**International Journal of Technical Research and Applications** e-ISSN: 2320-8163,
www.ijtra.com Special Issue 41 (AVALON) (March 2016), PP. 50-53

# COMPARATIVE STUDY ON KEYWORD SEARCHING TECHNIQUES OVER ENCRYPTED DATA IN CLOUD COMPUTING

**Kshitija Jagtap[#1], Shravani Kulkarni[#2], Shuchita Kapoor[#3], Vishakha Thakur[#4], Rashmi Dhumal[*5]**

[#]Computer Engineering, Mumbai Univesity
Terna Engineering College, Nerul
[1]jagtapkshitija40@gmail.com
[2]shrau94@gmail.com
[3]shuchitakapoor99@gmail.com
[4]vishakhat98@gmail.com
Assistant Professor, Terna Engineering College.Nerul
[5]rashmisalvi@gmail.com

***Abstract*- In cloud computing, clients outsource their data to the cloud storage servers. That data may contain sensitive or personal information and the cloud servers cannot be fully trusted in protecting that data. Confidentiality, integrity and availability of that data is a major issue. Encryption is a way to protect the confidentiality of the data. But, encryption makes it difficult to perform effective searches over encrypted data. Traditional search schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest. But the problem with it is that these only support exact keyword search. That is, tolerance for minor typos and format inconsistencies is not considered. Which can easily happen by a typical user. This drawback makes existing techniques unsuitable in Cloud Computing as it affects system usability, making user searching experiences very frustrating. Fuzzy keyword search is used in order to consider the minor typos or format inconsistencies that may happen by the user. So, even if the exact match of the keyword provided by the user is not found in the files, the keywords which are closest possible match are considered. And, the file ID having those closest matched keywords are returned. In this paper, we compare three fuzzy keyword search schemes including wildcard based technique, gram based technique and symbol-based trie-traversal scheme.**

***Keywords*-Fuzzy keyword search, Gram-based search, Wildcard based search, Symbol based search, Encryption.**

## I. INTRODUCTION

In remote storage, the data of the users is stored into the storage server to ensure confidentiality, integrity and availability of the data. Later, if the user wants to receive his data securely through the cloud network, the user may receive the whole data from the server. This induces the communication overload. The user may want to retrieve the data related to a specific keyword only. In this case, we need keyword searching schemes over the encrypted data which are secure as well as efficient and could be easily implemented. In a keyword search technique, a user queries the server to find a particular keyword or a phrase. That keyword is then matched with the fuzzy keyword sets. Fuzzy keyword here means that those keywords approximately match with the actual keyword present in the files stored on the cloud. Fuzzy keyword search is used in order to consider the minor typos or format inconsistencies that may happen by the user. So, even if the exact match of the keyword provided by the user is not found in the files, the keywords which are closest possible match are considered. And, the file ID having those closest matched keywords are returned. There are several techniques for fuzzy keyword generation that are considered in this paper.

## II. DIFFERENT SEARCHING TECHNIQUES

From all the files stored on the cloud, to search desired file, the user enters words or phrases of which fuzzy keywords are generated to make the searching easier and quick. There are various techniques for the generation for fuzzy keywords: Wildcard-based Technique, Gram- based Technique and Symbol-based trie-traversal Scheme. The files are first encrypted and then stored into the cloud server to ensure security. Along with these encrypted files a table is constructed which

contains the keywords present in the file and the generated fuzzy keywords. When a user searches a keyword it is matched with the fuzzy keywords present in the table. So, even if the exact keyword is not matched or there are any typing mistakes done by the user keyword match will be found based on fuzzy keywords and the desired files will be retrieved after decryption.

