p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 03
March 2017

# An approach for Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data

[1] Dr. K. Praveen Kumar, [2] K. Sree Ranganayaki
[1] Associate Professor, School of Electrical Engineering & Computing, Department of Computing, Adama Science &Technology University, Adama, Ethiopia
[2] M.Tech Student, Chaitanya Institute of Technology and Science, Warangal, India

**Abstract:** Due to the cumulative admiration of cloud computing, more and more data owners are interested to outsource their data to cloud servers for great expediency and condensed cost in data management. Nonetheless, sensitive data should be encrypted before outsourcing for privacy requirements, which obsoletes data utilization like keyword-based document retrieval. We use security as a parameter to create trust. Cryptography is one way of forming trust. Searchable encryption is a cryptographic method to provide security. In literature numerous researchers have been working on developing efficient searchable encryption schemes. In this paper we discover some of the operative cryptographic techniques based on data structures like CRSA and B-Tree to enhance the level of security, henceforth trust. We tried to contrivance the search on encrypted data using Azure cloud platform.

**Keywords**- Searchable encryption, multi-keyword ranked search, dynamic update, cloud computing.

## I. INTRODUCTION

Cloud computing is a conversational phrase used to express a variety of dissimilar types of computing ideas that occupy large number of computers that are connected through a real-time communication network i.e Internet. In science, cloud computing is the capability to run a program on many linked computers at the same time. The fame of the term can be recognized to its use in advertising to sell hosted services in the sense of application service provisioning that run client server software on a remote location. Cloud computing relies on sharing of resources to attain consistency and financial system alike to a utility (like the electricity grid) over a network. The cloud also centres on maximize the effectiveness of the shared resources. Cloud resources are typically not only shared by multiple users but as well as dynamically re-allocated as per demand. This can perform for assigning resources to users in dissimilar time zones. For example, a cloud computing service which serves American users during American business timings with a specific application (e.g. email) while the same resources are getting reallocated and serve Indian users during Indian business timings with another application (e.g. web server). This mechanism must take full advantage of the use of computing powers thus decreasing environmental damage as well, since less power, air conditioning and so on, is necessary for the same functions. The expression "moving to cloud" also explains to an organization moving away from a traditional CAPEX model i.e buy the devoted hardware and decrease in value it over a period of time to the OPEX model i.e use a shared cloud infrastructure and pay as you use it. Proponents maintain that cloud computing Permit Corporation to avoid direct infrastructure costs, and focus on projects that distinguish their businesses as an alternative of infrastructure.

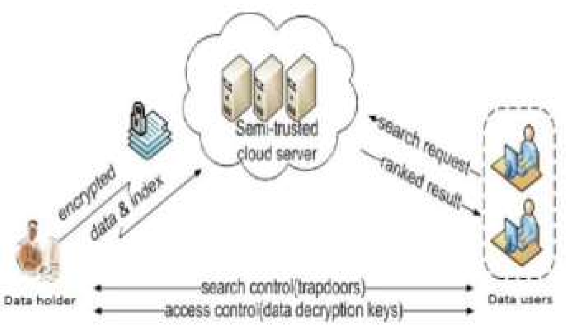
Semi-trusted cloud server
encrypted data & index
search request
ranked result
Data holder
Data users
search control(trapdoors)
access control(data decryption keys)

Fig. 1. Architecture of the search over encrypted cloud data

Proponents also maintains that cloud computing permit schemes to get their applications should run faster, with better manageability and less maintenance, and enable IT to more quickly adjust resources to meet random and changeable business demand.

