Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 4, April 2016, pg.465 – 474

Multi-Keyword Ranked Searched on Encrypted Data using Searchable Symmetric Encryption in Cloud Storage

Bhooshan Waghmare[1], Nilesh Sambhe[2]
[1,2]Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India
[1] bhooshanw27@gmail.com, [2] nilesh.sambhe@gmail.com

Abstract— With the increasing advent of cloud computing a large amount of data is currently being stored on cloud and it is rapidly increasing. However ensuring cloud data security is major concern for cloud subscribers as they are outsourcing there sensitive information on cloud. One simple way to ensure data security is to encrypt the data before outsourcing to cloud. But encrypting the data limits the capability to search over that data, as user cannot simply perform a plain text keyword search over cipher text. This makes the retrieval of information inefficient. Searchable encryption is the solution for this problem. Searchable encryption provides secure and efficient retrieval of data with the ability to search over encrypted data without revealing the contents and searched keyword. In this paper we discuss the work that has been done in the field of searchable encryption and also propose a new approach.

Keyword- Cloud Computing, Searchable Encryption, Data Confidentiality, Privacy Preserving Search.

I. INTRODUCTION

Cloud computing, the term has gained enormous popularity in the current age and users now tend to access their files from remote cloud storage more often. The main advantage of cloud storage is its ubiquitous user accessibility and also its virtually unlimited data storage capabilities. Despite such benefits provided by the cloud, the major challenge that remains is the concern over the confidentiality and privacy of data while adopting the cloud storage services [1]. For example unencrypted user data stored at remote cloud servers can be vulnerable to external attacks initiated by unauthorized outsider and internal attack initiated by untrustworthy cloud service providers(CSPs) [2].This raises a concern for many cloud subscribers as the outsourced data might be contain very sensitive information.

Vast research has been done to ensure confidentiality and privacy of cloud data without compromising the user functionality. Here confidentiality means secrecy of stored data so that only the authorized user can read the contents of data. To solve the problem of confidentiality use of data encryption schemes is done, which provide some control over the secrecy of stored data.

But by making use of standard encryption schemes we lose the ability of searching over the data, because after encrypting the data user cannot simply perform plain text keyword search over cipher text. Thus encryption makes the retrieval of information inefficient. The keyword search functionality enables the user to search for a keyword on remote cloud data [3].

Consider an application that uses cloud to store user's data. Here user can use traditional encryption scheme for confidentiality of contents. The natural way for retrieving encrypted contents for certain keyword would require user to download all the data, and then decrypt it and perform search on the local machine. However this solution is not feasible from a practical point of view, as user needs to download all the contents rather than the contents containing the searched keyword. For example consider user's data is of 1GB but only 1MB of data from that is related to search keyword, which is inefficient. Another way is to store plaintext keyword index on cloud server and use it while searching or retrieving data. But this way the keywords are known to server which is not expected. Thus to provide to secure and efficient retrieval of data, we need to ensure that the contents and keywords are not revealed while performing search over encrypted data. The technique which provides this feature is called as searchable encryption (SE).

In this paper we are first going to study what is searchable encryption, then we will study what work has been carried out in this field by various researchers. After that we will propose our technique using searchable encryption and finally the conclusion.

II. Searchable Encryption

Searchable encryption (SE) enables the users to generate a search token from the searched keyword in such way that given a token, the cloud server can retrieve the encrypted contents containing the searched keyword. Basically, the search token represents an encrypted query over the encrypted data and can be generated only by users with the appropriate secret key.

Figure 1 shows the basic architecture and working principle of a searchable encryption scheme. The architecture comprises mainly four entities: data owner, data user, cloud service provider and key generator. A brief description of the entities and their operations are given below,

1. Data owner

The data owner is the entity which generates and encrypts the data and uploads them to the cloud server. It can be either an organization or an individual. To use the service, the data owner uses its application which consists of a data processor for uploading new contents to the cloud. It encrypts the data and metadata with a cryptographic scheme that enables searching capability.

2. Data user

This entity is also a subscriber to the cloud storage which sends encrypted queries to the cloud service provider to search for a specific encrypted data. There may be more than one data user in the system and in some scenario, the data owner and the data user might be the same entity.

3. Cloud service provider

This entity provides the data storage and retrieval service to the subscribers. The cloud service provider consists of cloud data server and cloud service manager. The first entity is used to store the outsourced encrypted data whereas the latter one is used for data management in the cloud. Upon receiving the encrypted search queries from the data user, the cloud service provider tests on the encrypted queries and encrypted metadata in the cloud storage. The encrypted data that satisfies the search criteria is retrieved and sent back to the data owner upon completion of the test. The cloud service provider should not learn any information from the operation.

