International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-4 Issue-6, August 2015

Achieving Efficient Ranked Multi keyword Search over Outsourced Cloud Data

Lekshmi Balakrishnan, Soja Salim

Abstract— With the arrival of cloud computing, sensitive data are being centralized into the cloud. Data owners are allowed to store their complex data from their local systems to the public cloud. For the protection of data privacy, sensitive information is encrypted using any of the cryptographic algorithms before outsourcing it to cloud. Data owners outsource their data in encrypted form onto the cloud which makes effective data utilization based on plain text keyword search a challenging task. There are many traditional searchable encryption schemes which allow users to search over the encrypted data but most of them only support single keyword search. In this paper, multikeyword ranked searching technique is used which supports multikeyword searching. This method uses the concept of coordinate matching which captures the similarity between query and documents. This paper proposes a ranking method which uses the principle of coordinate matching and adds additional security functions to protect the data stored in the cloud. The proposed scheme introduces low overhead on computation and communication.

Index Terms—Cloud computing, Coordinate matching, Searchable encryption, Ranked Search.

I. INTRODUCTION

Cloud Computing is a pervasive technology which provides storage facility to users and also allows users to use applications without installing the applications in their local systems. Data owners are allowed to outsource their files in the cloud so that they can access their files from any systems. Also by storing their files in the cloud, data owners no longer have to worry about storage facilities and their maintenance. Cloud computing [1] has several advantages which includes cost efficient, unlimited storage, flexibility etc. It is flexible in the sense that it has the ability to meet the demands quickly. However the main issue in cloud computing is its security. Data stored in the cloud is not secure since they are vulnerable to external threats. Also since sensitive information is stored in cloud, cloud servers are not allowed to read the stored information. So in order to protect the sensitive information, many security approaches have been adopted. One such approach is to encrypt the sensitive information before storing them to cloud [2]. Thus storing documents in encrypted form in cloud improves security by allowing only authorized users to view the original documents.

As already mentioned, sensitive data has to be encrypted before outsourcing them to cloud for data privacy and to avoid unsolicited accesses. Storing encrypted data in cloud make effective data utilization a challenging task. There are many searching methods [3] which focus only on plain text search. But such methods are useless in case of encrypted cloud data. There are many traditional searchable encryption algorithms that allow users to search over the encrypted data. But most of them supports only single keyword query. Usually searching with a single keyword over large collaborative collection of cloud data will retrieve thousands and thousands of documents, thus making it difficult for users to find the most relevant documents matching their interest. In other words, we can say that single keyword query is inefficient since they results in too many answers. To solve this issue, multikeyword query is supported in this paper. Each keyword in the search query narrows down the results. To retrieve the top most relevant files, ranked search is performed by using a ranking function. For each document matching the search query, a score is calculated using this ranking function and based on this score, documents are ranked. Ranked search greatly support efficient data retrieval by returning the matching files in a ranked order based on some relevance criteria (for example, keyword occurrences), thus making one step closer towards deployment of privacy-preserving data hosting services in cloud computing. While uploading a file onto cloud, a unique id is given to the file. A set of keywords along with its file id is stored in an index file. Both the index file and the document collection is stored in the cloud in the encrypted form. When any user submits a query (single or multikeyword) to the cloud server, the cloud server first searches the index file and finds the matching documents. Then using the ranking function, it will compute the score and then ranks the documents. The ranked list is returned to the user so that the user can select a document and then decrypt it .Only authorized users are allowed to decrypt the document. In this method ranking is done using coordinate matching. Coordinate matching uses the principle of inner product similarity, which is a measure of number of occurrence of a keyword in the document divided by total number of keywords in the document.

II. RELATED WORKS

Cloud computing allows data owners to outsource their information onto the cloud. There are many traditional searching algorithms to search over the encrypted cloud data. Efficient Similarity Search technique [4] allows to users to search over the encrypted data. It not only finds the exact matching documents but also finds the documents that are similar to search query. This method first computes the similarity between the document and the query word. If the computed similarity is greater than or equal to a threshold value, the document is retrieved.

Manuscript published on 30 August 2015.
* Correspondence Author (s)
Lekshmi Balakrishnan*, Computer Science, University of Kerala, SCT College of Engineering, Trivandrum, India.
Soja Salim, Computer Science, University of Kerala, SCT College of Engineering, Trivandrum, India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

For easier searching, an index is constructed from the document collection using Locality Sensitive Hashing (LSH). This method is free from typographical errors thus supporting efficient keyword search. This method performs poor when the number of data items is large.

"Practical techniques for searches on encrypted data" proposed by D. W. D. Song and A. Perrig [5] describes a sequential scanning search technique which searches on the encrypted data without losing data confidentiality. Since it just stores the data on the cloud in encrypted form, the server doesn't know anything about the sensitive information thus providing security. Probabilistic searching is used to find whether a search query word appears in the document or not. It will return all those matching document along with the location where the word appears. This method has the disadvantage that searching involves high time complexity.

"Encrypted phrase searching in the cloud "proposed by S. Zit-trower and Cliff C. Zou[6], provides a new method in the field of encrypted searching .It performs proximity ranked keyword searching over encrypted data stored in the cloud. In this method, a third party ie a trusted client side server can only do encryption and decryption. This trusted server generates the searchable index which stores the keywords along with their location. Then the trusted server uploads the encrypted documents and index to untrusted cloud. Using proximity ranked searching method, it ranks the documents based on a function which is directly proportional to the distance between the keywords in the search query. In this method, if the trusted server is compromised, the whole security is broken.

Jiadi proposed a searching technique that uses Two- round searchable encryption (TRSE)[7]. The TRSE uses the concept of vector space model and homomorphic encryption. In this method, a user can submit encrypted query and a trapdoor are created. The trapdoor is submitted to the cloud server. The cloud server searches on the index file and computes the document scores and then returns the scores to user. The document score is also encrypted. In the second round, user decrypts the scores and then requests the documents having higher scores. In this method user performs the ranking part. Most of the computing part is done on server side so that information leakage is eliminated ensuring security. But this method suffers from high communication overhead since searching is completed in two rounds.

Public-Key Encryption with Keyword Search (PEKS)[8] proposed by D. Boneh introduces the first predicate encryption scheme. User first creates a trapdoor for his query word. The server then matches with its existing keywords and returns the matching file. The file is in encrypted form. There exists a secure channel between owner , server and user. Through that channel, the necessary authentication is made by user so that he can decrypt the file. The main limitation of this method is that trapdoor contains meaningful keywords which helps the cloud server to do a one-to-one mapping between trapdoor and keyword. Thus user privacy is not considered here. Also this method only supports single keyword searching and does not support multikeyword searching.

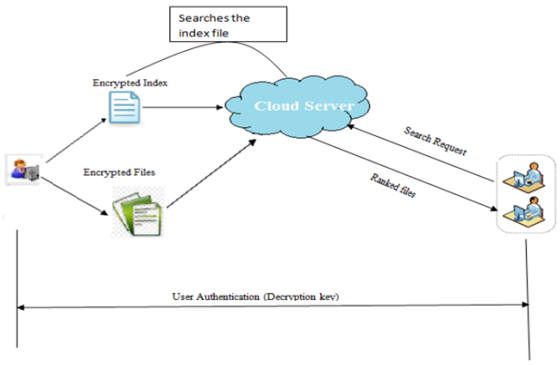

Fig.1 : Architecture of our model

III. PROBLEM FORMULATION

A. *System architecture.*

Consider a cloud data system as shown in the Fig.1. Cloud data system consists of data owner, cloud server and data users. Data owners upload their documents onto the cloud. For each uploaded document, a unique id is assigned. While uploading the document, a set of keywords is also given by the owner. The set of keywords are stored in an index file corresponding to the document id. After creating the index file, both index file and the documents are encrypted and then stored in the cloud. To search for a matching document against search query, the cloud server first performs a search on the searchable index and then performs a ranked search over the matching encrypted documents and then returns the top most relevant documents. User then selects a document and sends a request to the owner for the decryption key. The owner can either accept or reject the request .If the owner accepts the request, and then key is send to the user so that user can decrypt the document.

B. Threat Model

We usually consider cloud server as "honest-but-curious" [9] in our model. Cloud server is considered as honest since it correctly follows the designated protocol specification. But cloud server is curious in the sense that it try to learn additional information from the document collection and from user search query. Actually cloud server has no intention to modify or delete any information but still it behaves as honest but curious model .Here we consider two threats model:

- **Known Ciphertext Model:** The cloud server is only supposed to know the encrypted documents and the index file which the owner uploads to the cloud.
- **Known Background Model:** The cloud server can use previous searching history and the document frequency to know what can be accessed from an encrypted document.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

C. Design Goals

Ranked search is performed for effective data retrieval. The main goal of this method is to retrieve the top most documents matching the user search query. Also it is necessary to prevent the cloud server from learning anything from the user search query. It means that cloud server is not able to see what a user is searching. For that purpose, when a user submits a query to the server, a trapdoor is created. Using that trapdoor, the server searches on the index file. By creating trapdoor privacy preserving is achieved.

D. Notations

- D – Set of documents, denoted by D= {D_1 , D_2 , D_3 ,……, D_n }.
- C-Set of encrypted documents (cipher texts) denoted by C= {C_1, C_2, C_3,……, C_n }.
- W –Set of keywords, denoted by W = { w_1, w_2, w_3, … w_m }.
- I – Searchable index.
- $\overline{W}$ - Set of keywords in the search request denoted by $\overline{W}$ ={ $\overline{w}_1$, $\overline{w}_2$, $\overline{w}_3$,……, $\overline{w}_i$ }.
- R –ranked list if documents.

E. Coordinate matching.

For effective data retrieval, ranked search is performed. Documents are ranked based on some score calculated from a ranking function. In this method coordinate matching [10] is used. Score is computed using the below equation:

$$Score(D_i) = \frac{\text{Number of occurrence of keywords in document } D_i}{\text{Total number of keywords in that document } D_i} \text{ --(1)}$$

IV. FRAMEWORK

This method includes two phases: Setup phase and retrieval phase:

A. Setup:

The data owner initializes the secret key and other parameters. The data owner also generates the index file by executing Build index () which generates a searchable index for the document collection. Then the data owner encrypts the document collection and index file and uploads them to cloud. The data owner also generates a key and distributes the key to authorized users to create the trapdoor. The following steps are given:

i. Key Generation: Key is generated randomly .This Key is used for encryption and decryption.
ii. Encryption: The documents are encrypted using AES algorithm using the above key.
iii. Build Index: An index file is generated to store the important keywords related with the documents. This index file is used to make the searching easier.

B. Retrieval:

The user submits their query to the cloud server. In order to protect the privacy of the query, the user first generates a trapdoor using the key from the owner. The trapdoor is then sent to the server. When the cloud server receives the trapdoor, server performs a ranked search and then returns the ranked list to the user. The following steps are given:

i. Trapdoor Generation: The trapdoor is created using the keywords from the search query.
ii. Search index: The cloud server searches the index file for the matching documents.
iii. Ranking: The documents are ranked and the ranked list is returned to the user.
iv. Decryption: The documents are decrypted using AES algorithm.

V. IMPLEMENTATION DETAILS

Data owner can upload the documents in any cloud providers such as Openstack, CloudStack, IBM cloud, Smartfile etc. As already mentioned, to protect the sensitive information's, documents are encrypted before uploading. For easier searching, an index file is created .The index file stores the important keywords from the document collections. The implementation of the existing work is given below:

A. Key Generation.

The keys are generated randomly. The keys are used for encrypting and decrypting the documents. The keys are stored in the database.

B. Encryption and Decryption.

RSA (Rivest-Shamir-Adleman) algorithm is used for encryption and decryption. RSA is a public (asymmetric) key algorithm. Two different keys are used for encryption and decryption. The public key is used for encryption and private key is used for decryption. For giving more security, RSA algorithm is replaced by AES algorithm which is explained in next section.

C. Build Index.

Before uploading the document, the whole document is scanned and all keywords (words) are extracted. The extracted keywords are stored in an index file with the document id. For each uploaded document, an unique id is created. The important keywords are mapped to this document id.

D. Searching.

A user can submit a query to the server. The query consists of single or multikeyword. But in order to protect the privacy of the user query, a trapdoor is created. This trapdoor is submitted to the cloud server. The cloud server upon receiving the trapdoor, searches on the index file to find the matching documents. Then the matching documents are ranked using coordinate matching. Here the ranking function focuses on the number of times the keywords occurring in a document. Based on it, score is computed for each matching document. Based on the score, the documents are ranked and the ranked list is returned to the user.

Algorithm 1: Search Index

Input: A set of keywords K in the user search query and the searchable index file R.

Output: A list of document id's matching user query along with their temporary scores.

1. Extract all keywords from the keywod set K=k_1,k_2,k_3,…k_n.
2. Scan the index file R.
3. For each id in index file.
4. If any of the extracted keywords matches with those stored in the corresponding row id(Rid).
5. Then
6. Find the number of keywords in the search query that matches with the stored keyword in each row.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

$Temp_{score}[i]$= N , where N is the number of keywords that matches with the stored keywords.

The algorithm 1 describes how server performs searching when a user submits a query. Server first searches on index file to find the matching documents against the query. After that, the server will find out how many keywords in the search query occur in each matching document. A temporary score for the matching documents are calculated.

The algorithm 2 describes steps to rank the documents in ascending order and outputs the ranked list to user.

Algorithm 2: Ranking

Input: A list of document id's (Fid) matching user query along with their temporary score.

Output: A ranked list of matching documents.

1. **For each** matching document of Fid
2. Count the number of occurrence of keywords. Let it be N.
3. Calculate score of the document.

   $Score(id)=Temp_{score}[id]+\frac{N}{T}$,where T=total number of keywords in the document.

4. **End for**
5. Rank the documents in ascending order based on the calculated scores.

E. Decryption.

The user can decrypt the document using the private key. The user requests the owner for the private key .The owner can either reject or accept the request. If accepted, the private key is sent to the user's mail. The user can then get the key from his mail and decrypt the document.

VI. Improving Security

The main issue in cloud computing is security. Even if the documents are stored in the encrypted form in the cloud, it is less secure. Suppose an attacker gets the private key, then the attacker can then easily decrypt the document. Also an attacker may upload some illegal files using data owner's identity. So for additional security, usb authentication is used. The data owner while registering stores the usb id also to the database. Every usb has a unique id. Then upon login using his username and password, he must insert the same usb he has used during the registering process. After inserting the usb, the unique id is extracted and then matches it with that in the database. If it matches then the owner completes the authentication process and is allowed to login and upload files. In the proposed method RSA algorithm was used. But AES algorithm is more efficient than RSA since AES consumes less encryption time compare to RSA. So in the suggested method rsa algorithm is replaced with AES. AES (Advanced Encryption Standard) is a symmetric key encryption algorithm which uses a single key for both encryption and decryption.

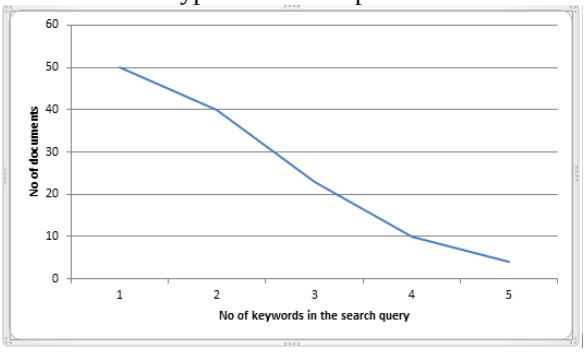

Fig 2: Single keyword vs Multikeyword

In the suggested method, coordinate matching is also modified to make the searching more efficient. In the proposed method, ranking function focuses only on the number of times a keyword occurs in the document. Consider an example: A user submits a query which consists of three keywords. Suppose two matching documents are found. Also imagine that the documents contain equal number of keywords. The first document contains all the three keywords but only occurs once. And the second document contains only two keywords but occurs several times. According to our ranking function, the second document will be given higher rank .But note that the second document only contains two keywords in the search query. This problem must be solved. So in our suggested method, ranking function is modified. It will give preference to those documents which have all keywords in the search query. Modified equation is given below:

Let N_i be the number of keywords in the search query occurring in document D_i

$$Score(D_i) = N_i + \frac{\text{Number of occurrence of keywords in document } D_i}{\text{Total number of keywords in that document } D_i} \quad \text{-----(2)}$$

VII. Result Analysis and Evaluation

Searching with a single keyword retrieves too many answers, many of them may be irrelevant. In order to retrieve top relevant documents, users must be allowed to search with more keywords to express their interest more accurately. Fig. 2 explains how the number of matching documents decreases as number of keywords in the search query increases, thus returning the top most matching documents.

Consider that about 500 documents are uploaded in the cloud. Suppose a user submits a query which consists of multiple keywords. For example, suppose the user request is as follows:

No of keywords in the search query: 4

A. Existing MRSE method:

Assume that 4 documents matched the query i.e. 4 documents contains either all keywords in the query or some keywords in the query. For simplicity assume that total numbers of keywords in all documents are equal, say 40.Using (1), score of matching documents are calculated.

Table .I

	D_1	D_2	D_3	D_4
No of occurrence of document D_i	10	20	5	4
Score(D_i)	.25	.5	.125	.1

From the calculated score, the documents are ranked as follows:

Table .II

Rank	Document	Score
1	D_2	.5

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

2	D_1	.25
3	D_3	.125
4	D_3	.1

But this method has a disadvantage: Suppose document D_2 consists of only three keywords in the search query, documents D_3 and D_1consists of only two keywords and document D_4 consists of all four keywords in the search query. So the user is expected to get the documents matching all keywords in his search query. But by above method Document D_2 is given higher rank and D_4 is given lower rank. So, ranking function has to be modified to overcome this problem.

B. Modified method:

First the server will find the matching documents.

No of matching documents: 4

Table III

	D_1	D_2	D_3	D_4
N_i (No of keywords in the search query appearing in the document D_i)	2	3	2	4

Using (2),score of the documents can be calculated. The document D_4 is given higher rank followed by D_2 .Observe that D_1 and D_3 contains two keywords. Here ranking function is applied.

Table IV

	D_1	D_2	D_3	D_4
No of occurrence of document D_i	10	20	5	4
Score(D_i)	.25	.5	.125	.1

The documents are ranked as follows:

Table V

Rank	Document	Score
1	D_4	4.1
2	D_2	3.5
3	D_1	2.25
4	D_3	2.125

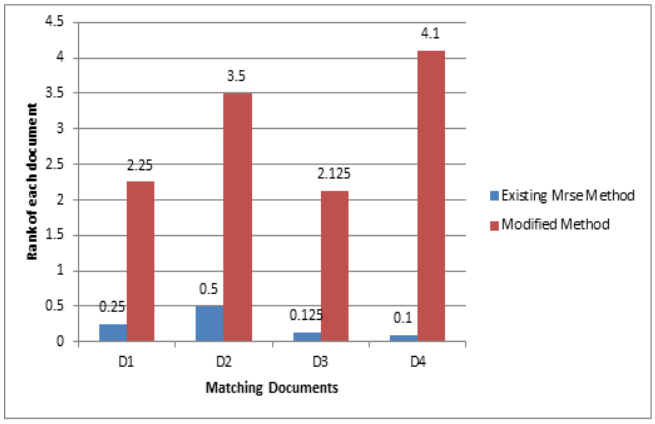

Figure 2: Existing Method v/s Modified Method

VIII. CONCLUSION

Large volumes of information are stored into the cloud nowadays. To protect privacy, information is encrypted before storing them to cloud. This makes searching a challenging task. There are various searching techniques to overcome this issue. This paper proposes a searching method which uses the concept of coordinate matching. This method extends the concept of coordinate matching for supporting more efficient multikeyword searching. Documents are ranked based on a ranking function. Ranked search enhances the system performances since it return only the top matching documents. Also the cloud server is unable to know what a user is searching thus preserving user privacy. The proposed method involves low communication and computation cost.

REFERENCES

1. 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.
2. S. Kamara and K. Lauter, "Cryptographic cloud storage," in RLCPS, January 2010, LNCS. Springer, Heidelberg.
3. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in Proc. of ACM CCS'06, 2006.
4. M. S. I. M. K. Mehmet Kuzu, "Efficient similarity search over encrypted data," IEEE 28th International Conference on Data Engineering, 2012.
5. D. W. D. Song and A. Perrig, "Practical techniques for searches on encrypted data," in Proc. of S&P, 2000.
6. S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in IEEE Symposium on Security and Privacy, 2012.
7. Y. Z. G. X. J. Y. P. Lu and M. Li, "Toward secure multikeyword top-k retrieval over encrypted cloud data," IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, vol. 10, Aug 2013.
8. O. R. P. G. Boneh D, Crescenzo G, "Public key encryption with keyword search," In: Proceedings of Eurocrypt,, 2004.
9. J. L. K. R. C. Wang, N. Cao and W. Lou, "Secure ranked keyword search over encrypted cloud data," Proc. IEEE 30th Int'l Conf. Distributed Computing Systems (ICDCS '10), 2010.
10. I. H. Witten, A. Moffat, and T. C. Bell, "Managing gigabytes: Compressing and indexing documents and images," Morgan Kaufmann Publishing, San Francisco, May 1999.
11. S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k retrieval from a confidential index," in Proc. of EDBT, 2009.

Lekshmi Balakrishnan is currently doing her post-graduation in Computer Science and Engineering at Sree Chitra Thirunal College of Engineering under Kerala University, Trivandrum, Kerala, India. Lekshmi received her under graduation in Computer Science and Engineering from College of Engineering, Perumon under CUSAT, Kerala, India in 2012. Her area of interest includes network security, distributed computing and cloud computing.

Soja Salim is working as assistant professor in the department of computer science and engineering, Sree Chitra Thirunal College of Engineering, Trivandrum, Kerala. She did her graduation degree from Kerala University and post-graduation from CUSAT.

© Copyright: All rights reserved.