A Novel Technique for Relevant Data Retrieval from a Set of Encrypted Documents

Giri S M
Assistant Professor
Department of Computer Science & Engineering
Musaliar College of Engineering and Technology, Kerala, India

Abstract— Data owners can outsource their data in cloud servers rather than purchasing their own storage devices. The Amount of trust provided by the third party cloud server will threaten security of our data as anyone who has access to the data server can view our sensitive information. Solution is to encrypt the data before outsourcing it. Searching on encrypted domain is a prominent research problem. Usually a search request will prompt the server to upload the entire file to the user and user has to decrypt the whole data at its local host. This is not an efficient solution for remote storage as it needs decrypting the entire data for retrieving a small part from it. Searchable encryption algorithms can be used to search on the cipher text. When a search request arrives then the cloud server attempts to search on the encrypted domain making a simpler solution. This paper introduces a new technique to search on the encrypted domain.

Keywords: Cloud, AES, Hash-map, tf-idf

I. INTRODUCTION

Due to the increased popularity of mobile devices such as smart phones, laptops .., are used by many people for various computing purposes. The ad- vantage of mobile cloud computing makes it possible to access information stored anywhere in the world with an Internet connection. There are powerful computing platforms available in the remotely placed servers, the mobile cloud enables the users of smart phones to access the data through a wireless connection. Mobile commerce, mobile learning, mobile health care are widely used mobile applications, in this situations the data that the user wants are stored in the remote server placed somewhere around the world. The users only want to authenticate their self to the server and then they can start accessing the data. Thus the mobile applications reduces the computing capability in accessing information.

Cloud computing is a widely accepted technique for many purposes such as data storage, website hosting, application running etc. Storage of large amount of data in a private machine incurs extra effort for the data management and maintenance.

So data owners now a day's stores the large amount of data in cloud by availing storage as a service rather than purchasing their own storage devices. Data sharing is fast and convenient in cloud. If the data user uploads the data directly into the cloud then any user having access to the storage device can view our data. So it is not possible to use storage as a service if our data is sensitive as it threatens the confidentiality and Integrity. The solution to this problem is to encrypt the data before storing it in the cloud.

Suppose user A has a set of documents with him and he wants these set of documents to be stored in the cloud so that his friends or some other authorized users can use it. But most of his data is sensitive so he cannot outsource it directly to the cloud as it is. He encrypts the data and then outsources to the cloud. Suppose an authorized user wants a small portion of the data only then he needs to search, in order to achieve this we need to design a scheme so that it should give the required encrypted data and it should not reveal any other data.

Encryption may be a powerful technique for safeguarding confidential information keep on Associate in Nursing un- trusty server, like in cloud computing. Encoding may be a technique for remodeling info in such the way that it becomes unclear. So, even though somebody is in a position to achieve access to non-public information, they seemingly won't be ready to do something with the information unless they need sophisticated, overpriced code or the initial information key.

Since the data in the cloud is encrypted, legitimate users should decrypt it before using the data. In the naive approach the user takes the entire data from the server and decrypts the data at his local machine for using it, but this is not practical when the data is too large and the user only wants a subset of the whole data stored in the cloud. In such a situation, searching the required data in the server and making that data only available to the data user is a good solution

Searching is done by the cloud Server. Only required data is given tothe device with less computational capability, so Smart Phones only have to decrypt the required information. Searching on encrypted data is not an easy task, searchable encryption techniques make the user to search on cipher text.

II. RELATED WORK

Searchable encryption allows users to search on encrypted data stored in cloud servers, searching on encrypted domain is a difficult task, some of the existing approaches are discussed below.

Inverted index based approach [1] allows search on cipher text by keeping an inverted index of all the keywords in the document. Data owner first creates a public key private key pair, an invertible matrix and random permutations. For each keyword and document the data owner creates an inverted index and then represents each keyword as a polynomial in terms of the document tag. The index is created by encrypting the coefficients of the polynomial created before using the public key. Data owner encrypts dictionary matrix consisting of all the keywords in the document, outsources both these to the cloud server. All the authorized users will be having a trapdoor given by the data owner for searching, keywords in the query also converts to a polynomial form based on the key word tags, when a query reaches the server it checks the trap door for authorization and then compares it with the encrypted index, if a match is found

then it will retrieve the encrypted content back to the data user, he then needs the help of data owner to decrypt the content. Advantage of this method is that the cloud server can't infer anything about the content of the data as everything is in polynomial form. Computational overhead is the disadvantage.

