International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 6, Issue 5, September- October 2017 ISSN 2278-6856

A special Tree based index structure for efficient data searching in cloud data

P. Dileep Kumar Reddy

Lecturer, Department of CSE
JNTUA College of Engineering, Anantapur
Andhra Pradesh, India.

Abstract: ***Cloud computing yields a variety of advantages like high storage space, shared resources and less management etc. Searchable encryption schemes are implemented to store the encrypted data. But this approach suffers from lack of security. To overcome this problem, a new tree based structure having Greedy Depth First Search algorithm is implemented to perform search process. Vector space and term frequency in addition to inverse document frequency are the two mechanisms responsible for efficient index and query generation. This approach involves UDMRS schemes which are more secure and efficient and it performs various operations like insertion and deletion of documents.***

Keywords: cloud data, encryption, keywords, data outsourcing, data privacy, security.

1. INTRODUCTION

Protecting the information [1] is the main task to the data owners. To obtain this advantage, data has to be encrypted before storing in the cloud [2] [3]. Encryption mechanisms are applied to provide security to sensitive information. Homomorphic encryption mechanism [4] is applied to gain security. But these schemes provide low security. So to overcome this drawback searchable encryption schemes are used. Basically, vector space technique and term frequency are the two techniques responsible in the process of both index and query generation [6]. To obtain better search efficiency, we implement a tree-based index structure. Greedy Depth-first search algorithm is majorly implemented to reduce search time and to carry out operations like insertion and deletion of documents effectively. UDMRS schemes which are introduced in this approach provide more security and better functionality.

The rest of the paper is organized as follows: Section 2 presents the Related work. Section 3 discusses the proposed approach. Section 4 presents the Performance Evaluation of the proposed approach. Finally Section 5 concludes the paper.

2. RELATED WORK

Our related work involves searchable encryption schemes to assure security.

2.1. SEARCHABLE ENCRYPTION SCHEMES

SE schemes are majorly designed to protect data by converting it into encryption manner. These approaches involves Symmetric searchable encryption scheme [7] [8], Single keyword search schemes, fuzzy keyword searchable encryption scheme [9]. These methods does not achieve better ranking [10] [11] [12] functionality and also it does not supports operations like insertion and deletion of documents.

2.1.1. Single Keyword Searchable Encryption

This scheme is one of the searchable encryption mechanism and is responsible to construct searchable index in encryption format [13]. By that, data is hidden to the server, only authorized persons will be given keys to access the information. The persons who are having public key can only write to the data stored on server . But for accessing the information which is present in the cloud, private key is necessary. The main drawback of this approach is, it is not suited for complex or large data and ensures low security [14].

Algorithm 1 BuildIndexTree($\mathcal{F}$)

Input: the document collection $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ with the identifiers $\mathcal{FID} = \{FID|FID = 1, 2, ..., n\}$.
Output: the index tree $\mathcal{T}$

1: **for** each document f_{FID} in $\mathcal{F}$ **do**
2: Construct a leaf node u for f_{FID}, with $u.ID =$ GenID(), $u.P_l = u.P_r = null$, $u.FID = FID$, and $D[i] = TF_{f_{FID},w_i}$ 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(), $u.P_l = u'$, $u.P_r = u''$, $u.FID = 0$ 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 u_1 for the $(2h-1)$-th and $2h$-th node, and then create a parent node u for u_1 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 $\mathcal{T}$;

Algorithm 1: Index Tree Construction.

3. PROPOSED APPROACH:

To overcome the drawback by the existing approaches, we design UDMRS schemes.

UNENCRYPTED DYNAMIC MULTI-KEYWORD RANKED SEARCH

UDMRS schemes provide better functionality, efficiency, security and perform dynamic updation operations. The UDMRS scheme consists of two efficient search schemes (BDMRS and EDMRS) and these are designed to ensuring more security.

3.1. Index Construction of UDMRS Scheme

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.

3.2. Search Process of UDMRS Scheme

The search process can be implemented by applying "Greedy Depth first search" algorithm. RList, is the result list, RScore is a relevance score obtained for each document and RList is responsible for storing the documents and it stores the document which has largest relevance. The rank of elements can be given in descending order based on the RScore, and will be updated whenever required. GDFS algorithm clearly is explained in Algorithm 2.

Algorithm 2 GDFS(IndexTreeNode u)

```
1: if the node u is not a leaf node then
2:   if RScore(D_u, Q) > k^th score then
3:     GDFS(u.hchild);
4:     GDFS(u.lchild);
5:   else
6:     return
7:   end if
8: else
9:   if RScore(D_u, Q) > k^th score then
10:    Delete the element with the smallest relevance
       score from RList;
11:    Insert a new element ⟨RScore(D_u, Q), u.FID⟩ and
       sort all the elements of RList;
12:  end if
13:  return
14: end if
```

Algorithm 2: Search Process.

