IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. I (Nov – Dec. 2015), PP 103-112
www.iosrjournals.org

Secure Multi-Keyword Top-K Retrieval Over Encrypted Cloud Data Using Homomorphic Encryption

Sruthi V [1], Surekha Mariam Varghese [2]
[1, 2(]*Dept. of CSE, Mar Athanasius College of Engineering, Kothamangalam, Kerala, India)*

Abstract: *Using Cloud Storage, users can remotely store their data and enjoy on-demand high quality applications and services. To ensure safety of stored data, it becomes must to encrypt data before storing in the global space. In cloud data, search arises only with plain data. But it is essential to invoke search with encrypted data. The specialty of cloud data storage is that it allows copious keywords in a solitary query and sorts the resultant data documents in relevance order. The proposed multi keyword search based on ranking over an encrypted cloud data uses feature of similarity and inner product similarity matching. The vector space model helps to provide sufficient search accuracy and homomorphic encryption enables users to involve in ranking while majority of computing work is done on server side by operations only on cipher text. Thus in this method for Top-K retrieval user gets an interested/used link in top. To selectively share documents fine-grained attribute-based access control policies can be used.*

Keywords: cloud storage**,** vector space model, homomorphic encryption, cosine similarity.

I. Introduction

Cloud refers to internet or network or as a synonym to something with so many capabilities. Cloud computing describes a variety of different types of computing concepts that involve a large number of computers connected through a communication network. Broader concept of converged infrastructure and shared services layed the foundation of cloud computing. The term cloud is not simply the synonym term for the Internet rather, the cloud is where we go to use technology when we need it, for as long as we need it and does not need to install anything on our desktop/pc. Moreover there is no need to pay for the technology when we are not using it. The cloud thus can be both software and infrastructure. Security is one of the most often-cited objections to cloud computing. Cloud users face security threats both from outside and inside the cloud. In the cloud, this responsibility is divided among potentially many parties, including the cloud user, the cloud vendor, and any third-party vendors that users rely on for security-sensitive software or configurations. The cloud user is responsible for application-level security. The cloud provider is responsible for physical security. When users outsource their private data onto the cloud, the cloud service providers can control and monitor the data without having permission from the data owner.

To ensure privacy, users thus needed to encrypt the data before outsourcing it onto cloud. But encrypting the data to be uploaded will greatly affect the data utilization. Moreover, data owners may share their outsourced data with a number of users. The data users on the other hand want to retrieve the data files they are interested in. One of the most popular ways for this is through keyword-based retrieval. Keyword-based retrieval is a typical search technique widely applied in plaintext scenarios, in which users retrieve relevant files based on keywords. But searching is a difficult task in ciphertext scenario as limited operations can only be performed on encrypted data. Besides it is also required that the retrieved files must be in the order of relevance. The existing searchable encryption techniques do not suit for cloud computing scenario because they support only exact keyword search and does not consider the significance of retrieved file with the queried keyword. Moreover it employs server side ranking which causes the leakage of sensitive information. This significant drawback of existing schemes signifies the importance of a new method that supports multikeyword search.

In this paper Secure Multikeyword Top-K Retrieval Over Encrypted Cloud Data Using Homomorphic Encryption is proposed. This paper is organized as follows: in section II, the related work is discussed. Section III describes implementation of multikeyword top-k retrieval in encrypted cloud using homomorphic encryption. Security and performance analysis of the proposed system is analyzed in section IV. Section V and VI gives the result and conclusion for the work.

II. Related Work

As the communication services are growing tremendously, outsourcing sensitive data into a global storage space has become common. Documents containing private sensitive data are uploaded into the trusted third party so as to perform operations such as sharing, editing, deleting, searching, retrieval etc on them. Preserving the confidentiality of the data while information retrieval, is now a big challenge. To overcome this challenge, data owners encrypt their data before uploading into the global space. This encryption in turn affect

the search and retrieval of data based on keywords. Since the data is uploaded into the global space maintained in the remote server as encrypted, the challenge is that how to enable efficient and secure search and retrieval over such encrypted data. Multi keyword searching in plain text scenarios is very common e.g. google search. But the condition is different in case of cloud. In cloud searching is a difficult task and time consuming as the data exist in encrypted form. To facilitate search on cloud a number of searchable symmetric encryption schemes exist. At earlier times the data owner has to download all the encrypted documents uploaded by him on the cloud, to retrieve a particular document. This way of document retrieval becomes infeasible as the number of uploaded documents increases. To overcome this scenario and to share the documents with others, proposed single keyword search.

To meet this need, several searchable symmetric encryptions[10] are available such as boolean keyword search, fuzzy keyword search etc. Order Preserving Encryptions [OPE] based on SSE scheme are there, but that technique breaches the privacy of sensitive information. Moreover, they allow only single keyword in search query. In order preserving encryption keyword frequency is used as ranking criteria, i.e. instead of all the files containing that keyword, gives retrieval result as the most relevant files. In OPE the server can obtain additional information through statistical analysis. Relative distribution of terms, co-occurrence of terms, searching patterns, access patterns etc of the plain text are revealed even it is in encrypted form in OPE. Access pattern refers to which keywords and the corresponding files have to be retrieved during each search request and Search pattern refers to whether the keywords retrieved between two requests are the same. According to [1], there are two possible statistic leakages: term distribution and inter distribution.

