International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 2, Issue 9, September 2015

An Efficient Multi keyword Top-K Retrieval With Historical Scoring Over Encrypted Cloud Data

[1]Vidhu Edavana , [2]Lallu Anthoor
[1]P.G. Scholar, [2]Guide,
Department Computer Science and Engineering,
MIT,Anjarakandy ,Kannur, Kerala, India
[1]vidhu145cs@gmail.com ,[2]lalluanthoor@gmail.com

Abstract: **Cloud computing is recent commonly used method for data outsourcing through web services. We propose a concept to search any encrypted document in cloud store based on multiple keywords specified with the document. Keywords for the document can be of contents from the document, which are selected while uploading that file to the cloud server. The result retrieved after searching is a list of k files (index file). File selection is based on the score calculated by using TRSE (Two Round Searchable Encryption) algorithm. The TRSE is a combination of vector space model and homomorphic encryption technique. Since the homomorphic encryption allows users to do majority of their work on server side by operation only on cipher text. We can eliminate information leakage and can ensure data security. In this paper we propose a concept to score these documents more efficiently on the basis of search history of the document with TRSE scheme. The search history may include the location where the document is being accessed or the number of access to the document by the same user. Besides this when the owner try to update document, with the help of hashing we notify the owner to update index file to avoid data inconsistency and ensure efficiency.**

Index Terms- Cloud, data privacy, ranking, similarity relevance, homomorphic encryption, vector space model

I. INTRODUCTION

Cloud computing [4] is widely used method for data outsourcing over web stores. As the data spread over the internet we must take care of our data, for this we should encrypt our data before upload it to cloud space. Controversies on privacy, however, have been incessantly presented as outsourcing of sensitive information like e-mails, health history and personal data is explosively expanding. Reports of data loss and privacy breaches in cloud computing systems appear from time to time [2], [3].

Furthermore, in cloud computing, data owners may share their outsourced data with a number of clients, who might want to only use the data files, which they are interested in. The most popular ways to retrieve is the keyword-based processing. Keyword-based retrieval is a type of service and widely applied in plain text context, in which clients download relevant files from a file set based on some keywords. However, since we store our encrypted document on cloud, if we want to search any document by using its content is very difficult. Besides, to improve efficiency and save on the expense in the cloud paradigm, it is preferred to get the retrieval result with the most relevant files that match clients' interest instead of all the files, which indicates that the files should be ranked in the order of priority by clients interest and only the files with the highest priority are sent back to clients.

A series of searchable symmetric encryption (SSE) schemes was implemented to enable search on cipher text. Traditional SSE schemes [7], [8] enable users to securely retrieve the cipher text, but these schemes allows only Boolean keyword search, i.e., whether a keyword exists in a file or not, here they do not consider the difference of relevance with the queried keyword of these files in the result. To improve security without sacrificing efficiency, schemes presented in [5], [9] show that they support top-k single keyword retrieval under various scenarios. The authors of [10], [11] tried to solve the problem of top-k multi keyword over encrypted cloud data. These schemes, however, suffer from two problems boolean representation and how to strike a balance between security and efficiency. In the former, files are ranked only by the number of retrieved keywords, which impairs search accuracy. In the latter, security is implicitly compromised to tradeoff for efficiency, which is particularly undesirable in security-oriented applications.

Some approaches have been made by the authors of [1] to solve the above problem, they introduce the concepts and to formulate the privacy issue in searchable encryption schemes, and then correct the security issues by implementing a TRSE scheme, the majority of computing

work is done on the server while the user takes part in ranking, which ensures the retrieval process with more security and practical efficiency. Here also some problem of efficiency reductions can be found.

This paper allow professionals to solve such issues related to efficiency up to an extent. Here we improve efficiency by implementing search logs and keep search history with document. Our contributions can be summarized as follows:

[1]We bring the concepts of similarity relevance and scheme robustness. We, thus, perform the first attempt to formulate the privacy issue in searchable encryption, and we show server-side ranking based on order-preserving encryption (OPE) inevitably violates data privacy.

[2]We propose a history based scoring to improve efficiency of ranking of files on the basis of user behavior, apply TRSE scheme, which fulfills the secure multi keyword top-k retrieval over encrypted cloud data. Specifically, for the first time, we employ relevance score to support multi keyword top-k retrieval. Besides this the efficiency of index file can be ensured by frequently updating the index file with every update in the document, for this any hashing techniques can be used.

[3]Detailed analysis on security demonstrates the proposed scheme guarantees high data privacy and improved efficiency.

