International Journal of Current Advanced Research
ISSN: O: 2319-6475, ISSN: P: 2319-6505, Impact Factor: 6.614
Available Online at www.journalijcar.org
Volume 7; Issue 7(H); July 2018; Page No. 14414-14418
DOI: http://dx.doi.org/10.24327/ijcar.2018.14418.2613

Research Article

DESIGN AND ANALYSIS OF SEARCH ALGORITHM WITH B-TREE AND COMMUTATIVE KEY RSA FOR DYNAMIC UPDATION IN CLOUD COMPUTING

Pawan Kumr Tanwar[1]., Ajay Khunteta[2] and vishal Goar[3]

[1,2]Department of Computer Science Poornima University, Jaipur
[3]Department of Computer Application Govt. Engineering College, Bikaner

A R T I C L E I N F O

Article History:

Received 7th April, 2018
Received in revised form 16th May, 2018 Accepted 3rd June, 2018
Published online 28th July, 2018

Key words:

Searchable Encryption, Multi keyword, B tree, CKRSA .

A B S T R A C T

To access the data anytime from anywhere in an easy way and as an advantage of service storage at very low cost, organizations are keeping their important data at cloud space. For the establishment of trust between service provider and data user, we have used cryptography. For providing the security a method of cryptography is encryption scheme. Further encryption schemes are also have different types as per requirement and one of them is searchable encryption. Researchers have worked upon development of effective schemes for searchable encryption. Here we have explored some techniques of cryptography like B-tree and CKRSA for the enhancement of security level which leads to trust. We have uses the cloud platform Microsoft Azure for the searching upon encrypted clod data.

Copyright©2018 ***Pawan Kumr Tanwar., Ajay Khunteta and vishal Goar.*** *This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.*

INTRODUCTION

For the haring of resources by multiple users cloud computing is the best way. The advantage of cloud may extend from single user to big organizations due to cloud storage and access. Software and hardware resource virtualization saves the financial overhead of maintaining the warehouse of data. Privacy and security are very big issues in cloud computing. Various types of software and hardware security methods are used by the cloud service provider but the solutions are not sufficient for the protection of data from non authorized data users. Valuable data should be encrypted before uploading at cloud space. Encryption of data gives assurance of integrity and confidentiality of data. For the privacy preservation of data at cloud server, there is a need of designing an algorithm for searching which works upon enciphered data.

Number of researchers has contributed to search over encrypted cloud data. The searching is also classified in some categories like Boolean search, single or multi keyword search. In large amount of data the searched keyword may be matched in number of documents which leads to a tedious work for the data user for searching a large number of documents. Rank based search is a solution of this type of work where documents are ranked according to the relevance of the keywords.

Corresponding author:* **Pawan Kumr Tanwar
Department of Computer Science Poornima University, Jaipur

For an economical encrypted data searching method the researchers club the process of ranking the documents and multi keyword searching which becomes multi keyword rank search.

The time and cost of computation are two important criteria the researchers are using for performance evaluation in the searching of encrypted cloud data. Time of computation is the time taken for keyword searching, trapdoor generation etc. Similarly, cost of computation is the overhead of allocation of resource and utilization of CPU.

Here we have analyzed the problem of security in cloud space and proposed a remedy for it. Our solution is abstracted as follows: -

1. We have defined the problem of multi keyword rank search upon enciphered cloud data and an effective solution has been provided by us which satisfies the function of rank search securely where keyword privacy is maintained without the leakage of information regarding relevance score.
2. Our B-tree and CKRSA (Commutative Key RSA) based encryption and search schemes provides guarantee of security in comparison of already available searchable symmetric encryption schemes.
3. Thorough experimental output shows the efficiency and viability of the solution made available by us. In the remaining part of the paper various sections elaborates various thing like, review of literature, formulation of problem, performance analysis and conclusion.

Review of Literature

For protecting the data integrity and confidentiality the encryption of data is an effective method but at the time of searching it leads to low efficiency. For searching of complex data queries available schemes are not very effective according to literature available and therefore important data may be leaked and can be accessed by unauthorized persons. Li *et al.*[7] suggested the cryptography based symmetric searchable scheme. According to this method the word by word encryption of document is done. For searching of keyword same key is being sent by the data user to cloud server. The limitation of this scheme is that the frequency of keyword will be revealed. Goh *et al.*[23] suggested much better scheme than Song's method by secure table index construction. For his scheme Goh [23] used pseudorandom functions and bloom filter. Bosch *et al.* extended the scheme provided by Goh *et al.*[23] and provided the wild card searching concept but the limitation of the scheme is that false positives may be introduced by the bloom filters. Chang *et al.* suggested a scheme where an index is formed for every file. This is much secured scheme in comparison of Goh's scheme because words quantity in a document is not disclosed. The drawback of the method is that it does not support random updates of keywords. Golle *et al.*[24] suggested multi keyword searching with single encryption query but practically it is tedious to execute. Shareef *et al.*[16] provided the symmetric searching encryption scheme later Kamara *et al.* provided dynamic symmetric searching scheme where dynamic update (deletion and addition) of files can be applied. These schemes are for unit (single) keyword searching.

