Turkish Journal of Computer and Mathematics Education Vol.12 No.9 (2021),1538-1546

Research Article

Intelligent Multi-Keyword Ranked Searchable Security Algorithm

K. Madan Mohan[1], Dr. B V Ram Naresh Yadav[2]

[1]Asst.Professor, Dept of Computer Science and Engineering,CMR Institute of Technology, Kandlakoya (V). Medchal (M). Hyderabad. Pin : 501401,TS.Research Scholar, Dept of Computer Science and Engineering, JNTUH, Kukatpally, Hyderabad,500085,TS.
[2]Associate Professor,Department of Computer Science &EngineeringJNTUH College of Engineering Jagtial (JNTUHCEJ) ,Nachupally, Jagtial,505 501,TS.
[1]madan.keturu@gmail.com, [2]bvramnaresh@gmail.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 20 April 2021

Abstract: - enterprises are moving their valuable information to the cloud because of the advantage that storage as a service brings. Reliability is at the heart of the cloud service and relationship. We use the sense of security when establishing trust. Trust is established through cryptography. Searchable encryption is a method to provide security. Literature research workers have been studying searchable encryption schemes. We will focus on advanced techniques that leverage CRSA and B-Tree to enhance level of trust. We conducted a search on encrypted data using the Azure cloud platform.

Keywords: Searchable Encryption, Multi keyword, CRSA, B tree, Azure

1. Introduction

Cloud computing is one of the ways that computing can be done. Here the computing resources are shared by a large number of people. The benefits of cloud can be extended to corporations and organizations in general. An important feature of cloud is its storage. Cloud service makes hardware and software resources available as a null form. Many companies like Google Drive, Dropbox, iCloud, SkyDrive, Amazon and Azure provide storage services. Cloud Computing's major challenges have been security and privacy concerns. Cloud providers have used a variety of security measures to protect themselves from hacking and other computer issues. These approaches will not sufficiently protect your data because of small degree of transparency. Because two cloud user and two cloud providers are in separate trusted domain, some data may become vulnerable. Before storing highly sensitive data in cloud, that data needs to be encrypted. Data encryption assures the information security and confidentiality of data. To protect the privacy, we need to design an algorithm that works on encrypted data [13].

A lot of researchers are focused on looking in encrypted data. Searchers may perform a single keyword search or a keyword request search. Searches in huge database may reveal many matching documents and by-products. It makes it hard for cloud users to locate their needed documents. Various search strategies like ranking and sorting are also acceptable solutions to search documents. The economical encryption technology helps cloud users such as pay-as-you-use model. The researchers advanced several effective ways of encrypting electronic documents while using keywords of different significance. In the searchable encryption literature, the performance of cryptographic schemes is determined by speed and low-level complexity. Computation time is the length of time required to complete a task such as searching a keyword, generating trapdoor and so on. CPU utilization is measured in a terms of resource allocation measured in time.

In this research work, I investigated the security issues in cloud storage and proffered a resolution for the same. We contributed as follows:

1. For the first time, we define the problem of secure ranking of keywords over encrypted data in the cloud, and provide a search algorithm to resolve this privacy-related issue.
2. It is proved that ranked searchable asymmetric searchable encryption scheme coupled with CRSA and B-tree has "practically perfect" security.
3. Experimental results clearly demonstrate the effectiveness and efficiency of this.

In Section II of this paper, previous studies are discussed. Section III discusses problem definition. Section IV provides our search strategies. Security and performance assessments are presented in Section V and VI, and finally, in Section VI, the paper ends with some solutions for future papers.