II.LITERATURE SURVEY

A.Scenario

Fig1 illustrates a cloud computing system that hosting data service, in which three different entities are involved: cloud server, data owner, and data user.

The cloud server is responsible for hosting third-party data storage and retrieval services. Since data may contain sensitive information, the cloud servers cannot be fully entrusted in protecting data. So outsourced files must be encrypted by the data owner. Any kind of information leakage that would affect data privacy are regarded as unacceptable.

Figure 1: Scenario of retrieval of encrypted cloud data.[1]

The data owner has a collection of n documents C=$\{d_1; d_2; \ldots; d_n\}$ to outsource onto the cloud server in encrypted form and needs the cloud server and want to provide keyword retrieval service to data to authorized users. To achieve this, the data owner needs to build a searchable index I from a collection of l keywords $W = \{w_1; w_2; \ldots; w_l\}$extracted out of C, and then outsources both the encrypted index I and encrypted files onto the cloud server.

The data user is authorized to process multi keyword retrieval over the outsourced data. The computing power on the user side is limited, so operations on the user side should be simplified. The authorized data user at first generates a query $REQ = \{(w_1'; w_1'; \ldots; w_1') | w_i' \in W, 1 \le i \le s \le l\}$. For privacy consideration, which keywords the data user has searched must be protected. Thus, the data user encrypts the keyword and sends it to the cloud server that returns the relevant files to the data user. Afterward, the data user can decrypt using corresponding data owners decryption algorithm and key and hence can use of the files.

B.Two Round Searchable Encryption

TRSE is a new searchable encryption scheme, in which novel technologies in cryptography and IR are implemented, including homomorphic encryption and the vector space model. In this scheme, the data owner encrypts the searchable index with homomorphic encryption. When the cloud server receives a query consisting of multi keywords, it computes the scores from the encrypted index stored on cloud and then returns the encrypted scores of files to the data user. Next, the data user decrypts the scores and picks out the top-k highest scoring filesâ€™ identifiers to request to the cloud server.

The retrieval takes a two-round communication between the cloud server and the data user. We, thus, name the scheme the TRSE scheme, in which ranking is done at the user side while scoring calculation is done at the server side

III.PRACTICAL HOMOMORPHIC ENCRYPTION SCHEME

To reduce the computational overhead on the user side machine, computing work should be done at the server side, so we need an encryption methodology to ensure the operability and security at the same time on server machine. Homomorphic encryption allows specific types of computations to be carried out on the corresponding ciphertext. The result is the ciphertext of the result of the same operations performed on the plaintext. That is, homomorphic encryption completes computation of ciphertext without knowing anything about the plaintext to produce the correct encrypted result. Although it has such a fine property, the original fully homomorphic encryption scheme, which employs ideal lattices over a polynomial ring [6], is too complex and inefficient for practical utilization. Fortunately, as a result of employing the vector space model

to top-k retrieval[1], only addition and multiplication operations performed on integers are needed to produce the relevance scores from the encrypted searchable index. Therefore, we can reduce the original homomorphism in a full form to a simplified form that only supports integer operations, which more efficiency than the full form does.

IV.IMPROVING EFFICIENCY

A.Historical Scoring

As per the concept specified in [1] top-k ranking is performed on the basis of index file,here we propose a concept through which we store search history of documents in the server search history includes count for accessing the document by the same user and the location where the document accessed previously with location based access count.

B. Relevance

When a user searches documents using some keywords and if he accessed a document from top-k document list,the probability to search the same document using the same keyword is high, so we can provide high score to this document.Similarly accessing document from a location with some keywords have also high probability.

C.Implementation

For implementing Historical Scoring, initialization is as specified in [1].then some changes where made in retrieval phase.While retrieving the content we need to store the search history with the server, for that we must findout client location. The client location and client account are linked with a document id and kept it with server.

During next retrieval onwards this stored values are also compared with values computed from requested keywords and score may be calculated.

D.Enable Efficient Updation

The main challege of [1] occures while implement it in to practical cloud. When a data update like adding or deleting file lead issues. So to protect cloud system from this issue, this paper propose a method, by which if a data owner made any change on his data, this will show a warning based on any of the hashing techniques.When user try to delete a file it will be deleted only after changing index file content corresponding to that file.Similar in case of insertin and modification on a file.

V.IMPLEMENTATION

The frame work for implement this concept includes 3 phases:

Initialization : This phase includes the initialization phase specified in [1].

Updation : The updation phase includes UpdateFile(H; F; Fid), DeleteFile(H,F) algorithms.

