An Efficient and Privacy-Preserving Ranked Fuzzy Keywords Search over Encrypted Cloud Data

Shugeng Ding[1], Yidong Li[2], Jianhui Zhang[1], Liang Chen[1], Zhen Wang[1], Qunqun Xu[2]
[1]Shandong Luneng Software Technology, Shandong, China
[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

***Abstract*—As cloud computing becomes widespread, more and more users prefer to outsource their local sensitive data into the cloud. In order to protect data privacy, these sensitive data usually has to be encrypted before outsourcing, which makes effective data utilization a very difficult task. Although traditional searchable encryption techniques allow users to securely search over encrypted cloud data, they only support exact single keyword search, i.e. they do not allow any minor spelling errors or format inconsistencies. Besides, these traditional schemes support only Boolean search, without capturing any relevance of data files and rarely sort the search result. Recently, fuzzy keyword search over encrypted data techniques are introdeced to resolve the problem of spelling errors and format inconsistencis. However, they may incur large index size, search result inaccuracy and high search complexity, which greatly reduce the system usability and efficiency. In this paper, we propose the solution for privacy preserving ranked fuzzy keyword search over encrypted cloud data with small index. We use k-grams and Jaccard coefficient to constrcuct fuzzy keyword set and produce fuzzy results, and efficient relevance criteria (e.g., $TF \times IDF$) to capture the relevance between data files and search requests. Extensive experiment result shows the efficiency of proposed scheme.**

***Index Terms*—K-Gram; Fuzzy Keyword Search; Ranked Keyword Search; Searchable Encryption; Cloud Computing**

I. Introduction

Recently, with the rapid development of cloud computing and portable device market, in order to save storage costs and enjoy the on-demand high quality applications and services anywhere anytime in a pay-as-you-use way, more and more individuals and enterprises (collectively called data users) prefer to outsource their local data into the cloud server. However, the server is considered as "honest but curious" and is not fully trusted by data owners. To protect data privacy, sensitive data such as the patients' health information or personal health records, government documents, emails have to be encrypted before outsourcing, which makes efficient data utilization a very challenge task. Besides, in cloud computing, some data owners may want to share their outsourced data with many other data users. And the single user might intend to only search certain data files during a given occasion.

Although traditional searchable encryption techniques [2], [5], [8], [9] could achieve the guarantee of security and efficiency by creating an index for each keyword and associate the keyword with the data files which contain the keyword, they may have some drawbacks. First, they only support exact keyword search, i.e., they do not allow any minor spelling errors and format inconsistencies[14], which greatly reduce the system usability. It is quite normal that users may input keywords that are not precisely match the pre-set keywords due to some minor spelling mistakes or format inconsistencies. Second, these schemes only support Boolean search, without capturing any relevance between data files and search request, rarely sort the search result. Users have to retrieve all of the received files to find the ones they most interested in, which may incur large post-processing costs, sending back all the files that contain or not contain the keywords further brings large unnecessary network traffic. Although the state-of-the-art information retrieval techniques have already been utilizing a variety of scoring mechanisms [17] to rank the relevance of data files, but these plaintext-based schemes are not suitable in the context of encrypted data. Besides, they are too slow to be used on a large data collection, and they are only support single keyword search, i.e., they are not scalable.

According to the first drawback, Li et al. [14] first solved the problem of fuzzy keyword search over encrypted cloud data. They proposed the "Wild-card-based Fuzzy Set Construction", in which each keyword needs to build a fuzzy set. An improved method, "Dictionary-based Fuzzy Set Construction" is proposed in [15], in which each keyword is corresponding with much less fuzzy keywords. According to the second drawback, Wang et al. [18] proposed a secure ranked keyword search scheme over encrypted cloud data, they and Cao et al. [7] proposed several improvements such as multi-keyword search feature. Our early work [20] has been proposed a solution to the ranked fuzzy keyword search over encrypted cloud data problem. However, it only support single keyword search, and it may meet with the problem of search result inaccuracy: when a user inputs a keyword and the exact match fails, and there are more than one fuzzy keyword set contain search keyword's fuzzy set, the system may cannot be sure which files should be returned. All the problems mentioned above will greatly reduce the system usability and efficiency

