International Journal of Security and Its Applications
Vol.8, No.6 (2014), pp.1-16
http://dx.doi.org/10.14257/ijsia.2014.8.6.01

Privacy-preserving Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Dynamic Update

Xingming Sun*, Lu Zhou, Zhangjie Fu and Jin Wang

College of Computer and Software & Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing210044, China

sunnudt@163.com, zl_0713@163.com, wwwfzj@126.com, wangjin@nuist.edu.cn

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. Considering the huge number of outsourced data, there are three problems we are focused on to enable efficient search service: multi-keyword search, result relevance ranking and dynamic update. In this paper, we propose a practically efficient and flexible searchable encrypted scheme which supports both multi-keyword ranked search and dynamic update. 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 which supports insertion and deletion update well without privacy leakage. We propose a secure search scheme to meet the privacy requirements in the threat model. Finally, experiments on real-world dataset are implemented to demonstrate the overall performance of the proposed scheme, which show our scheme is efficient.

Keywords: *Multi-keyword search, ranked search, dynamic update, encrypted cloud data*

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, 24, 25].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, as the cloud server possesses powerful functionality and flexibility. 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, three 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

ISSN: 1738-9976 IJSIA

outsourced to the cloud is so large that the cloud should have the ability to perform search result ranking to meet the demand for 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. The last but not least, dynamic update is a useful functionality for a good data management system which should provide as more as possible convenience for the data owner. It is common that sometimes the data owner wants to add a document to the dataset or delete a document from the dataset. So, a data management system that supports insertion and deletion update is a more integrated one.

In recent years, many researchers have engaged in the field of searchable encryption over encrypted cloud data and put forward a series of outstanding achievements. Nevertheless, there are still many challenging and important problems need to be solved. There does not exist scheme that supports both multi-keyword ranked search and dynamic update. How to design a scheme supporting both multi-keyword ranked search and dynamic update is still a challenging open problem.

In this paper, we propose a practically efficient and flexible searchable encrypted scheme supporting dynamic update. To address multi-keyword search and result ranking, we use Vector Space Model (VSM) [13] 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 3.5). We construct the searchable index tree based on the document index vectors. And our tree-based index tree also can support dynamic update, including insertion update and deletion update, without any privacy leakage. Our encryption scheme can meet the privacy requirements in the threat model. Our contributions are summarized as follows:

(1) For the first time, we study the problem of multi-keyword ranked search supporting dynamic update over encrypted cloud data while supporting strict privacy requirements.

(2) With the index tree designed in this paper, our scheme can support dynamic update. And the time complexity is $O(n \log m)$ in the worst case for updating a document. (n is the size of the keyword dictionary and m is the whole number of documents in the dataset).

(3) Our scheme can meet the privacy requirements in the threat model. 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.

The rest of this paper is organized as follows: Related work is summarized in Section 2. Then, we introduce the system model, threat model, design goals, notations and preliminaries in Section 3. The detailed description of our secure index scheme is given in Section 4, followed by Section 5, which introduces the process of dynamic update in detail. Section 6 and Section 7 give the security analysis and performance analysis of our proposed scheme respectively. Finally, Section 8 gives the conclusion and future work.

2. Related Work

2.1. Single Keyword Searchable Encryption

Searchable encryption schemes [3-9, 21] usually build an encrypted searchable index based on the keywords within document set, by which its content is hidden to the cloud server. Given appropriate search trapdoors generated by authorized user(s) (who has the secret key given by the data owner), the server can search the index and return corresponding search result. Song, *et al.*, [4] propose the first searchable encryption scheme in the symmetric setting. After this work, some schemes [5-6] are proposed to improve the security definition and search efficiency [21]. Solves the fuzzy keyword search which utilizes edit distance to extend keyword set. Schemes in [3, 9] solve the result ranking search utilizing order-preserving techniques. With frequency related information, they can rank search result and

return more accurate result. Boneh, *et al.*, [7] propose the first searchable encryption scheme based on public key cryptography. However, public key methods are always computationally expensive. Kamara, *et al.*, [12] propose a dynamic searchable encrypted scheme which supports both keyword search and document update. However, all the schemes expressed above only provide single keyword search.

2.2. Multi-keyword Searchable Encryption

As an attempt to improve system usability, a lot of works have been done in the public key setting to enrich search functionalities, including conjunctive keyword search, range queries and subset search [14, 15]. But these schemes usually result in large computational burden caused by some significant operation (for example, bilinear map [14]). [16-18] are focused on predicate encryption which supports both conjunctive and disjunctive search. Although these schemes can provide more general search, they does not support result ranking. Cao, *et al.*, [10] propose a privacy-preserving multi-keyword ranked search scheme which adopting "coordinates matching" to realize ranked search. This scheme ranks search result by the number of matched keywords, without considering more accurate ranked result. Since the scheme [10] uses the inverted index as the index structure, it leads to traverse all indexes of document set executed by the cloud server for each search query. Sun, *et al.*, [11] also propose a secure multi-keyword ranked search scheme based on vector space model (VSM). The VSM can measure the similarity between document index vector and query vector and hence support more accurate ranked search result. Schemes in [11] use an MDB-tree [22] as its index structure and propose a search algorithm based on the tree structure, which improve the search complexity in that the cloud server only need to search the part of the tree.

