International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 8, August 2014 **ISSN 2319 - 4847**

Homomorphic Encryption for Multi-keyword based Search and Retrieval over Encrypted Data

Jiss Varghese[1], Lisha Varghese[2]

[1] PG Scholar, Dept of Computer Science, Amal Jyothi College of Engineering, Kanjirappally, Kerala, India

[2] Assistant Professor, Dept of Computer Science, Amal Jyothi College of Engineering, Kanjirappally, Kerala, India

ABSTRACT

With growing communication networks and digital communication, data outsourcing has turn out to be an advanced data service for users to store sensitive data into a global storage space. Data owners can upload their private data into these space, and do many operations on them. Performing information retrieval tasks while preserving data confidentiality is a big challenge when data is stored in global space maintained by a third-party service provider. To avoid confidentiality problem, the sensitive data have to be encrypted before uploading into global space. This leads to a problem to perform efficient keyword based search and retrieval over these encrypted data. Ranking based on order preserving encryption (OPE) has statistic information leakage problem. In order to avoid this, a Two Step Searchable Encryption (TSSE) scheme has been proposed which supports multi-keyword based top-k data retrieval. It enables us to get the retrieval result as the most relevant files that match users' interest. It indicates that files are ranked in the order of relevance and only the files with high relevance are sent back to users. In TSSE, the concept of homomorphic encryption, relevance scoring, vector space model and key management are introduced. Since the search operation is performed over encrypted data, information leakage can be eliminated and data can be searched and retrieved efficiently.

Keywords: Homomorphic Encryption, Relevance Scoring, Vector Space Model, Data Retrieval

1. INTRODUCTION

Data outsourcing can be considered as an advanced data service for users to store sensitive data into a global storage space. The sensitive data is managed on remote servers maintained by trusted third party outsourcing vendors. The distributed nature of today's data management services give assurances to detect and correct faulty behaviour. This is relevant for outsourced data frameworks in which data owners place their sensitive data into specialized storage spaces. Data owners outsource their data without assurances of confidentiality and security. Confidentiality can be achieved by encrypting the data. But the challenge here is that how to enable search and retrieval over such encrypted data. Several searchable symmetric encryptions(SSE) [14] are available which enable both search and retrieval over encrypted data, but each have their own drawbacks. Traditional SSE schemes allow users to retrieve the ciphertext in a secure way, but most of them work based on Boolean keyword search [2], [13]. Boolean keyword search gives results based on whether a keyword exists in a file or not, without considering the relevance with the queried keyword of these files in the result. Some SSE schemes based on "order preserving encryption" breaches the privacy of sensitive information and allows only single keyword in search query [3], [9]. There are some multi-keyword based searching schemes which enables secure indexing and ranked searching [10], [11], but have issues on how to strike a balance between security and efficiency. So a new searchable encryption scheme called TSSE is been proposed, in which new technologies in cryptographic system and IR community are used. It enables us to get the retrieval result as the most relevant files that match users' requirement. It indicates that files are ranked in the order of relevance, and only the files with high relevance are sent back to users. In TSSE, the concept of homomorphic encryption, vector space model and key management are introduced. Since the search operation is performed over encrypted data, information leakage can be eliminated and data can be searched and retrieved efficiently.

2. BACKGROUND

Data owners generally share their outsourced documents with a number of users, and users want to retrieve only the files that they are interested in. One simpler way to do so is by keyword-based retrieval. The keyword-based retrieval deals with plaintext data, which means users may retrieve required files from a file set based on queried keywords. Since a limited number of operations are possible in ciphertext, it appears to be a complex task in such scenario. Several searchable symmetric encryption (SSE) schemes have been proposed which enable search and retrieval operations on ciphertext. The SSE schemes described in [13],[7] enable users to securely retrieve the ciphertext, but these schemes support only "Boolean keyword search". Documents are retrieved based on whether a searched keyword is present in a file or not, without considering the degree of relevance of these documents with the search query. The schemes presented

in [3],[9] show that they support top-k "single keyword based retrieval" under various scenarios. The schemes proposed in [10],[11] tries to solve the problem of multi-keyword based search over encrypted data. These scheme follows boolean representation for index and have the issue of how to strike a balance between security and efficiency. Thus to improve efficiency, the data files should be in encrypted form before outsourcing into server. Also it should allow only authorized users to perform search and retrieval over these encrypted data. It is favoured to get the retrieval result as the most relevant files that match users requirements instead of all the files containing that keywords. The files should be ranked in the order of relevance and only the files with high relevance are sent back to users. To improve efficiency, the entire system should be scalable which support multiple data owners and associated set of authorized users. The keys should be properly managed and should deal with search control and access control.