In single keyword search, the keyword in the query issued by the data user is compared with the documents. Documents matching the keyword are only retrieved in this case. Here the relevance of the queried keyword with the document is not considered hence ranking is not possible. To broaden the search criteria, later introduced boolean keyword search, fuzzy keyword search, single keyword search with ranking, fuzzy keyword search with ranking etc. Boolean operators such as AND, OR, NOT are used for boolean keyword search along with the keyword to be searched. For example, a Boolean search could be "Apple" AND "Computers". This would limit the search results to only those documents containing the two keywords.

Fuzzy keyword search [2] retrieve the documents where the queried keyword is present, if exact match is not obtained it will return the documents matching the closest to the queried keyword i.e. find matches even when users misspell words. Semantic context is considered in fuzzy search. Mainly 3 techniques are there for fuzzy keyword search viz: Wild Card based technique, Gram-based technique and symbol-based trie traversed scheme. All these techniques uses edit distance to quantify the severity of violation. Insertion, deletion and substitution are used for determining the edit distance. Suppose the queried keyword is APPLE; then with edit distance 1, the set of fuzzy keywords to be searched become: {APPLE,*APPLE, *PPLE, A*PLE,AP*LE,APPL*E, APPLE*}. Substrings of the queried keyword are used in gram based fuzzy search technique. Each substring thus formed is called grams. Grams for the keyword APPLE are {APPLE, APLE, APPE, APPL, PPLE}. The concept of queried keywords sharing common prefix has common nodes. Based on this a multi-way trie is constructed for every keyword and depth-first search is performed. Embedding of edit distance into hamming distance and combining it with private identification scheme [6] is more efficient can fuzzy keyword search as this technique does not require a predefined set of keywords relevant to the current queried keyword.

The scheme introduced by [3], [4] support topk single keyword based retrieval. To make the search efficient and relevant to the user it is necessary to have multikeyword search in encrypted data along with ranking. Indexing [7] [8] is the efficient technique for efficient search. Index management scheme [5] make use of two techniques viz: Proxy re-encryption function and searchable encryption function. However each data owner requires the index build for his document to be secure and at the same time must facilitate searching. Homomorphic encryption [6] with additive and multiplicative properties can be applied to the secure index build and can perform any operation on the ciphertext, will map to the same operation on the plain text. Major limitations of the existing system thus include: single keyword search and boolean keyword search without ranking, single keyword search with ranking, ranking on server-side with leakage of sensitive information and does not support multikeyword search with ranking.

So there arise the need for new searchable encryption scheme which uses new technologies in information retrieval community and cryptographic community. Thus Two Round Searchable Encryption (TRSE) scheme has been proposed. TRSE enables multi-keyword search and top-k retrieval. It enables us to get the retrieval result as the most relevant files that match users' interest. It means that files are ranked in the order of relevance of the interest of the user. The concept of homomorphic encryption, relevance scoring, vector space model and key management are used in this top-k retrieval of data. Since the all operation is performed on encrypted data, information leakage is eliminated.

III. Implementation

The proposed secure multikeyword top-k retrieval over encrypted cloud data make use of the techniques of information retrieval community i.e. vector space model and cryptographic community i.e. homomorphic encryption. In proposed system to ensure security and efficiency most of the work is done by cloud, but ranking is left to the cloud user. As most users want the documents that are most relevant to them, in top-k multi keyword retrieval the cloud calculate the score for each document as per the keywords in the query. Term frequency-inverse document frequency weighing scheme is used to assign score for each document and cosine similarity to find the similarity. Cloud server then returns the calculated scores to the data user and data user send back the top-k scores to the cloud server. Thus there is a two round communication between cloud server and data user, hence called Two Round Searchable Symmetric Encryption Scheme [TRSE].

Large organizations require sharing a number of documents of large size within group. Cloud provides suitable platform for sharing documents as well as resources. Usually to prevent leakage of sensitive information, the uploaded documents are encrypted. Retrieval of interested document is done with the help of a secure index. Data owners itself build an index for each documents he uploads into the cloud. The index so build is encrypted using homomorphic encryption scheme. Here Apache's Lucene API is used to extract keyword. Apache Lucene is a high performance full featured text extractor which extracts the keywords from the encrypted uploaded document. The cloud server stores the encrypted documents and corresponding encrypted index. Cloud users who want to retrieve the documents of his interest in relevant order issues query, which is also homomorphically encrypted. Cloud server on receiving the query extract the key terms and stem it. To find the documents satisfying all the keywords in the query, both the documents and the query is represented as vector. Each dimension of the vector corresponds to the presence and absence of the keyword. The presence of a term in the vector is denoted by 1 otherwise 0. Term frequency-inverse document frequency is used to calculate the score of each term and cosine similarity to rank the documents. After ranking top-k documents scores matching the user's interest is returned back to the data user.

