International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

IJIEMR Transactions, online available on 17 April 2019.

Link : http://www.ijiemr.org

Title:- A Practical And Secure Multi-Keyword Ranked Search Technique In Cloud Computing.

Volume 08, Issue 04, Pages: 203 - 210.

Paper Authors

CHINTA MOSES RAJU,' CH.N.D.CHAMUNDESWARI.
Department of MCA, SKBR PG College.

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Approvals We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

A PRACTICAL AND SECURE MULTI-KEYWORD RANKED SEARCH TECHNIQUE IN CLOUD COMPUTING

[1]CHINTA MOSES RAJU, [2]CH.N.D.CHAMUNDESWARI
[1]PG Scholar Department of MCA, SKBR PG College, Amalapuram
[2]Lecturer, Department of MCA, SKBR PG College, Amalapuram

ABSTRACT: Due to the increasing popularity of cloud computing, more and more data owners are motivated to outsource their data to cloud servers for great convenience and reduced cost in data management. However, sensitive data should be encrypted before outsourcing for privacy requirements, which obsoletes data utilization like keyword-based document retrieval. In this paper, we present a secure multi-keyword ranked search scheme over encrypted cloud data, which simultaneously supports dynamic update operations like deletion and insertion of documents. Specifically, the vector space model and the widely-used TF_IDF model are combined in the index construction and query generation. We construct a special tree-based index structure and propose a "Greedy Depth-first Search" algorithm to provide efficient multi-keyword ranked search. The secure kNN algorithm is utilized to encrypt the index and query vectors, and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. In order to resist statistical attacks, phantom terms are added to the index vector for blinding search results. Due to the use of our special tree-based index structure, the proposed scheme can achieve sub-linear search time and deal with the deletion and insertion of documents flexibly. Extensive experiments are conducted to demonstrate the efficiency of the proposed scheme.

KEY WORDS: Searchable encryption, multi-keyword ranked search, dynamic update, cloud computing.

I.INTRODUCTION

Cloud computing has been considered as a new model of enterprise IT infrastructure, which can organize huge resource of computing, storage and applications, and enable users to enjoy ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources With great efficiency and minimal economic overhead [1]. Attracted by these appealing features, both individuals and enterprises are motivated to outsource their data to the cloud, instead of purchasing software and hardware to manage the data themselves. Despite of the various advantages of cloud services, outsourcing sensitive information (such as e-mails, personal health records, company finance data, government documents, etc.) to remote servers brings privacy concerns. The cloud service providers (CSPs) that keep the data for users may access users' sensitive information without authorization. A general approach to protect the data confidentiality is to encrypt the data before outsourcing [2]. However, this will cause a huge cost in terms of data usability. For example, the existing techniques on keyword-based information retrieval, which are widely used on the plaintext data, cannot be directly applied on the encrypted data. Downloading all

the data from the cloud and decrypt locally is obviously impractical.

In order to address the above problem, researchers have designed some general-purpose solutions with fully-homomorphic encryption [3] or oblivious RAMs [4]. However, these methods are not practical due to their high computational overhead for both the cloud sever and user. On the contrary, more practical special purpose solutions, such as Searchable Encryption (SE) schemes have made specific contributions in terms of efficiency, functionality and security. Searchable encryption schemes enable the client to store the encrypted data to the cloud and execute keyword search over ciphertext domain. So far, abundant works have been proposed under different threat models to achieve various search functionality, such as single keyword search, similarity search, multi-keyword boolean search, ranked search, multi-keyword ranked search, etc. Among them, multi keyword ranked search achieves more and more attention for its practical applicability. Recently, some *dynamic* schemes have been proposed to support inserting and deleting operations on document collection. These are significant works as it is highly possible that the data owners need to update their data on the cloud server. But few of the dynamic schemes support efficient multikeyword ranked search.