3. PROBLEM DEFINITION

The search method described in [9] introduces a new framework for confidentiality preserving rank ordered search and retrieval over large encrypted documents. They have used an order preserving encryption to encrypt documents and searchable index, and the keyword frequency is used as ranking criteria. The scheme give the retrieval result as the most relevant files instead of all the files containing that keyword. They follows OPE encryption which leads to statistic leakage problem(explained below). The search scheme described in [10] proposes a privacy preserving multi-keyword ranked search over encrypted documents. They allow multiple keywords in the search request and return documents in the order of their relevance to these keywords. They have used the similarity matching and inner product similarity to measure and evaluate the relevance of documents. This have employed Boolean representation for searchable index. Since OPE is used for encryption, the nature of plaintext is not hidden, so statistic analysis is possible here also[6].

3.1 Statistic Leakage

Although all data files, searchable indices, and user requests are in encrypted form before being outsourced, if the encryption method used is OPE, then the server can still obtain additional information through statistical analysis. The nature of plaintext like relative distribution of terms, co-occurrence of terms, searching patterns, access patterns etc are revealed even it is in encrypted form. The possible information leakage is termed as statistic leakage. According to [1], there are two possible statistic leakages: term distribution and inter distribution. The term distribution of term t is t's frequency distribution of scores on each file i (i € C). Here score means, a way to calculate the relevance of a term with respect to a file. The inter distribution of file f is file f's frequency distribution of scores of each term j (j € f). They can be deduced either directly from ciphertext or indirectly via statistical analysis over access and search pattern [9]. Here, access pattern refers to which keywords and the corresponding files have been retrieved during each search request, and search pattern refers to whether the keywords retrieved between two request are the same. These distribution information implies a similarity relationship among searched keywords and files. On the one hand, terms with same or similar term distribution may have simultaneous occurrence. To explain this, take a scenario which is given in [1]. Here, the term "states" are very likely to co-occur with "united" in an official paperwork from the White House, and their term distribution are very same in a series of such a kind of paperwork. Given that this paperwork is encrypted but term distribution is not concealed, once an attacker somehow cracks out the plaintext of "united", he can reasonably guess the term that shares a similar term distribution with "united" may be "states". On the other hand, files with similar inter distribution are always the same category, e.g., two student records from a college office surely are of same category, and they are very likely to share a similar inter distribution (such as the titles of each entries are the same). Therefore, this specificity should be hidden from an untrusted server.

4. RELATED WORK

The SSE schemes allows us to outsource the storage of its data to another party (a server) in a private manner, while maintaining the ability to selectively search over it. Traditional searchable encryption are investigated in [13],[14],[2] focus on security definitions and encryption efficiency, these are based on Boolean keyword retrieval without ranking. Swaminathan et al. [9] has built a framework for privacy-preserving top-k file retrieval, including secure indexing and ranking based on OPE. Wang et. al [3] explored the data retrieval over encrypted data. On the basis of SSE, these schemes employed one-to-many OPM to improve the efficiency. Here the security and retrieval accuracy are slightly weakened. Considering the large number of data users and documents, it is necessary to allow multi-keyword in the search request and return the most relevant documents in the order of their relevancy with these keywords. Cao et al.[10] made the first attempt to define and solve the problem of top-k multi-keyword retrieval over encrypted data. They employed coordinate matching and inner product similarity to measure and evaluate the relevance scoring. Hu et. al [11] have proposed homomorphic encryption to accure data privacy. These two schemes support Boolean representation for searchable index. Thus, files that contain queried terms have the same score, which leads to poor data utilization. Ranked search greatly enhances system usability by returning the matching files in a ranked order regarding to certain relevance criteria (e.g., keyword frequency) [3]. One simple ranked keyword search is implemented using the order-preserving

symmetric encryption (OPSE)[6]. The main drawback of OPE scheme is that it inescapably leaks data privacy. Since all these schemes employ ranking based on OPE, security is compromised.