3.1 Techniques Used

For greater flexibility, data owners outsource their data from local systems to remote servers. Security issues are to be considered if sensitive data is outsourced into the global space. For protecting data, they are encrypted before uploading. Search on plain text is an easy task. But it is complicated in cipher text scenarios. Several searchable symmetric encryption schemes are there for cipher text search. But they do not consider the relevance of document and allow only single keyword in a search query. For a large data management environment, multiple keyword based search is necessary. Existing schemes that supports ranked search have low efficiency and security. So for having security in information retrieval, an encryption mechanism called Homomorphic Encryption is being used.

Homomorphic Encryption Scheme

Encryption scheme that provides security at both user side and on server side is needed. Moreover only addition and multiplication operations over integers are needed as a result of using the vector space model and relevance scoring, thus homomorphic encryption is employed here. Homomorphic encryption maps specific operations in cipher text to that to plain text. Pailier cryptosystem shows additive homomorphic property. A Homomorphic encryption is additive, if:

$Enc(x \oplus y) = Enc(x) \otimes Enc(y)$

$Enc(\Sigma\, mi) = \Pi\, Enc(mi)$

$i=1 \qquad i=1$

Suppose C1 and C2 be the two cipher text such that:

$C_1 = g^{m1}. r_1^n \bmod n^2$

$C_2 = g^{m2}. r_2^n \bmod n^2$

Then,

$C_1.C_2 = g^{m1}. r_1^n\, g^{m2}. r_2^n \bmod n^2 = g^{m1+} m_2 (r_1 r_2)n \bmod n^2$

An example for additive homomorphic property is shown below:

Let (n, g) = (2501, 92)

m1 = 34 and m2 = 16

$C_1 = 92^{34}\, 5^{2501} \bmod n^2 = 1129735 \quad (r_1 = 5)$

$C_2 = 92^{16}\, 7^{2501} \bmod n^2 = 5140305 \quad (r_2 = 7)$

$C_1 . C_2 = 2010769$

m1 + m2 =50

$C_{1+2} = 92^{50} .\ 35^{2501} \bmod n^2 = 2010769$

Thus Paillier cryptosystem shows the property of additive Homomorphic encryption. Electronic voting is an example of additive Homomorphic encryption

Scoring Scheme

To weight the relevance of a document with respect to a keyword the simplest way is scoring. For scoring different ranking models exists; of these most common is TF-IDF weighing scheme. Many variations of the tf-idf exist. Term frequency and Inverse document frequency are the two attributes of this scoring scheme. How many times a particular keyword occurs in a document are defined by Term Frequency (tf) whereas in how many document a particular keyword exists is defined by Document frequency (df). The inverse document frequency is calculated as follows:

$Id_f = \log [N/ d_f]$, where N is the number of files.

Then, by using tf-idf weighting scheme a term t in file f is given a score as:

$tf - idf_{t,f} = tf_{t,f} X idf_t$

A combination of the Vector Space Model and the Boolean model is used in Lucene Scoring. Vector space model determines how many times a query keyword appears in a document with respect to the number of times the keyword appears in all the documents in the collection. Boolean model on the other hand narrows down the documents that need to be scored based on the use of boolean logic.

Vector Space Model

To find the similarity of queried keyword with the existing documents, vector space model is used. In vector space model both documents and query is represented in the form of vector. For example consider the following two texts:

Text 1: Ram loves me more than Krishna loves me

Text 2: Lakshmana likes me more than Ram loves me

The keywords that occur in this are: me Ram loves Krishna than more likes Lakshmana

Then count the number of times each keyword exists in the text as:

me 2 2

Ram 1 1

likes 0 1

loves 2 1

Lakshmana 0 1

Krishna 1 0

than 1 1

more 1 1

The two vectors that correspond to the text are:

a: [2, 1, 0, 2, 0, 1, 1, 1]

b: [2, 1, 1, 1, 1, 0, 1, 1]

Here an 8 dimensional vector is formed. Cosine similarity is further used to find the similarity.

Cosine Similarity

To rank the documents different ranking techniques are available. Here to retrieve the top-K documents matching the user's interest, cosine similarity is used. An example for calculating cosine similarity is given below:

Consider a small collection of documents, consisting the following three documents:

document1: "very good times"

document2: "very good post"

document3: "los angeles times"

Step1: Calculating Term frequency

For all the documents, then calculate the *tf* scores for all the terms in C. Assign the score 1 if the keyword appear in that particular document, otherwise assign 0:

Table 3.1 Term Frequency

	angles	Los	very	post	times	Good
document1	0	0	1	0	1	1
document2	0	0	1	1	0	1
document3	1	1	0	0	1	0

Step2: Inverse Document Frequency

The total number of documents is N=3. Therefore, the *idf* values for the terms are:

angles $log_2(3/1)=1.584$

los $log_2(3/1)=1.584$

very $log_2(3/2)=0.584$

post $log_2(3/1)=1.584$

times $log_2(3/2)=0.584$

good $log_2(3/2)=0.584$

Step 3: TF x IDF

Then multiply the tf scores by the idf values of each term, obtaining the following matrix of documents-by-terms:

Table 3.2 TF-IDF Matrix

	angles	Los	very	post	times	Good
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

Step 4: Vector Space Model And Cosine Similarity

Let the query given by the user be: "very very times", calculate the *tf-idf* vector for the query, and compute the score of each document in C relative to this query, using the cosine similarity. When computing the *tf-idf* values for the query terms, divide the frequency by the maximum frequency (2) and multiply with the *idf* values.

