International Journal of Engineering Technology and Management Sciences[IJETMS]

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

An Efficient and Secure Multilevel Keyword Ranked Search for Encrypted Data

Prasanth K. Baby[1], Nikhil Samuel[2]
[1]*Christ College of Engineering, irinjalakuda, Thrissur*
[2] *Christ College of Engineering, irinjalakuda, Thrissur*

*(E-mail: [1]seprasanthkbaby@gmail.com,[2]*nikhilsamuelkf@gmail.com*)*

***Abstract*— with the growing digital communication and networks, the data owners are motivated to outsource their complex data to the global storage space. Greater flexibility and economic saving are the advantages of this global storage space. Before outsourcing the sensitive data, it has to be encrypted in order to enforce the data privacy. In the encrypted data, search service is important to get the necessary data. The stored data is relatively large so it requires multiple keywords in the search query and it will return document in the order of their relevance to these keywords searched. Related works on searchable encryption focus on single keyword search or Boolean keyword search and rarely sort the result and for the multi-keyword search coordinate matching, i.e., as many matches as possible, to effectively capture the relevance of outsourced documents to the query keywords and inner product similarity to evaluate such similarity measure. In Multi-keyword Ranked Search under the coordinate matching, the ranking helps for the efficient retrieval. The multilevel keyword ranked search is implemented by using the cache to reduce the search time.**

Keywords— **information retrieval; keyword search; multilevel; ranked search; searchable encryption.**

I. INTRODUCTION

With the evolution of different technologies the amount of data handled by users also increased. Hackers and malicious programs all pose a threat to your computer and the information it contains. Advanced data storage service for user is data outsourcing, store complex data to global storage space provider. The outsourced data's are managed on remote servers are maintained by the trusted third party outsourcing vendors. In today's distributed nature of data management, gives assurances to detect and correct faulty behavior. For the relevance of outsourced data, data owners place their sensitive data into specialized storage area. Data owners outsource their data without assurances of confidentiality and security. Achieving confidentiality by encrypting the data, the major challenge is that how to enable search and retrieval over such encrypted data.

In standard data searching service is basically on plain text keyword search, the solution of downloading all the data and decrypting locally impractical. This is due to the huge amount of band width required. For avoiding the local storage management, the data are storing to the storage servers. Easily searched and utilized, explores security and effective search service for encrypted data is of paramount importance. For the useful data retrieval in the large number of documents, request the server to perform result relevance ranking, rather than returning undifferentiated results. This ranked search helps to avoid the unnecessary network traffic. For the privacy protection, such ranking does not leak any keyword related information.

To improving accuracy in search result and enhance the user search experience, ranking system has to support multiple keyword search. As a user routine practice, provide multiple keywords as a search user interest to retrieve data, the each keyword in search request helps to restrict the search result. For this coordinate matching is used. This coordinate matching is used widely in the plaintext information retrieval (IR). Coordinate matching means as many matches as possible.

The proposed method describes the technique for reducing the searching time by using multilevel keyword ranked search. This method will help the user for consuming less band width, reduce the searching time over large encrypted document, and also help the user for searching multi-keyword (set of keywords). For restricted the search result, each keyword in the search query helps. Multilevel keyword ranked search uses coordinate matching, i.e. as many matches as possible. It is an efficient similarity measure among multi-keyword semantics to refine the result relevance, and has been used in plain text information retrieval (IR). The search query consists of keyword and the top k retrieval (top k rank). Here the search result is on the rank ordered search. There are using several searching technique for searching on encrypted data. For performing better search reduced time, here cache is implemented in multilevel keyword ranked search. This cache helps for the better search reduction time. In the cache there store the document and there weight.

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

The rest of the paper is organized into four sections. Section II deals with the related works. Methodology is discussed in section III. Section IV involves the analysis of the work and the conclusion of the work is given in section V.