5. PROPOSED APPROACH

With the advent of data outsourcing, data owners are encouraged to outsource their data from local systems to global space for great flexibility. When sensitive data are outsourced to an external space, the need of keeping data as secure arise. For protecting data, they are encrypted before uploading into such global space which obsoletes search and retrieval based on "plaintext keyword search". So allowing search and retrieval over encrypted data is of paramount importance. Several searchable symmetric encryption schemes are available which allows search and retrieval operations on encrypted data but most of them are boolean keyword search, which do not consider the relevance of document. Also some schemes allow only single keyword in a search query. Considering a large data management environment, it is necessary to allow multiple keyword based search and return the most relevant documents with respect to given search request. Some schemes support ranked search but have low efficiency and security is not guaranteed. So for implementing secure information retrieval, a novel encryption mechanism called Homomorphic Encryption is been proposed.

5.1 Homomorphic Encryption Scheme

Here a fully homomorphic encryption scheme is introduced, solving a central open problem in the area of cryptography[2]. Homomorphic encryption allows specific types of computations to be carried out on the corresponding ciphertext. The result will be the ciphertext of the result of the same operations that performed on the plaintext. That is, homomorphic encryption permit computation of ciphertext without knowing anything regarding the plaintext to get the correct encrypted result [1]. It is an encryption transformation which maps set of operations on plaintext to another set of operations on ciphertext. It enables complex operations on ciphertext without decrypting it[12]. An example shows how homomorphic encryption could be used to solve addition of two numbers 5 and 3 which are in encrypted form. The data is encrypted locally, 5 becomes some 279 and 3 becoming say some 394. The encrypted numbers are sent to the server where the encrypted numbers are added. Server then returns the encrypted answer as 673. It is then decrypted locally and the decrypted result is 8. The homomorphic encryption enables private queries to a search engine. The user submits an encrypted query and the search engine computes a concise encrypted answer without looking at the query in its plain form. It also enables searching on encrypted data. More broadly, fully homomorphic encryption improves the efficiency of secure multiparty computation[4]. The original fully homomorphic encryption scheme, which makes use of ideal lattices over a polynomial ring [5][7], is too complicated. These are applied widely in banking systems, voting systems etc [15][16]. To enable secure search and retrieval over encrypted data, we need an encryption scheme to assure security at both user side and on server side. Fortunately, as a result of employing the vector space model and relevance scoring, only addition and multiplication operations over integers are needed in On the basis of homomorphism property, the encryption scheme can be described as four stages [8]:

1. Key Generation: The secret key p is an odd ɳ-bit number randomly selected from the interval $[2^{\eta-1},2^{\eta};)$. The public key pk is derived from p as p*q+ x*r. The noise factor x is selected randomly from the interval $(2^{\mu},2^{\mu+1}]$. Where μ represents the bit length of plaintext.
2. Encryption: Ciphertext is given by xr + m + k_i
3. Evaluation: Apply the binary addition and multiplication gates to the t ciphertext C_i, perform all necessary operations, and then return the resulting integer .
4. Decryption: Output M = (Ƙ mod p) mod x

5.2 Relevance Scoring

Scoring is a simple way to weight the relevance of a document with respect to a keyword. Based on the relevance score, files can be ranked either in ascending or descending order. There exist several ranking models to score and rank files in IR community. Among these schemes, the most widely used one tf-idf weighting is used here. It is a numerical statistics that specifies how much important a word is to the document in a file collection. Many variations of the tf-idf weighting scheme are used by search engines as a major tool in scoring and ranking the relevance of a document with respect to a search query. It involves two attributes- Term Frequency and Inverse Document Frequency. Term frequency ($tf_{t,f}$) denotes the number of occurrences of a term t in file f. Document frequency (df_t) denotes the number of files that contains term t, and the inverse document frequency (idf_t) is defined as:

$$idf_t = \log \frac{N}{df_t}$$

where N denotes the total number of files. Then, the tf-idf weighting scheme assigns to term t a weight in file f given by

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

The IDF value is used here so that the weights of terms that occur very frequently in the collection are diminished and the weights of terms that occur rarely are increased. This helps to reduce stop words that very frequently appear in the file collection.