3.3. BASIC DYNAMIC MULTI-KEYWORD RANKED SEARCH

This scheme is designed to satisfy the privacy concerns. This scheme protects both index query confidentiality. The same search requests based on the same visited paths and the same relevance scores are linked by the cloud server. The search process involves inner product computing of encrypted information. It does not leak any information about Specific keyword. By that, the keyword privacy is protected.

3.4. ENHANCED DYNAMIC MULTI-KEYWORD RANKED SEARCH

The EDMRS scheme is almost similar to BDMRS scheme. But this scheme achieves more security. Index query confidentiality are inherited from BDMRS scheme. This scheme introduces a random value in search process. By that, different query vectors can be generated for same search results and it receives different relevance score. Thus, the query unlinkability is protected.

4. PERFORMANCE EVALUATION

The performance analysis can be determined by evaluating the efficiency of index construction and updation operations. All the measurements during section are performed using reference website [15].

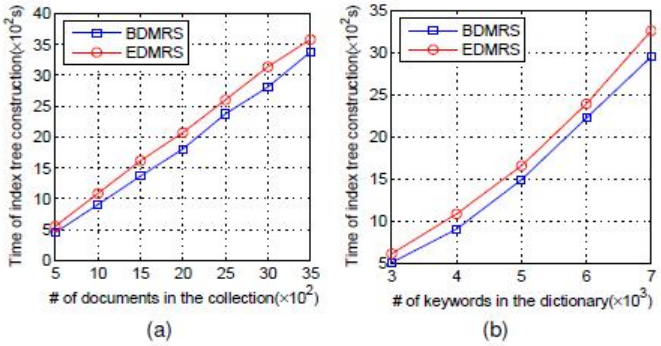

Fig. 1: Time cost for index tree construction: (a) for the different sizes of document collection with the fixed dictionary, $m = 4000$, and (b) for the different sizes of dictionary with the fixed document collection, $n = 1000$.

Fig 1 (a) shows that the time cost of index tree construction is almost linear to the data size. Fig 1 (b) shows that the construction of index is proportional to the number of keywords. EDMRS scheme takes more time that that of BDMRS scheme in case of dimension extension. Parallel search process is used to improve the search efficiency.

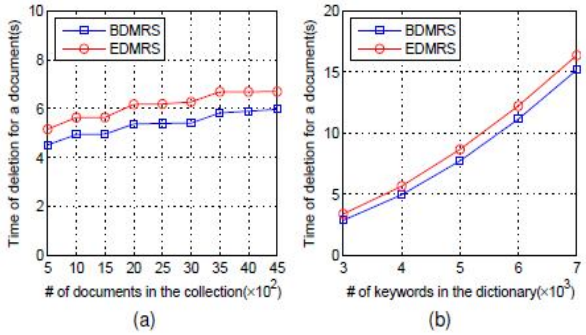

Fig. 2. Time cost for deletion of a document: (a) for the different sizes of document collection with the same dictionary, $m = 4000$, and (b) for the same document collection with different sizes of dictionary, $n = 1000$.

Fig. 2 (a) shows that the deletion of document takes nearly logarithmic time to the data size. Fig. 2(b) shows that the update time is proportional to the size of dictionary. The time complexity for update operations is denoted as $O(m^2 \log n)$.

5. CONCLUSION

UDMRS schemes are very secure and efficient. It efficiently performs the operations like insertion and deletion of documents in the cloud environment. We introduce a special tree based index structure to create index and proposes "Greedy Depth First Search "algorithm to improve better functionality and efficiency. Parallel Search process is helpful to reduce search time.

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]. C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," Parallel and Distributed Systems, IEEE Transactions on, vol. 23, no. 8, pp. 1467–1479, 2012.

[3]. C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 451–459.

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

[5]. E.-J. Goh et al., "Secure indexes." IACR Cryptology ePrint Archive, vol. 2003, pp. 216, 2003. [6]. D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in Proceedings of the 4th conference on Theory of cryptography. Springer-Verlag, 2007, pp. 535–554.

[7]. D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Ros¸u, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology CRYPTO 2013*. Springer, 2013, pp. 353–373.

[8]. 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.

[9]. 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.

[10]. 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. 506–522.67.

[11]. B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," Journal of Network and Computer Applications, vol. 34, no. 1, pp. 262–267, 2011.

[12]. 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. ACM, 2007, pp. 7–12.

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

[14]. D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Dynamic searchable encryption in very large databases: Data structures and implementation," in *Proc. of NDSS*, vol. 14, 2014.

[15]. Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang, "A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data", IEEE transactions on Parallel and Distributed systems, Vol: Pp no: 99, 2015.