3. Problem Formulation

3.1. System Model

Like in [10], we consider there are three different entities in the system model as showed in Figure 1: the data owner, the user, and the cloud server respectively. The data owner encrypts document collection DC in the form of C 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 I based on a set of distinct keywords W extracted from DC. 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 I 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 k together with search query to get top-k most relevant documents. The update stage includes three phases. At first, with the update related information inputted by the data owner, the cloud server deals the searchable index I and returns a portion of I, denoted as $T(u)$, which is adequate to execute the update. And then the data owner performs the update and returns the updated portion $T'(u)$ to the server. At last, the cloud server just copies $T'(u)$ to the index I, and new encrypted dataset is also generated after adding the document to it or deleting the document from it. 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.

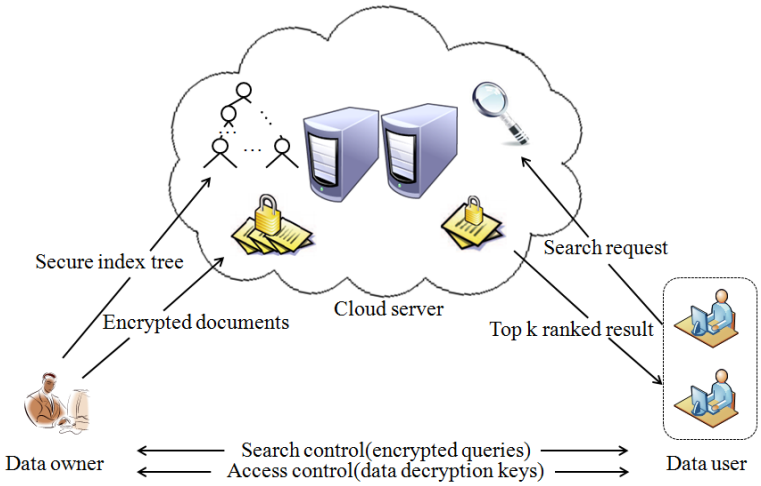


Figure 1. Framework of the Search over Encrypted Cloud Data

3.2. Threat Model

In our system model, we consider that the cloud server is "honest-but-curious" adopted by most previous searchable encryption schemes [3, 10, 11]. That is to say, the cloud server honestly implements the protocol and correctly returns the search results, but it is also curious to infer and analyze the outsourced data set, searchable index and messages that received during execute protocol. In the known ciphertext model, only the encrypted data set C , the encrypted search query and the searchable index I are available to the cloud server.

3.3. Design Goals

Dynamic Update: The tree-based searchable index designed in our scheme can support dynamic update well, only accessing a portion of the index tree for updating a document. Our designed scheme can provide insertion and deletion update.

Multi-keyword Ranked Search: To design a search scheme over encrypted cloud data ,which is not only capable of effective multi-keyword search, but also, with the use of vector space model, supports search result similarity ranking.

Privacy Preserving: The main privacy goal is to protect user's sensitive data by preventing cloud server from learning additional information by analyzing dataset, searchable index and search queries.

3.4. Notations

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

• DC – the plaintext document collection, expressed as a set of m documents $DC = \{d \mid d_1, d_2, ..., d_m\}$.

• C – the encrypted document collection for DC stored in the cloud server, expressed as $C = \{c \mid c_1, c_2, ... c_m\}$.

• W – the dictionary, including n keywords extracted from DC , expressed as $W = \{w \mid w_1, w_2, ... w_n\}$.

• $\tilde{W}$ – a subset of W, representing the keywords in a search request, expressed as $\tilde{W} = \{w_{i1}, w_{i2}, ..., w_{it}\}$.

• I – the searchable index tree generated from the whole document set DC. Each leaf node in the index tree is associated with a document in DC.

• D_d – the index vector of document d for all the keywords in W.

• Q – the query vector for the keyword set $\tilde{W}$.

• $\tilde{D}_d$ – the encrypted index vector for D_d.

• $\tilde{Q}$ – the encrypted query vector for Q.

• $f(key, \cdot)$, $g(key, \cdot)$ –pseudorandom function (PRF), defined as: $\{0,1\}^* \times key \rightarrow \{0,1\}^k$.

• $Enc(key, \cdot)$, $Dec(key, \cdot)$ –symmetric encryption/decryption function.

• $T_{\tilde{W}}$ – the encrypted form of $\tilde{W}$.

3.5. Preliminaries

Keywords Hash Table: A static hash table for all the keywords in W, denoted as λ. There are n entries in λ and each entity is a tuple $(key, value)$, in which the key is from a domain of exponential size, i.e., from $\{0,1\}^k$ representing a keyword in W, and $value$ is a boolean value which has been encrypted. For a key $x \in \{0,1\}^k$, the corresponding $value$ is denoted as $\lambda[x]$.