Wang *et al.*[10] proposed the scheme PKES (public key encryption with keyword search). The drawback of this scheme is inference attack over encryption of trapdoor. Security in th Wang's scheme has been improved by Baek *et al.* and Rhee *et al.* Keyword conjunction searching scheme has been proposed by Baek. The PKES schemes are complex and having high computation time, therefore these algorithms are not efficient.
Yang *et al.* proposed a scheme where searching is done by unique key allotted to data users. Key management is the major drawback of this scheme. Wang *et al.*[10] suggested the conjunction searching, subset and range queries within the light of function encryption. Katz *et al.* suggested an extension in Wang's scheme by proposing the inner products predicate encryption and supporting disjunction and conjunction searching over encrypted data.

A technique is suggested by Kamara S. *et al.* [17] which is deployed through link list called an inverted index having file identifiers is maintained for every keyword. Each node of the list keeps positional information and next node decryption key. The nodes of every index are encrypted through random keys and are uploaded in an array randomly. It is possible to locate all the files which have the relevant keyword from the position and decryption key of the first node of an inverted index. Top k single keyword extraction schemes are suggested in the material to improve the efficiency of the above scheme.

Wenhai Sun *et al.* suggested a multi- keyword ranked search over encrypted data (MRSE) scheme which works upon similarity ranking. In this scheme searching index is formed upon the basis of vector space and term frequency. Multi keyword searching and search output ranking is done through searching index. By deploying tree structure upon index the search efficiency can be improved.

Formulation of Problem

Privacy and confidentiality of data is maintained in the searchable encryption schemes by providing the facility of keyword searching directly upon encrypted cloud data. Encrypted data is uploaded to the cloud by data users. Further, authentic data users can do keyword searching over encrypted cloud data. Number of things like storage, index and cryptography etc. are clubbed for the derivation of secure, efficient algorithms upon encrypted documents.

In the searching model of a cloud three entities are important. These are owner, cloud server and user. Owner of data encrypts the documents and corresponding index documents based on keywords by using algorithms of cryptography. Encrypted and index documents both are loaded at the server. For searching the encrypted documents upon cloud server, the encrypted keywords (trapdoors) are used.

Existing systems

Already available encryption schemes used for secure searching upon encrypted cloud data with the help of keywords. Multi keyword searching is supported by these schemes. In MRSE the measure of similarity (coordinate matching) has few limitations when evaluating the rank order of files. First thing is that any keyword appears in the file will show binary digit 1 for that file in the index vector irrespective of times of appearance. Naturally it is failing to reflect the significance of frequently appearing keyword in the file. Hence it takes no accountability of term frequency. Second thing is that any keyword appears in only single file is more significant than a keyword appearing in several files. Hence it takes no accountability of the term scarcity.

Furthermore, large files with multiple terms shall be favored by the process of ranking because they may include more terms than small files. Therefore, due to these drawbacks, the heuristic function for ranking (coordinate matching) is unable to provide much accurate searching output. Highly advanced measure of similarity might be adopted from the community of plaintext data extraction. Moreover, the complexity of searching in MRSE is linear to the quantity of files in the set of data, that becomes non desirable and non efficient for large amount of files

Proposed system

For the proposed system we have selected a data structure called B-tree for indexation and to identify the matching between data files and query. Particularly we have used the quantity of queried keywords appear in file to calculate the similarity of the file with the searching query. Every file is changed to a balanced B-tree as per the keywords and encrypted using Commutative Key RSA (CKRSA). When the data user wants to do searching he forms a trapdoor.Our goal is to use CKRSA algorithm and B-tree data structure for design and analysis of multi keyword rank search with searchable index tree. We have used CKRSA for designing a scheme based upon secured rank multi keyword search upon encrypted cloud data and then analyzed the performance upon B-tree based searchable index tree. MS-AZURE platform has been used by us for the emulation of proposed system and for studying its performance.

Encryption Module

Information in a document can be dynamically modified by not affecting the total search performance upon B-tree by using CKRSA. Again indexation of complete information is not required if the enciphered indexed information is updated. In the same way it is not required to again encrypt the documents of the database when the document is updated. It is a desired property as it minimizes the time of computing.