[2] [DS-PEKS], two servers are used. First a key generation algorithm generates key pairs for front and back servers. The keywords are encrypted using the public keys of front server and back server respectively. Up on receiving the query from the receiver, the front server preprocesses the trap door and all the cipher texts using its private key, and then sends some internal testing states back to the back server with the corresponding trap door and hidden cipher text. The back server can now decide what all documents are required by the receiver and using its private key it processes the internal testing states given by the front server. After successful processing the back server retrieves required cipher text. The advantage is that this method prevents keyword guessing attack by the malicious server.

In [3] all the keywords are associated with a linked list constructed like a node consisting of identifier of the files where the key word appears and a state visibility parameter to indicate whether the file is public or private. For more privacy each node is randomly stored in an array and encrypted using different keys. They also keep an access control list to check whether the request made by the user has the right to access the file. During authorization phase the data owner provides a private key for proving his identity to the cloud server and two more keys for generating trap door and decrypt the cipher text. All users have a unique secret key, trap door is unforgeable are the advantages.

In [4] four parties are involved, data owner, data user, cloud server, a trusted third party. The data owner chooses keywords from his collection of plain text documents to be outsourced for the index creation. In the index, each keyword has a pruning code and a secret code. After encrypting the data and index, both outsources to the cloud server. Unless a valid trapdoor index will not retrieve anything. After getting the query the server searches in the index for a match by comparing the pruning code of all the keywords, a candidate list is created with the result and it sends to the trusted third party. The trusted third party compares the trap door with the secret code of each keyword in the list for finding an exact match, once match is found the third party sends the cipher text corresponding to the query back to the server. The advantage is that it preserves the privacy of the trap door, but third party trusting is the disadvantage.

In [5] they provide an easy search of query by introducing a compression function, this function divides the keywords in to different sets. Each key word in the document belongs to different sets, the set index is returned by the compression function corresponding to each keyword. Up on receiving a query the server finds the set index using the compression function and checks only in that set for a match instead of checking the entire key words in the index. More functional but less effective

III. Approach and Methodology

The entities in proposed system are data owner, cloud storage server and clients. Owner encrypts the data and stores the data in storage server. The data owner outsources the data in the server after encryption. The user of the data has to access the data from the server, but it is in an encrypted form, most probably he may not require the entire data, so he needs to perform a search on the encrypted data. Searching on encrypted data is a difficult task, in this paper we introduce a new technique in which the data owner not only keeps the encrypted data but also stores an efficient searchable index with least searching time along with the data to be outsourced so that the data user will get the required part of the encrypted text. In this approach server handles all computing when a query from the user is reached. The server searches the query with the index for a match, once a match is found the server retrieves the data and giving it back to the user.

A. Problem Statement:

Many data owners now a day's prefer to store their data in cloud in an encrypted format, users sometimes need only a portion of the encrypted data so they search what they want but searching on this is difficult. Efficient searching on the encrypted domain and retrieving only the relevant information instead of the whole files is the problem addressed in this work.

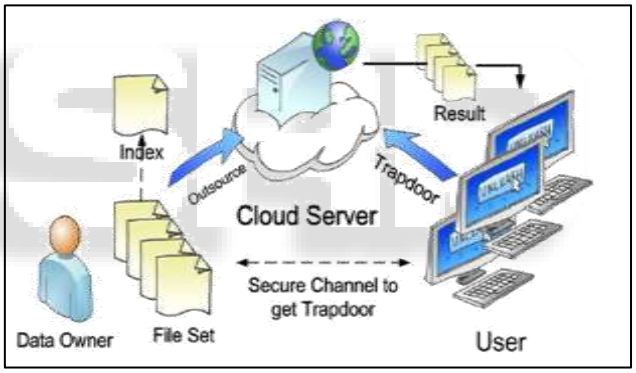

Fig. 1: Proposed model

IV. Solution Approach

Proposed solution approach contains mainly three stages.