Using the formula given below we can find out the similarity between any two documents.

Cosine Similarity (d1, d2) = Dot product(d1, d2) / ||d1|| * ||d2||

Dot product (d1,d2) = d1[0] * d2[0] + d1[1] * d2[1] * … * d1[n] * d2[n]

||d1|| = square root($d1[0]^2 + d1[1]^2 + ... + d1[n]^2$)

||d2|| = square root($d2[0]^2 + d2[1]^2 + ... + d2[n]^2$)

The query entered by the user can also be represented as a vector

q [0 0 (2/2)*0.584 0 (1/2)*0.584=0.292 0]

Calculate the length of each document and of the query:

Length of d1 = sqrt(0.584^2+0.584^2+0.584^2)=1.011

Length of d2 = sqrt(0.584^2+1.584^2+0.584^2)=1.786

Length of d3 = sqrt(1.584^2+1.584^2+0.584^2)=2.316

Length of q = sqrt(0.584^2+0.292^2)=0.652

Then the similarity values are:

cosSim(d1,q) = (0*0+0*0+0.584*0.584+0*0+0.584*0.292+0.584*0) / (1.011*0.652) = 0.776

cosSim(d2,q) = (0*0+0*0+0.584*0.584+1.584*0+0*0.292+0.584*0) / (1.786*0.652) = 0.292

cosSim(d3,q) = (1.584*0+1.584*0+0*0.584+0*0+0.584*0.292+0*0) / (2.316*0.652) = 0.112

According to the similarity values, the final order in which the documents are presented as result to the query will be: d1, d2, d3.

TRSE Scheme

Data management environment includes a group of data owners, their associated set of data user and a storage space. Both data owner and user can login into the environment by registering their username and password. On successful registration, data owner will have a unique ID in addition to his username and password. After approval by the admin, data owner can upload, share, search and delete the documents.

The data owner has a collection of documents which may be of extension .txt, .doc and .pdf to be uploaded into the trusted third party server for greater flexibility. For security purpose the uploaded documents are encrypted before uploading and a secure searchable index is prepared by the owner. The keywords are taken from documents, transformed into its root form by a process called stemming. Porter stemmer algorithm is commonly used for this. Moreover, data owner is has to generate keys for homomorphic encryption; both secret key and public key. The secure index so build is encrypted homomorphically. The ciphertext will be in integer form. Then score value is calculated using TF-IDF function. The selected documents can be encrypted using any symmetric encryption algorithm. Then both encrypted index and document are uploaded into remote global space. Data user issues the query after proper authentication. The query is represented in vector form; homomorphically encrypted and send to the remote server. Score for each document based on the query is calculated and returns the encrypted score to the data user. Data user then picks the top-k scored documents and requests the data owner for the decryption key. As the retrieval process takes a two round communication between server and the data user, the scheme named as Two Round Searchable Symmetric Encryption scheme. The score calculation is done by the server but ranking by the data user. At the time of data uploading, each data

owner generates keys for homomorphic encryption, using which both the index and user query is encrypted. To encrypt query vector and also to decrypt the retrieved documents, the data user needs keys from data owner. So that data owner has to handle the in a secure manner. Decrypting keys are provided by the owner to the user is by Mail-Id. The data files of extensions like .txt, .doc, .pdf etc are supported by this scheme.

TRSE scheme has 2 phases: Initialization phase and Retrieval phase. TRSE scheme setup and building of index is done in the initialization phase whereas trapdoor generation, score calculation and ranking of documents are done in the retrieval phase. Thus four algorithms are involved in the TRSE scheme such as for building index, trapdoor generation, score calculation and ranking. As the user has less power of computation, here most of the computation is done at the server i.e. score calculation by the server but ranking by the user.

In the initialization phase, data owners generates homomorphic keys for encryption viz secret key(sk) and public key(pk) by using a secret key λ. This setup algorithm is done only once by the data user. The public key pk is used to encrypt the index I build from the document being uploaded. Let the encrypted index denoted by $I^{'}$. In the retrieval phase operations are performed on the ciphertext. Data user issues multi-keyword query q. keywords are extracted from the query using stemming algorithms. Let query q contains $\{k_1, k_2, k_3 \ldots k_n\}$ keywords. Homomorphic encryption is also applied to the query q to generate $q^{'}$. Encrypted q' is then compared with the secure index to find a match and score is calculated. The data user picks the top-k scores send by the cloud server and send back it to the cloud server itself. To decrypt the file data user sends request for decrypting key.