*A. Wildcard-based Technique*

The wildcard-based fuzzy set of w with edit distance d is denoted as Sw,d. Each wildcard represents an edit function on w. The function for wildcard-based fuzzy set construction is described in Algorithm 1. For example, for the keyword APPLE with the pre-set edit distance 1, its wildcard-based fuzzy keyword set is {APPLE, *APPLE, *PPLE, A*PPLE,A*PLE, · ·· , APP*E, APPL*, APPLE*}. The total number of variants on APPLE constructed are 11 + 1, instead of 11 × 26 + 1. Generally, for a given keyword w with length l, the size will be only 2l + 1 + 1 where * is put at every position, between every letter, at the beginning and the end of the keyword. Also the keyword itself is considered as a fuzzy keyword. The storage overhead can be reduced if larger pre-set edit distance is taken i.e, with the respect to the traditional technique, this technique can help reduce the storage of the index from 30GB to approximately 40MB[5].

Algorithm 1: Wildcard-based Fuzzy Set Construction
1: procedure CreateWildcardSet(w, d)
2: if d >1 then // d = edit distance
2.1: Call CreateWildcardSet(w, d −1);
3: end if
4: if d = 0 then
4.1: Set S′w, d= {w};
5: else
5.1: for (k ←1 to |S′w,d−1|) do
5.1.1: for j ←1 to 2 ∗|S′w,d−1[k]| + 1 do
5.1.1.1: if j is odd then
5.1.1.1.1: Set fuzzy_word as (S′w,d−1)[k];
5.1.1.1.2: Insert ⋆at position ⌊(j + 1)/2⌋;
5.1.1.2: else
5.1.1.2.1: Set fuzzy_word as (S′w,d−1)[k];
5.1.2.2: Replace ⌊j/2⌋ character with ⋆;
5.1.1.3: end if
5.1.1.4: if fuzzy_word is not in S′w,d−1 then
5.1.1.4.1: Set S′w,d= S′w,d∪{fuzzy_word};
5.1.1.5: end if
5.1.2: end for
5.2: end for
6: end if
7: end procedure

*B. Gram-based Technique*

The Gram-based technique used for constructing fuzzy set is based on grams. The gram of a string is a substring that can be used as a signature for efficient and approximate search [1]. Gram can be used for constructing inverted list as well as for the matching purpose[2-4]. In this technique, any edit function will affect at most one specific character of the keyword, leaving all the remaining characters untouched. Thus, the fuzzy keyword set for a keyword w with l single characters supporting edit distance d can be constructed as Sw,d. For example, the gram-based fuzzy set for keyword APPLE with edit distance 1 is {APPLE, APLE, APLE, APPE, APPL}. Compared to the previous technique, the gram-based technique can further reduce the storage of the index from 40MB to approximately 10MB[5]. The function for gram-based fuzzy set construction is described in Algorithm 2.

Algorithm 2 Gram-based Fuzzy Set Construction
1: procedure CreateGramFuzzySet(wi, d)
2: if d >1 then
2.1: Call CreateGramFuzzySet(w, d −1);
3: end if
4: if d = 0 then
4.1: S′w,d= {w};
5: else
5.1: for (k ←1 to |S′w,d−1|) do
5.1.1: for j ←1 to 2 ∗|S′w,d−1[k]| + 1 do
5.1.1.1: Set fuzzy_word as S′w,d−1[k];
5.1.1.2: Delete the j character;
5.1.1.3: if fuzzy_word is not in S′w,d−1 then
5.1.1.3.1: Set S′w,d= S′w,d∪{fuzzy_word}
5.1.1.4: end if
5.1.2: end for
5.2: end for
6: end if
7: end procedure

*C. Symbol-based Trie-Traversal Search Scheme*

For achieving efficiency in searching results, we propose another technique known as symbol-based trie-traversal search scheme. In this technique, a multi-way tree is constructed for storing the fuzzy keyword set {Sw,d}w∈W over a finite symbol set. The basic idea behind this is that all trapdoors sharing a common prefix may have common nodes. The root is always associated with an empty set. The symbols in the trapdoor can be recovered in a search from the root to the leaf that ends the trapdoor. All fuzzy words can be found by a depth-first search. The encrypted file identifiers or the file id will be stored at the ending node or the leaf. With the

returned search results, the user may retrieve the files of his interest using those file ids. These files are then decrypted and provided to the users. The user access structure is created when a user types more than one keyword.

The following example for user access structure shows that, a user requires information about the student who is a girl, should be an Indian and he/ she either scored grade A or is studying in University X.

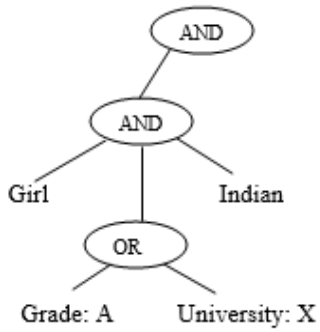


Fig1. Example for multi-way tree

The function for symbol-based trie-traversal search scheme is described in Algorithm 3.

Algorithm 3 SearchingTree
1: procedureSearchingTree({T′w})
2: for i←1 to |{T′w }| do
2.1: set currentNodeAs Root of Gw;
2.2: for j ←1 to l/n do
2.2.1: Set α as αj in the i T′w;
2.2.1.1: if no child of currentNodeContains α then
2.2.1.1.1: break;
2.2.1.2: end if
2.2.2: Set currentNodeAs child containing α;
2.3: end for
2.4: if currentNodeIsLeafNode then
2.4.1: Append currentNode.FIDS to ResultIDSet;
2.4.2: if i = 1 then
2.4.2.1: return resultIDSet;
2.4.3: end if
2.5: end if
3: end for
4: return resultIDSet;
5: end procedure

The main advantage of this technique is that it maintains keyword privacy by efficiently utilising encrypted data form the remote storage.[5]

## IV. COMPARISON BETWEEN DIFFERENT TECHNIQUES

TABLE I
COMPARISON

| Parameter | Fuzzy keyword search schemes | | |
|---|---|---|---|
| | **Wildcard based technique** | **Gram based technique** | **Symbol based trie-traversal scheme** |
| Storage size | approximately 40MB | approximately 10MB | approximately 13MB |
| Keywords generated | 2l+1+1<br>l= length of original Keyword. | l-t+1<br>l= length of original keyword<br>t= edit Distance. | Where a multi-way tree is constructed for storing the fuzzy keyword set. |
| Searching time | Moderate searching time required because of the presence of *. | Less searching time is required because keywords generated are less. | Depth first search over a multi-way tree requires a lot of time. |

Wildcard based technique reduces the storage of the index from 30GB to approximately 40MB as compared to the straightforward approach. For eg, for the keyword CAT, The total number of variants = (3*2)+1+1=8 instead of 7×26+1=183.

Compared to wildcard based construction, gram-based construction can further reduce the storage of the index from 40MB to approximately 10MB under the same setting as in the wildcard-based approach.

For the example CAT, the total number of variants for edit distance 1 are 3 i.e, 3-1+1.

## V. CONCLUSION

In this paper we have presented different searching techniques for fuzzy keyword generation: Wildcard-based search technique, Gram-based search technique, Symbol-based trie-traversal search technique. Wildcard based technique generates huge number of keywords so there is more probability of getting the file to the matching keyword and memory required to store the keyword is less, as compared to the traditional approach. Also the number of keywords generated are much more compares to traditional approach. Gram based technique is simple for large n i.e. the number of keywords generated. Symbol-based trie-traversal search technique uses depth first search for the traversal and matching of

keyword. Using depth first search it retrieves the file ID as per the matched keyword. As depth first search is time consuming, this technique requires more time for the execution.

We conclude that out of the three techniques, wildcard-based technique is the most efficient as it generates huge number of keywords for the searched word and yet takes less space and memory.

## ACKNOWLEDGEMENT

The authors would like to thank the college for giving them a platform to showcase their paper. They would also like to thank their assistant professor for supporting and guiding them in the creation and experimentation of this paper.

## REFERENCES

[1] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in Proc. of WWW'09, 2009.
[2] S. Chaudhuri, V. Ganti, and R. Motwani, "Robust identification of fuzzy duplicates," in Proc. of ICDE'05.
[3] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in Proc. of VLDB'06, 2006, pp. 918–929.
[4] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin, "An efficient filter for approximate membership checking," in Proc. of SIGMOD'06.
[5] Jin Li, Qian Wang, "Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing"