan the whole metadata set to extract the keywords and build the inverted index with corresponding scores. Fig. 3 shows that the whole index is nearly linear with the size of MS, namely the number of documents in the collection. The SRL is also built by scanning the MS, with the certain support threshold, the number of entries is the main factor to the efficiency. Fig.4 shows the time cost of building SRL against the increasing size of MS or dataset. In addition, taking into account the abundant computing resources on server, the performance of building index and SRL is practically efficient.



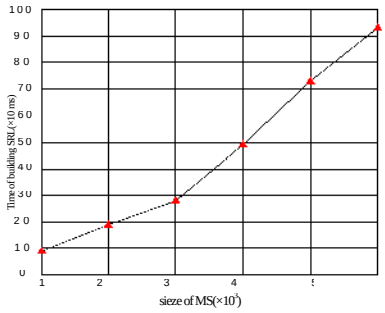
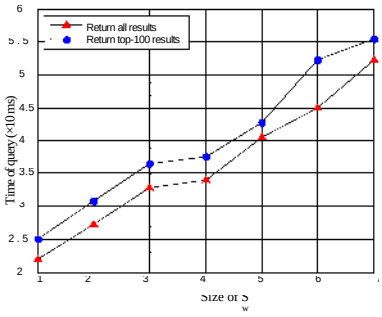


Fig. 3. The time cost for building index **Fig.4.** The time cost for building SRL

(3) **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 extended keywords set is a factor to the query efficiency. Fig. 5 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, Fig.6 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.



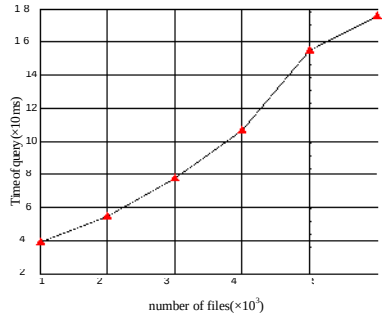


Fig.5. Time cost of query. **Fig. 6.** The overall query performance

(4) **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 in puts 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.

6 Conclusion

In this paper, as an initial attempt, we propose a secure search scheme over encrypted cloud data which support ranked semantic search. 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 dataset, 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.

Acknowledgements. This work is supported by the NSFC (61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61373133), GYHY201206033, 201301030, 2013DFG12860, BC2013012 and PAPD fund.

References

1. S. Kamara and K. Lauter, "Cryptographic cloud storage," in Financial Cryptography and Data Security, ed: Springer, 2010, pp. 136-149.
2. K. Ren, et al., "Security challenges for the public cloud," Internet Computing, IEEE, vol. 16, pp. 69-73, 2012.
3. D. X. Song, et al., "Practical techniques for searches on encrypted data," in Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, 2000, pp. 44-55.
4. E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, Report 2003/2162003.
5. D. Boneh, et al., "Public key encryption with keyword search," in Advances in Cryptology-Eurocrypt 2004, 2004, pp. 506-522.
6. Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in Applied Cryptography and Network Security, 2005, pp. 442-455.
7. R. Curtmola, et al., "Searchable symmetric encryption: improved definitions and efficient constructions," in Proceedings of the 13th ACM conference on Computer and communications security, 2006, pp. 79-88.
8. M. Bellare, et al., "Deterministic and efficiently searchable encryption," in Advances in Cryptology-CRYPTO 2007, ed: Springer, 2007, pp. 535-552.
9. C. Wang, et al., "Achieving usable and privacy-assured similarity search over outsourced cloud data," in INFOCOM, 2012 Proceedings IEEE, 2012, pp. 451-459.
10. J. Li, et al., "Fuzzy keyword search over encrypted data in cloud computing," in INFOCOM, 2010 Proceedings IEEE, 2010, pp. 1-5.
11. M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword

search over encrypted data," in Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on, 2011, pp. 273-281.
12. C. Liu, et al., "Fuzzy keyword search on encrypted cloud storage data with small index," in Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, 2011, pp. 269-273.
13. A. Ibrahim, et al., "Approximate Keyword-based Search over Encrypted Cloud Data," in e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on, 2012, pp. 238-245.
14. N. Cao, et al., "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in INFOCOM, 2011 Proceedings IEEE, 2011, pp. 829-837.
15. C. Wang, et al., "Secure ranked keyword search over encrypted cloud data," in Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, 2010, pp. 253-262.
16. K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," Computational linguistics, vol. 16, pp. 22-29, 1990.
17. A. A. MOFFAT and T. C. Bell, Managing gigabytes: compressing and indexing documents and images: Morgan Kaufmann, 1999.
18. A. Boldyreva, et al., "Order-preserving symmetric encryption," in Advances in Cryptology-EUROCRYPT 2009, ed: Springer, 2009, pp. 224-241.
19. Request For Comments Database [Online]. Available: http://www.ietf.org/rfc.html