3.2 Description Of Modules

The different modules involved in TRSE scheme are given below:

Preprocessing Module

Three different entities are involved in cloud computing environment: Cloud server, Data owner and Data user. Data owner at first has to register with the cloud. After that the user has to wait until the administrator approves him. The data owner can upload the documents. Third-party data storage and retrieval services are hosted by the cloud server. As the uploaded data may contain sensitive, data is needed to be encrypted before outsourcing. The admin also assigns a group to the data users.

Searchable Index Module

To effectively search documents, data owner has to build a searchable secure index for all the documents being uploaded into the cloud. All the terms relevant to the document is extracted, stemmed and counted. The document along with the index is then uploaded into the cloud.

Encrypt Module

The documents before uploading into the global space are encrypted using any encryption scheme, whereas for the secure index homomorphic encryption is applied. Homomorphic encryption maps the operations in cipher text and plain text domain. The additive and multiplicative property of homomorphic encryption is utilized here. Thus Pailier homomorphic encryption is applied to the index build. Homomorphic keys should be handled properly.

Processing Module

Processing module deals with how to get accurate search result based on the multiple keyword queries. The users can enter the multiple words query in the encrypted form; the server stems that query into single words. Then find the term frequency (tf) and inverse document frequency (idf) for each term.

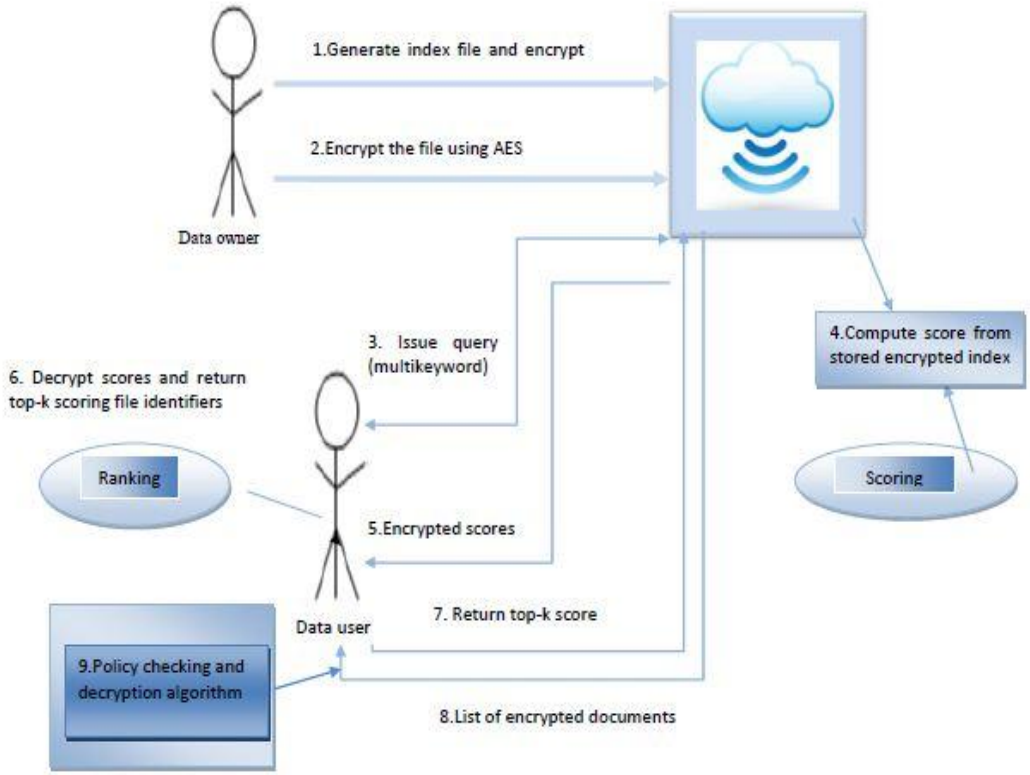
Query Ranking Module

Score for each document is calculated based on the tf-idf value. The number of documents to be retrieved is specified by the user. The query issued by the user as well as all the documents are represented in the form of vector. Each dimension of the vector represents the presence of the particular keyword. If a particular term is present then its value is 1, otherwise the value of that vector is set to 0. To rank documents cosine similarity between user query and all documents is found. After ranking top-k scores are returned to the user.

Log File Generation

For each user action a log file is generated at the cloud server. Log information is used by cloud admin to get all information regarding document uploading and downloading. Data owners on the other hand use this log file to get information about file's download.

3.3 System Architecture



IV. ANALYSIS

4.1 Security Analysis

Confidentiality of index is maintained

In the proposed scheme for the cloud server it does not possible to obtain information about data vector and multi-keyword query vector. Moreover TF-IDF values, access pattern and search Pattern cannot be deducted by the global server. Scoring of documents based on multikeyword query is done in the cloud but confidentiality of index is maintained as index is homomorphically encrypted.

Unlinkability of trapdoor.