4. Key generator

This entity is considered to be a trusted third party which is responsible for the generation and management of the encryption/decryption keys. User specific keys are generated and distributed during the setup of the system.

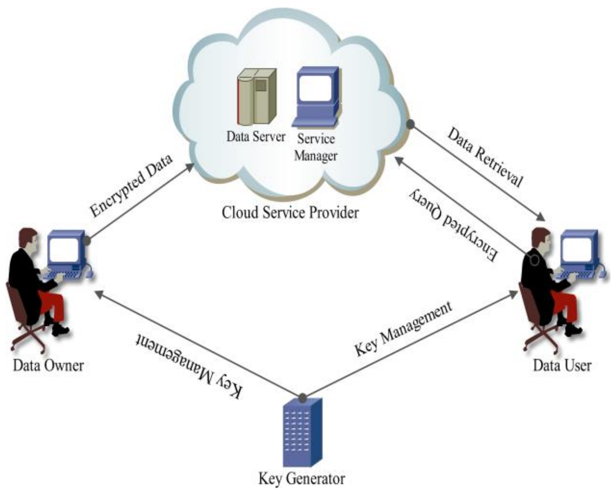

Fig 1. Searchable Encryption Architecture

A. Security Requirements

In general, the following requirements should be satisfied when constructing a searchable encryption scheme:

1. Retrieved data: Server should not be able to distinguish between documents and determine search contents.
2. Search query: Server should not learn anything about the keyword being searched for. Given a token, the server can retrieve nothing other than pointers to the encrypted content that contains the keyword.
3. Query generation: Server should not be able to generate a coded query. The query can be generated by only those users with the relevant secret key.
4. Search query outcome: Server should not learn anything about the contents of the search outcome.
5. Access patterns: Server should not learn about the sequences and frequency of documents accessed by the user.
6. Query patterns: Server should not learn whether two tokens were intended for the same query.

B. Design Approaches

Searchable encryption scheme can be built using either a non-keyword based approach or an index/keyword based approach. In the non-keyword based approach, the scheme scans the entire document word by word to find out the word W of interest. This provides the functionality to search any words in the document. However, it takes a long search time for a large number of document set. On the other hand, index/keyword based solution builds up an index, for each word W of interest and lists out the corresponding documents that contain W. This provides a faster search operation when the document set is large. However, storing and updating the index can be an overhead [3].

From the viewpoint of cryptographic algorithm selection, the SE scheme can mainly be modeled using either asymmetric/public key or symmetric/secret key setting. In the following, we briefly discuss the difference between these two settings.

1. Asymmetric Searchable Encryption (ASE)

In this setting, a user encrypts the data using asymmetric/public key encryption schemes (e.g. RSA) before outsourcing it to the cloud server. This setting is appropriate for a scenario where the user searching over the data is different from the user who generates it. For example, multiple users can use the public key of a certain user to encrypt and upload the data, however; only the owner with the corresponding private key can generate the search token and therefore can perform a search over the encrypted data. The main advantage of ASE is its functionality whereas the drawback is inefficiency. ASE schemes can be used in a larger number of settings since the reader and writer can be different for this case. On the other hand, all known ASE schemes require the evaluation of pairings on elliptic curves which is a relatively slow and costly operation compared to the hash functions or block ciphers [1].

2. Symmetric Searchable Encryption (SSE)

In this setting, a user encrypts the data using symmetric/private key encryption schemes (e.g. AES) before outsourcing it to the cloud server. This setting is appropriate when the user that searches over the data is also the one who generates it. The main advantage of this setting is the efficiency, but it lacks of functionality as it can only be used for a single user scenario. Moreover, most of the SSE schemes leaks the access patterns. The encryption is efficient because most SSE schemes are based on symmetric primitives like block-ciphers and pseudo-random functions and requires very less computational overhead. The SSE scheme was first proposed by Song et al. [8] which provides techniques for remote searching over encrypted data using symmetric key primitives. Later, security notions of SSE schemes were revisited and stronger security definitions were provided by Goh [4], Chang et al. [9] and Curtmola et al. [10].