5.3 Vector Space Model

While tf-idf ranking function computes the weight of a single term on a file, the vector space model is introduced to score a file based on multi-keyword. The vector space model is a mathematical model for representing a file as a vector. Each dimension of the vector represents a separate term, i.e., if a term occurs in the file, its value in the vector is nonzero, otherwise is zero. It enables a degree of similarity between search query and files, thus rank files according to their relevance. A query is represented as a vector while each dimension of the vector is assigned with 1 or 0 according to whether this term is queried or not.

5.4 TSSE Scheme

Consider a data management system hosting data service, as illustrated in Figure 1, in which three different entities are involved: data owner, data user and a storage server.

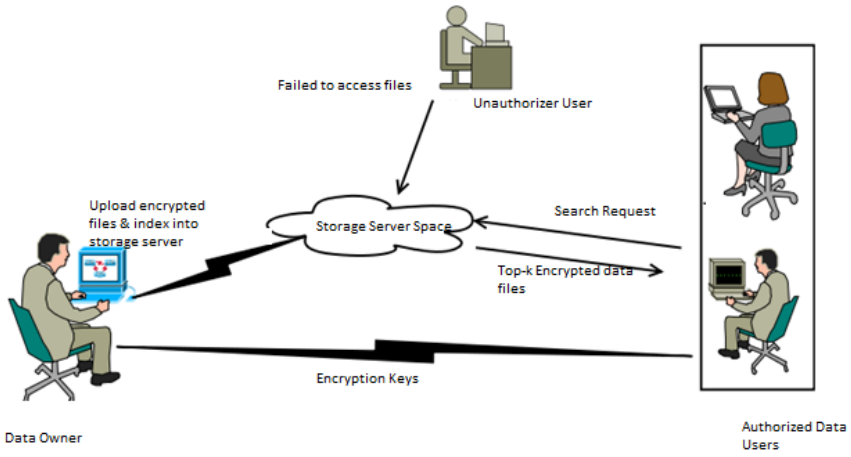

Figure 1 Scenario of search and retrieval over encrypted data

The data owner has a collection of data files. Data owners are encouraged to outsource their data from local systems to global space for great flexibility. For protecting data files, they are encrypted before uploading into such global space. Thus enabling search and retrieval over such encrypted data is of paramount importance. The data owner has a collection of n files say, C = {$f_1, f_2, ...f$n } which may be of extension .txt, .doc and .pdf. To build searchable index, the owner prepare a Word list. Word list is a collection of l unique keywords W = {$w_1, w_2, .$wl} , extracted out of file collection C. The words are taken from files, and are transformed into its root form. The process of converting a word into its root form is termed as Stemming. The most efficient stemming algorithm called "Porter-Stemmer" algorithm is used here. The whole unique stemmed words constitute the word list. Another function of data owner is to generate keys for homomorphic encryption. Both secret key and public key(which is derived from secret key) are used for encryption. From the word list, searchable index is been built. The first column corresponds to IDs of selected data files. The first row corresponds to keywords taken from word list. i.e, the index consists of (l+1) columns. Each entry in index represents the score value of a file with respect to a keyword. For example the I_{ij} (i: row of file ID, j:column of keyword) represents the score value of a file given in column i, with reference to the keyword at column j. The score value is calculated using TF-IDF function explained earlier. After the index is constructed, it is encrypted homomorphically. The ciphertext will be in integer form. The selected data files can be encrypted using any symmetric encryption algorithm like DES. Then the encrypted index and files are uploaded into server space. The data user is authorized to process multi-keyword based data retrieval over outsourced data. The authorized data user at first generates a query. For privacy consideration, which keywords the data user has searched must be concealed. For that the query should be encrypted and this encrypted query should be processable with searchable index. For this the query is to be converted into a vectorized form. The query vector T_w = {m_1, m_2, ...m_l} is generated in which m_i = 1 if t_i € REQ or m_i = 0 otherwise. The data user perform homomorphic encryption on query vector T_w and produce encrypted vector and send to server. For each file vector the server computes the inner product and returns the result vector to the data user. On receiving the encrypted score information, the data user decrypts the file-id and score values and build id score pattern. Then user rank these documents and pick the top-k highest scoring files' identifiers and sends it to server. After calculating the rank, the data user sends the set of ranked file-id back to server. On receiving these ids, the server send corresponding download-links back to user. Thus user can download the encrypted files, decrypt them using the homomorphic keys and make use of that files. The retrieval takes a two-step communication between server and the data user. Thus named as the TSSE scheme, in which ranking is done at the user side while score calculation is done at the server side.

5.5 Multi-owner Support

To improve efficiency, the secure search system should be scalable, i.e, the system should support multiple data owners and associated data users. The scheme proposed in [1] deals with only a single data owner and its associated set of users. Considering a large private data management system, the proposed scheme should address such scalability issues. So the TSSE is been developed with multi-owner support. There are multiple data owners and associated with each owner, there is a set of users. A user is authorised by a single data owner such that user can make use of data only from that owner. A set of documents owned by a person A can be accessed by another person B, only after B is authorised by A. The information regarding authorised set of users corresponding to each owner is maintained in a database. The database information is protected so no one can access the database except the owners.

5.6 Key Management

In order to deal each set of users by each data owners, key management is of paramount importance. The key management issue is not discussed in [1] because it was developed for single data owner. During data uploading, each data owner generate keys (secret key as well as public key) for homomorphic encryption. Both searchable index and user query are encrypted using this key. It is well known that the authorized users should construct a query vector to perform information retrieval. To encrypt this query vector and also to decrypt the retrieved files, the user need keys from owner. So keys should be properly managed in order to provide multi-owner support. The data owners transfer the homomorphic keys to authorised users in a secure manner. The generated key sets are stored in a database. In key database, there is entries for each owner, ownerID, secret key and public key. A user is authorized by only a single data owner and his owner's ID is stored along with user information. So the keys corresponding to his ownerID is securely transferred to user side. For protecting the keys, they are encrypted before storing into database. The key used for encrypting homomorphic-key-sets is termed as “Master Key”. The master key is known only to data owners which enables sufficient security and privacy.

6. DISCUSSIONS

The evaluation of proposed TSSE scheme has been conducted using a set of sample data files taken from Internet source. The scheme support data files of extensions like .txt, .doc , .pdf etc. The environment includes a group of data owners, their associated set of data user and a storage space. Both data owner and user can sign up by registering their username and password. On sign up, a data owner have a unique ID in addition to his username and password. On sign up, a data user should specify their data owner. A user can perform search and retrieval over his owner's data files only after getting approval from data owner. There is a login page for data owner/user where they can either sign up or sign in to the system. Data owner enters his username & password and logs into the system. He select a set of data files and prepare the word-list. Based upon the word-list a searchable index is prepared. The data files are encrypted using DES and index is encrypted with homomorphic encryption. Then these encrypted index and data files are uploaded to server space. A data user who logs in by entering his username and password can do search and retrieval. User can enter the search query on provided space. The user constructs corresponding query vector and is encrypted homomorphically. This is send to server. The server calculates scores of each files and send the encrypted score information to data user. On receiving, the data user decrypts the scores and picks out the top-k highest scoring files' identifiers, and sends file-request to server. The server responds with requested encrypted files. Users on receiving, can decrypt it and make use of them. There are a set of data owners and some groups of users are associated with each data owner. A data user must be under the authorization of any of the data owner. This ensures that only authorised users are making use of outsourced data. There is an authorisation page in system. For the user who sign up for first time, he should select a data owner and send a registration request to concerned owner. The user is said to be authorized only after he get approval from owner. After authorizing a user, the owner enter that information into database. This user can consume data only from this selected data owner. The keys used for homomorphic encryption is different for each data owners. Data owners generate homomorphic keys and are transferred to authorized user when such users logs into the system. So only users who are authorized by corresponding data owners are able to access the keys. Keys should be properly managed in order to give flexibility to users. Also these keys have to be protected from attackers. So in database, keys are stored in encrypted form. The key for encrypting/decrypting homomorphic key-set are called Master key. Only data owners know this master key which ensure privacy hence guarantee security. Only authorized user gets the homomorphic key from owner so that he can perform search and retrieval. Here the files are ranked in the order of relevance by users' interest and only the files with the highest relevance are sent back to users.

7. ANALYSIS

Here the security of proposed TSSE scheme is discussed by analysing its fulfillment of the privacy requirements of traditional SSE. It is mandatory that the server should not learn either the nature of plaintext of the data files, searchable index, or the searched keywords. Also, the statistic information including access pattern, search pattern, and term distribution should be concealed. The server should not learn the degree of similarity relevance of terms or files so that

the scheme gains high robustness. In short, statistic information leakage problem should be resolved. Also the index building must be efficient compared to existing SSE scheme.

7.1 Security Analysis

7.1.1 Access pattern and Search pattern

Access pattern defines which keywords and the corresponding files have been retrieved during each search request, and search pattern refers to whether the keywords retrieved between two request are same[1]. These distribution information implies a similarity relationship among terms or files. On the one hand, terms with similar term distribution may always have simultaneous occurrence [1]. Let Tw_1 and Tw_2 be the vectorized search query corresponding to queries REQ_1 and REQ_2. Suppose one same keyword t_i is requested in two queries REQ_1 and REQ_2, then the positions $m1_i = m2_i = 1$ in the corresponding query vector Tw_1 and Tw_2. The two vectorized queries are encrypted into different ciphertext by using 'Encrypt' function. Eventhough 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.

7.2 Performance Analysis

7.2.1 Index Building

In SSE scheme to build a searchable subindex for each document in the dataset, the first step is to map the keyword set extracted from the document to a data vector. Then every data vector is encrypted. The time cost of mapping or encrypting depends directly on the dimensionality of data vector. This dimensionality is determined by the size of the word list. The time taken for building the whole index is also depends on the number of subindex which is equal to the number of documents in the dataset. Figure 2 shows that, given a word list, since the time taken for building the subindex remain fixed, the time cost for building the whole index is nearly linear with the size of dataset. Figure 3 shows that the number of keywords in the word list determines the time cost of building a subindex. For generating the subindex, the SSE scheme include a series of matrix multiplications. These matrix dimension have direct relationship with the size of word list so the index construction time becomes large. The size of subindex is absolutely linear with the dimensionality of data vector which is determined by the number of keywords in the dictionary.

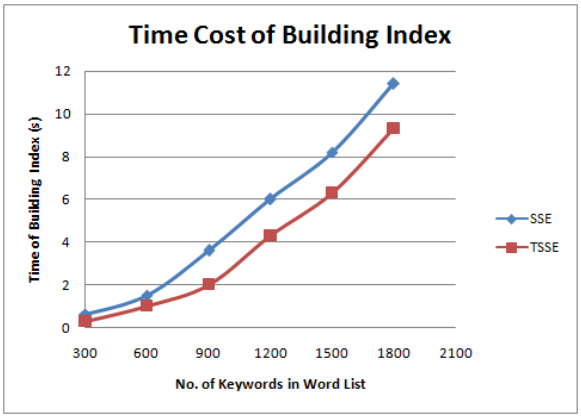

Figure 2 Time cost of building index for different size of dataset with the same word-list

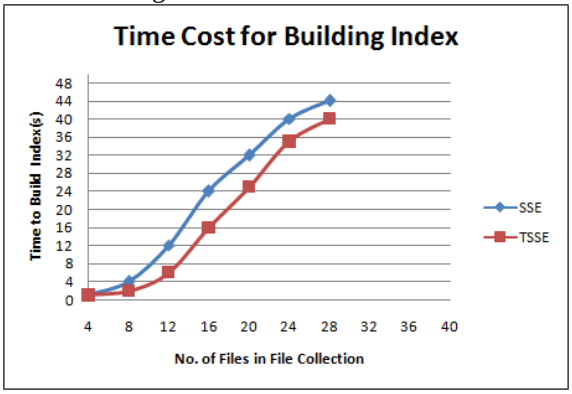

Figure 3 Time cost of building index for the same dataset with different size of word-list

In the proposed TSSE scheme also, the time cost for building the index is greatly influenced by both the number of documents in the file set and size of the wordlist. The index column corresponds to each keyword in word list and row corresponds to each document in the file set. As the size of data file set increases the row size of index also increases. Similarly when the word list size increases, the column size is get expanded. So for each document the score should be entered corresponding to each keyword. So, the time cost for generating the index grows nearly linear with the size of

data set and word list size. Comparing the complexity of operations involved in both scheme it can be observed that the TSSE scheme take less level of time than SSE scheme.

7.2.2 Query Vector Generation

Figure 4 shows that the time taken to generate a query vector is greatly affected by the number of keywords present in the word list. The figure compares the time taken for two schemes: one is the scheme described in [25] and other is proposed scheme. For the SSE in [25], every query vector generation involves two multiplication of a matrix and a split query vector. The dimensionality of matrix becomes larger with the increasing size of word list, so time taken will be increased. For the proposed scheme, as the number of keywords in word list increases the time to generate corresponding query vector also increases. The comparative studies on two schemes show that the proposed TSSE scheme is more efficient in this stage.

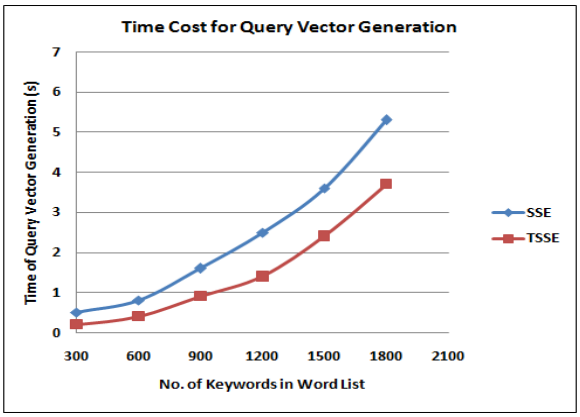

Figure 4 Time cost of generating query vector for the same query keywords within different sizes of word-list

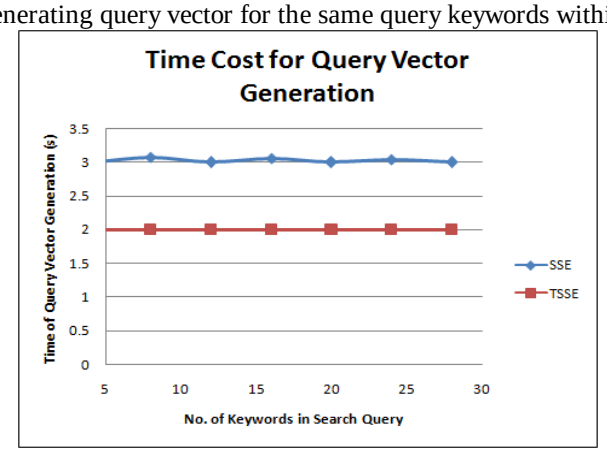

Figure 5 Time cost of generating query vector for different numbers of query words within the same wordlist

One point to be noted is that eventhough the search queries consists of different number of keywords, corresponding query vectors will be of l bits (l denotes the no. of keywords in word list). i.e, the length of query vector is fixed to l so the time taken to generate query vector is changeless when the number of queried keywords increases. For any number of keywords, a vector of l bits have to be generated. The comparison is as shown in figure 5. The TSSE take less time than SSE scheme in this stage when the word list count l is kept as fixed value. It shows that the number of queried keywords has little influence on the overhead of query vector generation, which is a significant advantage over related works on multi-keyword searchable encryption.

7.3 Complexity Analysis

7.3.1 Initialization Phase

The initialization phase must be processed only once by a data owner and it includes key generation, wordlist preparation and index building.

Key Generation

In fully homomorphic encryption several parameters like λ, x,r etc are to be kept as some default values. In this implementation, the security parameter (which is a fixed integer for a realistic scheme) is taken as 128 or above. Here for conceptual simplicity it taken as $\lambda = 7$. According to the parameter selection in the FHEI scheme, the complexity of key generation stage is $O(\lambda^{10})$ so the stage costs a fixed time.

Index Building

The index building stage includes building searchable index I and then encrypt homomorphically. To build I, the tf-idf values are calculated and rounded to integers, which does not affect the retrieve accuracy. This encryption needs only addition operation, so the complexity of encryption is O(nl), where n denotes the number of files and l denotes the

number of keywords. In two approaches, as the number of keywords in the file set increases, the time taken for building searchable index also increases.

7.3.2 Data Retrieval Phase

The data retrieval phase includes Query vector generation and Score calculation.