The user query I is homomorphically encrypted to I'. In this proposed scheme for same search query, the trap door generated will be different thus unlinkability is maintained.

Maintains keyword privacy

In traditional SSE scheme it is possible for the trusted third party to know about the querying keyword by analyzing the access pattern, search pattern and term frequency. Moreover, they supports only single keyword query. But the proposed system supports multikeyword query and conceal the term distribution details. Hence preserves keyword privacy.

4.2 Performance Analysis

4.2.1 Index Building

Whenever a data owner uploads a document into the cloud, keywords are extracted, converted to root form and encrypted using homomorphic encryption scheme. Keywords from the queried keyword is stemmed, encrypted and mapped to the data vector. Length of the wordlist (secure index for each document) and the number of documents in the files determines the data vectors dimensionality. On the other hand dimensionality of the data vector directly determines cost for mapping. That is as number of documents increases the file size increases, as document size increases the length of secure index increases. Figure below shows time cost for building secure index for various documents with different number of keywords.

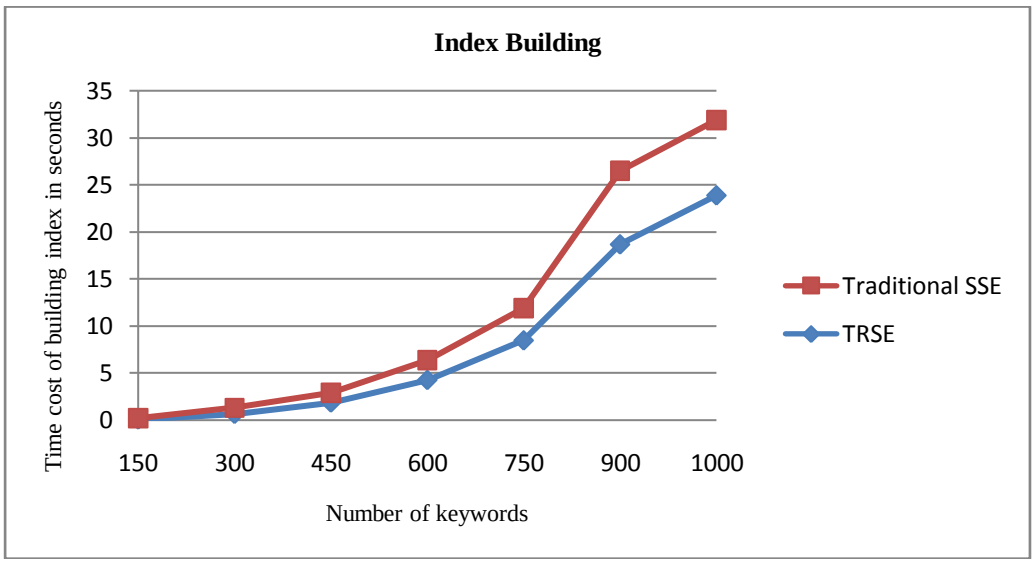


Figure 4.1: Index Building

From the above graph it can be inferred that the proposed scheme requires less time than traditional SSE scheme to build a secure index.

4.2.2 Query Vector Generation

The time to build a query vector depends on the length of the secure index build for each document. Thus the number of keywords in the user query has less influence on the vector generation.