III. RELATED WORK

In this section we discuss a brief summary of related work dealing with searchable encryption schemes. The first searchable encryption scheme based on public key algorithm was proposed by Boneh et al. [5]. This is known as the PEKS scheme which uses the public key of a user to encrypt and store the data in the server, and allows an authorized user with the private key to search and decrypt the corresponding content. This is a keyword based scheme ensuring faster search functionality; however, limiting the search capability. Also, the scheme is computationally expensive and it reveals the user access pattern. An extension of the PEKS scheme was proposed by Liu et al. [6]. This also uses the public key primitive to support the keyword searching on encrypted data. This scheme allows the cloud service provider to participate in the partial decipherment and claims to have reduced computational overhead on the client due to this partial decipherment.

Another variant of the PEKS scheme called iPEKS was proposed by Tseng et al. [7]. This scheme aims to accelerate the search time by looking into the previously searched keywords. For this, the cloud service provider caches the previously searched keywords to avoid the search on all the stored ciphertexts. However, this comes with the tradeoff of large storage overhead.

The first searchable encryption scheme based on symmetric key primitive was proposed by Song et al. [8]. This provides a non-keyword based solution. However, this scheme can search for only fixed length words and also the search time is linear in document size, since it needs to scan the whole document to complete the search. The scheme is too slow when searching for a large number of documents. Goh [4] addresses some of the issues of the above scheme by introducing the concept of a secure index. This scheme generates a search index which can be used to locate the encrypted content. Later, the security notions of searchable symmetric encryption were revisited and stronger security definitions were provided by Curtmola et al. [10]. This is a very simple scheme based on keyword indexing approach.

Besides these there are some other researchers who have addressed the problem of Searchable Encryption. We will now discuss them one by one. Mikhail Strizhov [11] have introduced three schemes to carry search on encrypted data, the first is a basic scheme which provides provable secrecy, the second scheme provides controlled searching and finally third scheme provides hidden searches. Also a fourth scheme is also mentioned to remove some inadequacy in the third and second scheme. Besides these scheme other practical consideration to implement searching on encrypted data are also discussed.

Jian Li et al [12] have developed a new architecture, TEES as an initial attempt to create a traffic and energy efficient encrypted keyword search tool over mobile cloud storages. They have started with the introduction of a basic scheme and then compared to previous encrypted search tools for cloud computing and showed their inefficiency in a mobile cloud context. Then they developed an efficient implementation to achieve an encrypted search in a mobile cloud. The security study of TEES showed that it is secure enough for mobile cloud computing, while a series of experiments highlighted its efficiency. TEES is slightly more time and energy consuming than keyword search over plain-text, but at the same time it saves significant energy compared to traditional strategies featuring a similar security level. Thus time and energy should be reduced and also multi-keyword searching feature should be also implemented in TEES.

Baojiang Cui et al [13] have proposed the concept of key-aggregate searchable encryption (KASE) and construct a concrete KASE scheme. In a KASE scheme, the owner only needs to distribute a single key to a user when sharing lots of documents with the user, and the user only needs to submit a single trapdoor when he queries over all documents shared by the same owner. However, if a user wants to query over documents shared by multiple owners, he must generate multiple trapdoors to the cloud. However this approach generates a lot of trapdoors which is not efficient and hence work should be done to reduce the number of trapdoors. Also KASE scheme cannot be applied on federated clouds.

IV. PROPOSED WORK

Currently developed techniques for searchable encryption focus on single keyword search and provide search functionality which is limited to the owner of the data that is being searched. Also with the use of symmetric and asymmetric encryption techniques there are various problems arising, concerning the security of data stored on cloud and also about the access control. Thus a system should be developed which can be able to search the data stored in encrypted form by multiple users.

The proposed system will enable multiple users belonging to same group to perform multi-keyword search on the data shared in same group. To achieve this symmetric encryption scheme is proposed which will encrypt the data before outsourcing it to cloud. For this cryptographic server (CS), a secured entity is proposed which will be residing in the cloud. The CS is responsible for security operations, such as key management, encryption, decryption, the management of the Access Control List (ACL) for providing confidentiality, and secure data forwarding among the group. ACL is used by CS to manage access rights.

In the proposed system, index of keywords is created for the documents which are going to be outsourced on cloud. The keywords which are stored in the index are encrypted using the same key by which the data is encrypted. Thus when a user submits the query, it is also encrypted with the same key and then it is searched in the index created for each document. If the match is found the user is returned with the relevant set of documents.