This paper proposes a secure tree-based search scheme over the encrypted cloud data, which supports multikeyword ranked search and dynamic operation on the document collection. Specifically, the vector space model and the widely-used "term frequency (TF) × inverse document frequency (IDF)" model are combined in the index construction and query generation to provide multi keyword ranked search. In order to obtain high search efficiency, we construct a tree-based index structure and propose a "Greedy Depth-first Search" algorithm based on this index tree. Due to the special structure of our tree-based index, the proposed search scheme can flexibly achieve sub-linear search time and deal with the deletion and insertion of documents.

The secure kNN algorithm is utilized to encrypt the index and query vectors, and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. To resist different attacks in different threat models, we construct two secure search schemes: the basic dynamic multi-keyword ranked search (BDMRS) scheme in the known ciphertext model, and the enhanced dynamic multi-keyword ranked search (EDMRS) scheme in the known background model. Our contributions are summarized as follows:

1) We design a searchable encryption scheme that supports both the accurate multi-keyword ranked search and flexible dynamic operation on document collection.

2) Due to the special structure of our tree-based index, the search complexity of the proposed scheme is fundamentally kept to logarithmic. And in practice, the proposed scheme can achieve higher search efficiency by executing our "Greedy Depth-first Search" algorithm. Moreover, parallel search can be flexibly performed to further reduce the time cost of search process.

II. RELATED WORK

Searchable encryption schemes enable the clients to store the encrypted data to the cloud and execute keyword search over ciphertext domain. Due to different cryptography

primitives, searchable encryption schemes can be constructed using public key based cryptography [5], [6] or symmetric key based cryptography [7], [8], [9], [10]. Song *et al.* [7] proposed the first symmetric searchable encryption (SSE) scheme, and the search time of their scheme is linear to the size of the data collection. Goh [8] proposed formal security definitions for SSE and designed a scheme based on Bloom filter. The search time of Goh's scheme is $O(n)$, where n is the cardinality of the document collection. Curtmola *et al.* [10] proposed two schemes (SSE-1 and SSE-2) which achieve the optimal search time. Their SSE-1 scheme is secure against chosen-keyword attacks (CKA1) and SSE-2 is secure against adaptive chosen-keyword attacks (CKA2).

These early works are single keyword boolean search schemes, which are very simple in terms of functionality. Afterward, abundant works have been proposed under different threat models to achieve various search functionality, such as single keyword search, similarity search, multi-keyword boolean search, ranked search, and multi-keyword ranked search, etc. Multi-keyword boolean search allows the users to input multiple query keywords to request suitable documents. Among these works, conjunctive keyword search schemes only return the documents that contain all of the query keywords. Disjunctive keyword search schemes return all of the documents that contain a subset of the query keywords. Predicate search schemes are proposed to support both conjunctive and disjunctive search. All these multikeyword search schemes retrieve search results based on the existence of keywords, which cannot provide acceptable result ranking functionality.

Ranked search can enable quick search of the most relevant data. Sending back only the top-k most relevant documents can effectively decrease network traffic. Some early works have realized the ranked search using order-preserving techniques, but they are designed only for single keyword search. Cao *et al.* realized the first privacy-preserving multi-keyword ranked search scheme, in which documents and queries are represented as vectors of dictionary size. With the "coordinate matching", the documents are ranked according to the number of matched query keywords. However, Cao *et al.*'s scheme does not consider the importance of the different keywords, and thus is not accurate enough. In addition, the search efficiency of the scheme is linear with the cardinality of document collection. Sun *et al.* presented a secure multi-keyword search scheme that supports similarity-based ranking.

The authors constructed a searchable index tree based on vector space model and adopted cosine measure together with TF×IDF to provide ranking results. Sun *et al.*'s search algorithm achieves better-than-linear search efficiency but results in precision loss. O¨ rencik *et al.* proposed a secure multi-keyword search method which utilized local sensitive hash (LSH) functions to cluster the similar documents. The LSH algorithm is suitable for similar search but cannot provide exact ranking. In, Zhang *et al.* proposed a scheme to deal with secure multi-keyword ranked search in a multi-owner model. In this scheme, different data owners use different secret keys to encrypt their documents and keywords while authorized data users can query without knowing keys of these different data owners.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

The authors proposed an "Additive Order Preserving Function" to retrieve the most relevant search results. However, these works don't support dynamic operations.