2. Literature survey

The encryption of data in cloud is an efficient way to safeguard sensitive data. In the game of searching, search efficiency gets low. In literature, research work has not effectively explored the complicated issues. This underperformance may allow sensitive information to leak out. Song et al proposed the practical cryptographic searchable method based on cryptography. This scheme allows encryption. Therefore, a keyword searches are available in the same manner. The drawback of this system is that it will disclose the word frequency. Goh and colleagues developed a technique for secure mail sorting by using pseudorandom functions and randomized Bloom filters. Bosch has worked to modify the concept given by Goh et al. This method may generate false positives. In Chang et al's schemes, indexes are built for each document. Key feature of this approach is that the number of words in the file is not disclosed. This scheme is less efficient and does not support arbitrary updates with new words. This system permits many different kinds of searches with a single encrypted query. This is impractical scheme. This was proposed by Curtmola et al and Kamara et al first time for symmetric searchable encryption (SSE), which is then expanded by the notion of dynamic symmetric searchable encryption (DSSE). All these methods focus on a single theme or keyword search.

The first implementation of public key search (PEKS) was implemented by Boneh et al. The scheme suffers from a problem related to the construction of trapdoor encryption systems. Baek et al, Rhee et al improved Boneh's scheme hardness. Baek's idea plays an active role in playing up keyword search. Public key encryption algorithms are very complex computationally and therefore they are slow. In Yang et al scheme, the encrypted text is searched by individual users by utilizing a secret key allocated for them. This scheme has some room for improvement with its management. Boneh et al. discussed functional encryption and connected subset search queries. Katz et al. scheme has updated version of Boneh's scheme and cover both conjunctions and disjunctions search on encrypted data.

Internet searching techniques are numerous. These techniques cannot do fuzzy searches. Using similar/trending terms is investigated with fuzzy search [8]. The primary goal of this project is to provide services to users for searching the data on the Internet with the help of fuzzy keyword. This work was proposed by Curtmola et al. [16], where the inverted index is implemented using linked list with document identifiers. Every node has a position and key information for the next node. The initial set of nodes are encrypted with random keys and are randomly inserted into a linked list. Knowing the position and decryption key of the first node of an inverted index is enough to find every document having the same keyword. Besides, this is something commendable in Islam as according to the legal rule "prevention is better than cure" and "there should be neither harm nor reciprocating injury".

Privacy preserving multikey word search on encrypted data has become a central area of work in the cloud computing sector. In [11], a model is constructed that effectively solves the problem of effective secure keyword search over encrypted cloud data. Here, they suggest an existing cryptographic primitive, order-preserving public-key cryptography (OPSE). Although it does not accept the multi-word keyword methodology, it does not use a diverse variety of approaches to determine scores.

Bricks et al [19] presents the first ranked results found from search queries over encrypted data. The linear scanning of the documents can be replaced by parallelizing operations and by shortening computation to the extent possible. The scheme makes keyword information about each document abstract by using a Bloom filter, and the catalog is encoded into a tree structure. Client will do query processing and retrieve all the dependencies. The topic is shielded from an internet service provider or cloud provider by using an effective private information retrieval protocol. This increases the performance of framework by running queries simultaneously on different CPUs.

Wenhai Sun and many others have proposed an MRSE scheme that works on a similarity-based ranking. Search engine rankings are decided on the basis of the frequency of occurrence and position of different terms in the phrase. Search indexes are used to rate the search result. Search performance is increased because of the inclusion of index dependent structure. Future work may also include work on a multi-keyword semantic search over the encrypted data. It will be better if you will allow multiple keywords in the query. In this paper, privacy-preserving keyword search over data stored in the cloud is discussed. Thus, LMIS lacks integrity check and strong privacy protection mechanism. Consider the issue of cloud data encryption using data synonym search. Here author uses symmetric encryption and uses binary search tree for indexing.

Unfortunately, it is not possible to guarantee that the data stored in the cloud are completely private. There are a number of research dilemmas which require further investigation. Thus, we choose to approach the problem.

