Advanced Science and Technology Letters
Vol.48 (CIA 2014), pp.141-151
http://dx.doi.org/10.14257/astl.2014.48.23

Secure and Efficient Multi-keyword Ranked Search over Encrypted Cloud Data

Xingming Sun, Lu Zhou, Zhangjie Fu, Zhihua Xia and Jiangang Shu

College of Computer and Software & Jiangsu Engineering Center of Network Monitoring,
Nanjing University of Information Science and Technology, Nanjing 210044, CHINA

Abstract: With the development of cloud computing, the sensitive information of outsourced data is at risk of unauthorized accesses. To protect data privacy, the sensitive data should be encrypted by the data owner before outsourcing, which makes the traditional and efficient plaintext keyword search technique useless. Hence, it is an especially important thing to explore secure encrypted cloud data search service. In this paper, we propose a practically efficient and flexible searchable encrypted scheme which supports multi-keyword ranked search. To support multi-keyword search and result relevance ranking, we adopt Vector Space Model (VSM) to build the searchable index to achieve accurate search result. To improve search efficiency, we design a tree-based index structure. We propose a secure search scheme to meet the privacy requirements in the threat model. Finally, experiments on real-world dataset show that our scheme is efficient.

Keywords: Multi-keyword search, ranked search, encrypted cloud data, security

1 Introduction

Cloud Computing is a new but increasingly mature model of enterprise IT infrastructure that provides on-demand high quality applications and services from a shared pool of configuration computing resources [1].The cloud customers, individuals or enterprises, can outsource their local complex data system into the cloud to avoid the costs of building and maintaining a private storage infrastructure. However, some problems may be caused in this circumstance since the Cloud Service Provider (CSP) possesses full control of the outsourced data. Unauthorized operation on the outsourced data may exist on account of curiosity or profit. To protect the privacy of sensitive information, sensitive data (e.g., emails, photo albums, personal health records, financial records, etc.) should be encrypted by the data owner before outsourcing [2], which makes the traditional and efficient plaintext keyword search technique useless. The simple and awkward method of downloading all the data and decrypting locally is obviously impractical. So, two aspects should be concentrated on to explore privacy-preserving effective search service. Firstly, ranked search, which can enable data users to find the most relevant information quickly, is a very important issue. The number of documents outsourced to the cloud is so large that the cloud should have the ability to perform search result ranking to meet the demand for

ISSN: 2287-1233 ASTL

effective data retrieval [3]. Secondly, multi-keyword search is also very important to improve search result accuracy as single keyword search often return coarse search results.

In this paper, we propose a practically efficient and flexible searchable encrypted scheme. To address multi-keyword search and result ranking, we use Vector Space Model (VSM) [6] to build document index. To improve search efficiency, we use a tree-based index structure which is a balanced binary tree (see the details in section 2.5). We construct the searchable index tree based on the document index vectors. Our encryption scheme can meet the privacy requirements in the threat model. Our contributions are summarized as follows:

(1) We study the problem of multi-keyword ranked search over encrypted cloud data while supporting strict privacy requirements.

(2) With the index tree designed in this paper, the search time complexity close to (is the number of documents including the search keywords and is the whole number of documents in the dataset).

(3) We make security analysis for our scheme which proves privacy guarantees. And experiments on the real-world dataset show that proposed scheme is indeed efficient.

2 Problem formulation

2.1 System model

Like in [4], we consider there are three different entities in the system model: the data owner, the user, and the cloud server respectively. The data owner encrypts document collection in the form of before outsourcing it to the cloud in order to protect the sensitive data from unauthorized entities. And for the purpose of searching interested data, the data owner will also generate an encrypted searchable index based on a set of distinct keywords extracted from . In the search stage, the system will generate an encrypted search trapdoor based on the keywords entered by the user (has been authorized by data owner). Given the trapdoor, the cloud server will search the index and then return the ranked search results to the user. As the search results are well ranked by the cloud server, the user can send a parameter together with search query to get top- most relevant documents. As the issue of key distribution is out of the scope of this paper, we assume that data users have been authorized by data owner.

2.2 Threat model

In our system model, we consider that the cloud server is “honest-but-curious” *ey* , *vaue*

adopted by most previous searchable encryption schemes [3-5]. That is to say, the cloud server honestly implements the protocol and correctly returns the search results, but it is also curious to infer sensitive information during execute protocol. In the known ciphertext model, only the encrypted dataset , the encrypted search query and the searchable index are available to the cloud server.

2.3. Notations

The main notations used in this paper are showed as follows:

- – the plaintext document collection, expressed as a set of documents .
- – the encrypted document collection for stored in the cloud server, expressed as .
- – the dictionary, including keywords extracted from , expressed as .
- – a subset of , representing the keywords in a search request, expressed as .
- – the searchable index tree generated from the whole document set . Each leaf node in the index tree is associated with a document in .
- – the index vector of document for all the keywords in .

- – the query vector for the keyword set .
- – the encrypted index vector for .
- – the encrypted query vector for .

–pseudorandom function (PRF), defined as:

–symmetric encryption/decryption function.

- – the encrypted form of .

2.4. Preliminaries

Keywords hash table: A static hash table for all the keywords in , denoted as . There are entries in and each entity is a tuple , in which the is from a domain of exponential size, i.e., from representing a keyword in , and is a boolean value which has been encrypted. For a

- ,
  .
- ,

D

, the corresponding is denoted as .

Similarity function: We employ the similarity evaluation function for cosine measure from [7]. Each document in the dataset is corresponding to an -dimension index vector , and each dimension of , denoted as , is related to a keyword in . If document contains keyword , stores the normalized TF weight of within document , otherwise . For a search request , an -dimension query vector is also generated. It is similar with document index vector that each dimension of is related to a keyword in . And if contains keyword , stores the normalized IDF weight of , otherwise . The notations used in our similarity evaluation function are showed as follows:

$S C$

$($ Q

$)^2$

q ,

W n i □1 □ n

)

2

1

(1)

where represents the TF weight of within document , represents the IDF weight of keyword . We use functions and to compute the value of TF and IDF weight respectively, where represents the TF of keyword within the document , represents the number of documents containing the keyword , represents the total number of documents in the document collection, represents the total number of keywords in the keyword dictionary. And hence, the vector and are both unit vectors.

2.5. Searchable index tree

Our searchable index is a balanced binary tree. In order to make it easy to understand, the document index vector in each leaf node only stores TF information rather than normalized TF weight. Given the document collection , we can build the index tree . The data structure is built using the procedure, denoted as , showed as follows:(1)For each document in , we generate a leaf node where stores document identifier and index vector . (2)Then we build the tree following a postorder traversal with all leaf nodes generated in (1). Each internal node of the index tree stores an -bit vector . If , then there is at least one path from to a leaf node of which corresponding document contains keyword . (3)In this step, we introduce how to generate vector in each internal node . Let and be the left child and right child of internal node respectively, then if or , otherwise . Note that when the node ( ) is a leaf node and stores identifier , ( ).

Tree-based search algorithm: The sequential search process for keywords in a search request conducts as follows: the procedure starts from the root node and when arrives at an internal node , if at least a keyword (is the order number of in) in leads to , it continues to search both subtrees of , otherwise stops searching in the subtree (denotes the tree whose root is)because none of leaf node in contains keyword in search query. When arrives at a leaf node, the process computes the cosine value between the index vector stored in the leaf node and the query vector as the similarity score. We denote the number of documents that contain the keyword in the search query as . In the sequential search, the procedure will traverse as many paths as . So, the search complexity is as the height of a balanced binary tree with leaf nodes is .

D

3 Secure Index Scheme

In order to achieve accurate multi-keyword ranked search, we propose to adopt VSM and cosine measure to evaluate similarity scores. By using the cryptographic methods similar to the techniques adopted in [4, 8], the document index vector and query are both well protected. The scheme is described as follows:

Setup: In this phase, we initialize our system. The data owner generates the secret key . The includes: 1) a -bit vector which is randomly generated; 2) two also randomly generated invertible matrices ; 3) two randomly picked key and . Hence, is in the form of a 5-tuple as .

GenIndex : The data owner calls procedure that defined in section 2.5. Then, every document index vector is split into two random vectors denoted as . The splitting procedure is expressed as follow: take as the splitting indicator, if the -th bit of is 0, and are set as the same as ; if the -th bit of is 1, and are set randomly so long as their sum is equal to . So, the encrypted index vector is denoted as . Store at the leaf node that stores correspondent and delete . For each internal node in the index tree, a static hash table is generated. There are tuples in , and for every , set . Store in internal node and delete . Finally, the encrypted searchable index tree is generated.