- Document Encryption.
- Index Construction.
- Searching On Index.

A. Document Encryption:

Cloud server allows data to be outsourced at any time and from anywhere. But if the data to be outsourced is sensitive then the data owner has to convert the data in another form which cannot be understood by an unauthorized person. So data owner go for encryption before outsourcing the data in the cloud storage. The basic function of encryption is basically to translate traditional text into cipher text. Encryption ensures that data doesn't get read by the unauthorized people, but can also ensure that data isn't altered in transit, and verify the identity of the sender. There are two different basic encryption methods: Symmetric Key encryption and Public Key encryption.

1) Symmetric Key Encryption:

Symmetric key encryption algorithm uses same cytological keys for both encryption and decryption of cipher text.

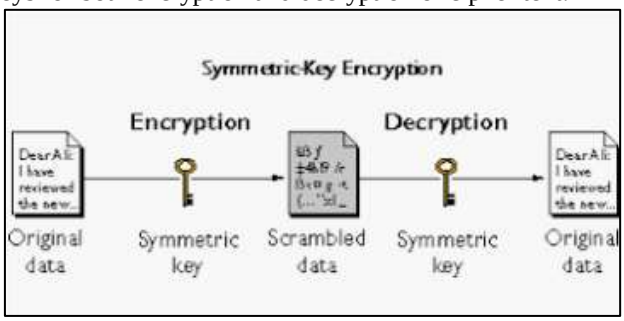

Fig. 2: Symmetric key encryption

2) Public Key Encryption:

Public key encryption algorithm uses pair of keys, one of which is a secret key and one of which is public.

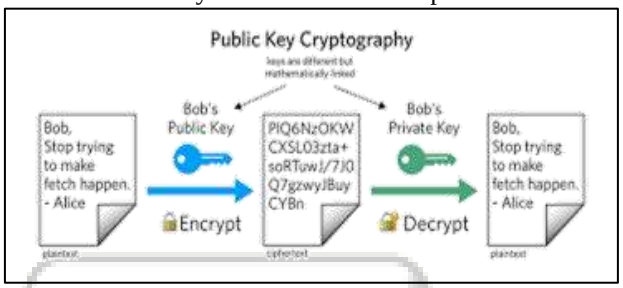

Fig. 3: Public key encryption

These two keys are mathematically linked with each other. Any public-key encryption schemes is bound to increase the size of the data that it enciphers: if it did not, there would be a single cipher text for any given plain text, and thus an adversary could test if the plain text is a certain value, simply by enciphering that value (using the public key) and comparing to the cipher text. Public-key encryption schemes typically increases the cryptogram size. Symmetric key algorithms provide large security with less key length.

One of the popularly used symmetric key algorithm is AES. One of the popularly used radially symmetrical key algorithmic rule is also AES. The Advanced encoding commonplace or AES could be a symmetrical block cipher employed by the U.S. Government to shield classified info and is enforced in code And hardware throughout the globe to cypher sensitive information. Symmetric key symmetric block cipher, 128-bit data, 128/192/256-bit keys, Stronger and faster than Triple-DES. Here the data owner encrypting the documents using AES algorithm before outsourcing.

B. Index Construction:

The second phase of the solution approach is Index construction. The index creation is done by the data owner, index reduces the searching time for a user when a query is given. Here the index is based on frequency of words occur in the document collection. We take those many number of words having highest weight in the document for index creation. But some words called stop words will occur everywhere in documents removed from the document collection because their weight or frequency adds no value for index creation. So the data owner first eliminates

Fig. 4: Paragraph retrieval

The stop words such as "is, has," from the data to be outsourced. Then finding the weights of the remaining terms for the index creation.