3. Formulation of the problem

Searchable Encryption (SE) schemes are very useful techniques to maintain confidentiality and privacy for individuals. Users can store their data on cloud storage. Moreover, this was endorsed.

people can privately perform keyword searches on the data in the cloud. Multiple domains like cryptography, indexing,

storage of data, computer network etc. are involved in developing secure, efficient, security enhanced algorithms over encrypted files. A stable search mechanism in the cloud generally includes a data owner, data user and the cloud server. Data owner secures the data on his or her computer with any common cryptographic algorithms. The encrypted and indexed files are both transferred to the cloud storage. The trap doors (encrypted keywords) are used to encrypt data.

A. System Model

Our system consists of three entities: a data owner, a data user and a cloud server.

1. Data owner encrypts the data files in the cloud for confidentiality. The users are allowed to access certain documents as per specified access rights. The access authorization is a two-state variable: yes or no. The data owner builds an index tree and encrypts it with CRSA.

2. The cloud server contains the data files that are encrypted and the file tree is encrypted too. It accepts the keywords in order to generate the most important results.

3. Data users may scan for encrypted files encrypted by keywords in cloud (trapdoor). The aim of applying an encrypted key is that even the cloud service provider itself must not be able to infer the contents of data stored in files.

Figure 1: Searchable Encryption Architecture using CRSA

B. Threat Model

The threat model for our search system adopts an "honest but curious" cloud server, that is, the "honest" cloud server follows the specification of the protocol, but it is "curious" to infer and analyse data (including indexes) in its storage and message flows obtained during the protocol in order to learn additional details.

C. Design Goals

The approach suggested addresses the following

1. The document/file search must be entirely safe and the Cloud Server cannot infer any document/file contents.
2. The results of this search have to be graded accordingly

Our device design should achieve the following safety and performance guarantee to enable confidential, searchable encryption for effective use of outsourced and encrypted cludge data in the above-mentioned model.

Specifically, we have the following goals: 1) Ranked keyword search: to explore various frameworks for developing successful ranked search schemes based on the current searchable encryption framework; 2) Security guarantee: to prevent cloud server from knowing the plaintext of either the data files or the searched keywords, and achieve the “as-strong-as-possible” security strength compared to existing searchable encryption schemes; 3) Efficiency: above goals should be accomplished with minimal communication and computation overhead.

Existing systems

Current scanning schemes [6] [38] allow the user to search for coded data with keywords in a safe way. These techniques help multi keyword search. When used to determine the order of document grade, the similarity test "coordinate correspondence" in MRSE [6] has many disadvantages. First, the term frequency is not taken into account so that any keyword in a document appears on the index vector, irrespective of how many of its appearances it is, as a binary value 1. Obviously, the value of a keyword always appearing in the text does not reflect this. Secondly, the word scarcity is not taken into consideration. Normally a keyword is more important than a keyword in several documents which appear in only one document. Moreover, the ranking method prefers

long documents that have several terms, because they undoubtedly include more terms than short documents. Therefore, the heuristic ranking function, "coordinate matching," cannot generate more accurate search results because of these constraints. More advanced similarity measure should be adopted from plaintext information retrieval community. On the other hand, MRSE's search complexity is linear to the amount in the data set of documents, which is unwanted and ineffective when large quantities of documents are present.

Proposed system

For our system, we choose the B-tree as indexing data structure to identify the match between search query and data documents. In order to determine the similarity of the document to the search query, we particularly use inner data correspondence, i.e., the number of query keywords appearing in the document. Each document is translated according to keywords to a balanced B-tree, and is encrypted with CRSA. Any time the user wants to find, a trapdoor for the keywords will be built. Our objective is to develop and evaluate the performance of a search system for multiple keywords using Commutative RSA and B-tree searchable index tree data structures.

We have developed a scheme that is based on a protected classification of several keywords over encrypted CRSA cloud data. We also analysed its performance over the searchable index tree based on B-tree. The authors analysed the RSA algorithm's output on B tree in [6][38]. We used the Azure Platform of Microsoft to emulate the framework proposed

and to research its efficiency.