In this paper, we propose the solution for privacy preserving ranked fuzzy keyword search over encrypted cloud data. We use k-gram to constrcuct fuzzy keyword set and Jaccard coefficient to quantify keyword similarity[21], to avoid enumerating all fuzzy keywords, and thus reducing the index space and search space, we eliminate keywords with Jaccard coefficient smaller than our threshold value, which may greatly reduces the index size, storage and communication costs. We use efficient relevance criteria (e.g., $TF \times IDF$) to capture the relevance between data files and search requests. In this paper, we call the criteria relevance score. For security consideration, we utilize One-to-many Order Preserving Mapping

(OPM) algorithm to encrypt the score and Order-Preserving Symmetric Encryption (OPSE) scheme in [3], [18]. And we will utilize some efficient algorithms to solve the problem of result unaccuracy which is mentioned above.

The rest of paper is organized as follows: Section 2 summarizes the features of related work. Section 3 introduces the system model, threat model, and our design goal. Section 4 formularizes existing keyword search schemes. Section 5 provides the construction and details of our proposed scheme. Section 6 presents the security and efficiency analysis. Finally, section 7 concludes the paper.

II. Related Works

A. Searchable encryption

Existing searchable encryption schemes [1], [2], [5], [9], [13] generally create an encrypted searchable index for each keyword and associate the keyword with the data files which contain the keyword. et al. Boneh et al. [4] proposed a public key encryption (PKE) scheme, which means that each file is encrypted using public key by data owners but the authorized data users can search the files using their private key, but this scheme fails regarding access policy and dictionary attack, and it takes too much time to calculate public key. [13] proposed a general search scheme called predicate encryption schemes, they proposed to support both conjunctive and non-conjunctive query. However, none of those existing Boolean keyword searchable encryption techniques support ranked or fuzzy keyword search.

B. Fuzzy keyword searchable encryption

Li et al. [14] first solved the problem of fuzzy keyword search over encrypted data. They proposed the "Wild-card-based Fuzzy Set Construction (WFSC)", in which each keyword needs to build a fuzzy set, they use edit distance to quantify keyword similarity. An improved method, "Dictionary-based Fuzzy Set Construction (DFSC)" is proposed in [15], in which each keyword is corresponding with much less fuzzy keywords. Our early work [20] proposed a privacy preserving ranked fuzzy keyword search scheme over encrypted cloud data, it combines the DFSC technique and OPM algorithm to construct a new index structure and implements both fuzzy keyword search and ranked keyword search functionality.

C. Ranked keyword searchable encryption

Ranked keyword search captures the relevance between data files and search request and ranks the search results. Boldyreva et al. [3] proposed a order preserving symmetric encryption (OPSE) scheme, it supports deterministic property in which a random coin generator and sampling function implimented. Wang et al. [18] introduced a more secure ranked keyword search scheme over encrypted cloud data, it uses Order Preserving Mapping (OPM) algorithm to encrypt the score. Cao et al. [7] proposed several improvements such as multi-keyword search feature.

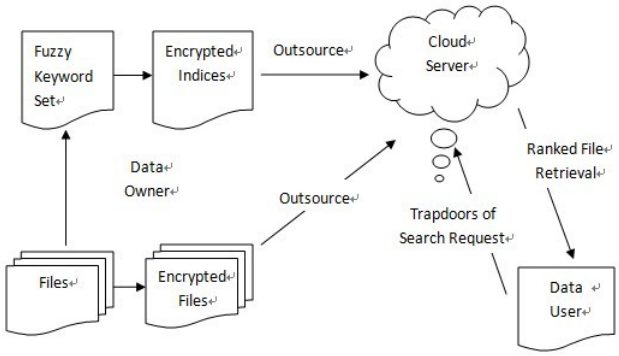

Figure 1. Frame of keyword search over encrypted cloud data

D. Multi-keywords searchable encryption

There are many literatures tried to improve the efficiency and security of a single keyword search scheme [9], [2]. Golle et al. [11] first proposed the conjunctive keyword search technique, and Byun et al. [6] introduced a more efficient conjunctive keyword search scheme. Cao et al. [7] presented a muti-keyword search scheme over encrypted cloud data and established various privacy requirements. Liu et al. [16] allows the cloud server to participate in the partial decryption of the data files.