When multiple documents are matching the search query then the documents are ranked based on their relevancy. For calculating the relevancy we are going to first find term frequency (tf) and document frequency (df).Term frequency is number of times a given term appear in that document, whereas document frequency is number of documents that contains the given term. In information retrieval system tf-idf weighting is calculated for each document with respect to the query which is called as the score of that document, and based on these scores the documents are ranked. Document is then decrypted before retrieving it to the user. Before searching the index the CS checks the ACL for user's access rights. This is a simple theory proposed with the use of Searchable Symmetric Encryption for searching over encrypted data which uses index mechanism. The proposed theory will full fill all the security requirements discussed in Section II.

V. IMPLEMENTATION DETAILS

A. Architecture of SSE

In SSE a user encrypts the data using symmetric/private key encryption schemes (e.g. AES) before outsourcing it to the cloud server. Figure shows the architecture of SSE. SSE is mostly suitable where the user who is outsourcing the file is searching those files. But in the proposed architecture we are going to SSE scheme where there are multiple users. For this the SSE is divided into three entities: the user, cryptographic server and finally the cloud service provider (CSP).

- *User*

In this scheme user can be a data owner or the individual who is searching over the encrypted data. It is not necessary that the owner of the document can only search the documents, other users whom the data owner have shared the files can also search over those shared documents. For sharing the files among other users the data owner need to create a group and share the files into the group.

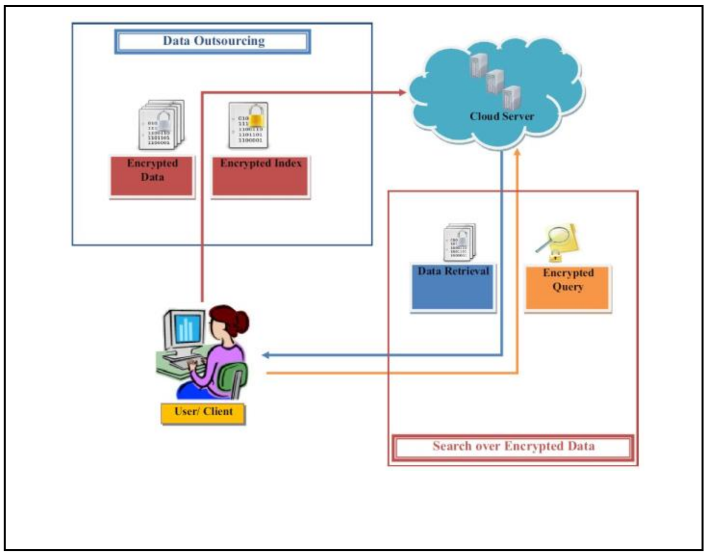

Fig 2. Architecture of SSE

- *Cryptographic Server (CS)*

Cryptographic Server (CS) is a third party server which is a mediator between user and CSP. No request from user goes directly to CSP; it always goes to CS. CS then checks the request, processes it accordingly and serves back the response to the user. Thus we can say that user never communicates with the CSP. CS is a trusted entity in proposed scheme which is responsible for various tasks like:

i. Registering new user into the system,
ii. Authenticating the registered user,
iii. Managing Groups,
iv. Creating encrypted index of the file to be outsourced,
v. Encrypting the document being outsourced,
vi. Sending encrypted index and file to CSP,
vii. Encrypting the search query,
viii. Getting the search results from CSP,
ix. Calculating the score of each document and ranking them,
x. Sending search results to user,
xi. Decrypting the file to be downloaded.

- *Cloud Service Provider (CSP)*

CSP is responsible to storing the encrypted index and encrypted file. Files are directly stored into the space provided by CSP. But to store index we have made use of MongoDB, which is a NoSQL database. MongoDB is very reliable and scalable database and there is no form of overhead even though millions of records are stored in it.

B. Creating Users

To be able access the system for a user, he/she first need to register into the system. For registering, a new user may have to submit some details like his name, address, email address and password. These details are sent to CS which is stored into a MySQL database. Each user is assigned a unique ID which is also stored in the database. After registering the user can now access the system by providing appropriate credentials which he/she submitted while registering. After logging into the system a session is created and then user can upload, download and search the documents which he/she has outsourced onto the cloud or which are shared with him/her.

C. Group Management

Once user has registered into the system the files uploaded by him/her are private. If he/she wants to share the file he/she needs to create the group and add other users into it. When a group is created it is stored in MySQL database which is on CS. It contains the details like group name, creator of group, and the list of members. Also every group is given a unique ID which is stored in database. When a group owner adds a new member an invite is sent to the user and when that user accepts the invite the user gets added into the group. Member of a group can leave or group admin can remove the user from the group at any time, after leaving/removing from the group user is no longer accessible to the files which are shared in that group, this feature is called forward access control. Also when a new user is added into the group he cannot access the files which are added before his/her joining, this feature is called backward access control. All the group management functions which we have discussed above are implemented by CS.

