RESEARCH ARTICLE OPEN ACCESS

Fuzzy Keyword Search Over Encrypted Data in Cloud Computing

Yogesh K. Gedam, Prof. Mrs. Varshapriya J.N.
M. Tech (Software Engineering) Veermata Jijabai Technological Institute Mumbai, India
Veermata Jijabai Technological Institute Mumbai, India

Abstract
As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud. For the protection of data privacy, sensitive data usually have to be encrypted before outsourcing, which makes effective data utilization a very challenging task. Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest, these techniques support only exact keyword search. This significant drawback makes existing techniques unsuitable in cloud computing as it is greatly affect system usability, rendering user searching experiences very frustrating and system efficiency very low. In this paper, for the first time we formalize and solve the problem of effective fuzzy keyword search over encrypted cloud while maintaining keyword privacy. In our solution, we exploit edit distance to quantify keyword similarity and develop new advanced technique on constructing fuzzy keyword sets which greatly reduces the storage and representation overheads. In this way, we show that our proposed solution is secure and privacy preserving, while realizing the goal of fuzzy keyword search.
Key Words: String Matching Algorithm, System Architecture

I. INTRODUCTION

As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud such as emails, personal health records, government documents, etc. By storing their data into the cloud the data owners can be leaved from the burden of data storage and maintenance so as to enjoy the on demand high quality data storage service. In Cloud Computing, data owners may share their outsourced data with a large number of users. The individual users might want to only retrieve certain specific data files they are interested in during a given session. One of the most popular ways is to selectively retrieve files through keyword based search instead of retrieving all the encrypted files back which is completely impractical in cloud computing scenarios. Such keyword based search technique allows users to selectively retrieve files of interest and has been widely applied in plaintext search scenarios, such as Google search. Unfortunately, data encryption restricts user's ability to perform keyword search and makes the traditional plaintext search methods unsuitable for Cloud Computing. Besides this, data encryption also demands the protection of keyword privacy since keywords usually contain important information related to the data files. Although encryption of keywords can secure keyword privacy, it further renders the traditional plaintext search techniques useless in this scenario.

To securely search over encrypted data, searchable encryption techniques have been developed in recent years. Searchable encryption schemes usually build up an index for each keyword of interest and associate the index with the files that contain the keyword. By integrating the trapdoors of keywords within the index information, effective keyword search can be realized while both file content and keyword privacy are well preserved. Although allowing for performing searches securely and effectively, the existing searchable encryption techniques do not suit for cloud computing scenario since they support only *exact* keyword search.

In this paper, we focus on enabling effective yet privacy preserving fuzzy keyword search in Cloud Computing. To the best of our knowledge, we formalize for the first time the problem of effective fuzzy keyword search over encrypted cloud data while maintaining keyword privacy. Fuzzy keyword search greatly enhances system usability by returning the matching files when users' searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when *exact* match fails. More specifically, we use edit distance to quantify keywords similarity and develop a novel technique, i.e. an wildcard based technique, for the construction of fuzzy keyword sets. This technique eliminates the need for enumerating all the fuzzy keywords and the resulted size of the fuzzy keyword sets is significantly reduced. Based on the constructed fuzzy keyword sets, we propose an efficient fuzzy keyword search scheme. Through rigorous security analysis, we show that the proposed

solution is secure and privacy preserving, while correctly realizing the goal of fuzzy keyword search.

II. SERVICE ARCHITECTURE

We consider a high level architecture for cloud data utilization services illustrated in Fig. 1. At its core, the architecture consists of three entities: *data owner*, *user*, and *cloud server*. Under the cloud paradigm, the data owner may represent either an individual or enterprise customer, who relies on the cloud server for remote data storage and maintenance, and thus is relieved from the burden of building and maintaining local storage infrastructure. Assume the data owner has a collection of *n* data files C = (*F*1, *F*2, …, *FN*) to be stored in the cloud server, where a predefined set of distinct keywords in C is denoted as *W* = {*w*1, *w*2, ..., *wp*}.

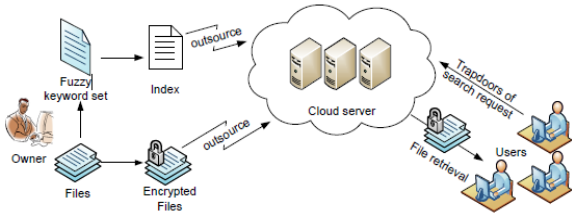


Fig. 1: Architecture of the fuzzy keyword search

Plaintext fuzzy keyword search:

The importance of fuzzy search has received attention in the context of plaintext searching in information retrieval community. They addressed this problem in the traditional information access paradigm by allowing user to search without using try-and-see approach for finding relevant information based on approximate string matching. At the first glance, it seems possible for one to directly apply these string matching algorithms to the context of searchable encryption by computing the trapdoors on a character base within an alphabet. However, this trivial construction suffers from the dictionary and statistics attacks and fails to achieve the search privacy.

Searchable encryption:

Traditional searchable encryption has been studied in the context of cryptography. Among those works, most are focused on efficiency improvements and security definition formalizations. The first construction of searchable encryption was proposed by Song et al, in which each word in the document is encrypted independently under a special two-layered encryption construction. Goh proposed to use Bloom filters to construct the indexes for the data files. To achieve more efficient search, Chang et al. and Curtmola et al.both proposed similar "index" approaches, where a single encrypted hash table index is built for the entire file collection. In the index table, each entry consists of the trapdoor of a keyword and an encrypted set of file identifiers whose corresponding data files contain the keyword. As a complementary approach, Boneh et al. presented a public-key based searchable encryption scheme, with an analogous scenario to that of. Note that all these existing schemes support only exact keyword search, and thus are not suitable for Cloud Computing.

Others:

Private matching , as another related notion, has been studied mostly in the context of secure multiparty computation to let different parties compute some function of their own data collaboratively without revealing their data to the others. These functions could be intersection or approximate private matching of two sets, etc. The private information retrieval is an often-used technique to retrieve the matching items secretly, which has been widely applied in information retrieval from database and usually incurs unexpectedly computation complexity.

III. THREAT MODEL

We consider a semi-trusted server. Even though data files are encrypted, the cloud server may try to derive other sensitive information from users' search requests while performing keyword-based search over C. Thus, the search should be conducted in a secure manner that allows data files to be securely retrieved while revealing as little information as possible to the cloud server. In this paper, when designing fuzzy keyword search scheme, we will follow the security definition deployed in the traditional searchable encryption. More specifically, it is required that nothing should be leaked from the remotely stored files and index beyond the outcome and the pattern of search queries.

IV. DESIGN MODEL

In this paper, we address the problem of supporting efficient yet privacy-preserving fuzzy keyword search services over encrypted cloud data. Specifically, we have the following goals: i) to explore different mechanisms for constructing storage-efficient fuzzy keyword sets; ii) to design efficient and effective fuzzy search schemes based on the constructed fuzzy keyword sets; iii) to validate the security and evaluate the performance by conducting extensive experiments.

V. PRELIMINARIES

Edit Distance:

There are several methods to quantitatively measure the string similarity. In this paper, we resort to the well-studied edit distance for our purpose. The edit distance ed(*w*1, *w*2) between two words *w*1 and *w*2 is the number of operations required to transform

one of them into the other. The three primitive operations are 1) Substitution: changing one character to another in a word; 2) Deletion: deleting one character from a word; 3) Insertion: inserting a single character into a word. Given a keyword *w*, we let *Sw,d* denote the set of words *w_* satisfying ed(*w,w_*) ≤ *d* for a certain integer *d*.

Fuzzy Keyword Search:

Using edit distance, the definition of fuzzy keyword search can be formulated as follows: Given a collection of *n* encrypted data files *C* = (*F*1, *F*2, . . . , *FN*) stored in the cloud server, a set of distinct keywords *W* = *{w*1, *w*2, ...,*wp}* with predefined edit distance *d*, and a searching input (*w, k*) with edit distance *k* (*k* ≤ *d*), the execution of fuzzy keyword search returns a set of file IDs whose corresponding data files possibly contain the word *w*, denoted as *FIDw*: if *w* = *wi* ∈ *W*, then return *FIDwi* ; otherwise, if *w* _∈ *W*, then return *{FIDwi }*, where ed(*w,wi*) ≤ *k*. Note that the above definition is based on the assumption that *k* ≤ *d*. In fact, *d* can be different for distinct keywords and the system will return *{FIDwi }* satisfying ed(*w,wi*) ≤ min*{k, d}* if exact match fails.

VI. FUZZY KEYWORD SEARCH SCHEME

The architecture of our scheme is shown in Figure 1. Fuzzy keyword search scheme is consist of Data Owner, Data User, and Cloud Storage Provider.

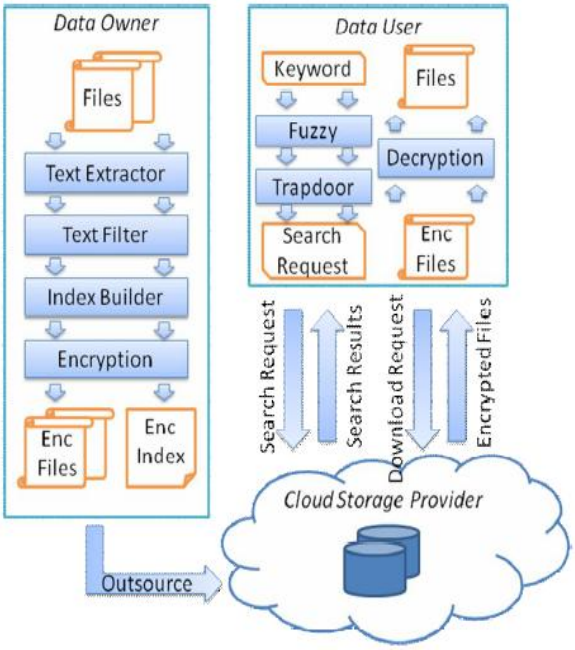


Figure 1 The architecture of fuzzy keyword search scheme based on "Dictionary-based Fuzzy Set Construction".

MODULES IN DATA OWNER

Definition 3.1 (Files): The set of files ready to outsource is denoted as :

FILE = {< fid_1, $file_1$ >, <fid_2, $file_2$ >,.....,<fid_n, $file_n$ >} $file_1$ is the file body, fid_1 the identifier of fid_1.

Definition 3.2 (Text Extractor): *file* has a variety of formats, such as doc, xls, and txt. Text Extractor is used to extract the text part of file.
Input: *FILE* ;
Output:
TEXT = {< fid_1, $text_1$ >, <fid_2, $text_2$ >, >,.....,<fid_m, $text_m$ >}

Definition 3.3 (Text Filter): Except for the English words in *text* , it also contains a lot of non-critical information such as punctuation, separators, etc. Text Filter is used to filter those non-critical information and generate a list of words.
Input:*TEXT* ;
Output:
LIST=
{< fid_1, $LIST_1$ >, <fid_2, $LIST_2$ >,.....,<fid_n, $LIST_n$>}
$LIST_i$ is the set of distinct words in $text_i$.

Definition 4 (Index Builder): Based on the Dictionary-based Fuzzy Keyword Set Construction, we define the index builder as follows:
Input: *LIST* , a dictionary D_1 , and the string edit distance d_1;
Output:
INDEX=$\{< F_{word_1 d_1}^{D_1}, FID_{word_1} >\}_{word_1 cLIST_1 "....cLIST_n}$
FID_{word_i} is the set of *fids* whose corresponding *files* contain the word $word_i$.

Definition 5 (Encryption): Let *MasterKey* be the data owner's secret key, *MasterKey* can generate $K_{FILE} = \{key_1, key_1 \ ....., key_1 \ \}$ and K_{INDEX} , which are used to encrypt *FILE* and *INDEX* , respectively.
Input: *FILE* , *INDEX* , *MasterKey*
Key Generation:
$K_{INDEX} = f_1(MasterKey)$
$key_i = f_2(MasterKey||fid_i), key_i \varepsilon K_{FILE}$
Output:$< INDEX_{Enc}, FILE_{Enc} >, INDEX_{Enc} =$
$\{< \{T_{K_{INDEX}}(W)\}_{W \varepsilon F_{word_1 d_1}^{D1}}, Enc_1(K_{INDEX}, FID_W) >$
$\}wordI\varepsilon LIST1 \cup LIST2 \ldots . \cup LIST3$
$FILE_{Enc} = \{Enc_2(key_i, file_i)\}_{key_i \in K_{FILE}}, file_i \in FILE$
$f_1$and f_2 can be implemented by hash functions; Enc_1 and Enc_2can be implemented by blockcipher such as AES and DES. *T* can be implemented by the one-way function, which is similar as. We assume Data Owner shares *MasterKey* with Data User.

MODULES IN DATA USER

Definition 6 (Fuzzy): To search with keyword w , Data User need to generate the fuzzy keyword set of w in string edit distance d_2 . Input: keyword w , a dictionary D_2 , and the string edit distance d_2 ;
Output:
FUZZY=$F_{w_1 d_2}^{D_2} = \{fuzzy_1, fuzzy_2, \ldots., fuzzy_p\}$

Definition 7 (Trapdoor): The trapdoor function is required to be the same with *T* in definition 3.5.
Input: $FUZZY$;
Output:$FUZZY_{Enc} = \{T_{K_{INDEX}}(fuzzy_i)\}_{fuzzy\ i \epsilon FUZZY}$
Definition 3.8(Decryption):
Input:$FILE'_{Enc}$ C$FILE_{Enc}$;
Output:
$FILE'$C$FILE, FILE' = \{Dec_1(key_i, cipher_i))\}_{cipher_i \epsilon FILE'_{Enc}}$

VII. FUZZY KEYWORD SEARCH PROCESS

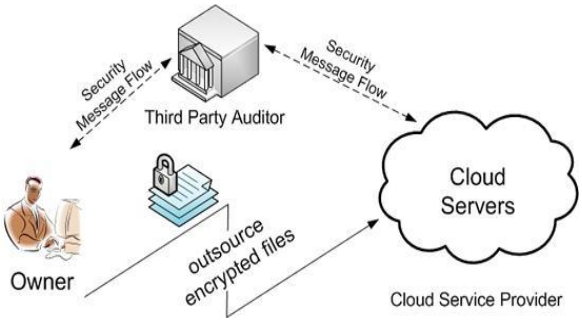
Let's begin with Data Owner. First, Data Owner uses "Text Extractor", "Text Filter" and "Index Builder" to build an index for the entire files, then he/she can generate K_{FILE} and K_{INDEX} by *MasterKey* to encrypt files and index respectively. Finally, Data Owner outsources $FILE_{Enc}$ and $INDEX_{Enc}$ to the cloud storage provider. When searching with keyword w , Data User generates the fuzzy keywords of w and computes their trapdoors ($FUZZY_{Enc}$) with K_{INDEX}. Then Data User sends $FUZZY_{Enc}$ as search request to Cloud Storage Provider. Cloud Storage Provider locates $FUZZY_{Enc}$ within $INDEX_{Enc}$, and then returns all the matched encrypted *fids* as search result. Afterward Data User uses *INDEX K* to decrypt search result. If he/she wants to download some files on the basis of the search result, he/she just sends the corresponding encrypted *fids* to Cloud Storage Provider, and then Cloud Storage Provider will return the corresponding encrypted files. Finally, Data User can use K_{FILE} decrypt them.

PROPOSED SYSTEM:
Main Modules:
1. Wildcard – Based Technique
2. Gram - Based Technique
3. Symbol – Based Trie – traverse Search Scheme

1.WILDCARD-BASED TECHNIQUE:

In the above straightforward approach, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we proposed to use an wildcard to denote edit operations at the same position. The wildcard-based fuzzy set edits distance to solve the problems.

For example, for the keyword CASTLE with the pre-set edit distance 1, its wildcard based fuzzy keyword set can be constructed as
SCASTLE, 1 = {CASTLE, *CASTLE,*ASTLE, C*ASTLE, C*STLE, CASTL*E, CASTL*, CASTLE*}.
Edit Distance:
a. Substitution
b. Deletion
c. Insertion
a) **Substitution** : changing one character to another in a word;
b) Deletion : deleting one character from a word;
c) **Insertion**: inserting a single character into a word.



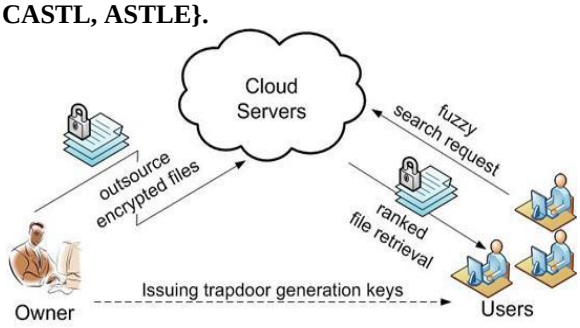
Algorithm 1 Wildcard-based Fuzzy Set Construction
1: procedure CreateWildcardFuzzySet(wi, d)
2: if d > 1 then
3: Call CreateWildcardFuzzySet(wi, d − 1);
4: end if
5: if d = 0 then
6: Set S′wi,d = {wi};
7: else
8: for (k ← 1 to |S′wi,d−1|) do
9: for j ← 1 to 2 * |S′wi,d−1[k]| + 1 do
10: if j is odd then
11: Set fuzzyword as S′wi,d−1[k];
12: Insert ⋆ at position [(j + 1)/2];
13: else
14: Set fuzzyword as S′wi,d−1[k];
15: Replace [j/2]-th character with ⋆;
16: end if
17: if fuzzyword is not in S′wi,d−1 then
18: Set S′wi,d = S′wi,d ∪ {fuzzyword};
19: end if
20: end for
21: end for
22: end if
23: end procedure
24: end procedure

2. GRAM-BASED TECHNIQUE:

Another efficient technique 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 approximate search. While gram has been widely used for constructing inverted list for approximate

string search, we use gram for the matching purpose. We propose to utilize the fact that any primitive edit operation will affect at most one specific character of the keyword, leaving all the remaining characters untouched. In other words, the relative order of the remaining characters after the primitive operations is always kept the same as it is before the operations.
For example, the gram-based fuzzy set SCASTLE, 1 for keyword CASTLE can be constructed as

{CASTLE, CSTLE, CATLE, CASLE, CASTE, CASTL, ASTLE}.



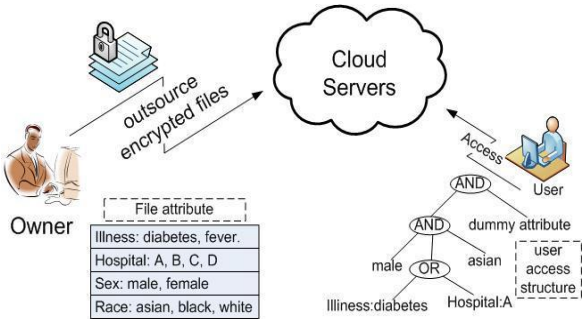
Algorithm 2 Gram-based Fuzzy Set Construction

```
1:  procedure CreateGramFuzzySet(wi, d)
2:        if d > 1 then
3:          Call CreateGramFuzzySet(wi, d − 1);
4:       end if
5:       if d = 0 then
6:         S′wi,d = {wi};
7:       else
8:         for (k ← 1 to |S′wi,d−1|) do
9:           for j ← 1 to 2 * |S′wi,d−1[k]| + 1 do
10:            Set fuzzyword as S′wi,d−1[k];
11:            Delete the j-th character;
12:            if fuzzyword is not in S′wi,d−1 then
13:              Set S′wi,d = S′wi,d ∪ {fuzzyword}
14:            end if
15:          end for
16:       end for
17:    end if
18:  end procedure
19:  end procedure
```

3. SYMBOL-BASED TRIE-TRAVERSE SEARCH SCHME

To enhance the search efficiency, we now propose a symbol-based trie-traverse search scheme, where a **multi-way tree** is constructed for storing the fuzzy keyword set over a finite symbol set. The key idea behind this construction is that all trapdoors sharing a common prefix may have common nodes. The root is associated with an empty set and the symbols in a trapdoor can be recovered in a search from the root to the leaf that ends the trapdoor. All fuzzy words in the trie can be found by a depth-first search.

In this section, we consider a natural extension from the previous single-user setting to multi-user setting, where a data owner stores a file collection on the cloud server and allows an arbitrary group of users to search over his file collection.



Algorithm 3 SearchingTree

```
1:  procedure SearchingTree({T′w})
2:        for i ← 1 to |{T′w }| do
3:          set currentnode as root of Gw;
4:          for j ← 1 to l/n do
5:             Set α as αj in the i-th T′w;
6:             if no child of currentnode contains α then
7:               break;
8:             end if
9:              Set currentnode as child containing α;
10:          end for
11:          if currentnode is leafnode then
12:                        Append currentnode.FIDs to
resultIDset;
13:            if i = 1 then
14:              return resultIDset;
15:            end if
16:         end if
17:       end for
18:       return resultIDset;
19:  end procedure
20:  end procedure
```

VIII. CONCLUSION

In this paper, for the first time we formalize and solve the problem of supporting efficient yet privacy-preserving fuzzy search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We design two advanced techniques (i.e., wildcard-based and gram- based techniques) to construct the storage-efficient fuzzy keyword sets by exploiting two significant observations on the similarity metric of edit distance. Based on the constructed fuzzy keyword sets, we further propose a brand new symbol-based trie-traverse searching scheme, where a multi- way tree structure is built up using symbols transformed from the resulted fuzzy keyword sets. Through rigorous security analysis, we show that our proposed solution is secure and privacy- preserving, while correctly realizing the goal of fuzzy keyword search. Extensive experimental results demonstrate the efficiency of our solution.

As our ongoing work, we will continue to research on security mechanisms that support 1) search semantics that takes into consideration conjunction of keywords, sequence of keywords, and even the complex natural language semantics to produce highly relevant search results. and 2) search ranking that sorts the searching results according to the relevance criteria.

REFERENCE

[1] D. Parkhill, "*The challenge of the computer utility*,"Addison-Wesley Educational Publishers Inc., US, 1966.

[2] P. Mell and T. Grance, "*Draft nist working definition of cloud computing*," Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009.

[3] M. Armbrust and et.al, "*Above the clouds: A berkeley view of cloud computing*," Tech. Rep., Feb 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[4] Google, "*Britney spears spelling correction*," Referenced online at http://www.google.com/jobs/ britney.html, June 2009.

[5] M. Bellare, A. Boldyreva, and A. O'Neill, "*Deterministic and efficiently searchable encryption*," in Proceedings of Crypto 2007, volume 4622 of LNCS. Springer-Verlag, 2007.

[6] D. Song, D. Wagner, and A. Perrig, "*Practical techniques for searches on encrypted data*," in Proc. of IEEE Symposium on Security and Privacy'00, 2000.

[7] E.-J. Goh, "*Secure indexes*," Cryptology ePrint Archive, Report 2003/216, 2003, http://eprint.iacr. org/.

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "*Public key encryption with keyword search*," in Proc. of EUROCRYP'04, 2004.

[9] B. Waters, D. Balfanz, G. Durfee, and D. Smetters, "*Building an encrypted and searchable audit log*," in Proc. of 11th Annual Network and Distributed System, 2004.

[10] Y.-C. Chang and M. Mitzenmacher, "*Privacy preserving keyword searches on remote encrypted data*," in Proc. of ACNS'05, 2005.

[11] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "*Searchable symmetric encryption: improved definitions and efficient constructions*," in Proc. of ACM CCS'06, 2006.

[12] D. Boneh and B. Waters, "*Conjunctive, subset, and range queries on encrypted data*," in Proc. of TCC'07, 2007, pp. 535–554.

[13] F. Bao, R. Deng, X. Ding, and Y. Yang, "*Private query on encrypted data in multi-user settings*," in Proc. of ISPEC'08, 2008.

[14] X. Yang, B. Wang, and C. Li, "*Cost-based variable-length-gram selection for string collections to support approximate queries efficiently*," in Proc. of ACM SIGMOD'08, 2008.

[15] C. Li, J. Lu, and Y. Lu, "*Efficient merging and filtering algorithms for approximate string searches,"* in Proc. of ICDE'08, 2008.