III. Problem Formulations

A. System model

We consider a cloud system consisting of three entities in this paper: cloud server, data owner and data user, as Fig.1 illustrates.

In our system model, data owner has a collection of n files $F = (F_1, F_2, F_3, \ldots, F_n)$ and intends to outsource them to the cloud server in encrypted form C, a predefined set of distinct keywords $W = (w_1, w_2, ..., w_m)$ is extract from F. In order to search on the encrypted data effectively, before outsourcing, data owner will first build a secure searchable index I for each keyword in W, then outsources all the encrypted indices I_{Enc} and data files C to the cloud server.

In this paper we assume the authorization between data owner and data users has been appropriately done. An authorized user can input a search request and selectively retrieve files which he/she interests. To search the file collection for a given search request such as Q, the authorized user first computes a fuzzy set of K and then acquires a corresponding trapdoor T_Q through search control mechanisms such as broadcast encryption [9]. Upon receiving T_Q, the cloud server is responsible for searching the index I_{Enc} and return corresponding set of encrypted file IDs. To improve the search accuracy, the search result should be ranked by the cloud server according to some ranking criteria.

The proposed ranked fuzzy keyword search scheme returns the results according to the following rules:

1) If the user's search request T_Q exactly matches the pre-set keyword, the cloud server returns corresponding ranked files containing the keyword;

2) If exact match fails, i.e., there exists minor spelling errors or format inconsistencies in the search request, the cloud server will return the closet possible ranked results based on pre-specified similarity semantics.

B. Threat Model

The cloud server is considered "honest but curious" and can not be fully trusted by users in our model, this is consistent with existing searchable encryption schemes[7], [14], [18], [20], [9]. Even though data files are encrypted, the cloud server may try to capture extra sensitive information from user's search request , encrypted files, and indices while performing keyword-based search over C. So the search should be performed in a secure manner that allows data files to be securely retrieved while revealing as little information as possible to the cloud. We will follow the security definition proposed in the existing searchable encryption [9].

C. Design Goals

In this paper, we should achieving the following security and performance guarantee:

i) Storage-saving ranked fuzzy keyword search, which means that the returning results are ranked according to some ranking criteria and the fuzzy keyword set consumes low storage costs.

ii) Muti-keyword search, which means our proposed scheme supports multiple keywords search.

iii) Privacy-preserving search, which means the cloud server is prevented from capturing extra useful information from the encrypted data files and the indices and the trapdoors;

iv) Access accuracy search, which means that when a user inputs a keyword and the exact match fails, and there are more than one fuzzy keyword set contain this keyword's fuzzy keyword set, the server is sure which data files should be returned.

v) Efficiency, which means above-mentioned goals should be achieved with low storage size, communication and computation overhead.

D. Preliminaries

Edit distance String similarity can be measured by several measures. In [14], [15], they use edit distance technique [12] to implement their scheme. 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) Insertion: inserting a single character into a word;

2) Deletion: deleting one character into a word;

3) Substitution: changing one character to another in a word.

Files The file set of size n is denoted as

$$F = \{F_1, F_2, ..., F_n\}$$

$ID = \{id_1, id_2, ..., id_n\}$

$Files = \{< id_1, F_1 >< id_2, F_2 >, ...,< id_n, F_n >\}$

F_i is the i-th file, id_i is the unique identifier of F_i.

Encrypted files Let sk be the data owner's secret key, sk can generate $K_F = \{k_1, k_2, ..., k_n\}$ which are used to encrypt data files .