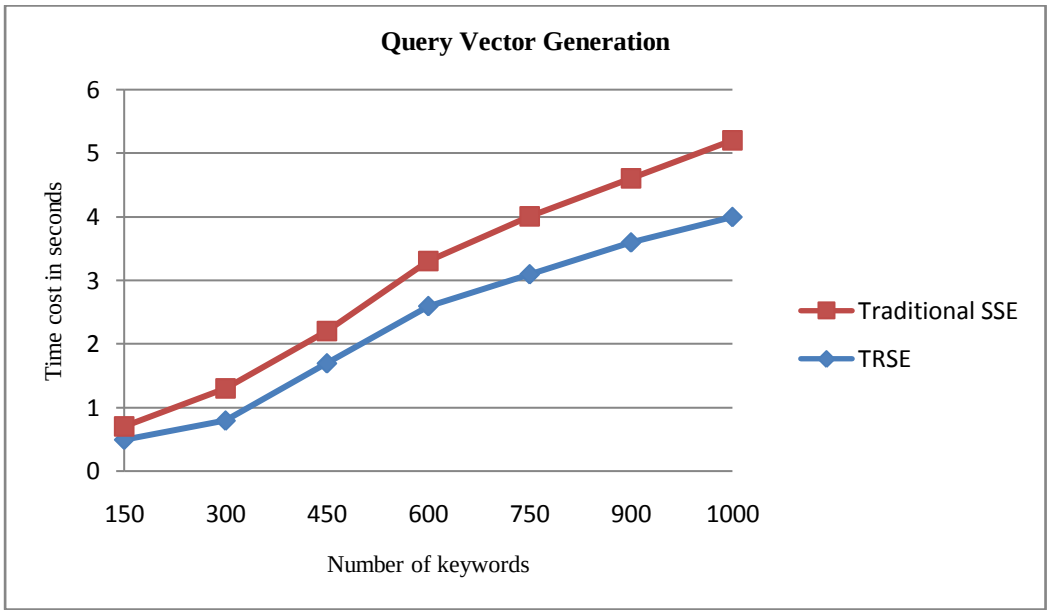


Figure 4.2: Query Vector Generation

V. Results

There are so many searching techniques implemented in the cloud but the drawback with these techniques is that they supports only exact keyword search. Moreover typical users searching behavior is also revealed. The proposed system overcomes all these limitations. Distribution information from access pattern and search pattern implies a similarity relationship among terms or files. Let Q1 and Q2 be the vectorized search query corresponding to queries R1 and R2. Suppose one same keyword t_k is requested in two queries R1 and R2, then the positions $m_{1i} = m_{2i} = 1$ in the corresponding query vector Q1 and Q2. By using 'Encrypt' function the two vectorized queries are encrypted into different ciphertext. Even though same keywords are present in different queries, the ciphertext of two queries are independent of each other, so that which keywords have been retrieved are concealed thus, the access pattern and search pattern are secure in this TRSE scheme.

Each data owners can login after providing proper credentials to perform operations like uploading, searching, sharing and deleting. The figure3.1 shows the documents uploaded by the user bob and figure 3.2 shows the search by bob based on keyword "Big Data". Compared to the existing TRSE scheme, the proposed system enhances security. Data users can directly access the documents either within the group or shared to him by the data owner. In the existing scheme data owner can share his/her documents with a number of users. The data user on the other hand can download the documents shared to him by the user from the cloud server directly. Once the data owner shares his document, he will lose control over that document. Document once shared can't be withdrawn. But in proposed system a data user needs decrypting key from the data owner even if the document is shared to him by the data owner. This enhances the security.