Commutative Key with RSA (CKRSA)

One among the optimum public key cryptography methods is RSA. Although maximum of available RSA schemes suffering from reordering issues and due to one sided encryption its robustness is very limited. Hence, for making the system more efficient and least complicated a protocol known as Commutative Key RSA (CKRSA) has been provided. The benefit of using this scheme is that if encryption and decryption would be done in same order the performance would not be affected. Standard method of making the communication private is encryption. Within the number of available cryptography schemes our system uses CKRSA. The mathematical model of the algorithm for deploying encryption is presented as follows:

RSA encryption algorithm can be stated as follows

1. Prime numbers: p ,q
1. 2.Compute n: n = p X q
2. Plain text : M < n
3. 4.Cipher text: C=Mecrsa (Mod n)

B-Tree

Figure – 1 shows the complex data structure called B-tree. Leaf and index vertex are included in the tree. Total leaf vertex depth is equal. Every index vertex includes links and keywords. Every vertex excluding root vertex in the B-tree of n order must have keys from n to 2n. Every vertex also includes (total keys +1) links to its child vertices. If root vertex is an index vertex then it is necessarily having minimum two children. The searching, insert and delete methods taking time complexity of only log.

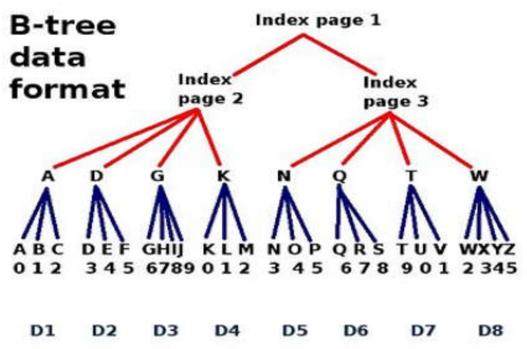

Figure 1 B-tree Data Structure Scheme of searchable encryption

Dynamic updation of data is possible through CKRSA but the total search performance upon B-tree will not be affected. Re-index of entire data is not required, if the modification is done on encrypted index. Moreover, when the modification is done on document, re-encryption of documents in database is not required. It is desired property because it minimizes time of computing.

By using CKRSA (public key encryption scheme) the owner of data first of all produces public key twin (SK, PK).

The owner of data keeps the key SK public and key PK private. Files {F| F1, F2……Fn} were enciphered through SK outputs in an encrypted (cipher) text {CP | CP_1, CP_2, ….CP_n}. The produced CP is saved in the database of cloud. The produced index on the basis of B-tree is enciphered through CKRSA, such that every produced word {K| k1, k2…….kn) by a file is index in the tree and enciphered through CKRSA. This outputs in a group of encryptions {E| E_1,E_2,..E_n} for every ej (1<= i <= m) will be defined as e_kj = CKRSA_Enc (SK, kj), where e_kj denoting enciphered keyword.

Structures of index for big sets of data could not be kept in primary memory. An alternate is usage of disk. Keeping data on disk needed distinct method. Using maximum branch of tree for decreasing the tree height is a possible solution. To implement the solution we apply complex data structure B-tree of n order for every file. For filling the vertex of tree the key are used from n to 2n.Vertex are minimum ½ filled of keys. Almost every vertex has the key. A links list is appended between keys. The links are used to traverse within tree. Generally a vertex of k keys has k+1 link.

Protocol-1, Protocol-2 and Protocol-3 are used to provide the design for forming and searching the index tree. Protocol-1 and Protocol-2 are used to form the index tree and Protocol-3 explains about the searching process upon index tree.

Protocol -1
Insert_Btree (Rt, K, Item_val)

```
Input: Rt (Root pgID of the B tree), K (the key) and the Item_val(Item value).
// Insertion at the time when B-tree have no Item_val

1. VERTEX = Read_Disk (Rt).
2.ifVERTEX_xisfull
(a) b = Page_ Allocate (),c = Page_ Allocate ().
(b) Find the mid item i saved in VERTEX_a.
 Shift the items to the left of item i into VERTEX_b.
 Shift the items to the right of i into VERTEX_c.
 If VERTEX_a is an index page,
 Then shift the child links of VERTEX_a asrequired.
(c) VERTEX_ a: child [1] = VERTEX_b, VERTEX_a: child [2]=VERTEX_b.
(d)Write_Disk(VERTEX_a);Write_Disk(VERTEX_b);
Write_Disk(VERTEX_c).
3. Endif
4. Full_Not_ Insert (VERTEX_a; K; Item_val).
```