GenQuery : With the interested keywords in , the -dimension query vector is generated. Each dimension of is a normalized IDF weight of corresponding keyword. Specifically, if -th keyword of is in , , otherwise . Next, is also split into two random vectors as using the similar splitting procedure used for document index vector. The difference is that if the -th bit of S is 0, and are set randomly so long as their sum is equal to ; if the -th bit of is 1, and are set as the same as . Then, the encrypted query vector is in the form of . Next, is produced by encrypting each item in . Finally, the is sent to the cloud server.

Search : The cloud server follow the search algorithm expressed in section 2.5. Let be an internal node in , and let for each item in . If exist at least one satisfies , the procedure continue to search all children of . When arrive at a leaf node, the procedure obtains the encrypted document vector and compute the similarity of and using Eq(2). Finally, the cloud server will rank the searched documents based on their similarity scores.

[illegible] (2)

4 Performance Analysis

In this section, we estimate the overall performance of our proposed scheme by implementing the secure search system using C# language on a Windows7 server with Intel(R) Core(TM)2 Quad CPU 2.83GHz. The document set is built from the real-world data set: Request for comments database (RFC) [9], which includes about 6500 publications.

4.1. Index tree construction

It is obvious that the time cost of the index tree construction is mainly affected by the number of documents in the dataset and keywords in the dictionary. For each internal node in the searchable index tree, the major computation is the encryption of the hash table, the time cost of which is proportional to the number of keywords in the dictionary. And for each leaf node in the index tree, the main computation is the encryption of the document index vector, which mainly depends on the time cost for two multiplications of a matrix and an -dimension vector where is in our scheme. And the whole number of nodes in the index tree is related to the number of documents in the dataset. Fig. 1(a) shows that, given the same dictionary with 4000 keywords, the time cost for building the index tree is nearly linear with the number of documents in the dataset. Fig. 1(b) indicates that the time cost for constructing index tree is proportional to the number of keywords in the dictionary with the same size of dataset. Although the time cost for constructing index tree is not an ignorable overhead for the data owner, it is a one-time operation before data outsourcing.

4.2. Query generation

The time cost for generating the search query is greatly affected by the size of keywords dictionary. Two multiplications of a matrix and a split query vector are conducted in every query generation phase. The dimensionality of matrix (-dimension) and query vector (-dimension) depends on the size of keywords dictionary. Fig. 2 shows the relationship between the time of generating search query and the number of keywords in dictionary.

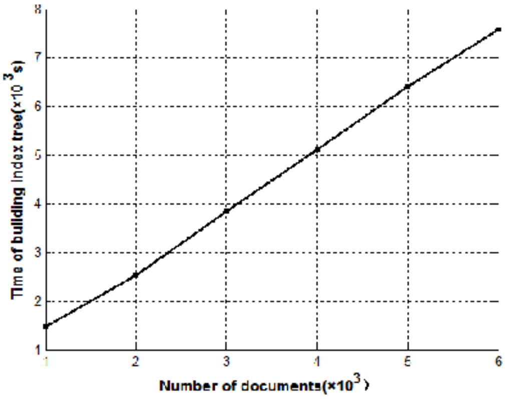
4.3. Search efficient

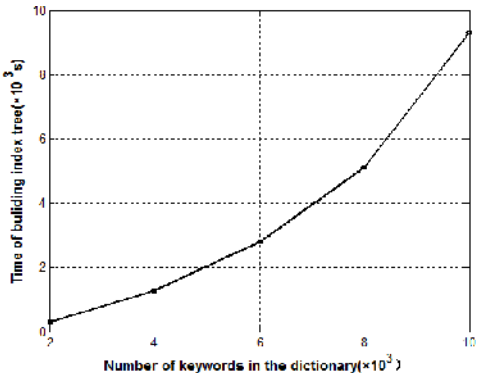
The search process, which is implemented by the cloud server, is composed by computing the similarity scores of relevant documents and result ranking based on these scores. Fig. 3(a) shows the search time for our scheme with different size of dataset. Let represent the number of documents including the search keywords. From Fig. 3(a), we can know that the search time is mainly depends on the number of documents in the dataset when is fixed. Fig. 3(b) shows the relationship between search time and with same dataset, in which search time is almost linear to .

5 Conclusion and Future Work

In this paper, we propose secure search scheme supporting multi-keyword ranked search over encrypted cloud data. We make contributions mainly in two aspects: similarity ranked search for more accurate search result and tree-based searchable index for more efficient searching. In term of accuracy, we adopt the vector space model combined with cosine measure to evaluate the similarity between search request and document and acquire accurate search result instead of undifferentiated result. For the efficiency aspect, we propose a tree-based index structure. We propose a secure scheme to meet privacy requirements in the threat model. Finally, we analyze the performance of our scheme in detail by the experiment on real-world dataset. But, there still exist some problems, such as dynamic update for searchable index. We will do more research in the future.