D. Uploading a Document

Figure 3 shows a flowchart which describes the intermediate operations that are performed when user uploads a new document. At first it is checked whether it is a document or not. If it is document then both index and document encryption starts. In index encryption firstly keywords are extracted, and then stopwords are removed. After that frequency of each keyword in that document is analyzed and an encrypted index is created which is stored in MongoDB. In document encryption simply the document contents are encrypted and stored in cloud.

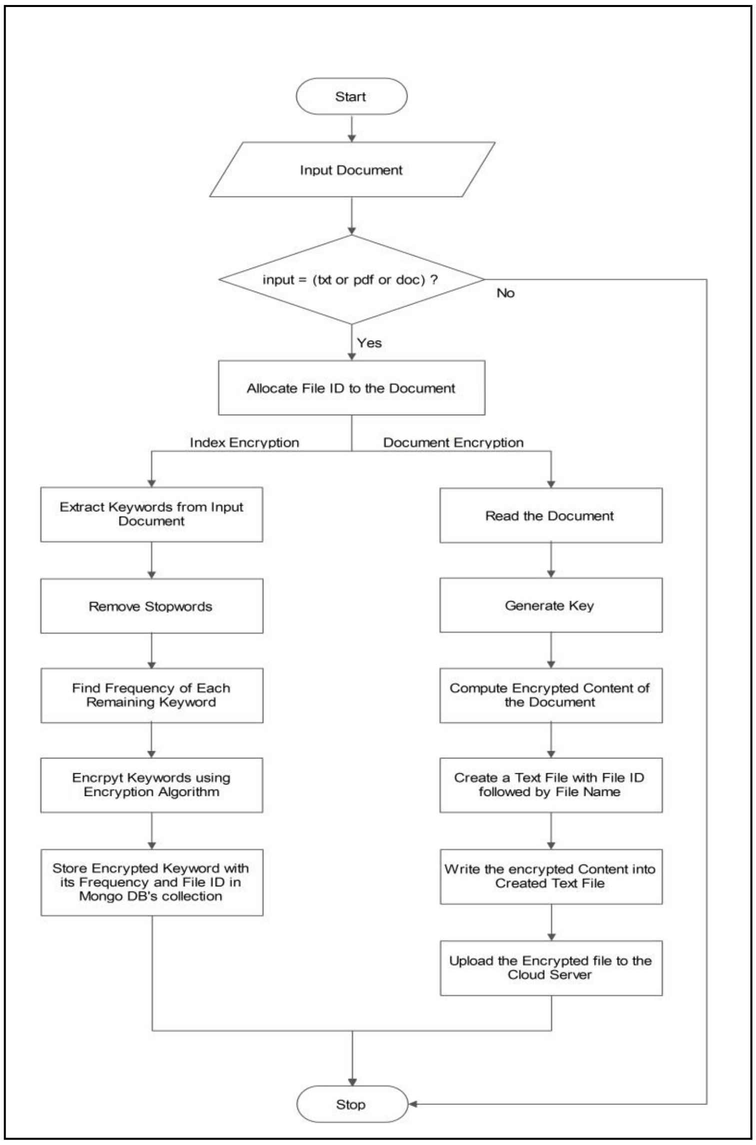
Start
Input Document
input = (txt or pdf or doc) ?
No
Yes
Allocate File ID to the Document
Index Encryption
Document Encryption
Extract Keywords from Input Document
Remove Stopwords
Find Frequency of Each Remaining Keyword
Encrpyt Keywords using Encryption Algorithm
Store Encrypted Keyword with its Frequency and File ID in Mongo DB's collection
Read the Document
Generate Key
Compute Encrypted Content of the Document
Create a Text File with File ID followed by File Name
Write the encrypted Content into Created Text File
Upload the Encrypted file to the Cloud Server
Stop

Fig 3 Flowchart for Upload Operation

E. Searching Documents

Figure 4 shows the flow while performing the searching operation. Firstly the stopwords are removed from the query and then it is encrypted. Now the encrypted query terms are matched with the index stored in MongoDB. If the contents are matched then term frequency and document frequency is calculated. From that the score of each document is calculated. And based on the score the results are ranked and served to the user.