D. Preliminaries

Commutative Encryption (CRSA): One of the ideal public key encryption approaches is the RSA cryptography system. Its overall robustness is nevertheless restricted by encryption in one way, and reordering problems affect most of the existing RSA systems. Therefore, an approach called Commutative RSA was proposed to make this method less complicated and more efficient. The order in which encryption was performed would not affect decryption in this scheme if it were carried out in the same order. Encryption is the traditional way to make private contact. Our system follows the switching RSA algorithm with the many cryptographic approaches. A pseudo algorithm presented below explains the mathematical scheme for performing this encryption.

B- Tree: A B-tree is a knowledge structure as shown in Figure 2. The tree comprises index and leaf nodes. Both leaf nodes are on equal footing (same depth). Keywords and pointers are included in each index nodes. Each node in a B-tree with order n must contain keys between n and 2n keys except root node. Each node contains also (key number + 1) indicators on its child nodes. The node must have at least 2 children if the root node is an Index node. It takes logarithmic time for insertion, deletion, search operations.

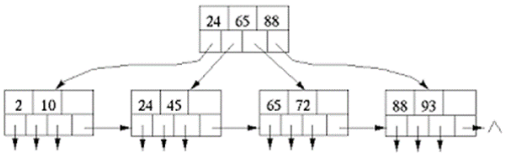

Figure 2: B tree data structure

4. Searchable encryption scheme

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

Encryption Module: Data can be modified in a file dynamically by the use of CRSA without affecting the overall search efficiency of B-tree. Re-indexing of the entire data is not required if the encrypted indexed data is updated. Similarly, when the password is updated, it is not important to re-encrypt the files in the folder. This is a desirable function since the calculation time is reduced. The database owner uses a standard public key encoding method (CRSA) as the first step towards creating a secret and public key pair (EK, DK). Then the owner makes DK public and keeps EK private hidden keys. The {D|D1, D2, Dn} documents are encrypted and the ciphertexts {C |C1, C2,.Cn} are generated using EK. C is created in the cloud database.

The B-built index is also encrypted via CRSA. That means each derived {W|w1, w2} keyword is indexed to a tree and encrypted by CRSA from a text. This results in a set of encryptions {e| e1, e2, en} in which the E wj = CRSA Enc (EK, wj, E wj denotes encrypted keywords) is specified by each ej (for).

Index Module: index structures cannot be saved in main memory for large datasets. Disk is a potential

substitute. It needs a different method to store on disk. To reduce the height of the tree, the alternative is to use more branches. For this reason, for each text, we used B-tree data structure. B-tree is an order n data structure. Keys from n to 2n are filled in. At least half of the knots are always keying full. The keys are inside each node. Between keys is a list of pointers inserted. These indicators allow you to browse the forest. A node with k keys usually has pointers (k+1). ALGORITHM-1, ALGORITHM-2 and ALGORITHM-3 can be used to construct and query the index tree. To build the index tree, ALGORITHM-1 and ALGORITHM-2 are used and ALGORITHM-3 explains how search on the index tree can be carried out.

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.

- Move the objects to the left of object o into NODE_y.
- Move the objects to the right of o into NODE_z.
- If NODE_x is an index page,
- 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

- NODE_y = Allocate_Page ().
- Locate the middle object o stored in NODE_w. Move the objects to the right of o into NODE_y.
- If NODE_w is an index page, move the child pointers accordingly.
- Move o into NODE_x. Add a right child pointer in NODE_x pointing to NODE_y
- Disk_Write (NODE_x); Disk_Write (NODE_y); Disk_Write (NODE_w).
- 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