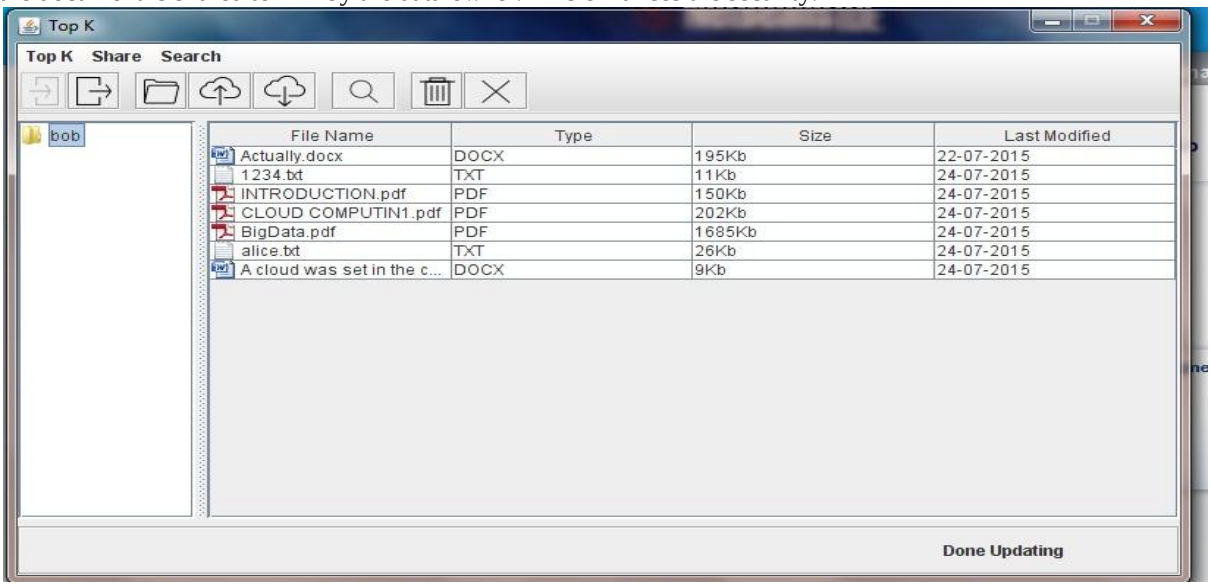


Figure 5.1: Documents uploaded by bob

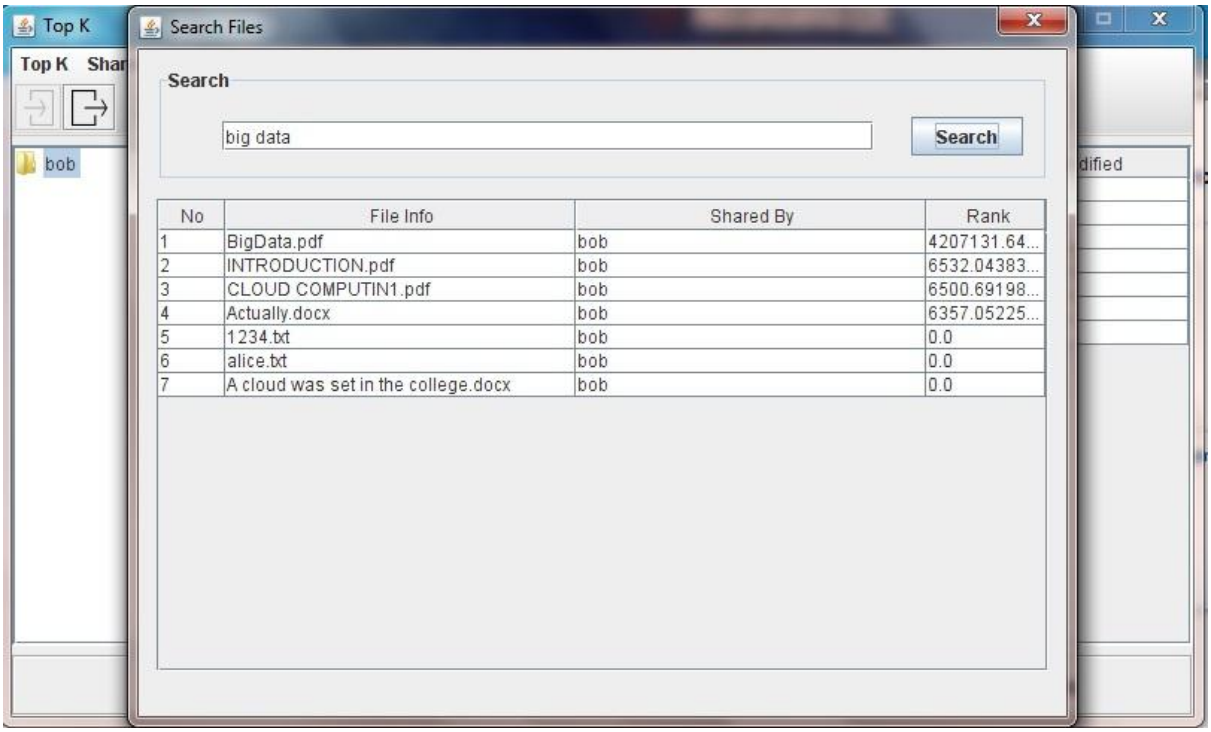


Figure 5.2: Ranking of documents Based on "Big Data" keyword

VI. Conclusion

Now a day it's become common to access global storage space over the Internet. The main problem of storing data in a trusted third party is regarding security. Even data is encrypted before outsourcing; effective

information retrieval is a big challenge. To overcome these challenges, proposed a scheme [TRSE Scheme] which enables data owners to upload encrypted data files into global storage and allow several authorized users to perform operations such as uploading, search, share and retrieval over them. Instead of all the files, proposed method enable users to obtain the result with the most relevant files that match users' requirement. Documents of highest relevance are only sending back to the user. In TRSE, the concept of homomorphic encryption, relevance scoring, vector space model and key management are used. Since the search operation is performed over encrypted data, information leakage can be eliminated and data can be searched and retrieved efficiently. TRSE scheme supports multiple data owners and their associated set of users. Separate space for each data owner is maintained at storage side. To facilitate search on encrypted index homomorphic encryption is used.

Homomorphic encryption maps operation in the ciphertext to that in plain text. Moreover, homomorphic encryption enables users to involve in the ranking while the majority of computing work is done on the server side by operations only on ciphertext. Since statistical analysis is not possible in homomorphic ciphertext, the proposed scheme gives security to certain extends. Vector space model helps to provide sufficient search accuracy. Each uploaded document is represented as vector with dimensions corresponding to the presence or absence of keywords. The system can be further improved by implementing fully homomorphic encryption which supports reduced ciphertext and key size.

References

[1] Jiadi Yu, Member, IEEE, Peng Lu, Yanmin Zhu, Member, IEEE, Guangtao Xue, Member, IEEE Computer Society, and Minglu Li, Toward Secure Multikeyword Top-k Retrieval over Encrypted Cloud Data IEEE Transactions On Dependable And Secure Computing, Vol. 10, No. 4, July/August 2013

[2] S. Ji, G. Li, C. Li, and J. Feng, Efficient interactive fuzzy keyword search. In Proc. of 18th International World Wide Web Conference(WWW'09), Madrid, Spain. ACM, April 2009

[3] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data", Proc. IEEE 30th Intl Conf. Distributed Computing Systems (ICDCS), 2010.

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

[5] Sun-Ho Lee and Im-Yeong Lee,"Secure Index Management Scheme on Cloud Storage Environment" ,International Journal of Security and Its Applications Vol. 6, No. 3, July, 2012

[6] Craig Gentry" Fully Homomorphic Encryption Using Ideal Lattices" In the 41st ACM Symposium on Theory of Computing (STOC), 2009.

[7] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-kRetrieval from a Confidential Index," Proc. 12th Int'l Conf. Extending DatabaseTechnology: Advances in Database Technology(EDBT), 2009

[8] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive,Report 003/216,2003, http://eprint.iacr.org/.

[9] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. S, "Public key encryption that allows PIR queries" CRYPTO conference 2007.

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