Protocol-2
Full_Not_ Insert (VERTEX_a, K, Item_val)

```
Input: an in-memory page VERTEX_a of a B-tree, the key K and the value Item_val of a new item.

// This protocol appends when VERTEX_ a page have space.

// Insert the new Item_val into the sub-tree rooted by VERTEX_a.

1.If VERTEX_a is a leaf page
```

```
(i) Append the fresh Item_val into VERTEX_a, keeping Item_valin sorted order.
(ii)Write_Disk(VERTEX_a).
2.else
(i) Seek the child pointer VERTEX_a: child[j] whose key range includes key.
(ii) VERTEX_d = Read_Disk (VERTEX_a: child [j]).
(iii) if VERTEX_disfull VERTEX_b=Page_Allocate().
Find the mid item i saved in VERTEX_a.
Shift the items to the right of i into VERTEX_b.
If VERTEX_d is an index page, shift the child links as required.
Shift i into VERTEX_a. Add a right child link in VERTEX_a pointing to VERTEX_b
Write_Disk (VERTEX_a); Write_Disk (VERTEX_b); Write_Disk(VERTEX_d).
If (key < i.K), call Full_Not_ Insert (VERTEX_d; K; Item_val);
else, call
Full_Not_Insert (VERTEX_b;K;Item_val).
(iv)elseFull_Not_Insert(VERTEX_d;K; Item_val).
(v)endif
3. end if
```

Protocol-3
Query_ Search (Rt, trapdoor)

Input: Rt, trapdoor including keyword for searching.

Output: link to the files including keywords; if not available then NULL.

```
1. VERTEX_a = Read_Disk (Rt).
2. if VERTEX_a is an index vertex
(a) If an item I exist in VERTEX_a such that i:K = keyword, returni:value.
(b) Seek the child link a: child [j] whose range of key includes K.
(c)ReturnQuery_Search(VERTEX_a:child[j],K).
3. Else If item I exists in VERTEX_a such that i:K= keyword, outputi:value.
Else,outputNULL.
4.End If.
```

The Read_Disk in Protocol-1 study the relevant page from disk to memory and outputs the address in memory which is saved in vertex VERTEX_a. If the vertex VERTEX_a is totally filled then provide memory for two vertices and save the relevant locations in VERTEX_b and VERTEX_c. Seek the mid item save in VERTEX_a. Partition the vertex VERTEX_a by shifting the digit at left of mid item i in to VERTEX_b and right digit of mid item i to VERTEX_c. If VERTEX_a is indexed page then shift the links as required i.e. VERTEX_a: child [1] = VERTEX_b, VERTEX_a: child [2] = VERTEX_c. The VERTEX_a is upgraded to upper level. It raises the tree height. Write back total values at disk from memory by applying Write_Disk method. Else if VERTEX_a is having space then call Full _Not_ Insert function. Full _Not_ Insert function seeks the path through root to leaf, and appends the Item_val in to leaf. Using child link key range where the new item key available, the protocol follows the link. The vertices which are not filled along the path up to the depth of leaf the protocol loops in recursive way on each of those vertices. The item is appended at leaf depth.

Analysis of Performance

CKRSA is used to provide security to the proposed system. The service provider could not able to retrieve the set of files or index tree till the encrypted (private) key is not disclosed. CKRSA is also used to encrypt the trapdoor so to maintain the query and index level security the keywords will not be extracted from the trapdoor by the service provider. CKRSA is also used to protect the files so decryption of documents is impossible due to non availability of key for decryption, which leads to storage level security. The operations like searching, inserting and deleting of files must be supported for usability of database. For big enterprises the large size databases are used and could not be completely managed by memory.

The efficiency of searching can be improved with the help of B-tree (balanced) to form the index. Disk input output operations are minimized through B-tree by copy of page (data block) having number of records in one span in the memory. Hence efficiency of searching improves. Moreover, search in a non sorted and non indexed database results a time complexity of O(n), here n is quantity of keywords. We get the time complexity O(log n) for same data, if we use B-tree to index the data.

C# with visual studio framework is used to develop the system. Microsoft Azure has been used for implementation.

CONCLUSION

It is finally concluded that for encryption of documents and index tree CRSA algorithm has been used. Basis of index tree is basically B-tree. Due to commutative property of CRSA security of data has been increased and simultaneously privacy of data also improved. Dynamic updation of documents is possible here by the use of CRSA and the overall searching performance over B-tree will not be affected. Moreover, re-encryption of entire documents is not required if modification is applied in encrypted documents in the system proposed by us. Due to this desired property the time of computing is reduced. In future work, our intention is to check the performance of the system in the multiuser cloud environment.

