ENHANCE SECURITY AND ACTIVE UPDATION USING MULTI-KEYWORD RANKED SEARCH OVER CLOUD

R.Pavithra , Dr.R.Asokan

P.G Student, Department of Computer science and Engineering, Kongunadu College of Engineering and Technology .
Professor & Principal , Department of Computer science and Engineering, Kongunadu College of Engineering and Technology. Tamilnadu. India.

pavipsmn@gmail.com
principalkncet@gmail.com

----------***----------

Abstract - ***In 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, concerns of sensitive information on cloud potentially cause privacy problems. The security will be maintained at both the sides of data owner and data user and also provide fast retrieval of the documents. Proposed system, will focus on addressing data privacy issues to implement Asymmetric Searchable Encryption (ASE) allow retrieval of encrypted data over cloud. The KNN Algorithm is utilized to encrypt the index tree vectors and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. To sub-linear search time and deal with the deletion and insertion of documents flexibly. To supports dynamic update operations like deletion and insertion of documents in the cloud server through data owner.***

***Key Words*: keyword search, cloud computing, dynamic update, Greedy breathe first search, security.**

1. INTRODUCTION

In enterprise IT infrastructure which can organize the large amount of computing resources uses the concept of cloud computing. Those types of features will attracting features both individuals and organizations are motivated to outsource their data to the cloud instead of purchasing mechanism. More sensitive information (e-mails, health records, govt documents...etc)to be outsourced to the cloud. Cloud computing have the cloud service providers (CS)will keep the information to the users may access the owner sensitive information's without any authentication. For those type of security concern to improve the general approach to protect the data in confidential manner. In order to resolve the problem to include the authenticate checking for both the data owner and data user. Encryption scheme will used to encrypted data as both the sides and applied the keyword based searching. Some dynamic approach will be developed to support insert and deletions on document collection. This paper will propose the tree based search algorithm and also the keyword search using greedy breath first search and dynamic operation supporting secure ranked search.

1.1 Existing System

With the occurrence of cloud services progressively receptive data information are being integrated into the cloud server to guard seclusion and combat unsolicited accesses, susceptible data has to be encrypted earlier than outsourcing so as to offer end-to-end data privacy declaration in the cloud and beyond. However data encryption creates efficient data utilization a very challenging assignment specified that there could be a huge amount of outsourced data files. Besides, in cloud computing data owners may contribute to their subcontracted data with a huge number of users, who might give anything for only retrieve assured specific data files they are interested in during a given session. One of the most admired techniques to do so is through keyword-based investigates. Such keyword search system permits users to selectively related in plaintext investigate scenarios. Regrettably, data encryption, which restricts users ability to execute keyword search and further claims the shield of keyword privacy, creates the conventional Plaintext search techniques fail for encrypted cloud data.

1.2 Proposed system

This paper proposes a protected tree-based search format over the encrypted cloud information, which maintains dynamic operation and multi keyword ranked explore on the document set. Specially, the vector space replica and the generally-used "term frequency (TF) × inverse document frequency (IDF)" representation are collective in the index structure and query creation to offer multi keyword ranked search. In order to obtain high search efficiency, we construct a tree-based index constitution and offer a "Greedy Breath-first Search" technique based on this index tree. Due to the particular arrangement of our tree-based index, the proposed search system can flexibly attain secondary-linear search time and treaty with the insertion and deletion of documents. The protected kNN algorithm is used to query vectors and encrypt the index, and in the meantime make certain accurate significance score calculation between query vectors and encrypted index.

Advantages of Proposed System:
1.To offer not only accurate result ranking and multi-keyword query , but also active modernize on information data document set.
2.Enhanced searching competence with solitude preserving.

2. SYSTEM ARCHITECTURE

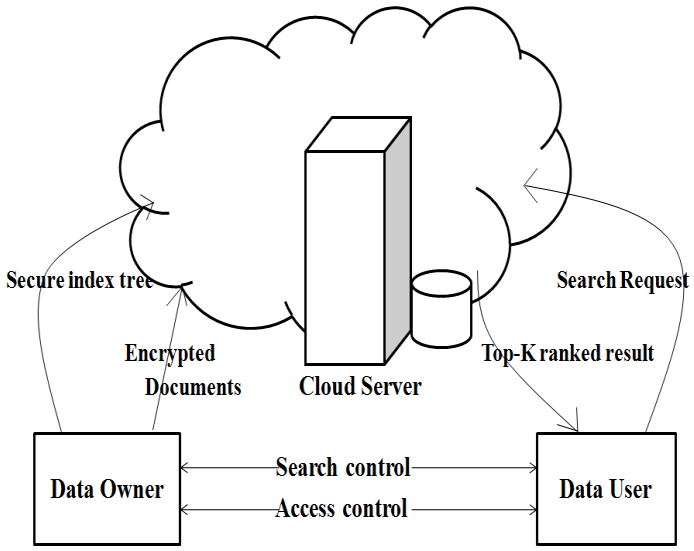

Fig -1: Architecture diagram

Cloud Data owner is insist to provide their details such as Name, Password, Email, Gender and Location. After Registration, a Unique key is generated and it sent to the Corresponding Email Id. Using this unique Key and Login Credentials, Data Owner can login their Account to File Upload .Data Owner can upload files such as Image, Documents such as .txt, .pdf etc…After uploading file to Database, Files are arranged in Tree Index manner using Tree index algorithm. After the File is uploaded user can update or delete the file. Every Deletion or Updation tree index gets re-arranged.

3. ALGORITHMS

3.1 Tree Index Algorithm

B-Tree is a self-balancing explore tree. In most of the supplementary self-balancing explore trees (like Red Black and AVL Trees), it is presumed that everything is in main memory. To appreciate employ of B-Trees, we have to think of enormous amount of information that cannot fit in main memory. When the number of keys is high, the data is examined from disk in the variety of blocks. Disk access instance is extremely high evaluated to main memory access time. The major idea of using B-Trees is to diminish the number of disk accesses. the majority of the tree processes (search, insert, delete, max, min..etc) require O(h) disk accesses where h is height of the tree. B-tree is a fat tree. Height of B-Trees is kept low by putting maximum achievable keys in a B-Tree node.

Properties

- If p is an internal node, it also holds $n[p] + 1$ pointers $c_1[p], c_2[p], \ldots, c_n[p]+1[p]$ to its children. Leaf nodes have no children, so their c_i fields are indeterminate.
- All leaf has the related intensity, which is the tree's height h.
- Every node p has the next fields:
  - a. $n[p]$, the number of keys currently stored in node x,
  - b. the $n[p]$ keys themselves, accumulated in non falling order: $key_1[p] \leq key_2[p] \leq \cdots \leq key_n[p][p]$, and
  - c. $leaf[p]$, a Boolean value that is TRUE if p is a leaf and FALSE if p is an interior node.
- The keys $key_i[p]$ separate the ranges of keys stored in every subtree: if k_i is any key accumulated in the subtree with root $c_i[p]$, then

$k_1 \leq key_1[p] \leq k_2 \leq key_2[p] \leq \cdots \leq key_{n[x]}[p] \leq k_n[p]+1$.

- There are lower and upper bounds on the number of keys a node can included. These bounds can be stated in terms of a connected integer $t \geq 2$ described the ***smallest amount degree*** of the B-tree:
  - a. Every node additional than the root should have as a minimum t - 1 key. Every interior node other than the root thus has least amount t children. If the tree is nonempty, the root should have as a minimum single key.
  - b. all node can hold at most $2t$ - 1 keys. Therefore, an inner node can have at most $2t$ children. We say that a node is ***full*** if it contains exactly $2t$ - 1 keys.
- The simplest B-tree occurs when t = 2. each inner node then has moreover 2, 3, or 4 children, and we have a ***2-3-4 tree.*** In put into practice, however, much superior values of t are characteristically used.

Search B-tree Algorithm:

B-TREE-SEARCH(p, k)

```
1  i ← 1
2  while i ≤ n[p] and k ≥ key_i[x]
3     do i ← i + 1
4  if i ≤ n[p] and k = key_i[p]
5     then return (p, i)
6  if leaf [p]
7     then return NIL
8     else DISK-READ(c_i[p])
9          return B-TREE-SEARCH(c_i[p], k)
```

3.2 Build index tree Algorithm

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

3.3 Greedy Breadth First Algorithm

Breadth First Search algorithm(BFS) traverses a graph in a breadth wards action and utilizes a queue to keep in mind to get the next vertex to start a search when a dead end happens in a few iteration.

Rules of Breadth First Search:

- ❖ Call adjacent unvisited vertex. Mark it visited. Display it. Insert it in a queue.
- ❖ If no adjacent vertex establish, take away the first vertex from queue.
- ❖ Repeat Rule 1 and Rule 2 pending queue is empty.

Algorithm:

IF root is NULL

Then create root node

Return

If root node exists then

Compare the data with note data

While until insertion position is located

If data is greater then node data

Go to Rightsubtree

Else

Go to Leftsubtree

End while

Insert data

Endif

4.MODULES

4.1 Login and Registration

- ➢ Used to checking the cloud user
- ➢ Register user only use this application for the security purpose.
- ➢ All the information about the user are stored in the cloud server and maintain.
- ➢ New user registers to the use this application.

4.2 File uploading

Once the registration completed owner candidates can upload the documents in the format name with the relevant keyword for the purpose of handling the searching process. Data Owner can upload files such as Image, Documents such as .txt,.pdf etc..After uploading file to Database, Files are arranged in Tree Index manner using Tree index algorithm. The file uploading finally will show the document upload successful message to the owner.

4.3 Encrypted index tree construction

In the process of index construction, to 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.

4.4 Search process

Greedy breath first search will provide the sub-linear search to fast access. They do the keyword based search in level by level manner in the index tree.

4.5 Dynamic update operation

To supports dynamic update operations like deletion and insertion of documents in the cloud server through data owner.

5. PERFORMANCE EVALUATION

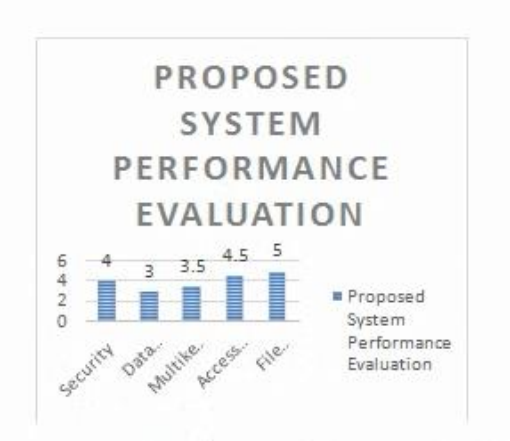

Chart -1: Performance implementation

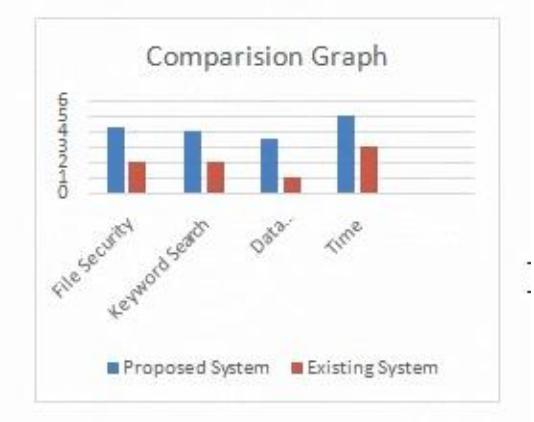

Chart -2: Comparison

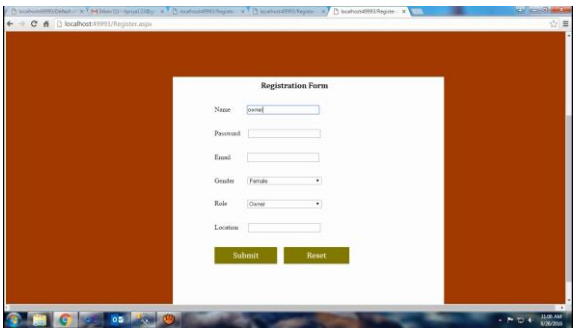

Fig-2: Registration

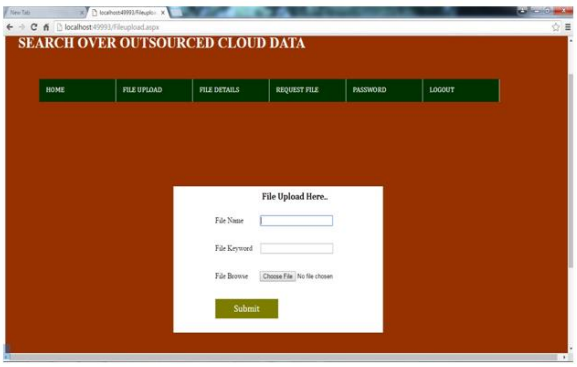

Fig -3: File uploading

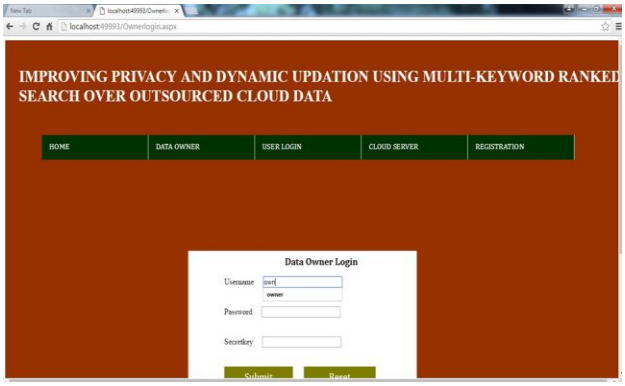

Fig 3: Data Owner login

6. CONCLUSION

In this paper, for the earliest time we define and solve the difficulty of multi-keyword ranked explore over encrypted cloud informatics data, and establish a diversity of solitude necessities. Among a variety of multi-keyword semantics, we choose the competent similarity compute of "Coordinate matching," i.e., as many matches as probable, to successfully capture the significance of outsourced documents to the uncertainty keywords, and use "inner product similarity" to quantitatively estimate such similarity determined. Then, we give two enhanced MRSE schemes to realize a variety of stringent privacy necessities in two dissimilar hazard models. We also examine some additional Enhancements of our ranked search method, as well as supporting extra explore semantics, i.e., TF × IDF, and dynamic data processes. Thorough study investigating privacy and efficiency assurances of proposed method is given, and experiments on the real-world information set demonstrate our proposed schemes initiate low overhead on both communication and computation .

REFERENCES

[1] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. secur., 2013, pp. 71–82.

[2] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in Proc. ACM Conf. Comput. Commun.Secur., 2012, pp. 965–976.

[3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in Proc. IEEE INFOCOM, Apr. 2011, pp. 829–837.

[4] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li "Verifiable Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Ranking" IEEE ,VOL. 25, No- 11, Nov2014.

[5] David Cash et al. "Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation" Rutgers University,University of California, Irvine ,IBM Research.

[6] S. Kamara, C. Papamanthou " Parallel and Dynamic Searchable Symmetric Encryption" Microsoft Research, senyk@microsoft.com

[7] 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.

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

[9] 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.

[10] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 50–55, 2009.

[11] S. Kamara and K. Lauter, "Cryptographic cloud storage," in RLCPS, January 2010,LNCS. Springer, Heidelberg.

[12] A Singhal, "Modern information retrieval: A brief overview," IEEE Data Engineering Bulletin, vol. 24, no. 4, pp. 35–43, 2001.

[13] I. H. Witten, A. Moffat, and T. C. Bell, "Managing gigabytes:Compressing and indexing documents and images," Morgan Kaufmann Publishing, San Francisco, May 1999.

[14] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proc. of S&P, 2000.

[15] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, 2003, http://eprint.iacr.org/2003/216.

[16] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in Proc. of
ACNS, 2005.