## II. RELATED WORKS

Kui Ren, Cong Wang, and Qian Wang, In this paper, Cloud computing speaks to today's most energizing figuring outlook change in data innovation. Not withstanding, security also, security are seen as essential difficulties to its wide appropriation. Here, the creators diagram a few basic security challenges and influence promote examination of security solutions for a dependable open cloud condition. cloud computing is the most up to date term for the since quite a while ago imagined vision of registering as an utility. The cloud provides advantageous, on demand network access to a centralized pool of configurable processing assets that can be quickly sent with incredible productivity and insignificant administration overhead. With its unprecedence favorable circumstances, distributed computing empowers a principal outlook change in how To convey and convey processing administrations that is, it makes conceivable registering outsourcing with the end goal that both people and undertakings can abstain from conferring extensive capital expenses when buying and overseeing programming and equipment, as Toll as dealing the operational overhead therein.[1]

S. Kamara and K. Lauter, In this paper, To consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. To describe, at a high level, several architectures that combine recent and nonstandard cryptographic primitives in order to achieve our goal.

To survey the benefits such an architecture would provide to both customers and service providers and give an overview of recent advances in cryptography motivated specifically by cloud storage. [2]

C. Gentry, In this paper, To propose the first fully homomorphic encryption scheme, solving a central open problem in cryptography. Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key { i.e., given encryptions E(ml); … ; E( ) of ml;… ; mt, one can efficiently compute a compact cipher text that encrypts f(ml;… ; m ) for any efficiently computable function f. This problem was posed by Rivest et al. in 1978. Fully homomorphic encryption has numerous applications. For example, it enables private queries to a search engine { the user submits an encrypted query and the search engine computes a succinct encrypted answer without ever looking at the query in the clear. It also enables searching on encrypted data { a user stores encrypted files on a remote file server and can later have the server retrieve only files that (when decrypted) satisfy some Boolean constraint, even though the server cannot decrypt the files on its own. More broadly, fully homomorphic encryption improves the efficiency of secure multiparty computation. Our construction begins with a somewhat homomorphic boostrappable" encryption scheme that works when the function f is the scheme's own decryption function. To then show how, through recursive self-embedding, bootstrappable encryption gives fully homomorphic encryption. The construction makes use of hard problems on ideal lattices.[3]

O. Goldreich and R. Ostrovsky, In this paper, To present a theoretical treatment of software protection. In particular, To distill and formulate the key problem of learning about a program from its execution, and reduce this problem to the problem of on-line simulation of an arbitrary program on an oblivious RAM. To then present our main result: an efficient simulation of an arbitrary (RAM) program on a probabilistic oblivious RAM. Assuming that one-way functions exists, To show how one can make our software protection scheme robust against a polynomial-time adversary who is allotted to alter memory contents during execution in a dynamic fashion. To begin by discussing software protection.[4]

D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, In this paper, To study the problem of searching on data that is encrypted using a public key system. Consider user Bob who sends email to user Alice encrypted under Alice's public key. An email gateway wants to test whether the email contains the keyword \urgent" so that it could route the email accordingly. Alice, on the other hand does not wish to give the gateway the ability to decrypt all her messages. To define and construct a mechanism that enables Alice to provide a key to the gateway that enables the gateway to test whether the word urgent" is a keyword in the email without learning anything else about the email. To refer to this mechanism as Public Key Encryption with keyword Search. As another example, consider a mail server that stores various messages publicly encrypted for Alice by others. Using our mechanism Alice can send the mail server a key that will enable the server to identify all messages containing some specific keyword, but learn nothing else. To define the concept of public key encryption with keyword search and give several constructions.[5]

## III. SEARCHABLE ENCRYPTION SCHEME

To design an efficient multi-keyword searchable encryption scheme based on public key cryptography, we included the following modules.

**Encryption Module:** By using CRSA, data in a file can be updated dynamically without affecting the overall performance of searching on B-tree. If the encrypted indexed data is modified, re-indexing for the whole data is not needed. Similarly there is no need of re-encrypting the files in the database whenever the file is modified. This is a desirable feature as it reduces the computation time.

Data owner first generates secret and public key pair (EK, DK) using a standard public-key encryption scheme ie CRSA. Then owner makes the public key DK public and keeps the secret keys EK private. Documents {D | D1, D2,…, Dn} are encrypted using EK resulting in a ciphertexts {C | C1,C2,….Cn}. The generated C is stored in cloud database.