Input: $Files$, sk

Key Generation: $k_i = f_1(sk||id_i), k_i \in K_f, 1 \leq i \leq n$

Output: $C = F_{Enc} = \{Enc_1(k_i, F_i)\}_{k_i \in K_F, F_i \in F}\}$

f_1 can be implemented by hash functions; and Enc_1 can be implemented by block cipher such as AES.

Fuzzy keyword search Existing schemes usually use edit distance technique to define the fuzzy keyword search: Given a set of n encrypted data files F, a set of distinct keywords $W = (w_1,w_2, ... , w_m)$ and edit distance d is pre-defined. Searching input is (Q, d'), Q is the target keyword for searching, d' is the threshold of fuzzy search based on edit distance, $d' <= d$. The fuzzy keyword search returns a set of file IDs whose corresponding data files probably contain the word Q. The corresponding file IDs that contain Q are denoted as id_{w_i}: if $Q = w_i \in W$, then return $id_{w_i} (id_{w_i} = id_Q)$; otherwise, return $\{id_{w_i}\}$,where $ed(w_i, Q) <= d'$. Note that the above definition is based on the assumption that $d' <= d$. In fact, d can be different for distinct keywords and the system will return $\{id_{w_i}\}$ satisfying $ed(Q, w_i) \leq min\,(d, d')$ if exact match fails.

Ranking function In information retrieval community, we generally use ranking function to rank files by computing score of file relavance to a given search request. $TF - IDF$ rule is widely used in statistical measurements for computing relevance score,where TF (term frequency) is simply the number of times a given term or keyword appears in a data file, and IDF (inverse document frequency) is achieved by dividing the number of files in the whole collection by the number of files containing the term. Among many $TF-IDF$ weighting techniques [22] analyzed, we choose the widely used relevance score defined in [19] as following:

$$Score\,(Q, F_i) = \sum_{t \in Q} \frac{1}{|F_i|} \cdot (1 + ln f_{i,t}) \cdot ln\left(1 + \frac{n}{N_t}\right) \quad (1)$$

Here Q denotes the searched keywords; $f_{d,t}$denotes the TF of term (keyword) t in file F_i; N_t denotes the number of data files that contain the term t; n denotes the total number of files in the collection, and $|F_i|$ is the number of indexed terms in file F_i, functioning as the normalization factor.

IV. Efficient ranked fuzzy keywords searchable symmetric encryption scheme

A. K-gram based fuzzy keyword set

K-gram K-gram refers to a substring which length is k. The substring meets "highly adjacent" feature. For example, "com", "omp", "mpu", "put", "ute", "ter" are all 3-grams of the word "computer", and each substring is called 3-gram. We can see that with regard to a string of length l , when we divide it into k-grams, we will get $l-k+1$ substrings, and each substring's length is k. In this paper, we use "#" to denote the beginning and the end of a word. Thus the set of 3-grams of the word "computer" is: "#co", "com", "omp", "mpu", "put", "ute", "ter", "er#". So with with regard to a string of length l, when we divided it into k-grams, we will get $l - k + 3$ substrings, and the total size of k-grams is $O(m * (l - k + 3))$ instead of $O(m * l^d)$ in DFSC approach mentioned in Section

4.1. In fact, there are many same k-grams in W, which may further reduce the index size.

K-gram based dictionary In this chapter, a k-gram based dictionary is a set of all the k-grams of distinct keywords $W = (w_1,w_2, ... , w_m)$. We define a dictionary of size N as $KD = \{G\{w_i\}\}$, where $G\{w_i\}$ denotes k-grams of keyword w_i, $1 \leq i \leq m$.

Jaccard coefficient There are many measures to quantify string similarity. In this paper, we choose Jaccard coefficient for our proposed scheme. The Jaccard coefficient is used to measure the similarity between finite sets, and is defined as the size of the intersection divided by the size of the union of the sets, i.e.

$$\lambda = J(A_w, B_K) = \frac{|A_w \cap B_K|}{|A_w \cup B_K|} \quad (3)$$

Here, sets A_{w_i} and B_K denote the set of k-grams for keyword $w_i (w_i \epsilon W)$ and K respectively. Here we specify that when both A and B are empty, $\lambda = 0$. If w_i is equal to K, w_i will have the highest Jaccard coefficient value ($\lambda = 1$) compared to the other keywords in the index.

K-gram based fuzzy keyword search In this paper, we adopt Jaccard coefficient to construct our fuzzy keyword set: Given a set of n data files F, a set of distinct keywords $W = (w_1,w_2, ... , w_m)$. We generate the dictionary $KD = \{G\{w_i\}\}$ for W , here $G\{w_i\}$ denote the k-grams of keyword $w_i (1 \leq i \leq m)$. For every gram $g_j \epsilon KD$ ($1 \leq j \leq |KD|$), we build a k-gram based index $I_{g_j} = g_j, \{w\}$, here $\{w\}$ denote a set of keywords which may contain the gram g_j. Thus, the whole k-gram based index can be expressed as $I = I_j (1 \leq j \leq |KD|)$.

We assume that the search keyword is Q, and λ_{min} is the threshold of fuzzy search based on Jaccard coefficient (λ_{min} can be determined in our experiments). First we generate the k-grams for Q, which are denoted as $G\{Q\}$. For every gram $g_i \epsilon G\{Q\} (1 \leq j \leq |G\{Q\}|)$, the server will match it in the k-gram index I introduced above and return relative keywords containong the k-gram g_i. To reduce our search space, we only want to search the keywords which is closely related with user's search request.

If the Jaccard coefficient λ_w of keyword w_i is bigger than our threshold value λ_{min} , i.e.

$\lambda_{w_i} = |A_{w_i} \cap B_Q| / |A_{w_i} \cup B_Q| \geq \lambda_{min}$

we add w_i to our fuzzy keyword set F_{fuzzy}.

B. K-gram based index

We assume that the user's search request contains mutiple keywords, denoted as $Q = \{Q_1, Q_2, ..., Q_t\}$. In our scheme, for each $Q_i \epsilon Q$, we computer its k-grams $G(Q_i)$, and then compute the fuzzy keywords set F_{Q_i} using the method mentioned in Section5.1. Thus the all fuzzy keyword set of Q is $F_{fuzzy} = \{F_{Q_i}\}$. And then cloud server can retrieve the inverted index to obtain corresponding data files and return them to users.

An example of k-gram based index is shown in Table 1.

According to the keyword w_i (such as *computer*), the posting list of w_i includes three enties: keyword, file ID and score, which is consistent with [18]. An example of the posting list is shown in Table 2.

k-gram	com				
keyword	computer	complete	complicated	...	come

Table I
AN EXAMPLE OF K-GRAM BASED INDEX ($k = 3$)

keyword	w_i					
file ID	id_{i_1}	id_{i_2}	id_{i_3}	...	$id_{i_{t-1}}$	id_{i_t}
score	2.34	1.46	14.36	...	4.77	5.45

Table II
AN EXAMPLE POSTING LIST OF PROPOSED K-GRAM BASED INDEX

Keyword denotes the whole keywords the in the k-gram based index. File ID denotes the file identifier containing the corresponding keyword.

C. The efficient ranked fuzzy keywords search scheme

Based on the storage saved fuzzy sets mentioned above and the secure OPM ranking funchtion in [18], our proposed ranked fuzzy keywords search scheme can be described as follows.

Initialization:

The data owner then scan F and extract distinct words $W = (w_1, w_2, ..., w_m)$ from F, for each $w_i \epsilon W$, buid $F(w_i)$, $F(w_i)$ is the set of file IDs which contains the word w_i;

In the Setup phase:

1) The data owner uses his/her secret key sk to generate $K_{KD} = \{k_1, k_2, ..., k_{|KD|}\}$ which are used to encrypt k-grams in dictionary KD, $k_i = f_1(sk||g_i), k_i \in K_{KD}, g_i \epsilon KD$. The data owner also calls $KeyGen\left(1^a, 1^b, 1^{b'}, 1^c, |D|, |R|\right)$ to generate random keys $x, y, z \leftarrow \{0,1\}^a$, and outputs $K = \left\{x, y, z, 1^b, 1^{b'}, 1^c, |D|, |R|\right\}$

2) The server then uses the index structure in Table 1 and Table 2 to build a k-gram index. The details are shown in Fig.4.

3) The data owner outsources the encrypted index table and encrypted data files C to the cloud server.

In the $Retrieval$ phase:

1) The authorized user inputs his/her search request Q. He/She first computes the fuzzy keyword set Q_{fuzzy} that satisfies with $\lambda_{w_i} = |A_{w_i} \cap B_Q| / |A_{w_i} \cup B_Q| \geq \lambda_{min}$, then computes the trapdoors $\{(\pi_x(Q_i), f_y(Q_i))\}_{Q_i \epsilon Q}$ and sends it to the cloud server.

2) Upon receiving the search request $\{(\pi_x(Q_i), f_y(Q_i))\}_{Q_i \epsilon Q}$, the cloud server compares them with the index table. He/She uses $\pi_x(Q_i)$ to locate the matching list of the index, and uses $f_y(Q_i)$ to decrypt the entries, and then he/she knows the file identifiers $< id(F_{ij}) >$ and their associated encrypted relevance scores $OPM_{f_z(w)}(S_{ij})$. The server fetches the files and sends back them in a ranked sequence according to the encrypted relevance scores $OPM_{f_z(w)}(S_{ij})$.

3) The user decrypts the returned results and retrieves their interested files.

V. PERFORMANCE ANALYSIS

We conducted a thorough experiment of our proposed scheme which is implemented by Java language. In our

Index (K, F)

1. Initialization:
   Scan F and extract the distinct words $W = (w_1, w_2, \ldots, w_m)$. For each word $w_i \in W$, build $F(w_i)$. $F(w_i)$ is the set of file identifiers in F that contains word w_i.
2. K-gram index
   a) For each $w_i \in W$, for $1 <= i <= m$, Build k-grams for w_i, denoted as $G(w_i)$, so we get the dictionary $KD = \{G(w_i)\}$
   b) For each$g_j \in KD$, $1 <= j <= |KD|$, build a k-gram index for g_j, denoted as $I_{g_j} = < g_j || F(g_j) >$, $F(g_j)$ denote a set keywords contain g_j.
3. Inverted index
   a) For each $w_i \in W$:
   b) For $1 <= j <= |f(w_i)|$, calculate the score for file F_j according to equation (1), denoted as $SC_{i,j}$. Compute $OPM_{f_z(w_i)}(SC_{i,j})$ according to Algorithm 1, and store it with F_j's identifier $<fid_j || OPM_{f_z(w_i)}(SC_{i,j})>$ in the posting list I_i';
   c) For eachI_i', where$1 \le i \le m$, encrypted all N_i entries with l' padding$0'$, $<0^{l'} || fid_j || OPM_{f_z(w_i)}(SC_{i,j})>$, with key $f_y(w_i)$, $1 <= j <= N_i$. Then set remaining $v - N_i$, if any, to random values of the same size as the existing N_i entries of I_i'.
   d) Replace w_i with $\pi_x(w_i)$
4. Output k-gram index $I=(I_1, I_2, \ldots, I_{|KD|}) = (g_j || F(g_j))$ and corresponding inverted index $I' = (I_1', I_2', \ldots, I_m') = (< 0^{l'} || fid_j || OPM_{f_z(w_i)}(SC_{i,j}) >)$

Figure 2. The details of $Index(\cdot)$ for proposed scheme

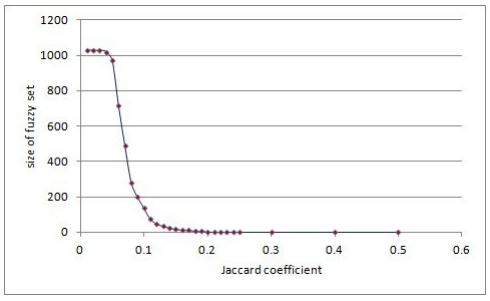

Figure 3. Relationship between Jaccard coefficient value and fuzzy set size

experiment, we select 10714 real data files from the website [10].

According to our k-gram based scheme, the threshold λ_{min} based on Jaccard coefficient controls the size of fuzzy keyword set. It indicates the lowest similarity between the search requests and the fuzzy keywords we generated. To find the best value of λ_{min} , we scan all the words in our downloaded files and use different Jaccard coefficient values from 0 to 0.5 to build fuzzy sets. As Fig.5 illustrates, when Jaccard coefficient value is greater than 0.21, the size of fuzzy set is almost 0, the fuzzy keyword search results will not be obvious. For example, when Jaccard coefficient value is 0.3, the fuzzy set of word "fuzzy" only contain the word "fuzzy". However, when the coefficient value is less than 0.15, the fuzzy set size will be greater than 20, it will waste storage and computation space, and some words in the set may have nothing to do with user's search request. According to Fig.5, that when the coefficient value is 0.18, when can get a reasonable fuzzy keyword size, which is consistent with [21].

1) Index construction: According to the complexity analysis

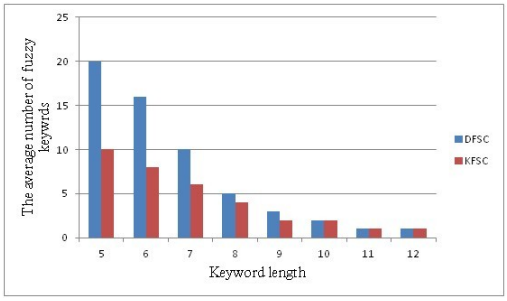

Figure 4. The comparison of fuzzy keyword number between KFSC and DFSC

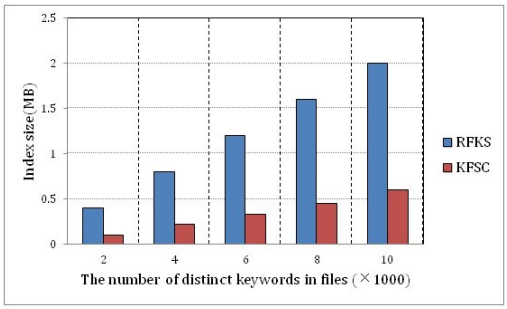

Figure 5. The comparison of index size between KFSC and DFSC

in Section 5.1, we know that with regard to a string of length l, when we divided it into k-grams, we will get $l - k + 3$ substrings, and the total size of k-grams is $O(m * (l - k + 3))$ instead of $O(m * l^d)$ in DFSC approach mentioned in Section 4.1. In fact, there are many same k-grams in W, which may further reduce the index size.

To allow efficient secure ranked fuzzy keyword search over encrypted cloud data, we adopt the k-gram based index illustrated in Table 1 and Table 2, when k is the same, the index size of our proposed scheme is smaller than DFSC scheme in [15]. We randomly select 10 words of length $5, 6, 7, 8, 9, 10, 11, 12$ respectively to consturct the fuzzy set, Fig.6 shows that the fuzzy keyword number of our k-gram based fuzzy set construction (KFSC) is relatively small. We randomly select different number of files and observe the sizes of indices when the keywords number are$1000, 2000, 4000, 6000$, respectively. We can see that our proposed index size is smaller than our early RFKS scheme in [20]. Here we assign $d = 2, k = 3, \lambda_{min} = 0.18$

2) Search: The search time includes trapdoor buiding time, posting list fetching time in the index, decrypting time for each entry. We can also consider the top-k retrieval. The cloud server can get the top-k retrieval as fast as the plain text search because of the order-preserved encrypted scores and our small index size. And we can also make some minor spelling errors and format inconsistencies when we input our search keywords, which may obviously improve the system usability and efficiency. Besides, the problem of result inaccuracy mentioned in Section 1 will not appear in this paper because of the k-gram based fuzzy set.

Besides, as [18] proved, the One-to-many Order-Preserving Mapping scheme is safe, so we can ensure that out proposed

scheme is also secure because our ranking scheme is consistent with [18]. We will no longer prove it in this paper.

VI. Conclusions

In this paper, we introduced a complete framework of an efficient and privacy preserving ranked fuzzy keyword search over encrypted cloud data. For the efficiency and security consideration, we adopted the k-gram and Jaccard coefficient to accomplish fuzzy keyword search and One-to-many Order-Preserving Mapping scheme to build the inverted index. Through thorough performance and security analysis, we showed that our proposed scheme is privacy preserving and efficient.

As our future work, we will study the problem of improving the efficiency of index building and searching. And we will also explore privacy preserving schemes under stronger threat models.

Acknowledgement

This work is supported by National Science Foundation of China Grant #61672088 and #61300175, Fundamental Research Funds for the Central Universities #2016JBM022. The corresponding author is Yidong Li.

References

[1] Feng Bao, Robert Deng, Xuhua Ding, and Yanjiang Yang. Private query on encrypted data in multi-user settings. In *Information Security Practice and Experience*, volume 4991, pages 71–85. Springer Berlin / Heidelberg, 2008.

[2] Mihir Bellare, Alexandra Boldyreva, and Adam O¡¯Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology - CRYPTO 2007*, volume 4622, pages 535–552. Springer Berlin / Heidelberg, 2007.

[3] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O¡¯Neill. Order-preserving symmetric encryption. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479, pages 224–241. Springer Berlin / Heidelberg, 2009.

[4] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027, pages 506–522. Springer Berlin / Heidelberg, 2004.

[5] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography*, volume 4392, pages 535–554. Springer Berlin / Heidelberg, 2007.

[6] Jin Wook Byun, Dong Hoon Lee, and Jongin Lim. Efficient conjunctive keyword search on encrypted data storage system. In *Public Key Infrastructure*, pages 184–196. Springer, 2006.

[7] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. In *INFOCOM, 2011 Proceedings IEEE*, pages 829 –837, april 2011.

[8] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security*, volume 3531, pages 391–421. Springer Berlin / Heidelberg, 2005.

[9] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 79–88, 2006.

[10] Request For Comments Database. http://www.ietf.org/rfc.html.

[11] Philippe Golle, Jessica Staddon, and Brent Waters. Secure conjunctive keyword search over encrypted data. In *Applied Cryptography and Network Security*, pages 31–45. Springer, 2004.

[12] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

[13] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology šC EUROCRYPT 2010*, volume 6110, pages 62–91. Springer Berlin / Heidelberg, 2010.

[14] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5, march 2010.

[15] Chang Liu, Liehuang Zhu, Longyijia Li, and Yuran Tan. Fuzzy keyword search on encrypted cloud storage data with small index. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 269 –273, sept. 2011.

[16] Qin Liu, Guojun Wang, and Jie Wu. An efficient privacy preserving keyword search scheme in cloud computing. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 2, pages 715–720. IEEE, 2009.

[17] Amit Singhal. Modern information retrieval: a brief overview. *BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING*, 24:2001, 2001.

[18] Cong Wang, Ning Cao, Jin Li, Kui Ren, and Wenjing Lou. Secure ranked keyword search over encrypted cloud data. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 253 –262, june 2010.

[19] I. H. Witten, A. Moffat, and T.C.Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. John Wiley & Sons, Inc., 1999.

[20] Qunqun Xu, Hong Shen, Yingpeng Sang, and Hui Tian. Privacy-preserving ranked fuzzy keyword search over encrypted cloud data. december 2013.

[21] Wei Zhou, Lixi Liu, He Jing, Chi Zhang, Shaowen Yao, and Shipu Wang. K-gram based fuzzy keyword search over encrypted cloud computing. *Journal of Software Engineering & Applications*, 6(1), 2013.

[22] Justin Zobel and Alistair Moffat. Exploring the similarity space. In *ACM SIGIR Forum*, volume 32, pages 18–34. ACM, 1998.