References

1. Singhal A.,“Modern information retrieval:A brief overview”,IEEE Data Engg. Bulletin,vol.24,no.4,pp.35–43,2001.
2. Cloud Secu. Alliance,"Security Guidance for Critical Areas of Focus in Cloud Computing",2009.
3. Armbrust M. *et-al.*,“Above the Clouds: A Berkeley View of Cloud Computing”,Feb 2009.
4. Lauter K. *et-al.*,“Cryptographic Cloud Storage”,in RLCPS, January 2010,LNCS. Springer, Heidelberg.
5. Wenjing Lou *et-al.*,"Privacy Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data", Parallel and Dist. Sys., IEEE Trans. on vol.25,no.1,pp.222-233,Jan.2014
6. Brinkman R., “Searching in encrypted data”, in Uni. of Twente, PhD thesis,2007.
7. Li J. *et al.*,“Fuzzy Keyword Search Over Encrypted

Data in Cloud Computing”,Proc.IEEE INFOCOM’10,San Diego,CA,Mar.2010.
8. Perrig A., Song Dawn Xiaoding, Wagner D.,“Practical techniques for searches on encrypted data”,Secu. and Privacy,2000, Proc. of IEEE Symp.2000,pp.44-55,2000.
9. Persiano G. *et- al*.,“Public key encryption with keyword search”,in Proc. of EUROCRYPT,2004.
10. Wang C. *et- al*.,“Secure Ranked Keyword Search Over Encrypted Cloud Data”,Proc. ICDCS ’10,2010
11. Li M. *et- al*. “Authorized Private Keyword Search over Encrypted Data in Cloud Computing”, 31st International Conf. of Distributed Computing Sys., pp.383-92, 2011.
12. Mamoulis *et-al*.,“Secure knn computation on encrypted databases”, in Proc. of SIGMOD, 2009.
13. Varna A.L *et-al*.,“Confidentiality-Preserving Image Search:A Comparative Study Between Homomorphic Encryption and Distance-Preserving Randomization”, Access, IEEE,vol.2,pp.125-141,2014
14. Fu Zhangjie *et-al*.,"Multikeyword Ranked Search Supporting Synonym Query over Encrypted Data in Cloud Computing", IEEE Conf.,2013.
15. Wang Q. *et-al*.,"Security Challenges for the Public Cloud", IEEE Internet Computing, vol.16,no.1,pp.69-73,2012.
16. Shareef *et-al*., "Implementation of Secure Ranked Keyword Search by Using RSSE", *Int. Journal of Engg. Research & Tech*., Vol.2, Issue-3, March 2013.
17. Kamara S.*et-al*.,“Searchable symmetric encryption: improved definitions and efficient constructions”,in ACM CCS,2006.
18. Bloom B. H.,“Space/time trade-offs in hash coding with allowable errors”,Comm. of the ACM, vol.13, no.7,1970,pp.422–426.
19. Bairas S. and Buyrukbilen S.,“Privacy preserving ranked search on public key encrypted data”,in Proc. IEEE Int. Conf. on High Performance Computing and Comm.,November 2013.
20. Li. H, Sun W., Wang B., Cao N., Li M., Lou W., Hou and Y.T.,“Privacy-preserving multikeyword text search in the cloud supporting similarity-based ranking”, Proc. of the 8th ACMSIGSAC symp. on Info.,comp. and comm. secu.,ACM,pp.71–82.,2013.
21. Ramzan Z. and Gentry C.,“Single-database private information retrieval with constant communication rate”, in ICALP, pp. 803–815,2005.
22. Akki C.B. and Prasanna B.T,“A Survey on Homomorphic and Searchable Encryption Security Algorithms for Cloud Computing”, *Int.Journal of Info.Tech. and Comp. Sci.*,November 2014.
23. Goh E.J., “Secure Indexes”, Cryptology ePrint Archive, 2003, http://eprint.iacr.org/ 2003/216.
24. Golle P. *et-al*.,“Secure Conjunctive Keyword Search over Encrypted Data”,Applied Crypto.and Net. Secu., LNCS 3089,Springer-Verlag,pp.31–45,2004.

How to cite this article:

Pawan Kumr Tanwar *et al* (2018) 'Design and Analysis of Search Algorithm With B-Tree and Commutative Key Rsa For Dynamic Updation in Cloud Computing', *International Journal of Current Advanced Research,* 07(7), pp. 14414-14418.
DOI: http://dx.doi.org/10.24327/ijcar.2018.14418.2613

****