The constructed index based on B tree is also encrypted using CRSA, i.e each derived keywords {W| w1,w2,….wn}from a document is indexed in a tree and encrypted using CRSA. This results in a set of encryptions {e| e1,e2,..en} where each ej (for ) is defined as E_wj = CRSA_Enc (EK, wj), where E_wj denotes encrypted keyword.

**Index Module:** Index structures for huge datasets cannot be stored in main memory. Disk is a possible alternative. Storing it on disk requires different approach. The solution is to use more branches to reduce the height of the tree. For this we used B-tree data structure for each document. B-tree is a data structure of order n. The nodes are filled from n to 2n keys. Nodes are always at least half full of keys. The keys are within each node. A list of pointers is inserted between keys. These pointers help to navigate through tree.

In general, a node with k keys has (k+1) pointers. The design for creating and querying the index tree can be given by Algorithm-1, Algorithm-2 and Algorithm-3. Algorithm-1 and Algorithm-2 are used to create an index tree and Algorithm-3 describes how search can be performed on index tree.

**Algorithm-1**

```
Btree_insert (root, Key, Object_value)
Input: root pageID of a B-tree, the key and the value of an object.
//Inserts when Object_value doesn't exist in a B-tree
1. NODE = Disk_Read (root).
2. if NODE_x is full
(a) y = Allocate_Page(), z = Allocate_Page().
(b) Locate the middle object o stored in NODE_x.
    o Move the objects to the left of object o into NODE_y.
    o Move the objects to the right of o into NODE_z.
    o If NODE_x is an index page,
    o Then move the child pointers of NODE_x accordingly.
(c) NODE_ x: child [1] = NODE_y, NODE_x: child [2] = NODE_z.
(d) Disk_Write (NODE_x); Disk_Write (NODE_y); Disk_Write (NODE_z).
```

```
3. end if
4. Insert_Not_Full (NODE_x;  Key; Object_value).
```

**Algorithm-2**

```
Insert_Not_Full (NODE_x,  key, Object_value)
Input: an in-memory page NODE_x of a B-tree, the key and the value Object_value of a new object.
// This algorithm inserts when page of NODE_ x is not full.
// Insert the new Object_value into the sub-tree rooted by NODE_x.
1. if NODE_ x is a leaf page
(a) Insert the new Object_value into NODE_x, keeping Object_values in sorted order.
(b) Disk_Write (NODE_x).
2. else
(a) Find the child pointer NODE_x:  child[i]  whose key range contains Key.
(b) NODE_w  = Disk_Read  (NODE_x:  child [i]).
(c) if NODE_w  is  full
    o  NODE_y  = Allocate_Page ().
    o  Locate the middle object o stored in NODE_w.
Move the objects to the right of o into NODE_y.
    o  If NODE_w is an index page, move the child pointers accordingly.
    o  Move o into NODE_x. Add a right child pointer in NODE_x  pointing to NODE_y
    o  Disk_Write (NODE_x);  Disk_Write (NODE_y);  Disk_Write (NODE_w).
    o  If (Key < o. key), call Insert_Not_Full(NODE_w;  KEY; Object_value); else, call
Insert_Not_Full(NODE_y;  Key; Object_value).
(d) else Insert_Not_Full(NODE_w;  Key;
Object_value).
(e) end if
3. end if
```

**Algorithm-3**

```
Search_Query (root, trapdoor)
Input: root, trapdoor containing keyword to be searched.
Output: pointer to the documents containing the keywords; NULL  if non-exist.
1. NODE_x  = Disk_Read  (root).
2. if NODE_x  is an index node
(a) If there is an object o in NODE_x such that o: key = keyword, return o: value.
(b) Find the child pointer x: child [i] whose key range contains key.
(c) Return Search_Query(NODE_x:child[i],  key).
3. else If there is an object o in NODE_x such that o:key = keyword, return o:value.
Otherwise, return NULL.
4.end if.
```

**Search Module:** Searching a B-tree is like searching a binary tree. Here instead of making a binary branching decision at each node, we make a multiway branching decision according to the number of the node's children. Let's suppose cloud server has received n encrypted documents of this form, so that it now holds a set of encrypted documents {C|c1,C2,…,Cn}. Now, if user wants to retrieve the documents with keyword , he just needs to generate a secret trapdoor encrypted using CRSA i.e Enc_CRSA (w1, w2, ..). The trapdoor containing the encrypted keywords is sent as token to the server. The server then uses this trapdoor to match the encrypted keywords in index tree node. If match found stores the pointer to that document in encrypted database. The search continues for other encrypted keywords. The following Algorithm-3 gives the stepwise information about how search will be done on B-Tree.

**Ranking Module:** In large databases, it is quite likely that the keyword might be matching with more number of documents. It is cumbersome for a user to decrypt and go through all the documents. Therefore there is a need for ranking the documents based on their relevance to the keywords. In our scheme we used (TF * IDF) to rank the documents. TF is the term frequency i.e. occurrence of keywords in a document and IDF is inverse document frequency i.e. total number of documents divided by number of documents containing the keyword. Similarity measure is used to find the rank based on relevance. For this, we maintain two vectors one for storing TF weight and other to store IDF weight.

**Platform Used:** Microsoft Azure is a cloud service provider. It provides storage as a service to the customers. Azure architecture contains roles, i.e. the worker role and the web role as shown in Figure 3. The web role is used for designing UI, whereas worker role is used to run background asynchronous applications. The workers in the B-tree provide search encryption services which support the multi-keyword search application.

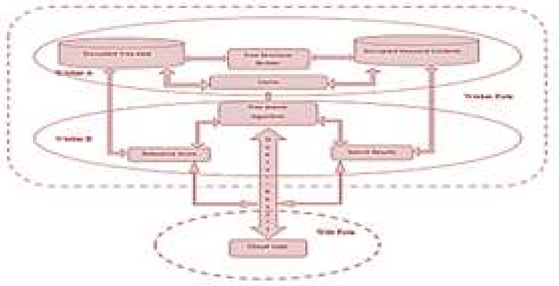
Figure 3: Architecture of searchable encryption scheme in Azure

## IV. CONCLUSION

In this paper, a secure, efficient and dynamic search scheme is proposed, which supports now not simplest the correct multi-keyword ranked search but additionally the dynamic deletion and insertion of documents.Using CRSA, information in a document may be updated dynamically without affecting the overall performance of searching on B-tree. In our proposed system, if encrypted records is changed, re-encrypting for the entire data isn't always desired. This is a proper task because it reduces the computation 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] 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, StanfordUniversity, 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] I.H. Witten, A. Moffat, and T.C. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann Publishing, May 1999.

[7] D. Song, D. Wagner, and A. Perrig, “Practical Techniques for Searches on Encrypted Data,” Proc. IEEE Symp. Security and Privacy, 2000.

[8] E.-J. Goh, “Secure Indexes,” Cryptology ePrint Archive, http:// eprint.iacr.org/2003/216. 2003.

[9] Y.-C. Chang and M. Mitzenmacher, “Privacy Preserving Keyword Searches on Remote Encrypted Data,” Proc. Third Int’l Conf. Applied Cryptography and Network Security, 2005.

[10] R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, “Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions,” Proc. 13th ACM Conf. Computer and Comm. Security (CCS ’06), 2006.

## BIOGRAPHY

**Dr.K.Praveen Kumar** received the PhD in Computer Science & Engineering in 2015, M.Tech in Software Engineering from Kakatiya Institute of Technology & Science Warangal, Telangana, India in 2010 and B.Tech in Information Technology from Kakatiya Institute of Technology & Science Warangal, Telangana, India 2007. Presently working as Assistant Professor in Computer Science Department at Adama Science and Technology University, Adama, Ethiopia.

**K Sree Ranganayaki** pursuing M.Tech in Computer Science & Engineering at Chaitanya Institute of Technology and Science. Her Area of Interest is Network Science and Cloud Computing.