Similarity Function: In the vector space model, cosine measure is the most popular method to measure the similarity of two vectors. The cosine measure supports accurate similarity ranking because it follows the "TF×IDF rule", where TF (Term Frequency) denotes the number of occurrences of a term within a document, and IDF (Inverse Document Frequency) is obtained by dividing the total number of documents in the dataset by the number of documents which contain the term. We employ the similarity evaluation function for cosine measure from [19]. Each document d in the dataset is corresponding to an n-dimension index vector D_d, and each dimension of D_d, denoted as $D_d[i]$, is related to a keyword w_i in W. If document d contains keyword w_i, $D_d[i]$ stores the normalized TF weight of w_i within document d, otherwise $D_d[i] = 0$. For a search request $\tilde{W}$, an n-dimension query vector Q is also generated. It is similar with document index vector that each dimension of Q is related to a keyword in W. And if $\tilde{W}$ contains keyword w_i, $Q[i]$ stores the normalized IDF weight of w_i, otherwise $Q[i] = 0$. The notations used in our similarity evaluation function are showed as follows:

$$SC(Q, D_d) = \frac{\sum_{i=1}^{n} w_{q,i} \cdot w_{d,i}}{\sqrt{\sum_{i=1}^{n} (w_{q,i})^2} \cdot \sqrt{\sum_{i=1}^{n} (w_{d,i})^2}} \qquad (1)$$

where $w_{d,i}$ represents the TF weight of w_i within document d, $w_{q,i}$ represents the IDF weight of keyword w_i. We use functions $w_{d,i} = 1 + \ln(f_{d,i})$ and $w_{q,i} = \ln(1 + m / f_i)$ to compute the value of TF and IDF weight respectively, where $f_{d,i}$ represents the TF of keyword w_i within the document d, f_i represents the number of documents containing the

keyword w_i, m represents the total number of documents in the document collection, n represents the total number of keywords in the keyword dictionary. And hence, the vector Q and D_d are both unit vectors.

Searchable Index Tree: Our searchable index is a balanced binary tree, a dynamic data structure, showed in Figure 2. In Figure 2, 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 $DC = \{d \mid d_1, d_2, ..., d_m\}$, we can build the index tree I. The data structure is built using the procedure, denoted as $buildIndex\ (DC)$, showed as follows:(1)For each document d_j in DC, we generate a leaf node where stores document identifier j and index vector D_{dj}. (2)Then we build the tree following a post order traversal with all leaf nodes generated in (1). Take Figure 2 as an example to explain the process of post order traversal: With m=8 documents, 8 leaf nodes generated ($d_1 \sim d_8$). Then internal node r_{21} is generated based on leaf nodes d_1 and d_2, and internal nodes r_{22}, r_{23} and r_{24} are also generated similarly. Next, the internal node r_{11} is generated based on nodes r_{21} and r_{22}, and the internal node r_{12} is generated based on nodes r_{23} and r_{24}. Finally, the root node r is generated based on nodes r_{11} and r_{12}. Each internal node u of the index tree stores an n-bit vector D_u. If $D_u[i] = 1$, then there is at least one path from u to a leaf node of which corresponding document contains keyword w_i. (3)In this step, we introduce how to generate vector D_u in each internal node u. Let v and w be the left child and right child of internal node u respectively, then $D_u[i] = 1$ if $D_v[i] \neq 0$ or $D_w[i] \neq 0$, otherwise $D_u[i] = 0$. Note that when the node v (w) is a leaf node and stores identifier j, $D_v = D_{dj}$ ($D_w = D_{dj}$).

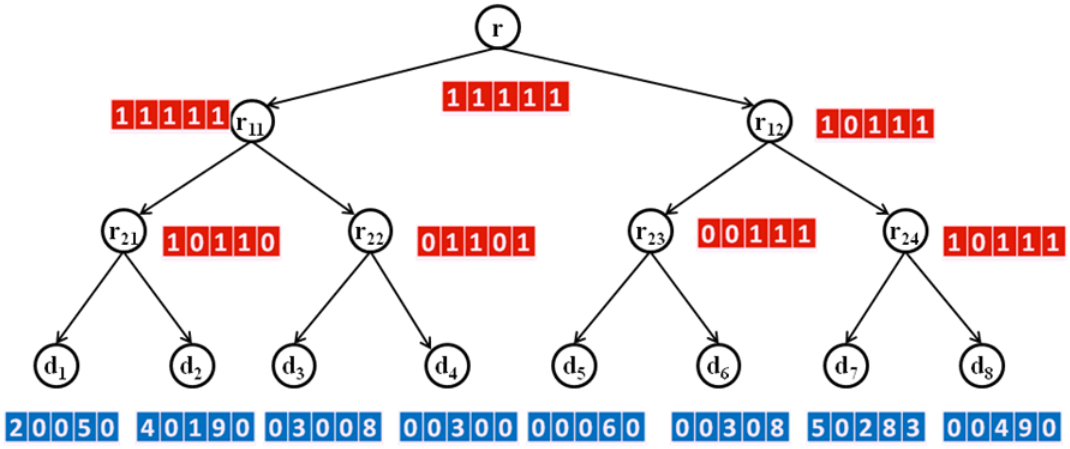


Figure 2. The Index Tree for a Document Collection of m=8 Documents and with n=5 Keywords

Tree-based Search Algorithm: The sequential search process for keywords in a search request $\tilde{W}$ conducts as follows: the procedure starts from the root node and when arrives at an internal node u, if at least a keyword w_k (k is the order number of w in W) in $\tilde{W}$ leads to $D_u[k] = 1$, it continues to search both subtrees of u, otherwise stops searching in the subtree t_u (t_u denotes the tree whose root is u)because none of leaf node in t_u 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 r. In the sequential search, the procedure will traverse as many paths as r. So, the search complexity is $O(r \log m)$ as the height of a balanced binary tree with m leaf nodes is $\log m + 1$.