Practically, the data owner may need to update the document collection after he upload the collection to the cloud server. Thus, the SE schemes are expected to support the insertion and deletion of the documents. There are also several dynamic searchable encryption schemes. In the work of Song *et al.* [7], the each document is considered as a sequence of fixed length words, and is individually indexed. This scheme supports straightforward update operations but with low efficiency. Goh [8] proposed a scheme to generate a sub-index (Bloom filter) for every document based on keywords. Then the dynamic operations can be easily realized through updating of a Bloom filter along with the corresponding document. However, Goh's scheme has linear search time and suffers from false positives. In 2012, Kamara *et al.* constructed an encrypted inverted index that can handle dynamic data efficiently. But, this scheme is very complex to implement. Subsequently, as an improvement, Kamara *et al.* proposed a new search scheme based on tree-based index, which can handle dynamic update on document data stored in leaf nodes. However, their scheme is designed only for single keyword Boolean search. In [32], Cash *et al.* presented a data structure for keyword/identity tuple named "TSet". Then, a document can be represented by a series of independent T-Sets. Based on this structure, Cash *et al.* proposed a dynamic searchable encryption scheme. In their construction, newly added tuples are stored in another database in the cloud, and deleted tuples are recorded in a revocation list. The final search result is achieved through excluding tuples in the revocation list from the ones retrieved from original and newly added tuples. Yet, Cash *et al.*'s dynamic search scheme doesn't realize the multi-keyword ranked search functionality.

III. PROPOSED SYSTEM

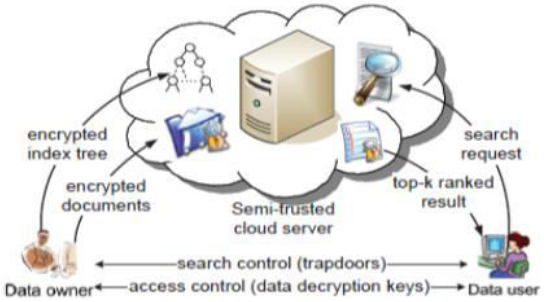

Fig. 1: PROPOSED SYSTEM

The system model in this paper involves three different entities: data owner, data user and cloud server, as illustrated in Fig. 1.

Data owner has a collection of documents $F = \{f1; f2; :::; fn\}$ that he wants to outsource to the cloud server in encrypted form while still keeping the capability to search on them for effective utilization. In our scheme, the data owner firstly builds a secure searchable tree index I from document collection F, and then generates an encrypted document collection C for F. Afterwards, the data owner outsources the encrypted collection C and the secure index I to the cloud server, and securely distributes the key information of trapdoor generation (including keyword IDF values) and document decryption to the authorized data users. Besides, the data owner is responsible for the update operation of his documents stored in the cloud server. While updating, the data owner generates the update information locally and sends it to the server.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Data users are authorized ones to access the documents of data owner. With *t* query keywords, the authorized user can generate a trapdoor *TD* according to search control mechanisms to fetch *k* encrypted documents from cloud server. Then, the data user can decrypt the documents with the shared secret key.

Cloud server stores the encrypted document collection *C* and the encrypted searchable tree index *I* for data owner. Upon receiving the trapdoor *TD* from the data user, the cloud server executes search over the index tree *I*, and finally returns the corresponding collection of top-*k* ranked encrypted documents. Besides, upon receiving the update information from the data owner, the server needs to update the index *I* and document collection *C* according to the received information.

The cloud server in the proposed scheme is considered as "honest-but-curious", which is employed by lots of works on secure cloud data search. Specifically, the cloud server honestly and correctly executes instructions in the designated protocol. Meanwhile, it is curious to infer and analyze received data, which helps it acquire additional information. Depending on what information the cloud server knows, we adopt the two threat models proposed by Cao *et al.* we firstly describe the unencrypted dynamic multi-keyword ranked search (UDMRS) scheme which is constructed on the basis of vector space model and KBB tree. Based on the UDMRS scheme, two secure search schemes (BDMRS and EDMRS schemes) are constructed against two threat models, respectively.