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





(a) (b)

Fig.1. Time cost for building index tree. (a)For the different number of documents in the dataset with the same dictionary, n=4000. (b) For the different number of keywords in the dictionary with the same dataset, m=1000.

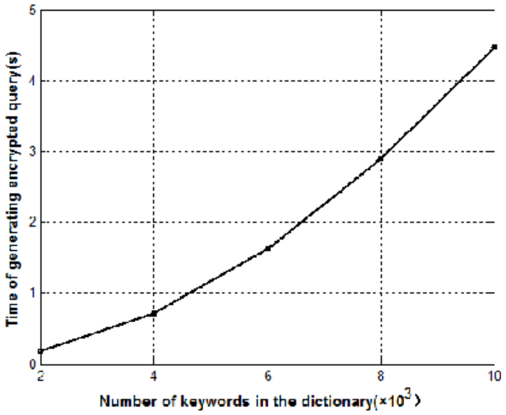
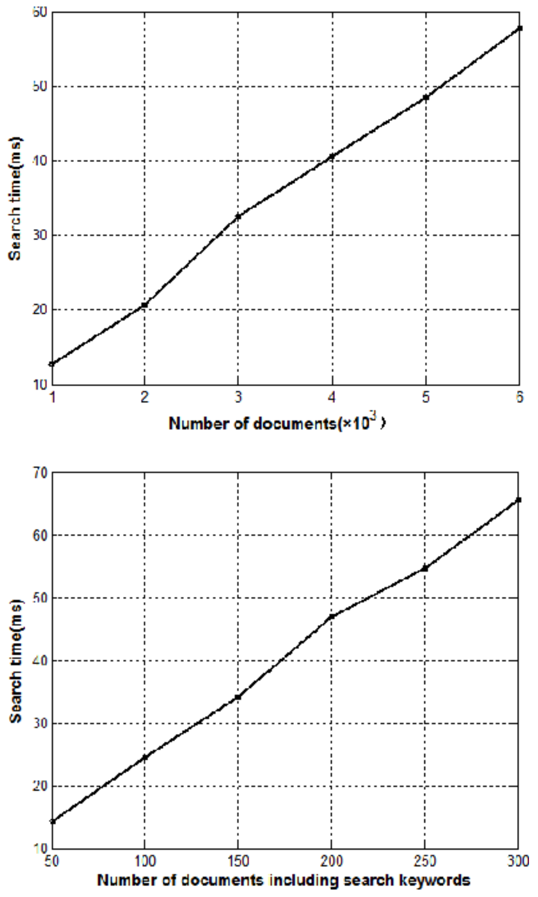


Fig.2. Time cost of generating encrypted query.



(a) (b)

Fig.3. Search efficiency. (a) For the different size of dataset with the same number of documents including search keywords, r=90. (b) For the different number of documents including the search keywords with the same size of dataset, m=2000.

Reference

1. L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner. A break in the clouds: towards a cloud definition [J]. ACM SIGCOMM Computer Communication Review, 2009, 39(1): 50–55.
2. S. Kamara and K. Lauter. Cryptographic cloud storage[C]//Financial Cryptography and Data Security, Springer Berlin Heidelberg Publishing, 2010: 136-149.
3. C. Wang, N. Cao, K. Ren and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data [J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(8): 1467–1479.
4. N. Cao, C. Wang, M. Li, K. Ren and W. Lou. Privacy-preserving multi-keyword ranked

search over encrypted cloud data [J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(1): 222-233.

5. W. Sun, B. Wang, N. Cao, M. Li, W. Lou, YT. Hou and H.L. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking[C]//Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, 2013: 71-82.
6. I. H. Witten, A. Moffat and T. C. Bell. Managing gigabytes: Compressing and indexing documents and images [M]. San Francisco: Morgan Kaufmann Publishing, 1999.
7. D.A.Grossman and O. Frieder. Information retrieval: Algorithms and heuristics[M]. Springer Publishing, 2004.
8. W. K. Wong, D. W. Cheung, B. Kao and N. Mamoulis. Secure knn computation on encrypted databases[C]//Proceedings of SIGMOD, 2009: 139–152.
9. RFC (Request For Comments Database) [DB/OL]. http://www.ietf.org/rfc.html.