4. 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 [10, 20], 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 SK. The SK includes: 1) a n-bit vector S which is randomly generated; 2) two $n \times n$ also randomly generated invertible matrices $\{M_1, M_2\}$; 3) two randomly picked key sk_1 and sk_2. Hence, SK is in the form of a 5-tuple as $\{S, M_1, M_2, sk_1, sk_2\}$.

GenIndex (DC, SK): The data owner calls procedure *buildIndex* (DC) that defined in section 3. 5. Then, every document index vector D_d is split into two random vectors denoted as $\{D'_d, D''_d\}$. The splitting procedure is expressed as follow: take S as the splitting indicator, if the j-th bit of S is 0, $D'_d[j]$ and $D''_d[j]$ are set as the same as $D_d[j]$; if the j-th bit of S is 1, $D'_d[j]$ and $D''_d[j]$ are set randomly so long as their sum is equal to $D_d[j]$. So, the encrypted index vector $\tilde{D}_d$ is denoted as $\tilde{D}_d = \{M_1^T D'_d, M_2^T D''_d\}$. Store $\tilde{D}_d$ at the leaf node that stores correspondent D_d and delete D_d. For each internal node u in the index tree, a static hash table λ is generated. There are n tuples $(key, value)$ in λ, and for every $i = 1,2,..., n$, set $\lambda_u[f(sk_1, w_i)] = Enc(g(sk_2, w_i), D_u[i])$. Store λ_u in internal node u and delete D_u. Finally, the encrypted searchable index tree I is generated.

GenQuery $(\tilde{W}, SK)$: With the interested keywords in $\tilde{w}$, the n-dimension query vector Q is generated. Each dimension of Q is a normalized IDF weight of corresponding keyword. Specifically, if i-th keyword of w is in $\tilde{w}$, $Q[i] = w_{q,i}$, otherwise $Q[i] = 0$. Next, Q is also split into two random vectors as $\{Q', Q''\}$ using the similar splitting procedure used for document index vector. The difference is that if the j-th bit of S is 0, $Q'[j]$ and $Q''[j]$ are set randomly so long as their sum is equal to $Q[j]$; if the j-th bit of S is 1, $Q'[j]$ and $Q''[j]$ are set as the same as $Q[j]$. Then, the encrypted query vector $\tilde{Q}$ is in the form of $\{M_1^{-1}Q', M_2^{-1}Q''\}$. Next, $T_{\tilde{W}} = \{f(sk_1, w_{i1}), g(sk_2, w_{i1}), f(sk_1, w_{i2}), g(sk_2, w_{i2}),..., f(sk_1, w_{it}), g(sk_2, w_{it})\}$ is produced by encrypting each item in $\tilde{w}$. Finally, the $\{T_{\tilde{W}}, \tilde{Q}\}$ is sent to the cloud server.

Search $(I, \tilde{Q}, T_{\tilde{W}}, k)$: The cloud server follow the search algorithm expressed in section 3.5. Let u be an internal node in I, and let $a_t = \lambda_u[f(sk_1, w_t)]$ for each item $f(sk_1, w_t)$ in $T_{\tilde{W}}$. If exist at least one a_t satisfies $Dec(g(sk_2, w_t), a_t) = 1$, the procedure continue to search all children of u. When arrive at a leaf node, the procedure obtains the encrypted document vector $\tilde{D}_d$ and compute the similarity of $\tilde{D}_d$ and $\tilde{Q}$ using Eq(2). Finally, the cloud server will rank the searched documents based on their similarity scores.

$$\begin{aligned} & SC(\tilde{D}_d, \tilde{Q}) \\ &= \{M_1^T D'_d, M_2^T D''_d\} \cdot \{M_1^{-1} Q', M_2^{-1} Q''\} \\ &= D'_d \cdot Q' + D''_d \cdot Q'' \\ &= D_d \cdot Q \end{aligned} \quad (2)$$

5. Dynamic Update

In this section, we describe the process of dynamic update in detail. Three steps should be executed to update a document, which are showed as follows:

UpdHelper (I, C, u, i) : As expressed above, the data owner outsources encrypted dataset C and searchable index I together into the cloud. So, when the data owner want to add or delete a document d_i, he or she should modify C and I at the same time. If let the cloud server modify the corresponding data in the searchable index I, the keyword privacy will be leaked as the cloud must know the plaintext data stored in the index. To protect data privacy, the operation conducted by the cloud server only based on the document identifier. For example, if the data owner want to delete document d_i, then the update operation implemented by the cloud server only based on the document identifier i. Hence, in this phase, the cloud server will only perform the structure update of the searchable index tree I and return a portion of I, denoted as $T(u)$, where the u is the symbol of update and is either an insertion of document d_i or a deletion of document d_i. To be specific, $T(u)$ is the portion of the index tree that is accessed during an update. The size of $T(u)$ is $O(n \log m)$, as a balanced binary tree update takes $O(\log m)$ time in the worst case and the size of encrypted index vector stored in each node of the tree is $O(n)$. We give an example to further illustrate the idea. In Figure 2, when u is "deletion of document d_7", the $T(u)$ includes nodes r, r_{12}, r_{23} and r_{24}.

UpdToken $(d_i, T(u))$: In this phase, the data owner will generate $T'(u)$ from $T(u)$. At first, if the u is an insertion of document d_i, the data owner encrypts d_i denoted as c_i. And then new index vector is generated for each node in $T(u)$ based on the original data stored in it and index vector of document d_i. Next, the modified $T(u)$ is encrypted using the same method showed above and $T'(u)$ is generated.

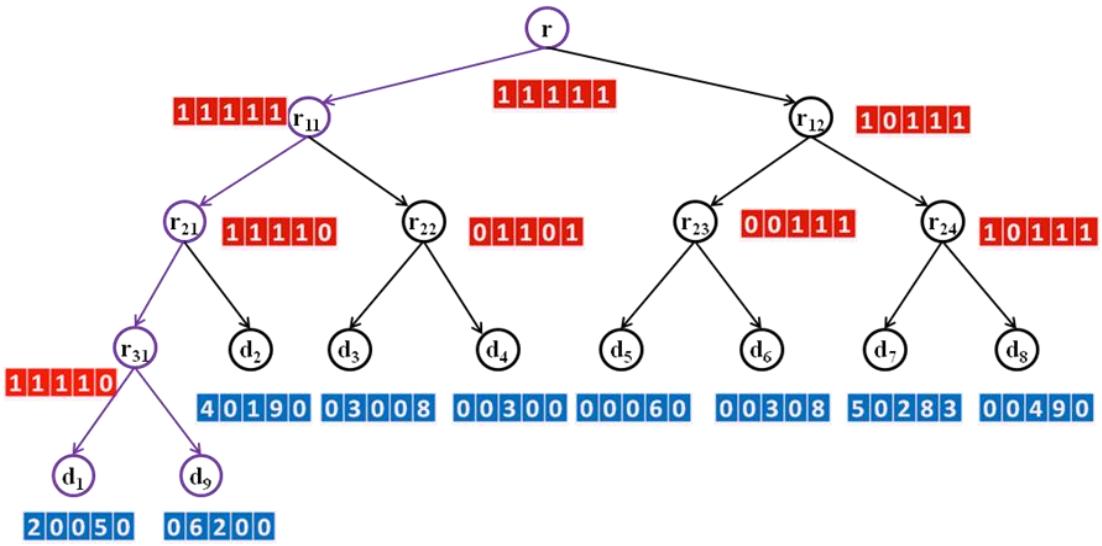


Figure 3. The Index Tree after Inserting the Document d_9

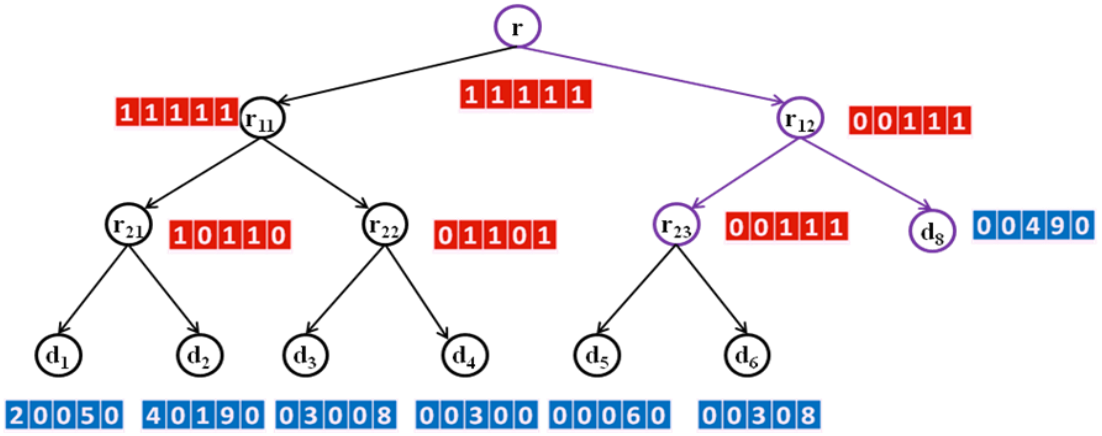


Figure 4. The Index Tree after Deleting the Document d_7

Update $(I, C, c_i, T'(u))$ **:** The cloud server just copies the new data $T'(u)$ to the already structurally updated searchable index tree I . And then the new searchable index and the new encrypted dataset are outputted. Two examples are given in Figure 3 and Figure 4, and Figure 3 shows the index tree after inserting document d_9, Fig.4 shows the index tree after deleting document d_7 .

6. Security Analysis

In this section, we analyze the security of the scheme considering the privacy requirements as expressed in section 3.

(1)Index Confidentiality and Query Confidentiality: $\tilde{D}_d$, $T_{\tilde{W}}$ and $\tilde{Q}$ are in encrypted form. As long as the secret key SK is well protected, the cloud server can not able to deduce

D_d , $\tilde{W}$ and Q . It is also impossible for the cloud server to infer the query keywords, term frequency related information (TF and DF) included in the documents or queries from the final similarity scores, which are random values for the server. This has been proven in the known ciphertext model in [20]. Therefore, index confidentiality and query confidentiality are well protected.

(2)Query Unlinkability: With the uncertainty of the vector splitting procedure, the vector encryption method that we employed provides non-deterministic encryption. Hence, different encrypted query vector $\tilde{Q}$ is generated even for the same search request (*e.g.,* same search keywords). The aim of non-link ability of search requests is achieved to this extent. However, on the one hand, as same search request will generate same $T_{\tilde{W}}$, the internal nodes visited during the search process and the outputted encrypted documents are same. On the other hand, the similarity score for same search request (though been encrypted to different vector) is equal (see Eq. (2)). With the two aspects information stated above, the cloud server is possible to link same search request. Under this circumstance the search pattern or the access pattern is leaked in the known ciphertext model.

(3) Keyword Privacy: In the known ciphertext model, only the encrypted data set C , encrypted search query and the searchable index I are available to the cloud server. As analyzed above, the dataset, search query and searchable index are all in encrypted form, which make it difficult for the cloud server and other unauthorized attackers to deduce a specific keyword. Although, the cloud can access similarity scores of searched documents, it can utilize these scores do nothing since it does not know related information about the outsourced dataset in the known ciphertext model. So the keyword privacy is protected to this extent.

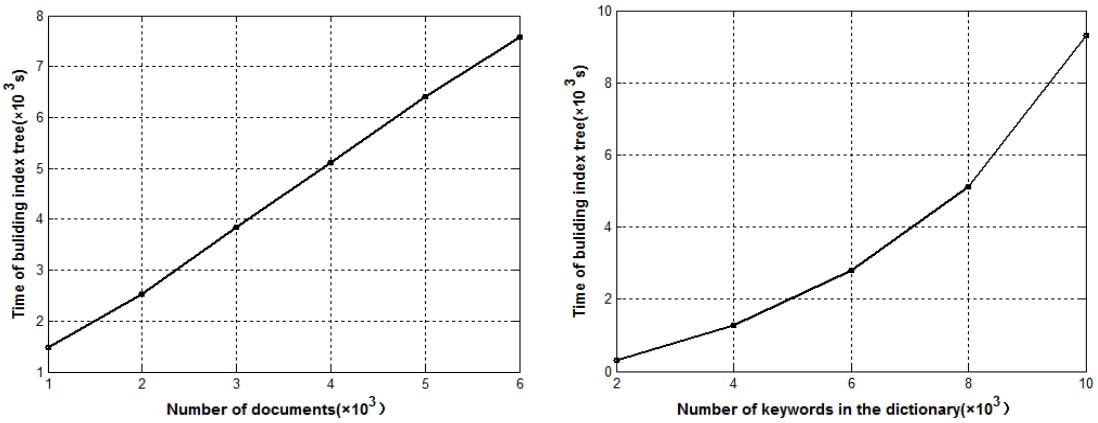
7. Performance Analysis

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) [23], which includes about 6500 publications.

7.1. 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 $H \times H$ matrix and an H -dimension vector where H is n in our scheme. And the whole number of nodes in the index tree (or the layers of the tree) is related to the number of documents in the dataset. Figure 5(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 because the time cost for encrypting each hash table and index vector is fixed. Figure 5(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. In

addition, the storage overhead of the index tree for the different sizes of dictionary with the fixed size of dataset m=1000 is shown in Figure 6.



(a) (b)

Figure 5. 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

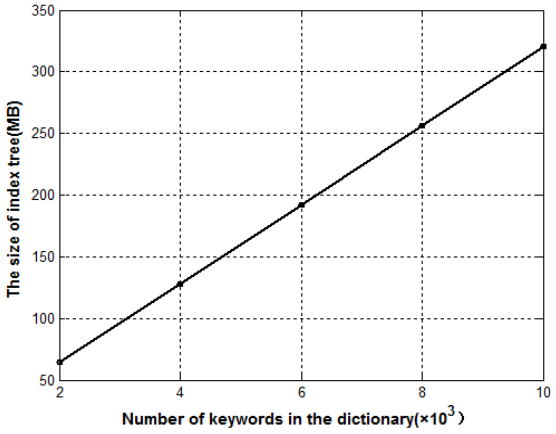
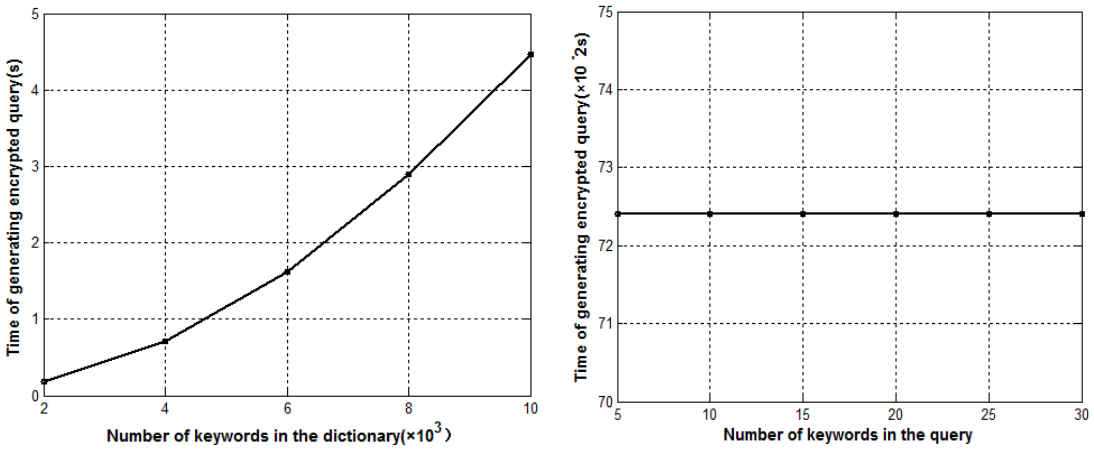


Figure 6. The Size of Index Tree for the Different Size of Dictionary with the Same Dataset, m=1000

7.2. 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 ($n \times n$ -dimension) and query vector (n -dimension) depends on the size of keywords dictionary. Figure 7(a) shows the relationship between the time of generating search query and the number of keywords in dictionary. Note that each search keyword is encrypted by a pseudorandom function in search query generation phase. The relationship between the time of generating search query and the number of search keywords is shown in Figure 7(b) , which indicates that the number of keywords in the query has little impact on the generation time. According to Figure 7(a) and Figure 7(b), it can be demonstrated that the difference of costs for search query generation is

mainly caused by the different size of matrices and vector, that is, the number of keywords in the dictionary.

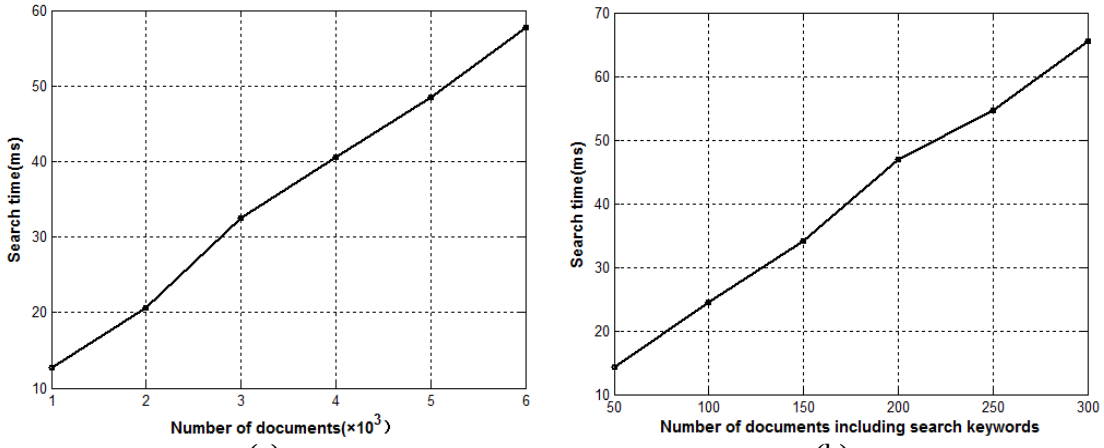


(a) (b)

Figure 7. Time Cost of Generating Encrypted Query (a) For the Different Number of Keywords in the Dictionary with the Same Query Keywords (b) For the Different Number of Keywords in the Query with the Same Dictionary, n=4000

7.3. Search Efficiency

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. Figure 8(a) shows the search time for our scheme with different size of dataset. Let r represent the number of documents including the search keywords.



(a) (b)

Figure 8. 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

From Figure 8(a), we can know that the search time is mainly depends on the number of documents in the dataset when r is fixed. Figure 8(b) shows the relationship between search time and r with same dataset, in which search time is almost linear to r.

8. Conclusions and Future Work

In this paper, we propose secure search scheme supporting both multi-keyword ranked search and dynamic update 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 and dynamic updating. In term of accuracy, we adopt the vector space model combined with cosine measure, which is popular in information retrieval field, 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. The tree-based searchable index designed in our scheme can support dynamic update well, only accessing a portion of the index tree. Our designed scheme can provide insertion and deletion update. 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 how to further reduce the time cost for index tree construction and so on. We will do more research in the future.

Acknowledgements

This paper is a revised and expanded version of a paper entitled “Secure and Efficient Multi-keyword Ranked Search over Encrypted Cloud Data” presented at CIA 2014, Angeles City (Clark), Philippines, April 24-26, 2014. This work is supported by the NSFC (61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61373133), GYHY201206033, 201301030, 2013DFG12860, BC2013012, PAPD fund, Hunan province science and technology plan project fund (2012GK3120), the Scientific Research Fund of Hunan Provincial Education Department (10C0944), and the Prospective Research Project on Future Networks of Jiangsu Future Networks Innovation Institute (BY2013095-4-10).

References

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, “A break in the clouds: towards a cloud definition”, ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, **(2009)**, pp. 50–55.
[2] S. Kamara and K. Lauter, “Cryptographic cloud storage”, Financial Cryptography and Data Security, Springer Berlin Heidelberg Publishing, **(2010)**.
[3] C. Wang, N. Cao, K. Ren and W. Lou, “Enabling secure and efficient ranked keyword search over outsourced cloud data”, IEEE Transactions on Parallel and Distributed Systems, vol. 23, issue 8, **(2012)** August, pp. 1467–1479.
[4] D. Song, D. Wagner and A. Perrig, “Practical techniques for searches on encrypted data”, In Proceedings of S&P, **(2000)** May 14-17, Berkeley, CA.
[5] E.-J. Goh, “Secure indexes”, Cryptology ePrint Archive, **(2003),** http://eprint.iacr.org/2003/216.
[6] R. Curtmola, J. A. Garay, S. Kamara and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions”, In Proceedings of the 13th ACM conference on Computer and communications security, **(2006)**, pp. 79-88.
[7] D. Boneh, G. D. Crescenzo, R. Ostrovsky and G. Persiano, “Public key encryption with keyword search”, In Proceedings of EUROCRYPT, **(2004)** May 2-6, Interlaken, Switzerland.
[8] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A. L. Varna, S. He, M. Wu and D. W. Oard, “Confidentiality-preserving rank-ordered search”, In Proceedings of the 2007 ACM Workshop on Storage Security and Survivability, **(2007)**, pp. 7–12.
[9] S. Zerr, D. Olmedilla, W. Nejdl and W. Siberski, “Zerber+r: Top-k retrieval from a confidential index”, In Proceedings of EDBT, **(2009)**, pp. 439–449.

[10] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data", In Proceedings of IEEE INFOCOM, **(2011)** April 10-15, Shanghai, China.
[11] 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", In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, **(2013)**, pp. 71-82.
[12] S. Kamara and C. Papamanthou, "Parallel and Dynamic Searchable Symmetric Encryption" Cryptology ePrint Archive, **(2013),** http://eprint.iacr.org/2013/335.
[13] I. H. Witten, A. Moffat and T. C. Bell, Managing gigabytes: Compressing and indexing documents and images, Morgan Kaufmann Publishing, San Francisco, **(1999)** May.
[14] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data", In Proceedings of the 4th Theory of Cryptography Conference, **(2007)** February 21-24, Amsterdam, Netherlands.
[15] P. Golle, J. Staddon and B. R. Waters, "Secure conjunctive keyword search over encrypted data", In Proceedings of ACNS, **(2004)** June 8-11, Yellow Mountain, China.
[16] J. Katz, A. Sahai and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products", In Proceedings of EUROCRYPT, **(2008)** April 13-17, Istanbul, Turkey.
[17] A. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption", In Proceedings of EUROCRYPT, **(2010)** May 30-June 3, French Riviera.
[18] E. Shen, E. Shi and B. Waters, "Predicate privacy in encryption systems", In Proceedings of the 6th Theory of Cryptography Conference, **(2009)** March 15-17, San Francisco, CA, USA.
[19] D. A. Grossman and O. Frieder, "Information retrieval: Algorithms and heuristics", Springer Publishing, **(2004)**.
[20] W. K. Wong, D. W. Cheung, B. Kao and N. Mamoulis, "Secure knn computation on encrypted databases", In Proceedings of SIGMOD, **(2009)**, pp. 139–152.
[21] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing", In Proceedings of IEEE INFOCOM'10 Mini-Conference, **(2010)** March 14-19, pp. 1-5, San Diego, CA, USA.
[22] P. Scheuermann and M. Ouksel, "Multidimensional B-trees for associative searching in database systems", Information systems, vol.7, issue 2, **(1982)**, pp. 123–137.
[23] RFC, "Request For Comments Database", http://www.ietf.org/rfc.html.
[24] J. S. Hwan, Y. E. Gelogo and B. Park, "Next Generation Cloud Computing Issues and Solutions", International Journal of Control and Automation, vol. 5, no. 1, **(2012)**, pp. 63-70.
[25] R. D. Caytiles, S. Lee and B. Park, "Cloud Computing: The Next Computing Paradigm", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 2, **(2012)**, pp. 297-302.

Authors

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

Lu Zhou, she received her BE in Software Engineering from Nanjing University of Information Science and Technology, China, in 2012. She is currently pursuing her MS in computer science and technology at the School Of Computer and Software, Nanjing University of Information Science and Technology, China. Her research interests include network and information security, steganography, digital watermarking, copyright protection technology.

Zhangjie Fu, he received his BS in education technology from Xinyang Normal University, China, in 2006; received his MS in education technology from the College of Physics and Microelectronics Science, Hunan University, China, in 2008; obtained his PhD in computer science from the College of Computer, Hunan University, China, in 2012. Currently, he works as an assistant professor in School of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include cloud computing, digital forensics, network and information security.

Jiangang Shu, he received his BE in Network Technology and Engineering from Nanjing University of Information Science & Technology (NUIST), Nanjing, China, in 2012. He is currently pursuing his MS in computer science and technology at the School of Computer & Software, Nanjing University of Information Science and Technology, China. His research interests include cloud security, steganography, network and information security.