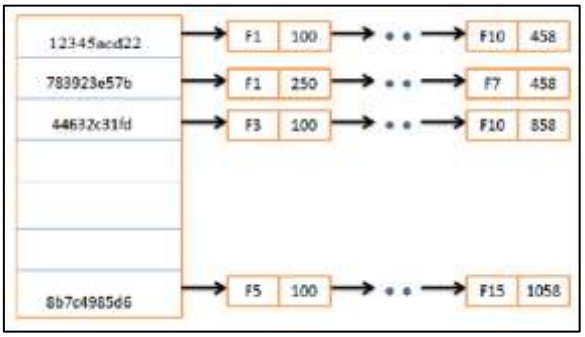

Fig. 5: Hash map

1) Algorithm 1: Algorithm for index creation and out sourcing the data

Input: A document set S= D1 D2. Dn
Output: Encrypted S and a searchable index for *S*

1) Step 1: Encrypt S using a key K
2) Step 2: for each document Di ε S do
3) Step 3: Bi[n] ← RemoveStopWords (Di)
4) Step 4: Ci[n] ← TFIDF (Bi[n])
5) Step 5: Read the whole file S and populate Ci[n] with word positions for all values of Ci[n] > threshold
6) Step 6: Set Index ← (hash÷ Size)+1
7) Step 7: Set position ← last˙ (index)
8) Step 8: for each i do
9) Step 9: Set Position n˙ ext← Ci[m]
10) Step 10: Return Hash Table and encrypted text

2) Algorithm 2: Algorithm for Searching

Input: keyword in the query
Output: Encrypted output

1) Step 1: Find the index corresponding to the Query using Index ← (hash÷ Size) +1
2) Step 2: while Index˙ Next = NULL do
3) Step 3: if index ˙next ˙key== query keyword then
4) Step 4: output the corresponding paragraph Using the positions else not found;

3) Term Frequency:

Usually we give weights for each term in a document based on how many times a word occur in the document. The term frequency of a term t in a document usually denoted by tf_t,d tf_t,=(no of occurrence of term "t" in document "d") / (total no of terms in d)Term frequency will only tell how important a word is in a document this alone is not a good measure to find number of words which are scattered in the document collection for creating the index.

4) Inverse Document frequency:

Sometimes we some word's term frequency will be very high if it occurs many times in the document collection in that case we need to reduce weight of that word to determine its relevance. For example the word auto will appear many times if we consider the automobile manufactures records. In such cases the weight of term frequency should be reduced for high collection frequency. An important measure in such cases is to scale down

The weight, Inverse document frequency is a good measure in such cases. Inverse document frequency (idf) of a term tis denoted by idf_t

idf_t = log [(total number of documents)/(no of documents contain the term t)]

$idf_t = \log [N/df_t]$

IDF of a word gives a good idea whether it is common in all documents or rare in all documents. The idf of a frequently occurring term or word will be less and that of rarely occurring word will be high.

5) Term frequency-Inverse document frequency (TF-IDF):

If we combine the term frequency and document frequency together then we will get a composite weight for all the terms in the document collection and this is a good measure to find the relevance of terms in a document collection for making an index for searching query terms.

$$tf\text{-}idf_{t,d} = tf_{t,d} \otimes idf_t$$

TF-IDF value of a term will be less if the term is occurs in many documents.

After finding the TF-IDF values of all the distinct terms in the document collection, we take a set of words having less tf-idf value as less number indicates more relevance. Since the index is also stored in the server, we either need to take the encrypted values of selected words or finding the hash value. SHA1 hash of these words are then taken for index construction. Index is built as a hash map for easy searching. The searching complexity in hash map is O (1) in average and best cases but *O* (*n*) in worst case. A hash map consists of a key value pair.

6) Hash Map:

The index is implemented as a hash map. A Hash Map contains values based on the key. It implements the Map interface and extends Abstract Map class. It contains only unique elements. It may have one null key and multiple null values. It maintains no order Hash Map maintains key and value pairs and often denoted

As Hash Map < Key, V value > or Hash Map < K, V >. Hash Map implements Map interface. Hash Map is similar to Hash table with two exceptions, Hash Map methods are un synchronized and it allows null key and null values unlike Hash table.