Query Vector Generation

The query vector generation stage needs O(l) time to construct the query vector of l-dimension from the multi-keyword request. To encrypt T to T0 , each dimension should be encrypted. Here encryption constitutes only addition operation, so the complexity is O(l).It should be noted that in SSE methods, the time for query vector generation increases as the number of keywords in index is large. i.e,for instance, it takes more time to generate query over a file set containing 500 keywords than for a file set containing some 200 keywords. It shows an exponential growth. Apart from this, the proposed approach shows the time cost as a linear growth against the increment of keyword set size. The length of query vector is fixed to l, so even if the number of queried keywords is increased, time to build corresponding query vector is constant.

Score Calculation

Score calculation involves computing the product of query vector value with each row in searchable index. To calculate the inner product, it take l multiplications and l-1 additions in each row. So the overall complexity is O(nl). Based upon the current studies and analysis, one issue remain to be addressed in multi-keyword based top-k data retrieval over encrypted data. The proposed scheme is comparatively simple and efficient approach. But it faces some issues regarding the size of encrypted index and keys. As the number of input files is very large, time required to build and encrypt the index is quite large. Also user encrypts the query vector and sends this ciphertext to server. Since the encrypted query-vector size is too large, the communication overhead will be large.

8. CONCLUSION AND FUTURE WORK

It is common for computer users to access public storage space over the Internet, generally to perform queries and find relevant information. The main problem in such information retrieval is that statistical information leakage is possible even the data are in encrypted form. Here a new scheme have been developed which enables data owners to upload encrypted data files into storage server and allow several authorised users to perform search and retrieval over them. It enable users to obtain the result with the most relevant files that match users' requirement instead of all the files. Data files are ranked in the order of relevance by users' interest and only the files with the highest relevance are sent back to users. Since statistical analysis is not possible in homomorphic ciphertext, the proposed scheme gives security to certain extend. The scheme supports multiple data owners and their associated set of users. There will be separate space for each data owner at storage side. The key management system ensures proper user authentication and access control. Here the homomorphic keys are managed efficiently so that only authorised set of users can make use of data of corresponding data owners. Several researches on cryptographic community tries to improve both security and efficiency of homomorphic encryption. This implies that efficiency of TSSE scheme can be further improved. The system can be further improved by implementing fully homomorphic encryption which supports reduced ciphertext and key size. It can also be extended by implementing in a distributed network environment.

References

[1] Jiadi Yu, Member IEEE, Peng Lu, Yanmin Zhu, Member IEEE, Guangtao Xue, Member IEEE Computer Society, 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] D. Song, D. Wagner, and A. Perrig, “Practical Techniques for Searches on Encrypted Data”, Proc. IEEE Symp. Security and Privacy, 2000.

[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] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully Homomorphic Encryption over the Integers,” Proc. 29th Ann. Intl Conf. Theory and Applications of Cryptographic Techniques, H. Gilbert, pp. 24-43, 2010.

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

[6] Santi Martinez, Josep M. Miret, Rosana Tomas and Magda Valls, “Security Analysis of Order Preserving Symmetric Cryptography”, Applied Mathematics & Information Sciences, Vol. 7, No. 4, 1285-1295, 2013.

[7] Nitin Jain, Saibal K. Pal and Dhananjay K. Upadhyay, “Implementation and Analysis of Homomorphic Encryption Schemes”, International Journal on Cryptography and Information Security(IJCIS),Vol.2, No.2,DOI:10.5121, June 2012.

[8] N. Smart and F. Vercauteren, “Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Size”, Proc. 13th Intl Conf. Practice and Theory in Public Key Cryptography (PKC), 2010.

[9] 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.
[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.
[12] J. S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully Homomorphic Encryption over the Integers with Shorter Public Keys" CRYPTO 11: Proc. 31st Ann. Conf. Advances in Cryptology, 2011.
[13] 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.
[14] 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.
[15] M. Naehrig, K. Lauter and V. Vaikuntanathan, "Can Homomorphic Encryption Be Practical?", ACM Workshop on Cloud Computing Security Workshop, pp. 113124, 2011.
[16] T. Graepel, K. Lauter and M. Naehrig, "ML Confidential: Machine Learning on Encrypted Data", Information Security and Cryptology, Lecture Notes in Computer Science, pp. 121, 2013.