The Disk_Read in ALGORITHM-1 reads the corresponding page from disk to memory and returns the location in memory that gets stored in node NODE_x. If the node NODE_x is full, allocate memory for 2 nodes and store the corresponding addresses in NODE_y and NODE_z. Find the middle object stored in NODE_x. Split the node NODE_x by moving the values to the left of middle object o in to NODE_y and right values of middle object o to NODE_z. If NODE_x is index page then move the pointers accordingly i.e. NODE_x: child [1] = NODE_y, NODE_x: child [2]=NODE_z. The NODE_x is promoted to higher level. This increases the height of the tree. Write all the values back to disk from memory by using Disk_Write operation. Else if NODE_x is not full then call Insert_Not_Full function. Insert_Not_Full function finds the path from root to leaf, and inserts the Object_value in to the leaf. Using the key range of the child pointer where the key of new object exists, the algorithm follows the pointer. The algorithm loops recursively on each of those nodes which are not full along the path till leaf level. The Object is inserted at the leaf level.

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.

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.

As the input and searches for the keywords in the cloud database, the ALGORITHM-3 uses trapdoors and root. The Disk Read will read the root page of the disk to memory, which is stored in the NODE x node. If the index node for the NODE x is, the keyword match for the trapdoor is verified. If found returns the corresponding document pointed by the node. In the event that the search is based on the keyword, NODE x child is pushed by pointers. Recursively, the search continues. If NODE x represents a leaf, then give it back.

Document indicator if otherwise NULL succeeds the quest.

Rating Module: It is extremely possible that the keyword fits more documents in large databases. A user must decrypt and read all documents. It is tedious. Therefore there is a need for rating the documents based on their importance to the keywords. We used (TF*IDF) to identify documents in our scheme. TF is called the frequency of keywords occurring in a document and IDF is inverse frequency, i.e., total number of documents, separated into a number of keyword documents. To find the rank based on significance, a calculation of similarity is employed. We hold two vectors, one for TF weight storage and Store weight for other IDF.

Used Platform: The cloud provider is Microsoft Azure. It provides customers with storage as a facility. Azure design involves functions like the position of the worker and the Web function, as shown in Figure 3. The function of the Web is used for user interface design, while the role of the worker is used for running asynchronous applications. The workers in the B-tree provide search encryption services which support the multi-keyword search application. The workers are defined as where is search provided by the worker. The encrypted index tree is built by tree builder feature using encrypted keyword contents (worker A) (worker A). Cloud (web role) users enter search keywords. The search algorithm for the B-tree dependent tree searches the encrypted keywords in the index bar. The search results can be obtained using an index tree query and an algorithm for tree search. The search results are ranked by search algorithm and index tree (worker B) as mentioned above in the rank module. Figure 3 shows the azure cloud framework architecture of a quest for encrypted worker data and web role.

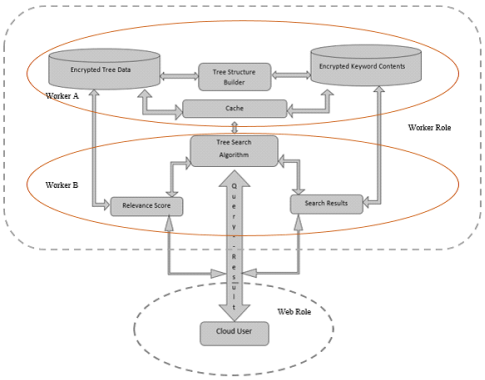

Figure 3: Architecture of searchable encryption scheme in Azure

5. Performance analysis

CRSA ensures the protection of the designed device. Where the cloud service providers cannot deduce index tree or document collection until the private key is secretly encrypted. The provider is not able to show in the trapping door the keywords preserving trust at index level and query level since trapdoor is encrypted by CRSA. Documents are also secured in cloud storage, as CRSA encrypts documents. It is very difficult to decrypt documents without using the decryption key to provide storage security. Databases must facilitate operations, for example search, deletion and insertion of data in order to be useful and usable. The databases are huge for large organisations and can't be stored in memory. We can improve search effectiveness by using balanced B-trees to create the index of data. B-tree minimizes the disk I/O (disk read and disk write) by copying a block of data (page) containing several records at a time into memory. This increases the efficiency of the hunt. Searching a non-sorted database without indexing would asymptotically have the worst run-time of O(n), where n represents the number of keywords. If BTree indexes the same data, the same logarithmic search operation is performed, i.e., O. (log n).