7) Key Value Semantics in Hash Map:

The purpose of a map is to store items based on a key that can be used to retrieve the item at a later point. Similar function will solely be achieved with a list in the limited case where the key happens to be the position in the list. In the hash map the key is the SHA1 code of the selected key words having less TF-IDF values. The value part is a linked list of positions of words in the cipher files. This makes the searching faster. Finally the encrypted set of documents and the index is outsourced to the server.

C. Searching

When the user enter a key word the SHA1 code is generated corresponding to the key word and is sent to the server. Server searches the key word in the index for a match. If a match occurs then the server retrieves the linked list for that key word which contains the cipher file name and the positions of the word in various files. Now the server knows the positions of keywords in the encrypted documents. Server now extract the paragraph of cipher text containing the given position by searching the cipher text for a new line (line break) above and below the position extracted from the list. After finding the first new line above and below the specified position it retrieves the cipher data in between and sent to user. If user is okay with the displayed paragraph he can download that file alone or search for another.

V. RESULT & EXPERIMENTATION

The proposed technique is implemented in Corei4 processor with 2GB RAM using Java Platform as front. The data set we used is Mien Kampf the auto biography of Hitler which is one of the most read text in the Internet. The data is of 1MB size. After implementing the proposed method we successfully execute a program for variety of key words and it outputs the real event paragraph in most cases.

VI. CONCLUSION

In this paper, a secure, efficient search scheme is proposed, which supports the accurate keyword search. We constructed special keyword index for searching. Index is maintained as a hash map with key and value. Value is a list of positions. Hah map returns the content in O (1) time. So the Retrieval is fast.

REFERENCES

[1] Bing Wan, Wei Song, Wenji Y. Thomas Hou,Inverted Index Based Multi-Keyword Public-key Searchable Encryption with Strong Privacy Guarantee, 2015 IEEEconference on computing.

[2] Rongmao Chen, Yi Mu,. Dual-Server Public-Key Encryption with Key- word Search for Secure Cloud Storage. IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 11, NO. 4,APRIL 2016

[3] Ayad Ibrahim, Hai Jin, Ali A. Yassin, Deqing Zou, Approximate Keyword-based Search over Encrypted Cloud Data,2012 Ninth IEEE International Conference on e-Business Engineering

[4] Bin Long, Dawu Gu, Ning Ding, and Haining Lu. On Improving the Performance of Public Key Encryption with Keyword Search. 2012 In- ternational Conference on Cloud Computing and Service Computing

[5] Mikhail Strizhov and Indrajit Ray,Multi-keyword Similarity Search Over Encrypted Cloud Data

[6] Joonsang Baek, Reihaneh Saavi-Naini,Willy Susilo Public Key Encryp- tion with Keyword Search Revisited

[7] Cheng-Kang Chu, Sherman S.M. Chow, Wen-Guey Tzeng, Jiany- ing Zhou,Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage,IEEE TRANSACTIONS ON PARALLEL AND DIS-

TRIBUTED SYSTEMS, VOL. 25, NO. 2, FEBRUARY 2014

[8] Bin Long, Dawu Gu, Ning Ding, and Haining Lu,On Improving the Performance of Public Key Encryption with Keyword Search,2012 In- ternational Conference on Cloud Computing and Service Computing

[9] Bharath K. Samanthula, Member, IEEE, Yousef Elmehdwi, and Wei Jiang, Member, IEEE,k-Nearest Neighbor Classication over Semanti- cally Secure Encrypted Relational Data,IEEE Transactions on Knowl- edge and Data Engineering Volume: 27 Year: 2015

[10] Li Chen, Li Chen, An Ecient and Privacy-Preserving Semantic Multi- Keyword Ranked Search over Encrypted Cloud Data,International Jour- nal of Security and Its Applications Vol.8, No.2 (2014), pp.323-332

[11] Seungmin Lee, Tae-Jun Park, Donghyeok Lee, Taekyong Nam, and Sehun Kim.Chaotic Order PreservingEncryption for Efficient and Secure Queries on Databases. IEICE Transactions on Information and Systems, E92-D (11):2207–2217, 2009.

[12] O. Pandey and Y. Rouselakis. Property preserving symmetric encryp- tion. In EUROCRYPT, 2012 Dawn Xiaodong Song

