JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

An Efficient Fuzzy Keyword Search Scheme over Encrypted Data Using Longest Common Subsequence in Cloud Computing

Jyoti Rani[1], Monika Poriye[2,] and Rajender Nath[3]
[1,2,3]Department of Computer Science and Applications, Kurukshetra University Kurukshetra,

Abstract— Cloud computing is growing exponentially; most data owners outsource their data to cloud server. To impart the security, data needs to be encrypted before outsourcing. Thus, demand of searching through encrypted data is occurs. While typing keywords for searching, it is very common to misspell them. Fuzzy keyword search technique on encrypted data is the essential requirement to handle spelling errors and format inconsistencies. In this paper, a new scheme is proposed for fuzzy keyword search on encrypted data. The proposed scheme uses length of longest common subsequence (LCS) to construct the fuzzy keyword set with minimum memory requirement as well as with efficient execution time. This scheme also supports result ranking to rank the files according to their relevance score to a given search request. The comparison of proposed scheme with two existing schemes is also done and the experimental results show the efficiency of the proposed scheme in terms of memory as well as execution time.

Keywords: Encryption, Search, Cloud Computing, Security, Longest Common Subsequence, Fuzzy Search.

1. INTRODUCTION

Cloud computing enable companies to consume computer resources without maintaining them by using pay per use scheme. Pay per use means companies have to pay the amount according to the usage of resources (like electricity) [1]. Various benefits of cloud computing are self-service provisioning, elasticity, workload resilience, migration flexibility and so on. Thus, most of the organizations want to outsource their data to the cloud. But due to the fear of data leakage, loss or theft, they hesitate to place their data in the cloud. Encryption of data before outsourcing it, is one of the solution of this problem. This result into necessity of searching on encrypted data otherwise cloud server have to send all the data to data user in encrypted form which is not an efficient solution. Another important thing is, users are prone to type misspelled word/s and the system should be able to return the original word/s. So, fuzzy based keyword searching is required. In this paper, a scheme is proposed for fuzzy keyword searching. The scheme uses length of longest common subsequence (LCS) to construct fuzzy keyword set. The longest common subsequence (LCS) of the given strings (usually two) is the longest substring that is present in all strings, letters in the LCS need not be contiguous, but they must be in the same order as in the given strings. For example, Let X is

XMJYAUZ and Y is MZJAWXU. The longest common subsequence between X and Y is MJAU [2]. An efficient relevance criterion that is TF*IDF is used to capture the relevance between data files and searching keyword/s. In this scheme, there is no need of any additional data structure (Like list, index, dictionary etc.) to construct fuzzy keyword set. It results into saving memory space and also enables scalability. This scheme is also efficient in terms of execution time.

The organization of this paper is in the following manner: Section 2 describes the various existing schemes for searchable encryption. Section 3 explains the system model for the proposed scheme. Section 4 explains the proposed scheme. Finally, Section 5 will conclude the paper.

2. RELATED WORK

In literature, there have been many schemes proposed for keyword-based searching. In keyword-based searching, the data user has to type the exact searching keyword as in keywords collection/index. Misspell/typing mistakes by data user while typing searching keyword/s would not be tolerable in these schemes. Seny et al. [3] had used keyword index for searching (The keyword index associates each keyword with its associated files) and the user had used pseudo-random bits to mask a dictionary-based keyword index for each file and send it to server in such a way that later user can use the short seeds to help server for recovering selective parts of the index. The problem with this scheme is that it required some additional storage overhead and did not rank the search result. The relevance score and k-nearest neighbour technique have been used to return the ranked search results based on the accuracy with blind storage (To create blind storage, all the documents were divided into fixed sized blocks and these blocks were indexed by random integers). Blind storage was developed to conceal the access pattern of search from cloud server by Hongwei et al. [4]. The inverted/keyword index had been used by Cheng Guo et al. [5] for searching as proposed by Kamara Seny et al. but it also ranked the search results and provided the facility for multi-phrase searching. These schemes were not dynamic. Two more schemes, Seny et al. [6], Wenjun et al. [7], used red-black tree data structure for searching instead of keyword index. Both schemes were dynamic (means addition, deletion or updation of data are possible). Multi-keyword multi-user searchable encryption was proposed by Huang Haiping et al. in [8] that also secured data against adaptive chosen keyword attack. But it also provided the access controllability of encrypted data by addition/ revocation of authorized users.

The various schemes for fuzzy keyword searching had been proposed. Fuzzy keyword searching means misspell/ typing mistakes by data user while typing searching keyword/s would be tolerable in these schemes. These schemes used different methods to create fuzzy keyword set. The wildcard based technique, Jianfeng et al. [9] and Xiaoyu et al. [10], had been used to construct fuzzy set (For example, word ”RAM” constructed fuzzy keyword sets with one letter mistake that is [?RAM ,?AM, R?AM, R?M, RA?M, RA?, RAM?, RA?]). While searching, if any keyword matched with any element of fuzzy set, then corresponding keyword came as matching one. But in that scheme, fuzzy keyword set grows exponentially high, with the increasing number of mistaken letters and both schemes were verifiable. But Xiaoyu et al. [10] had verified the search result based on the RSA accumulator and Jianfeng et al. [9] had used symbol tree for searching as well as for verification of searching results. The other scheme i.e. Jing et al. [11], was the combination of wildcard and dictionary based technique to create fuzzy set. Also the posting list associated with trie-traverse tree was used to build a secure index and to achieve the purpose of ranking. The main idea of Han Lidong et al. [12] is to reduce the edit distance between two keywords to hamming distance between their bloom filters using the technique of VGRAM. The gram dictionary was constructed to create fuzzy set and binary search tree was constructed so that it could be used for searching. The k-gram based index and k-gram based dictionary had been used to construct the fuzzy set in Shugeng et al. [13] whereas the

monogram set to construct fuzzy set was used in M A Manazir et al. [14].

3. SYSTEM MODEL

This system contains 3 entities as shown in Figure 1:

- Data Owner
- Data User
- Cloud Server

Data owner contains all the files in plaintext

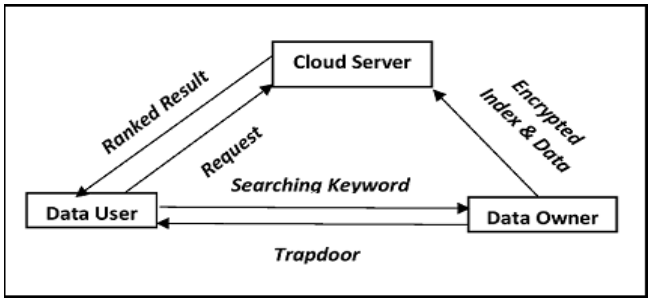


Figure 1. System Model

format. It extracts all the keywords from all files in the collection and constructs inverted index. An inverted index is a data structure that contains all the keywords and their corresponding posting list (The IDs of files which contains that keyword) [15].

Table 1. Inverted Index

Keyword	File IDs	Relevance Score
Kw_i	F_{i1}	1.07
	F_{i5}	0.08
	…	…
	F_{in}	2.06

With each ID of files, there is associated relevance score which defines the relevance of file with corresponding keyword. For example, inverted index for single keyword/term is shown in "Table 1". This will be helpful in result ranking. Relevance score is in order preserving encrypted form to hide actual score from cloud server. After construction of inverted index, it will be encrypted by data owner and the data files are also encrypted. Then encrypted data files and inverted index are outsourced to cloud server by data owner.

Data user is a user who wants to search over encrypted data residing at cloud server. For searching, it sends searching keyword to data owner and collect trapdoor from it. It sends this trapdoor to cloud server and collect resultant files in encrypted form from server.

Cloud server contains the data files and inverted index in encrypted form. Cloud server is honest and curious about data. So, security of data is needed from server.

4. PROPOSED SCHEME

The proposed scheme can handle one letter insertion/deletion and swapping of two consecutive letters. The scheme can be enhanced to cope with more than one letter insertion/deletion/swapping, just by changing a condition which is given in 7^{th} step of this section. Following are the steps of the proposed scheme:

1. Data owner contains all the files in plain text format. All the files having their unique IDs.
2. Data owner extracts all the keywords from files in the collection, which will help in searching through these files.
3. Using these keywords, inverted index will be constructed. Inverted index is a data structure which contains the keywords and their corresponding file IDs with the keywords. The relevance score is also associated with files IDs that states the relevance of file with given keyword. This score will help in ranking the resultant files, which will further help in reducing communication overhead. The relevance score is calculated by TF-IDF rule. Here TF (Term Frequency) is the total number of times a specific keyword appears in a data file and IDF (Inverse Document Frequency) is the ratio of number of data files in the given collection to the number of data files that

contains the given keyword. For calculating the relevance score, TF-IDF [13] is used, the scheme used the following formula which is defined as follow:

$$Score(Q,F_i) = \sum t \epsilon Q \frac{1}{|F_i|}.(1 + lnf_{i,t}).ln(1+\frac{n}{N_t}) \quad \textbf{(1)}$$

Here:

Q - Searched keyword/s;

$\mathbf{F}_{i,t}$ - The TF of term/keyword 't' in fie F_i;

$\mathbf{N}_t$ - Number of data files contain the term t';

n - Total number of files in the collection;

4. After construction of inverted index, it is encrypted by data owner.
5. The data files are also encrypted using any block cipher encryption technique like AES.
6. The encrypted inverted index and encrypted data files outsourced to cloud server for storage by data owner.
7. Searching: The data user sends searching keyword to data owner. To handle typing error or insertion/deletion/swapping of letter/s, fuzzy keyword set will be calculated by data owner. The searching keyword matches with all the keyword in the list (from step 2) and the LCS (Longest Common Subsequence) between them (searching keyword and keywords in the list) is calculated. The length of LCS is used for generation of fuzzy set. And, the following condition is check:

   If ((**L** >= **L2**-1) AND (**L2** <= **L1**+1) AND (**L2** >= **L1**-1)) is **TRUE**, then matching keyword will be added to fuzzy set;

   Here:

   L - Length of LCS for searching keyword and matching keyword in the list;

   L1 - Length of matching keyword;

   L2 - Length of searching keyword;

If the above condition is true for any keyword, then, that keyword will becomes the part of fuzzy keyword set. For example if user enters "INFOARMATION" as searching keyword then for "INFORMATION" keyword, in list of keywords, the above condition will be true. So, it will become the part of fuzzy keyword set. In this scheme, creation of any additional data structure (like index, dictionary etc.) to construct fuzzy keyword set is not required. So, memory will be saved and the scheme is scalable. After construction of fuzzy keyword set trapdoor of fuzzy keyword set will be calculated by hashing. This trapdoor is sent to data user and data user sends this trapdoor to cloud server for searching. Then, Cloud server uses inverted index for searching and finds the file IDs and their corresponding score of matched keyword from inverted index. Data user had also send the decryption key for decryption of matched entries (To know the file IDs and their corresponding score) to cloud server. The files IDs are ranked according to score for result ranking. Data user can also send an optional parameter 'M' with searching request. 'M' denotes the top M files according to score, to be returned to the data user. Then, the files corresponding to files IDs sends to data user in encrypted form. In case of parameter 'M', top M files will be send to data user.

If data user sends multiple keywords as searching keyword in searching request, then server finds the list of files for each keyword and at last, intersection (or union) of resultant files of each searching keyword will be done to get the final result.

4. RESULT AND COMPARATIVE ANALYSIS

4.1 Experimental Setup

We collected approximate 11,000 keywords from various sources like English dictionary, books, internet etc., refined them and conducted searching with some typos. To introduce typos, we took words with

different lengths and run different algorithms to tweak it differently such as- inserting extra letter(s), swapping adjacent letter(s), deleting letter(s), replacing one letter with another and sometimes multiple type of errors in a same word. Each time the proposed scheme finds out the original one with highest similarity. Python (2.2.14) language is used for implementing the proposed scheme. The platform used for execution of the proposed scheme is 32 bit OS (Window 7) with i3 processor and 4GB RAM.

4.2 COMPARATIVE ANALYSIS

In this section, the comparison of the proposed scheme will be done with two existing fuzzy keyword searching schemes [13, 14] and the corresponding results will be shown. The following tables and graphs shows the memory and execution time requirement of the proposed scheme and the two other existing schemes [13, 14]. The memory requirement is shown in MiB, where 1 MiB = 2^{20} bytes. Tables 2 , 3, 4 shows the both execution time and memory requirements and Figure 2(a), 2(b), 2(c) shows the execution time and Figure 3(a), 3(b), 3(c) shows the memory requirements of the proposed scheme, [14], [13] respectively.

Table 2: Memory and Execution Time Requirements of Proposed Scheme

Number of Keywords	Memory (in MiB)	Execution Time (in Seconds)
251	8.64	5.31
3366	9.44	5.76
6550	10.25	6.03
8152	10.29	7.99
10333	10.57	10.96

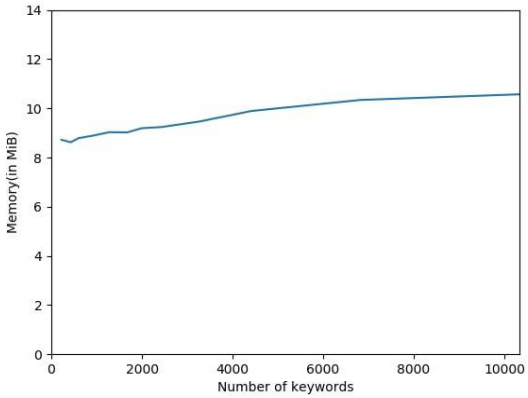
As shown in tables and graphs, in each case the proposed scheme performs better than other two schemes. Means the

Table 3: Memory and Execution Time Requirements[14]

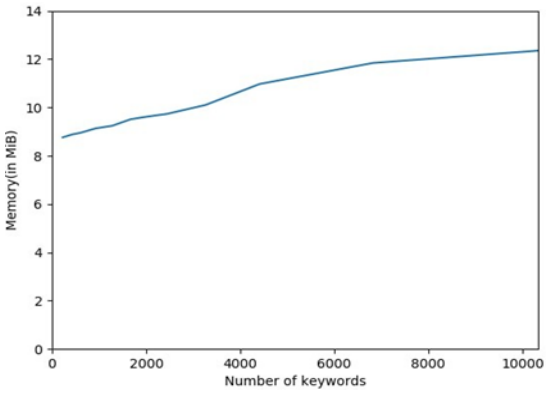
Number of Keywords	Memory (in MiB)	Execution Time (in Seconds)
251	8.80	5.89
3366	10.20	6.57
6550	11.58	7.08
8152	11.57	9.65
10333	12.35	12.20

Table 4: Memory and Execution Time Requirements[13]

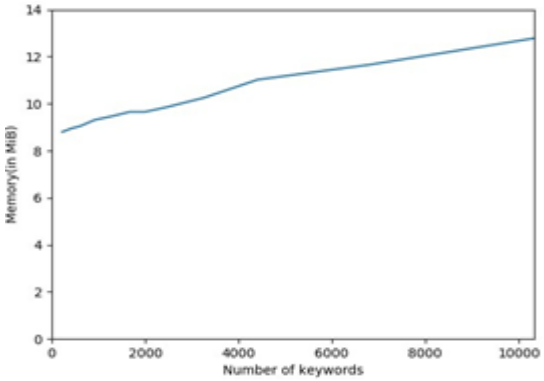
Number of Keywords	Memory (in MiB)	Execution Time (in Seconds)
251	8.86	5.72
3366	10.33	6.17
6550	11.69	8.28
8152	11.98	9.73
10333	12.78	15.67



(a) Proposed scheme

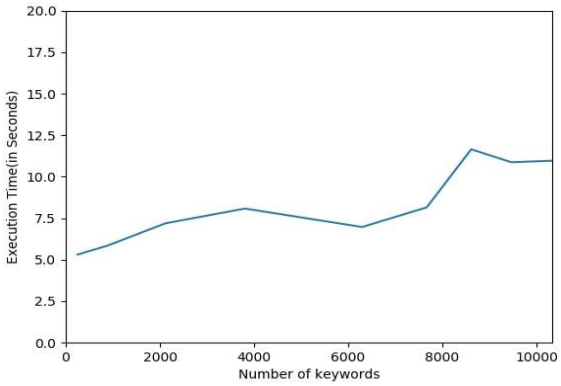


(b) [14]

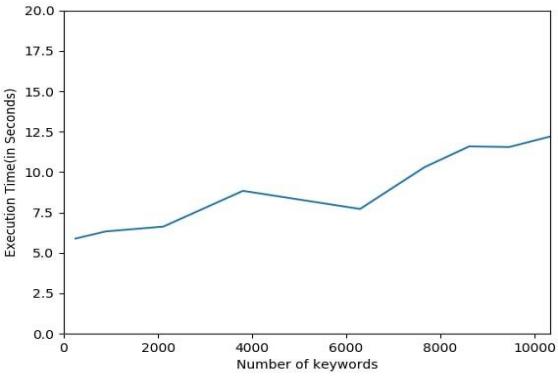


(c) [13]

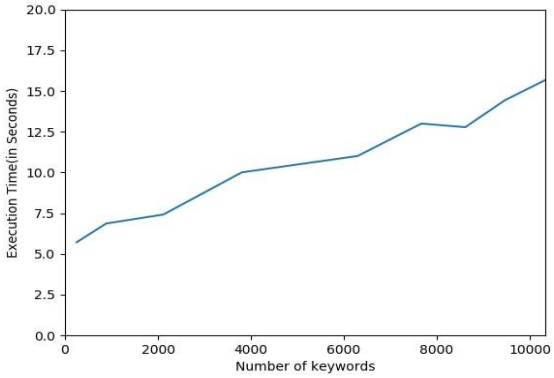
Figure 2: Memory Requirement



(a) Proposed scheme



(b) [14]



(c) [13]

Figure 3: Execution Time

execution time and memory requirements of proposed scheme is efficient than two other schemes. The memory and execution time requirements are shown according to the number of keywords in the whole collection.

5. CONCLUSION

In this paper, a ranked fuzzy keyword search scheme is proposed to enable efficient searching over encrypted data in cloud computing. Length of longest common subsequence is used for fuzzy keyword set construction in this scheme. Fuzzy keyword set construction doesn't need any additional data structure (like index, dictionary, list etc.) which results in saving memory space with efficient execution time. Also, the proposed scheme is compared with the two existing schemes. The results show the efficiency of the proposed scheme.

REFERENCES

[1] TechTarget "cloud computing. "Available at http://searchcloudcomputing.techtarget.com/definition/ cloud-computing.

[2] P. Y. (talk — contribs), "Longest com- mon subsequence problem." Available at https://en.wikipedia.org/wiki/Longest_common_subsequence_problem.

[3] S. Kamara and C. Papamanthou, "Privacy preserving keyword searches on remote encrypted data," Springer, 2005.

[4] H. Li, D. Liu, Y. Dai, T. H.Luan, and X. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," IEEE, 2015.

[5] C. Guo, X. Chen, Y. Jie, Z. Fu, M. Li, and B. Feng, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," IEEE, 2017.

[6] S. Kamara and C.Papamanthou, "Parallel and dynamic searchable symmetric encryption," Springer, 2013.

[7] W. Luo, Yaqiong, and Y. Zhou, "Dynamic searchable encryption with multi-user private search for cloud computing," IEEE, 2015.

[8] H. Huang, J. Du, H. Wang, and R. Wang, "A multi-keyword multi-user searchable encryption scheme based on cloud storage," IEEE, 2016.

[9] J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, and X. Chen, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," ComSIS, 2013.

[10] X. Zhu, Q. Liu, and G. Wang, "A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing," IEEE, 2016.

[11] J. Fang, J. He, and M.G. Muhammad Salman Pathan, "A novel storage and search scheme in cloud computing," IEEE, 2016.

[12] L. Han and C. Hu, "Efficient approximate search using vgram over encrypted data," IEEE, 2014.

[13] S. Ding, Y. Li, J. Zhang, L. Chen, Z.Wang, and Q. Xu, "An efficient and privacy-preserving ranked fuzzy keywords search over encrypted cloud data," IEEE, 2016.

[14] M. A. M. Ahsan, F. Z. Chowdhury, M. Sabilah, A. W. B. A. Wahab, and M. Y. I. B. Idris, "An efficient fuzzy keyword matching technique for searching through encrypted cloud data," IEEE, 2017.

[15] "Inverted index." Available at https://www.elastic.co/guide/en/elasticsearch/guide/current/inverted-index.html.

[16] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," IEEE, 2016