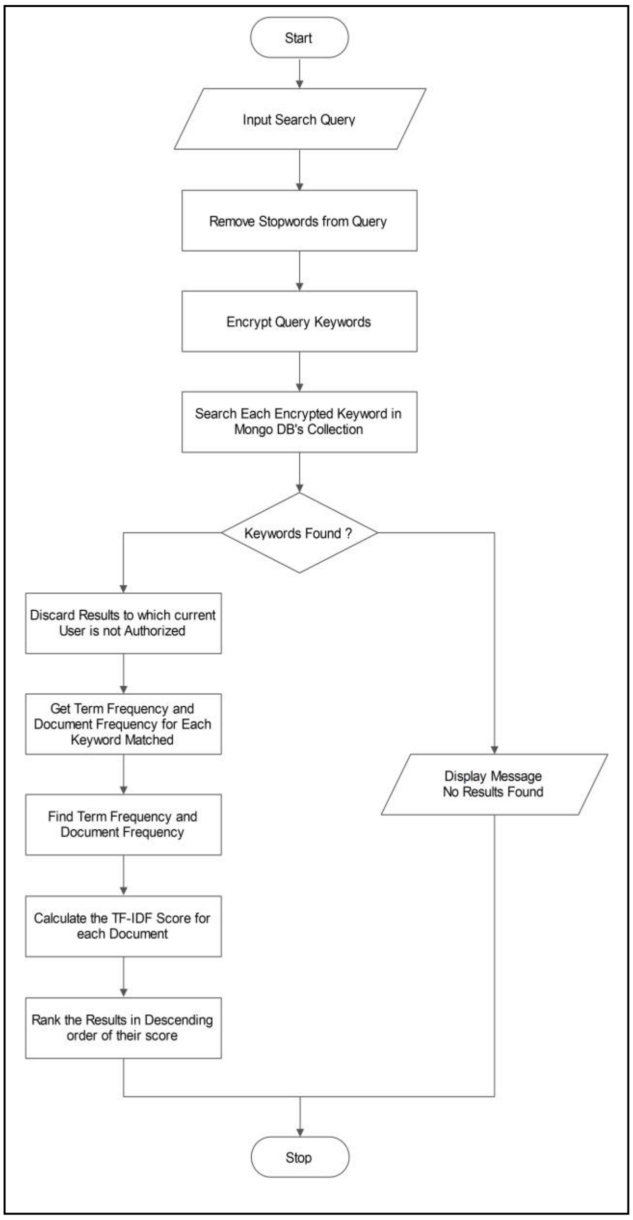
Start
Input Search Query
Remove Stopwords from Query
Encrypt Query Keywords
Search Each Encrypted Keyword in Mongo DB's Collection
Keywords Found ?
Discard Results to which current User is not Authorized
Get Term Frequency and Document Frequency for Each Keyword Matched
Find Term Frequency and Document Frequency
Calculate the TF-IDF Score for each Document
Rank the Results in Descending order of their score
Display Message No Results Found
Stop

Fig 4. Flowchart for Searching Operation

VI. EXPERIMENTAL RESULTS

- *Performance Analysis*

To measure the performance of the system a set of 10,000 keywords is used. All the keywords are unique and not repeated. While providing as input these keywords are not submitted one at a time instead they are divided and then submitted so as to measure the variations in the results which are recorded.

Figure 5 the Upload time required. We can see that it takes around 2 msec of time to upload 2,000 keywords, whereas to upload 10,000 keywords it takes 3.8 msec which is very small as compared to 2 msec.

Figure 6 shows index creation time which means the time required to extract keywords, find the frequency of each keyword and then encrypt the keyword and then create an index. We can see there is not much difference between time required for creating an index of 2,000 keywords and 10,000 keywords.

Figure 7 shows encryption time required to encrypt the files containing the samples. We can see that the encryption time for files is very less which is very good.