Index Construction of UDMRS Scheme

We have briefly introduced the KBB index tree structure, which assists us in introducing the index construction. In the process of index construction, we first generate a tree node for each document in the collection. These nodes are the leaf nodes of the index tree. Then, the internal tree nodes are generated based on these leaf nodes. The formal construction process of the index is presented in Algorithm 1.

Following are some notations for Algorithm 1. Besides, the data structure of the tree node is defined as *(ID;D; Pl; Pr; FID)*, where the unique identity *ID* for each tree node is generated through the function GenID().

• *CurrentNodeSet* – The set of current processing nodes which have no parents. If the number of nodes is even, the cardinality of the set is denoted as $2h(h \in \mathbf{Z}+)$, else the cardinality is denoted as $(2h + 1)$.

• *TempNodeSet* – The set of the newly generated nodes. In the index, if $Du[i] \neq 0$ for an internal node *u*, there is at least one path from the node *u* to some leaf, which indicates a document containing the keyword *wi*. In addition, *Du*[*i*] always stores the biggest normalized TF value of *wi* among its child nodes. Thus, the possible largest relevance score of its children can be easily estimated.

Search Process of UDMRS Scheme

The search process of the UDMRS scheme is a recursive procedure upon the tree, named as "Greedy Depthfirst Search (GDFS)" algorithm. We construct a result list denoted as *RList*, whose element is defined as *(RScore; FID)*. Here, the *RScore* is the relevance score of the document *fFID* to the query, which is

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

calculated according to Formula (1). The *RList* stores the k accessed documents with the largest relevance scores to the query.

The elements of the list are ranked in descending order according to the *RScore*, and will be updated timely during the search process. Following are some other notations, and the GDFS algorithm is described in Algorithm 2.

• RScore(*Du;Q*) – The function to calculate the relevance score for query vector Q and index vector *Du* stored in node u, which is defined in Formula (1).

• *kthscore* – The smallest relevance score in current *RList*, which is initialized as 0.

• *hchild* – The child node of a tree node with higher relevance score.

• *lchild* – The child node of a tree node with lower relevance score.

Since the possible largest relevance score of documents rooted by the node u can be predicted, only a part of the nodes in the tree are accessed during the search process. Search process with the document collection $F = \{fi|i = 1; :::; 6\}$, cardinality of the dictionary $m = 4$, and query vector $Q = (0; 0{:}92; 0; 0{:}38)$.

Algorithm 1 BuildIndexTree(*F*)

Input: the document collection $F = \{f1; f2; :::; fn\}$ with the identifiers $FID = \{FID|FID = 1; 2; :::; n\}$.

Output: the index tree *T*

1: **for** each document *fFID* in *F* **do**
2: Construct a leaf node u for *fFID*, with *u:ID* = GenID(), *u:Pl* = *u:Pr* = *null*, *u:FID* = *FID*, and $D[i] = TFfFID;wi$ for $i = 1; :::;m$;—
3: Insert u to *CurrentNodeSet*;
4: **end for**
5: **while** the number of nodes in *CurrentNodeSet* is larger than 1 **do**
6: **if** the number of nodes in *CurrentNodeSet* is even, i.e. $2h$ **then**
7: **for** each pair of nodes u' and u'' in *CurrentNodeSet* **do**
8: Generate a parent node u for u' and u'', with *u:ID* = GenID(), *u:Pl* = u', *u:Pr* = u'', *u:FID* = 0 and $D[i] = max\{u':D[i]; u'':D[i]\}$ for each $i = 1; :::;m$;
9: Insert u to *TempNodeSet*;
10: **end for**
11: **else**
12: **for** each pair of nodes u' and u'' of the former $(2h - 2)$ nodes in *CurrentNodeSet* **do**
13: Generate a parent node u for u' and u'';
14: Insert u to *TempNodeSet*;
15: **end for**
16: Create a parent node $u1$ for the $(2h - 1)$-th and $2h$-th node, and then create a parent node u for $u1$ and the $(2h + 1)$-th node;
17: Insert u to *TempNodeSet*;
18: **end if**
19: Replace *CurrentNodeSet* with *TempNodeSet* and then clear *TempNodeSet*;
20: **end while**
21: **return** the only node left in *CurrentNodeSet*, namely, the root of index tree *T* ;