II.Related Works

This session includes the related works that are done in the field of Searchable encryption are Single keyword Searchable Encryption and Boolean Keyword Searchable Encryption. Single keyword searchable encryption schemes usually build an encrypted searchable index such that its content is concealed to the server unless it is given appropriate trapdoors generated via secret keys [8]. In boolean keyword searchable encryption, conjunctive and disjunctive search are used. All these searchable encryption are not dealing with the searching time, considering with the basic introduction to cache is much better for those problems.as well, for math, etc.

A. Single Keyword Searchable Encryption

Song et al [4] introduced the notation of searchable encryption, based on symmetric key setting, where each word in the file is encrypted independently under a special two layered encryption construction. Thus, a searching overhead is linear to the whole file collection length. In this method the searching is done by, an index contains a list of keywords. With each keyword a list of pointers to documents where the key word appears. These keywords are words of interest that user want to search later. One possible advantage for this scheme is that the request could be embedded in other retrievals so that Bob might have uncertainty about the correlation of the search request and the retrieval request for cipher text. The disadvantage is that user has to spend an extra round-trip time to retrieve the documents. A general disadvantage for index search is that whenever user changes the documents, its index must be updated.

The limitations of [4] was the work load for each search request proportional to the number of files in the collection. To overcome this limitation E.J Goh [5] developed a Bloom filter based per-file index, reducing the work load for each search request proportional to the number of files in the collection. A secure index is a data structure that permits a queries with a trapdoor for a word X to test in O(1) time only if the index contains X; The index disclose no information about its contents without valid trapdoors, and trapdoors can only be generated with a secret key. Secure indexes [5] are a natural extension of the problem of constructing data structures with privacy ensures such as those provided by oblivious and history independent data structures. They use Bloom filter [1] as a per document index to track words in each document. In their scheme, a word is represented in an index by a code word obtained by applying pseudo-random functions twice once with the word as input and once with a unique document identifier as input. This non-standard use of pseudo-random functions ensures that the codewords representing a word X are different for each document in the set, and this technique together with blinding indexes with random tokens, ensures that there indexes are semantic security against adaptive chosen keyword attack (ind-cka) secure.

Kmara et al [8] has developed a similar per-file index scheme. In this method the files are stored in encrypted form, later the user U wants to retrieve files containing (indexed by) some keyword search, is not possible. There is no any straight forward way to do keyword search unless U leak the decryption key. Here U has to create keyword index associates each keyword with its associated files. All keyword searches by U are based on this index; hence their scheme does not offer full pattern-matching generality with the real text. In practice, this should be sufficient for most users. It is worth noting that in this system U can have complete control over what words are keywords and which keywords are associated with which files, a power that can be useful for many applications. let U use pseudo-random bits to mask a dictionary-based keyword index for each file and send it to S in such a way that later U can use the short seeds to help S recover selective parts of the index, while maintaining the remaining parts pseudo-random.

B. Boolean Keyword Searchable Encryption

In the Boolean keyword search conjunctive and disjunctive search are used. Conjunctive keyword search returns " all-or-nothing . Which means that returns those documents in which all the keywords specified by the search query appear. In disjunctive keyword search that returns undifferentiated results. This means it returns every document that contains a subset of the specific keyword, even only one key word of interest. The paper [10] gives a basic idea for privacy preserving Multi-keyword ranked search over encrypted data (MSRE). This is based on secure inner product computation, and give two significantly improved MRSE schemes to achieve various strict privacy requirements.

1) *Conjunctive & Disjunctive Keyword Search:* Golle et al [11] proposed conjunctive keyword search over encrypted data. Consider a user that stores encrypted documents on an untrusted server. Let p be the total number of documents, and assume there are m keyword fields associated with each document. For an example consider, documents were emails. They define the following 4 keyword fields: "From", "To", "Date" and "Subject". For the simplicity, they make the following assumptions:

- Assume that the same keyword never arise in two different keyword fields. The easiest way to satisfy this requirement is to prepend keywords with the name of the field they belong to. Thus for example, the keyword "From:James" belongs to the "From" field and cannot be confused with the keyword "To:James" that belongs to the "To" field.
- Assume that every keyword field is defined for every document, requirement can be easily satisfied. In our

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

email example, they may assign the key word " Subject:NULL " in the "Subject" field to the emails that have no subject.

The documents are identified with the vector of m keywords which characterize them. For (i = 1... n), they denote the ith document by $D_i = (W_{i;1} \ldots\ldots W_{i;m})$, where $W_{i;j}$ is the keyword of document Di in the jth keyword field. The body of the ith data document can be encrypted with a standard symmetric key cipher and stored on the server next to the vector of keywords D_i. For ease of presentation they may ignore the body of the document and concern themselves only with the encryption of the keyword vector D_i. When discussing a capability that enables the server to verify that a document contains a specific keyword in field j, they denote the keyword by W_j. In this conjunctive keyword search scheme there, it returns true if the expression $((W_{i;j1} = W_{j1}) \wedge (W_{i;j2} = W_{j2}) \wedge \ldots\ldots \wedge (W_{i;j1} = W_{j1}))$ holds and false otherwise. In predicate encryption [7] scheme are support both conjunctive and disjunctive search.

C. Keyword Ranked Search

In keyword ranked search Curtmola et al [12] discussed about vector space model working and similarity based ranking multi-keyword text search scheme. Vector space model is well known technique which provides TF-IDF weight rule through which we obtain accurate ranking result. In this they gives the revised Searchable Symmetric Encryption Scheme for, SSE-1 is efficient, it was only proven secure against non-adaptive adversaries. A second Searchable Symmetric Encryption SSE-2, which accomplishes semantic security against adaptive adversaries, and Multi-user Searchable Symmetric Encryption MSSE.

Searchable encryption permits data owner to outsource his data in an encrypted manner while maintaining the selectively searching capability over the encrypted data. Generally, searchable encryption can be accomplished in its full functionality using an oblivious RAMs [3]. Although hiding everything during the search from a malicious server (including access pattern), by the utilization of oblivious RAM usually brings the cost of logarithmic number of interactions between the user and the server for each search request. Thus, in order to accomplish more efficient solutions, almost all the relative works on searchable encryption writing resort to the weakened security guarantee, i.e., revealing the access pattern and search pattern but nothing else. Here, access pattern alludes to the result of the search query output, i.e., which files have been retrieved. The search pattern incorporates the equality pattern among the two search requests (whether two searches were performed for the same keyword), any information derived thereafter from this statement.

Curtmola et al [12] shows that following the exactly same security certification of existing SSE scheme, it would be very inefficient to accomplish ranked keyword search, which motivates us to further weaken the security guarantee of existing SSE appropriately (only leak the relative relevance order of the documents but not the relevance score of the documents) and understand an "as-strong-as possible" ranked searchable symmetric encryption..

In information retrieval (IR) [14], a ranking function is used to evaluating relevance scores of matching files to a given search request. The most commonly used statistical measurement for calculating relevance score in the information retrieval community uses the TF×IDF rule, where term frequency (TF) is the number of times a given term or keyword appears within a file (to measure the significance of the term within the particular file), and inverse document frequency (IDF) is obtained by dividing the number of files in the whole collection of document by the number of files containing the term (to measure the overall importance of the term within the collection of documents). Among several hundred variations of the TF×IDF weighting scheme, no single combination of them out performs any of the others universally [13]. Thus, without loss of generality, they choose an example formula that is commonly used and widely seen in the literature (see [6]) for the relevance score calculation is as follows:

$$TF_{(t)} = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

$$IDF_{(t)} = \log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}\right)$$

$$(TF \times IDF)_{weight} - TF_{(t)} \times IDF_{(t)}$$

In [2] focus on single keyword search, this case, the IDF factor is always constant with regard to the given searched keyword. Thus, search results can be accurately ranked based only on the term frequency and file length information contained within the single file.

D. Multi-keyword Ranked Search

In Multi-keyword ranked search [10], allows multiple keyword in the search request and return documents in the order of their relevance to these keywords, in the ranking principle uses coordinate matching. This means the presence of keyword in the document or the query is shown as '1' or else '0' in the data vector or the query vector. In the search query there are more factors which make impact on the search usability. For example, when one keyword appears in most documents in the data set, the significance of this keyword in the query is less than other keywords which appears in less documents. Similarly, if one document contains a query keyword in numerous locations, the user may prefer this to the other document which contains the query keyword in only one location. To capture these information in the search process,

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

we use the TF×IDF weighting rule within the vector space model to calculate such similarity measure. Here in the building index is a onetime process that means the index construction is done before outsourcing the document or data, i.e. if any modification is necessary then the index construction has to done again for all document. This is because that TF-IDF rule is related to whole document. In this similarity measure introduces high computation cost during the index construction and trapdoor generation; it captures more related information on the content of documents and query that returns better results for user's interest.

E. Server Cache

Server cache is techniques used for caching objects for reduce the server computation. This will help for the user to access the data easily. By the help of caching improve access speed and reduce the work load on the server. In [9] server caching provides better performance for the auctions and also suggests that cache at the application server can save a critical number of accesses to a backend database and thus reduce the server-side latency. In general, work of web caching can be classified into browser caching, client-side proxy caching, network caching (as in Content Delivery Networks), and server side caching. Any time request is performed, a cache hit occurs when keyword present, otherwise a cache miss occurs and the information about the request has to be retrieved from the storage. In traditional distributed file system, the server maintains a cache of blocks that have been accessed by the clients. The server cache is lower in the storage hierarchy than the client caches and therefore has a lower hit rate. Studies show that the server cache is still powerful in reducing server disk traffic and improving the performance of the file system.

In [15], the idea of cache memories is identical to virtual memory in that some active portion of a low-speed memory is stored in duplicate in a higher-speed cache memory. When a memory request is produced, the request is first presented to the cache memory, and if the cache cannot replies, the request is then presented to main memory. For READ operations that cause a cache miss, the item is retrieved from main memory and copied into the cache.

III.Procedure

In this session the multilevel keyword ranked search for encrypted data with caching feature is described. The main contents of this session are System model, Index construction and Search construction.

A. System model

The proposed system consist of 3 different entities, data user, trust server and storage server, as illustrated in figure 1. Here the users are of two type i.e. owner of the data document and the user who access or search these documents. The user (data owner) has a collection of data documents F to be outsourced to the storage server in the encrypted form E. For enable searching capabilities over E for the effective data utilization, the user (data owner) need to build an index I from the documents F. The index building comes before outsourcing the documents F. After building the index from the data document, the user (data owner) outsources encrypted document collection E and the index I to the storage server, for searching the document collection. In the proposed system it is assumed that another basic system exists for trust server which consists of user access control and search control.

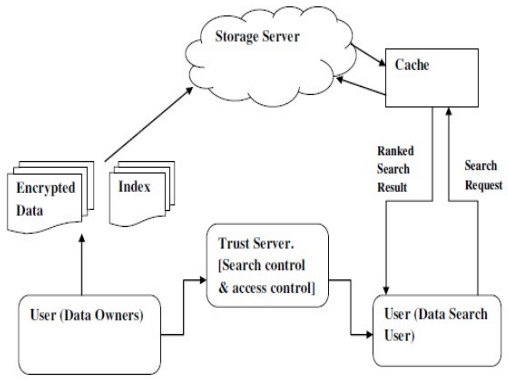

Fig. 1: Architecture of multilevel keyword ranked search for encrypted data.

After receiving the user search request from the user (data search user), the server is responsible to search the index I and return the corresponding set of encrypted documents (TF-IDF and document-id). For improving the document retrieval accuracy, search result should be ranked according to some ranking criteria. For reducing the communication cost, user (data search user) sends an optional number k (retrieve top rank up to k) along with search query and the storage server sends back the Top k documents that are most relevant to the search query.

B. Index Construction

In the proposed method the user U (data owner) has a collection of data documents. In Fig.2 shows the index construction of the proposed method. Before encrypting the document the user need to build an index, by using the index the user can search on the encrypted document. Each of these indexes (user interest keywords) is referred to the documents E. Before building up the indexes these documents need to go through several pre-processing stages. In information retrieval the pre-processing methods like Tokenization, Stemming process, Stop words removal are used and the rest is considered as the keyword. In normal cases these keywords are taken as indexes. The pre-processing is done using Natural Language Processing (NLP) tool-kit. For stemming process there are several stemming algorithms each of these have their own drawbacks and works on the basis of certain criteria or algorithms. For avoiding these draw backs, here in the stemming process dictionary based stemming is used.

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

Dictionary stemmers work quite differently from algorithmic stemmers. Instead of applying a standard set of rules to each word, they simply look up the word in the dictionary. Theoretically, they could produce much better results than an algorithmic stemmer. A dictionary stemmer should be able to do the following:

- Return the correct root word for irregular forms such as feet and mice.
- Recognize the distinction between words that are similar but have different word senses, for example, organ and organization.

The above features are the advantage of the dictionary based stemming.

After the stemming process stop words removal is done. Stop words are natural language words which have very little meaning, such as “and”, “the”, “a”, “an” and similar words. Stop words are filtered out before or after processing of natural language data (text). Though stop words usually refer to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools. Some tools specifically avoid removing these stop words to support phrase search.

After these processes have been completed the remaining words are taken as keywords. In normal cases these keywords are used as index for documents. For each documents there may be different keywords associated with the document. For selecting the index some criteria is used. The criteria are that the word which has a count more than a threshold value are selecting as the index for each document. After the index is generated, the term frequency (TF) for that word is calculated.

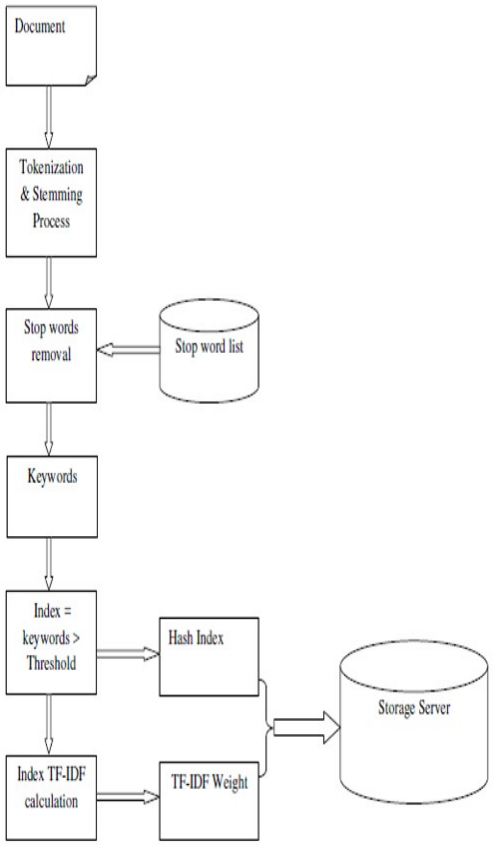

Fig. 2: Flow Chart for User build index and storage

The TF is calculated by the following expression:

$$TF_{(t)} = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}} \quad (1)$$

Inverse Document Frequency (IDF)is calculated using the expression:

$$IDF_{(t)} = \log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}\right) \quad (2)$$

Then the TF-IDF value is calculated for these indexes by using the following expression:

$$(TF \times IDF)_{weight} = TF_{(t)} \times IDF_{(t)} \quad (3)$$

The above expressions are used as the basic rules applied for how important the word is for a document. The TF-IDF is the weighting factor which is commonly used in information retrieval. A hashing is applied to encrypt these indexes; the hashed indexes and the corresponding TF-IDF are stored to the storage server. The data document are encrypted by AES encryption, and outsourced.

C. Search Construction

The keywords are the user interest words used for searching the documents. The generated keywords are the indexes constructed for the encrypted document. These keywords help the user while searching the encrypted document. Then the user has to enter the keyword in the search area with top *k* retrieval (rank order less than top *k*). When the user enters the keyword, pre-processing is done for each keyword and hashing is also done for these keyword. Then these keywords are send to the server, and if match found with the stored index, the corresponding TF-IDF weight is taken. If match occurs for more documents then each document is taken in the order of their TF-IDF weight. The top *k* documents are retrieved; these retrieved documents are relevant to the search query keyword. If a user searches a particular keyword repeatedly over a particular threshold, then that keyword relevance document id and TF-IDF value are cached, so on the next search for that keyword the document retrieval would be from the cache. By using the cache the searching could be done easily with reduced search time.

In the single keyword search, the user enters the keyword *M* and top *k*. The M will go through pre-processing steps and hashing is done for encrypting the keyword. If the user searches a particular keyword repeatedly over a threshold value(in this case uses 3 as threshold), then that keyword relevance document-id and TF-IDF weight are cached, the

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

user search keyword count is less than the threshold value then the keyword count is get incremented by 1. Consider, user search occurred when the keyword count less than threshold value 3, then the *M* relevance document-id and TF-IDF weight are retrieved from the storage, in this case there occur cache miss. The searched keyword count is greater than the threshold value 3, and then the *M* relevance document-id and TF-IDF weight are retrieved from the cache, the resultant documents would be ranked according to the TF-IDF weight. In single keyword search the TF-IDF weight of the document is the TF-IDF of *M* to that document.

As the user tends to outsource a large volume of encrypted document, search by multi-keyword is necessary which would help to retrieve relevant documents with user interest. In case of a multi-keyword, the user enters multi-keywords (M_1, M_2) in the search area, then pre-processing is done for M_1 & M_2 keyword. After that each Keyword is hashed (i.e. first hashing is perform for M_1 and then for M_2), then it is of the form $H(M_1)$ and $H(M_2)$, and send to the server. In the storage server the keyword $H(M_1)$ and $H(M_2)$ is checked with the stored hash keywords, if match occurred then the corresponding document TF-IDF is retrieved. Finally, the checking is done as:

- First check $H(M_1)$ in document D_1, if match found then the TF-IDF for that word in D_1 is taken.
- Then check $H(M_2)$ in document D_1, if again match found then the TF-IDF for that word in D_1 is taken.
- Finally, the TF-IDF obtained for different keywords of single document D1 is added up and the result obtained is the TF-IDF of that document, as given below:

$$(TF \times IDF)_{(D_1)} = (TF \times IDF)_{(D_1 \text{ for } M_1)} + (TF \times IDF)_{(D_1 \text{ for } M_2)} \quad (4)$$

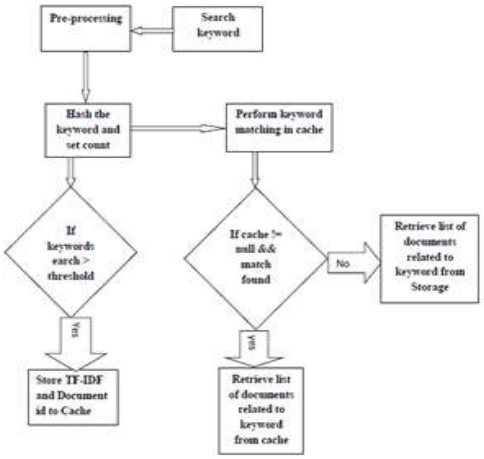

Fig. 3: Flow Chart for User Search

This $(TF \times IDF)_{(D1)}$ is the weight for the document D_1 in case of multi-keywords, similarly for the remaining documents. By using this calculated TF-IDF weights the ranking is done for the documents. The *k* value is used for retrieving the top *k* no of documents which are relevant to the keywords. The result is obtained for the keywords M_1 and M_2. If the user needs to search for multi-keywords (M_1, M_2,.....M_n), there are n keywords for searching, then a keyword search in the document D_1 for M_1, M_2,.....M_n is done, there after similar for keyword search is also done for the remaining documents. The resultant TF-IDF of document D_1 is the sum of TF-IDF of each keyword M_1, M_2 ...M_n for D_1. In general, there are n keywords and h documents of the form:-

$$(TF \times IDF)_{weight_{(Dj=1 \text{ to } h)}} = (TF \times IDF)_{(Dj \text{ for } M1)} + (TF \times IDF)_{(Dj \text{ for } M2)} + .. + (TF \times IDF)_{(Dj \text{ for } Mn)} \quad (5)$$

The equation (5) can be summarized as follows:-

$$(TF \times IDF)_{weight_{(Dj=1 \text{ to } h)}} = \sum_{(Mi=1)}^{n} (TF \times IDF)_{(Dj \text{ for } Mi)} \quad (6)$$

In figure 3 shows how the proposed system work with multilevel search and reduce the search time, multilevel keyword ranked search would help the users to cache the results. Here the cache is applied for each keyword. Each keyword is associated with the TF-IDF weight and document-id of the corresponding document. While performing the search operation again with the same keywords, the TF-IDF weight and document-id of the corresponding keyword would be retrieved from the cache, so that search time complexity could be reduced. The proposed method would be an efficient method to overcome the problem of search time.

IV.Results And Discussion

This session deals with the results and analysis of the proposed method. The analysis focuses on the reduced search time for document retrieval using search keywords. The proposed method with cache is compared with the method without cache. Based on the analysis, the results are presented below.

TABLE I: Effect of Search Time with Cache Vs without Cache

Query size (No of keyword searched)	Search Time (Milliseconds) without cache	Search Time (Milliseconds) with cache
7	57	2
8	69	2
9	73	2
10	79	2
11	84	2

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

The dataset consist of 67 files and each containing different no of keywords. The TF-IDF calculation is a onetime process, if modification is done to any of the files then keyword construction and recalculation of the TF-IDF Value is needed. The table I shows the search time with the effect of cache and without effect of cache. The caching is done for the keywords related with the document id and their TF-IDF weights. The analysis is done with the query size (No of keywords searched) of 7, 8, 9, 10, 11 and their related search time with cache and without cache is obtained. Consider that, when search occurs with query size 7 (i.e. no of keywords searched) the corresponding results are, with cache is 2 millisecond and without cache is 57 millisecond. From this it is seen that the search time could be reduced by using the multilevel cache method. The retrieved result is the relevant documents in the ranked order; ranking is done with the related keyword's TFIDF weights.

Similarly, for the remaining query size 8, 9, 10, 11 (i.e. No of keywords searched) their related search time with cache is 2 millisecond and without cache is 69, 73, 79, 84 respectively as shown in the table I. From the table it is clear that the search time without cache is higher than search time with cache. The proposed method with cache gives better search time as compared to that method without cache. The table II shows, the comparison between p + q keywords search time with cache and without cache. In this p and q are no of keywords with cache and without cache respectively, i.e. p keywords document id and their TF-IDF weight is in the cache, and q keywords TF-IDF weight and document id are taken from the storage. In this analysis, the value of p is made constant i.e. 7 keywords and the value q is varied from 1 to 4.

TABLE II: Effect of Search Time with p keywords Cache and q keywords not Cache Vs without Cache for p+q keyword Search.

Query size (No of keyword searched)	No of keywords cached (*p*)	No of keywords not cached (*q*)	Search Time (Milliseconds) without cache	Search Time (Milliseconds) with cache
8	7	1	70	11
9	7	2	84	22
10	7	3	116	25
11	7	4	129	37

The table II shows p + q keywords in the query search, in this p =7 and q =1.In case without cache, all the 8 keyword's TF-IDF weight and document-id are to be searched from the storage server with search time 70 milliseconds but by using cache, 7 keyword's TF-IDF weight and document-id is taken from the cache and 1 keyword's TF-IDF weight and document-id is taken from the storage server reducing the search time to 11 milliseconds. Now from the analysis it is clear that the search time in the proposed method is lower than the search time in the method without cache.

Similarly for the remaining p + q keywords, p =7 is made constant and q is given values 2, 3, 4 in this analysis and the search time without cache is shown as 84, 116, 129 millisecond and with cache is shown as 22, 25, 37 millisecond respectively. From these values it is clearly seen that with cache the search time can be reduced to a great extent.

V. Conclusion

In the proposed method of multilevel keyword ranked search, the problem of increased search time have been reduced. The coordinate matching i.e. as many matches as possible to effectively capture the relevance of outsourced documents to the query keywords and use inner product similarity to quantitatively evaluate such similar measures was used. Along with the coordinate matching the ranking mechanism by TF-IDF weight rule was used for the relevance of the data documents and also the cache was included in the proposed method. The TF-IDF weights and the document ids of the keywords that are frequently used are stored in the cache. Then after the keywords TF-IFD weight and document id is taken directly from the cache which reduces the search time rather than from retrieving the information from the storage. The analysis was done based on the performance with and without cache resulting in reduced search time in case of the method with cache.

As future work, enhancements using a clustering based search method, to reduce the search time and by adding multiple levels at multiple locations, to gain more efficiency could be added along with the proposed method.

References

[1] Steven M. and William R. Cheswick, "Privacy-Enhanced Searches Using Encrypted Bloom Filters", Cryptology ePrint Archive, Report 2004/022, http://eprint.iacr.org, 2004.

[2] 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, 2010,

[3] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data", IEEE Trans. Parallel and Distributed Systems, vol- 23(8), Aug 2012, pp. 1467–1479.

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

Website: ijetms.in Issue:4, Volume No.4, July-2020 DOI: 10.46647/ijetms.2020.v04i04.003

[5] E.J.Goh, "Secure Indexes", IEEE Transactions on Information Theory, Cryptology ePrint Archive, http://eprint.iacr.org/2003/216. 2003.

[6] I.H. Witten, A. Moffat, and T.C. Bell. , "Managing Gigabytes: Compressing and Indexing Documents and Images", Morgan Kaufmann, May, 1999,

[7] J. Katz, A. Sahai, and B. Waters, "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products",Proc. 27th Ann. Intl Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), 2008.

[8] S. Kamara and K. Lauter, "Cryptographic Cloud Storage", Proc. 14th Intl Conf. Financial Cryptography and Data Security, Jan, 2010,

[9] Daniel A. Menasc and Vasudeva Akula,"Improving the Performance of Online Auctions Through Server-side Activity Based Caching", http://cs.gmu.edu/ menasce/papers/menasce-akula-wwwj.pdf

[10] N. Cao, C.Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS,Vol-25, NO.1,JANUARY,2014.

[11] P. Golle, J. Staddon, and B.Waters,"Secure Conjunctive Keyword Search over Encrypted Data", Proc. Applied Cryptography and Network Security, 2004,pp:-31-45,

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

[13] J. Zobel and A. Moffat, "Exploring the Similarity Space", SIGIR Forum, 32(1), 1998, pp:-18-34.

[14] Singhal (2001). Modern information retrieval: A brief overview,. IEEE Data Eng, 24, 35–43.

[15] Hongqing Liu, Stacy Weng, and Wei Sun, "Introduction of Cache Memory", 2001 https://www.cs.umd.edu/class/fall-2001/cmsc411/proj01/cache/cache.pdf