Outcomes Analysis: Multi-keyword search for privacy has been built on the basis of encrypt cloud data. On Visual Studio 2010 Framework 4.0 with C# the presented system model has been developed. With the Microsoft Azure cloud platform, the entire framework was developed and implemented.

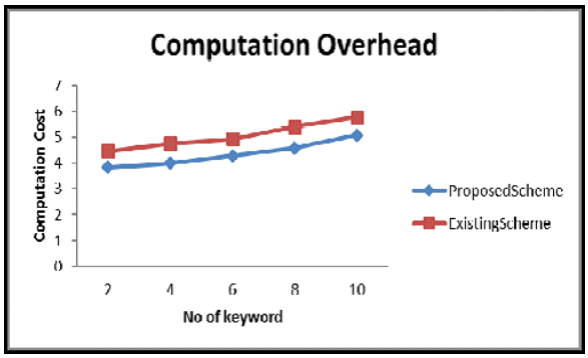

Figure 4: Computation Overhead

Figure 4 shows the overhead measurement in seconds based on keyword number. In this report, we compared our proposed system's efficiency with the proposed RSA-based system [15]. The results show clearly that overhead calculations with CRSA, compared to the RSA-based method, are low even in 10 keywords [15]. For instance, it takes approximately 4.5 seconds to find two keywords for RSA-based system, compared to only 4 seconds for our proposed system. The search measurement costs in both schemes rise linearly. But from Figure 4 it is evident that our proposed CRSA based scheme performs better even under increased number of keywords.

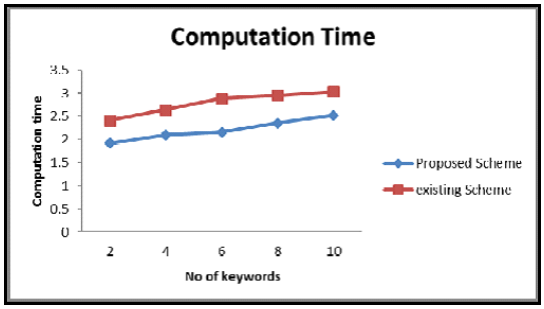

Figure 5: Time Comparison

The diagram above in Figure 5 compares our proposed system's search time to the RSA-based system in seconds. For two keywords quest, the time taken by the RSA based scheme is approximately 2.5 seconds, while our proposed method takes approximately 0.5 seconds less. As the number of keywords for search increases, the search time in both systems is also increasing linearly. But it is found that CRSA's method works better. Thus, it is obvious that the B Tree CRSA encoding algorithm is better than the RSA and B tree combinations as an index tree.

6. Conclusion and future work

This work uses CRSA asymmetric algorithm for encrypting data files and index tree based on B-tree. By its commutative nature, CRSA increases data protection and enhances data privacy. Data in a file can be dynamically modified with CRSA without affecting the overall search efficiency of the B-tree. If encrypted data is updated on our proposed framework, it is not appropriate to re-encrypt the entire data. This is a desirable function since the calculation time is reduced. The future work will focus on using Elliptic Curve Cryptography (ECC) encryption technique for better results. We also plan to examine the behaviour of the multi-user world of our proposed system(s).

References

A. M. Armbrust et al., 'Above the Clouds: A Berkeley View of Cloud Computing,' Feb 2009.
B. S. Kamara and K. Lauter, 'Cryptographic cloud storage,' in RLCPS, January 2010, LNCS. Springer, Heidelberg.
C. A. Singhal, 'Modern information retrieval: A brief overview,' IEEE Data Engineering Bulletin, vol. 24, no. 4, pp. 35–43, 2001.
D. Cloud Security Alliance, 'Security Guidance for Critical Areas of Focus in Cloud Computing,' http://www.cloudsecurityalliance.org, 2009.
E. R. Brinkman, 'Searching in encrypted data,' in University of Twente, PhD thesis, 2007.
F. Ning Cao; Cong Wang; Ming Li; Kui Ren; Wenjing Lou, 'Privacy- Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data,' Parallel and Distributed Systems, IEEE Transactions on , vol.25, no.1, pp.222,233, Jan. 2014
G. Dawn Xiaoding Song; Wagner, D.; Perrig, A., 'Practical techniques for searches on encrypted data,' Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on ,doi: 10.1109/SECPRI.2000.848445 vol., no., pp.44,55, 2000
H. J. Li et al., 'Fuzzy Keyword Search Over Encrypted Data in Cloud Computing,' Proc. IEEE INFOCOM '10 Mini-Conf., San Diego, CA, Mar. 2010.
I. M. Li et al., 'Authorized Private Keyword Search over Encrypted Data in Cloud Computing,' 31st Int'l. Conf. Distributed Computing Systems, 2011, pp. 383–92.
J. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, 'Public key encryption with keyword search,' in Proc. of EUROCRYPT, 2004.
K. C. Wang et al., 'Secure Ranked Keyword Search Over Encrypted Cloud Data,' Proc. ICDCS '10, 2010
L. Wenjun Lu; Varna, A.L.; Min Wu, 'Confidentiality-Preserving Image Search: A Comparative Study Between Homomorphic Encryption and Distance-Preserving Randomization,' Access, IEEE, vol.2, no., pp.125,141, 2014
M. W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, 'Secure knn computation on encrypted databases,' in Proc. of SIGMOD, 2009.
N. K. Ren, C. Wang, and Q. Wang, 'Security Challenges for the Public Cloud,' IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.

O. Zhangjie Fu et al, 'Multikeyword Ranked Search Supporting Synonym Query over Encrypted Data in Cloud Computing', IEEE Conference, 2013.

P. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, 'Searchable symmetric encryption: improved definitions and efficient constructions,' in ACM CCS, 2006.

Q. P. Naresh, K. Pavan kumar, and D. K. Shareef, 'Implementation of Secure Ranked Keyword Search by Using RSSE,' International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 2 Issue 3, March – 2013.

R. S.Buyrukbilen and S.Bairas, 'Privacy preserving ranked search on public key encrypted data,' in Proc. IEEE International Conference on High Performance Computing and Communications (HPCC), November 2013.

S. B. H. Bloom, 'Space/time trade-offs in hash coding with allowable errors,' Communications of the ACM, vol. 13, no. 7, 1970, pp. 422– 426.

T. C. Gentry and Z. Ramzan, 'Single-database private information retrieval with constant communication rate,' in ICALP, pp. 803– 815.2005.

U. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H., 'Privacy-preserving multikeyword text search in the cloud supporting similarity-based ranking,' Proceedings of the 8th ACMSIGSAC symposium on Information, computer and communications security, ACM, pp. 71–82.2013.

V. Prasanna B.T, C.B. Akki, 'A Survey on Homomorphic and Searchable Encryption Security Algorithms for Cloud Computing,' Communicated to International Journal of Information Technology and Computer Science, November, 2014.

W. Prasanna B.T, C.B. Akki, 'A Comparative Study of Homomorphic and Searchable Encryption Schemes for Cloud Computing,' Communicated to International Journal of Communication Networks and Distributed Systems, November, 2014.

X. Prasanna B.T, C.B. Akki, 'A Survey on Challenges and Security Issues in Cloud,' Presented in conference presented in Conference on Evolutionary Trends in Information Technology, May 20-22 2011, at Visvesvaraya Technological University, Belgaum, Karnataka.