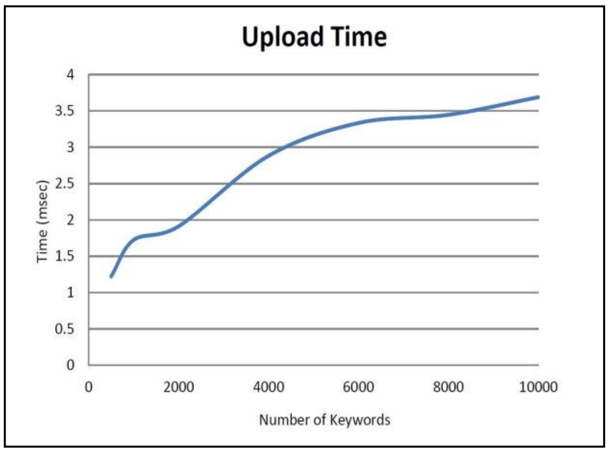

Fig 5. Upload Time

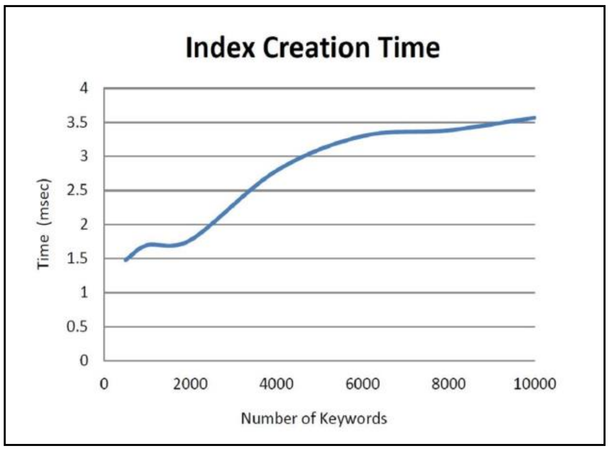

Fig 6. Index Creaation Time

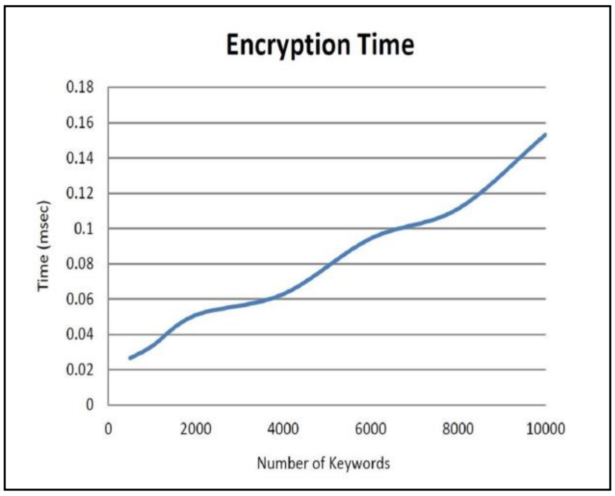

Fig 7. Encryption Time

Figure 8 shows the searching time required for 2,000 keywords to 10,000 keywords. We can see that the time required is almost same for any range of keywords. Thus we can say that even if the index size grows the searching time will not be affected greatly.

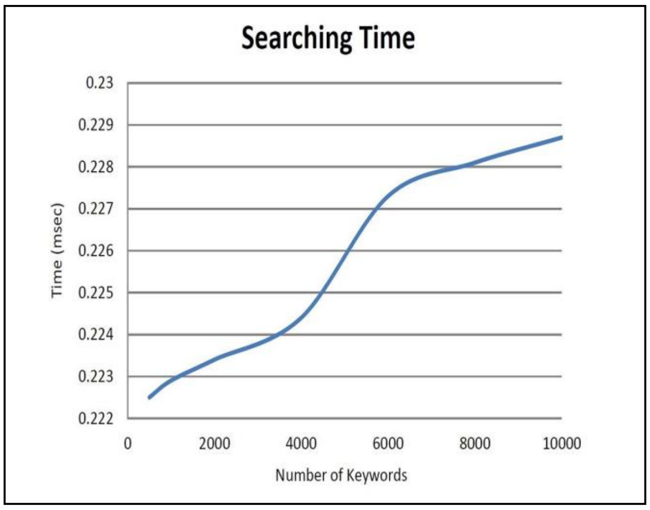

Fig 8. Searching Time

- Security Analysis

For encryption and decryption the proposed scheme uses AES algorithm with the key size of 256 bits. And from the research carried till date there is only way one can break the AES that is by using brute force attack. But practically it is impossible to perform brute force attack because it will require 2200 operations to break the AES algorithm. The largest successful brute force attack carried is against 64 bit key. Thus we can say by making use of AES the contents are secure and data confidentiality is achieved.

All the operations are performed on encrypted contents; hence there can be no leak of information. Also CSP is not able to learn any pattern from search queries because they are also encrypted. And also even if any outsider is able to get access to cloud he/she still cannot use that data, and decrypting that data without key is like impossible thing till current date.

Also the issue of forward and backward access control which was arising while joining and removing the user from the group is also solved by making access control list. Thus finally we can say the proposed system is secure and provides privacy preserving search.

- Searching Complexity

The search complexity of proposed technique is O(log8192n). We can see that the base of log is 8192 that is because the index is stored in MongoDB, and MongoDB stores the record using B-Tree where the bucket size of B-Tree is 8192. Thus first 8192 records are present on the first level of B-Tree and further records are placed in second level and so on. Thus we can say that the searching time is very fast.