Retrieval : The retrieval Phase include TrapdoorGen[1], ScoreCalculate and Rank[1], with in this Score calculate is quite different from [1].

A.Updation

Updation includes two types.The following are the algorithms for those process.

B.Update An Existing File

An update process which modifies the existing file content must change index file also for this first check whether the file modified or not. If file is being modified then warn data owner to modify index file also.

In ***UpdateFile(H,F,Fid)***: H is a hashset for all files in the cloud storage for a specific data owner. And F is modified file which we want to store by replacing the existing one, Fid is the file id corresponding to the existing file value.

Algorithm 1 UpdateFile(H; F; Fid)

```
1: Find Hashvalue as H:= Hash(F).
2: for all H1 in H do
3:    if H1 = H then
4:          print 'No modification found for the file';
5:          return.
6:    else if H1 = H[Fid] then
7:          loop
8:           notify to change the Index file for File with idFid.
9:          end loop(wait for changing the Indexfile)
10:         Update the store with modified file F.
11:         return.
12:  end if
13: end for
```

C.Delete An Existing File

This is another way to modify a document in cloud storage.Here a file is going to delete, before deleting that file, we must remove contents in the index file which corresponds to the file.

In ***DeleteFile(H,F)*** : H is a hashset for all files in the cloud storage for a specific data owner.and F is thefile which we want to delete from cloud server.

Algorithm 2 DeleteFile(H, F)

```
1: Find Hashvalue as H:= Hash(F).
2: for all H1 in H do
3:    if H= H1 then
4:       loop
5:          notify to change the Index file for corresponding
            file.
6:       end loop(wait for changing the Indexfile)
7:       Delete F from the Store. return
8:    end if
9: end for
```

D.Retrieval

Algorithm 3 ScoreCalculate

```
1: calculate score using algorithm specified in [1].
2: if History exists then
3:       compare scored documents as values with historical
         value.
4:       update score based on historical value.
5:       update historical data.
6: else
7:       create history file.
8: end if
```

While retrieving the document we first calculate the score and based on the score client rank the k files to download based on the rank, server prepare those k files to downloadable.In existing system scores are calculated using index file and homomorphic encryption techniques only. Besides that in this paper I introduce a new method that allow userspecific and location specific score calculation and through which for different users, different order of files are available.The filtering process is done on the basis of historical data.Historical data is a document which contains the details of previous search history for each and every document.The search history consists of location from where the document was accessed and the user who accessed the data and how many times did he access the document.

Here this process enables more efficiently than existing since user specific and location specific iteration is possible.

VI.CONCLUSION

In this paper,we try to solve issues present in [1].As per discussion by J.Yu. and et.al.they propose [1] and is a way to retrieve ecrypted document from cloud data using multiple keys.Through this paper we make concept of [1] more efficient. For this we use historical values of every access to score the document.Beside this index files for server document can be corrected with updation like modify, add, delete document from the store.As considering probablistic method we can say this paper is more efficent than existing.

REFERENCES

[1] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Towards secure multi-keyword top-k retrieval over encrypted cloud data," *IEEE Transactionson Dependable and Secure Computing*, vol. 10, no. 4, pp. 239-250, 2013.

[2] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," http://www.techcrunch.com/2006/12/28/gmaildisasterreports-of-mass-email-deletions/,Dec. 2006.

[3] Amazon.com,"Amazon s3 Availability Event: July 20, 2008,"http://status.aws.amazon.com/s3-20080720.html, 2008.

[4] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and M. Zaharia, " A View of Cloud Computing," *Comm. ACM*, vol. 53, no. 4, pp. 50-58, 2010.

[5]S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k Retrieval from a Confidential Index," *Proc. 12th Intl Conf. Extending Database Technology: Advances in Database Technology (EDBT)*, 2009

[6] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proc. 41st Ann. ACM Symp. Theory of computing (STOC)*, pp. 169-178, 2009.

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

[8] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public-Key Encryption with Keyword Search," *Proc. Intl Conf. Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2004.

[9] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A.L. Varna, S. He, M.Wu, and D.W. Oard,"Confidentiality-Preserving Rank-OrderedSearch," *Proc. Workshop Storage Security and Survivability*, 2007.

[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou,"Privacy-Preserving Multikeyword Ranked Search over Encrypted Cloud Data," *Proc. IEEE INFOCOM*, 2011.

[11] H. Hu, J. Xu, C. Ren, and B. Choi,"Processing Private Queries over Untrusted Data Cloud through Privacy

Homomorphism," *Proc. IEEE 27th Intl Conf. Data Eng. (ICDE)*, 2011.