Algorithm 2 GDFS(IndexTreeNode u)

1: **if** the node u is not a leaf node **then**
2: **if** RScore(*Du;Q*) > *kthscore* **then**
3: GDFS(*u:hchild*);
4: GDFS(*u:lchild*);
5: **else**
6: **return**
7: **end if**
8: **else**
9: **if** RScore(*Du;Q*) > *kthscore* **then**
10: Delete the element with the smallest relevance score from *RList*;
11: Insert a new element ⟨ RScore(*Du;Q*); *u:FID* and sort all the elements of *RList*;

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

12: **end if**
13: **return**
14: **end if**

IV. RESULTS

A. Efficiency

1) **Trapdoor generation**: The trapdoor generation process contains three major steps: stemming, the Bloom filter generation and the encryption shows the total time of trapdoor stemming and Bloom filter generation. The generation time increased linearly with respect to the number of the inserted keywords. As the number of keywords grew, the trapdoor generation time also increased.

2) **Index construction**: The index construction time was the same as that of trapdoor generation. Because the stemming and Bloom filter generation were linear in the number of the keywords, the index vector generation time could be large, but it was just a one-time effort shows that the encryption time is linear in the size of files because the index structure we constructed was a per file based index.

3) **Search time:** One important parameter that affected the search time was the number of the files n. Because our index was a per file based index, the search time increased linearly in the number of files, as illustrated

(a) The Bloom filter generation time of trapdoor & a single index file; The stemming time of keywords.

(b) The encryption time for all the indexes..

(c) The search time of different size of the file set. We set the query keyword number = 5;

(d) The search time of different number of query keyword. We set the size of document = 3000 we note that the number of the query keywords had a small impact on the search time. This is because regardless of the number of keywords, all of them were mapped into a query bloom vector. Hence, the search time was independent of the number of query keywords to a large extent. Another important parameter is the length of the bloom filter. The search efficiency of our scheme was the same as that of the original scheme because both the index and the trapdoor were built in the same manner.

B. Result Accuracy

We used precision to measure the result accuracy. We denoted the true positive by tp and the false positive by fp, and the precision was equal to tp tp+fp. To generate the fuzzy search, we randomly chose keywords and modified it into a fuzzy keyword.

1) Precision of Our Scheme: An important parameter in our proposed scheme is the number of the keywords in the query. For the exact search, the precision decreased slightly from 100% to 95% as the number of the keywords increased from 1 to 10. Although the accuracy of the fuzzy search was not greater than that of the exact match, it was still produce a high level of accuracy, greater than 85%. From we note that the precision of the exact match slightly decreased from 100% to 95% as the number of the query keywords increased from 1 to 10.

V. CONCLUSION

In this paper, a secure, efficient and dynamic search scheme is proposed, which supports not only the accurate multi-keyword ranked search but also the dynamic deletion and insertion of documents. We construct a special keyword balanced binary tree as the index, and propose a "Greedy Depth-first Search" algorithm to obtain better efficiency than linear search. In addition, the parallel search process can be carried out to further reduce the time cost. The security of the scheme is protected against two threat models by using the secure kNN algorithm. Experimental results demonstrate the efficiency of our proposed scheme.

VI.REFERENCES

[1] K. Ren, C.Wang, Q.Wang *et al.*, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 136–149.

[3] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[4] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.

[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology-Eurocrypt 2004*. Springer, 2004, pp. 506–522.

[6] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III, "Public key encryption that allows pir queries," in *Advances in Cryptology-CRYPTO 2007*. Springer, 2007, pp. 50–67.

[7] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000, pp. 44– 55.

[8] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[9] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of the Third international conference on Applied Cryptography and Network Security*. Springer-Verlag, 2005, pp. 442–455.

[10] R. Curtmola, J. 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*. ACM, 2006, pp. 79–88.

[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–5.

[12] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 1156–1167.