VII. Conclusion

Proposed system uses Searchable Symmetric Encryption (SSE) for searching over encrypted data using secure indexes. The system guarantees to be secure and also provides privacy preserving search. The system uses a third party trusted server called as Cryptographic Server (CS) as a mediator between user and the Cloud Service Provider (CSP). To share the documents among other users the system provides the functionality to create groups and also provides the search functionality over shared documents. The search support multi-keyword query and also generates ranked results based on the score of the documents. The system also deals with the issue of forward and backward access control.

Current system do not support wild card queries, it can used for substring search. Also in current system if user enters wrong keywords in search query unknowingly then no results are shown to user. For such situation the system should be fault tolerant and should correct the search query on its own. Boolean expression based query can be used to get more results and better frame the search query. We left all these features as a future work of the proposed system.

REFERENCES

[1] Kamara S, Lauter K, "*Cryptographic cloud storage*", Financial Cryptography and Data Security, 2010, LNCS 6054. Springer, Berlin, Heidelberg, pp 136–149.0

[2] Hacigümüs. H, Iyer B, Li C, Mehrotra S, "*Executing sql over encrypted data in the database‑service‑provider model*", In: Proceedings of SIGMOD, ACM, 2002,pp 216–227.

[3] Md Iftekhar Salam, Wei‑Chuen Yau, Ji‑Jian Chin, Swee‑Huay Heng, Huo‑Chong Ling, Raphael C‑W Phan, Geong Sen Poh, Syh‑Yuan Tan and Wun‑She Yap, "*Implementation of searchable symmetric encryption for privacy‑preserving keyword search on cloud storage*," Springer Open Journal on Human Centric Computing and Information Sciences, 2015, DOI 10.1186/s13673‑015‑0039‑9.

[4] Goh EJ, "*Secure indexes*," In: Cryptology ePrint Archive: Report 2003/216.

[5] Boneh D, Crescenzo GD, Ostrovsky R, Persiano G, "*Public‑key encryption with keyword search*," Advances in Cryptology EUROCRYPT, LNCS 3027. Springer, Berlin Heidelberg,2007, pp 506–522.

[6] Liu Q, Wang G, Wu J, "*Secure and privacy preserving keyword searching for cloud storage services*", Journal of Networkng and Computer Applications (JNCA),2012, pp 927–933.

[7] Tseng FK, Chen RJ, Lin BS (2013), "*iPEKS: Fast and secure cloud data retrieval from the public‑key encryption with keyword search*," International Conference on Trust Security and Privacy in Computing and Communications, IEEE, 2014, pp 452–458.

[8] Song DX, Wagner D, Perrig A, "*Practical techniques for searches on encrypted data*," Proceedings of the IEEE Symposium on Security and Privacy, IEEE, 2000, pp 44–55.

[9] Chang YC, Mitzenmacher M, "*Privacy preserving keyword searches on remote encrypted data*", Applied Cryptography and Network Security, LNCS 3531. Springer, Berlin Heidelberg, 2005, pp 442–455.

[10] Curtmola R, Garay J, Kamara S, Ostrovsky R, "*Searchable symmetric encryption: improved definitions and effi‑ cient constructions*", Proceedings of the 13th ACM conference on Computer and communications security, ACM, 2006, pp 79–88.

[11] Mikhail Strizhov, "*Towards a Practical and Efficient Search over Encrypted Data in the Cloud,*" IEEE International Conference on Cloud Engineering, 2015.

[12] Jian Li, Ruhui Ma, Haibing Guan, "*TEES: An Efficient Search Scheme over Encrypted Data on Mobile Cloud*," IEEE Transactions on Cloud Computing, DOI 10.1109/TCC.2015.2398426,2015.

[13] Baojiang Cui, Zheli Liu and Lingyu Wang, "*Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage*," IEEE Transactions on Computers, DOI 10.1109/TC.2015.2389959, January 2014.

[14] Rahul Lanjewar, Gaurav Pande, "*Implementation of AES-256 Bit: A Review*", Inventi Rapid: Information Security, Vol. 2015, Issue 3, 2015.

[15] Peter Mell, Timothy Grance, "*The NIST Definition of Cloud Computing*", National Institute of Standards and Technology, Technical Report, 2011.

[16] CSA, "*Security Guidelines for Critical Areas of Focus in Cloud Computing v3.0*", Accessed on 2011, [Online] Available https://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf.