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

Xianglai Yang
State Grid of China Technology College
Jinan, China

Jianmei Cao
State Grid of China Technology College
Jinan, China

Youjie Wang
State Grid of China Technology College
Jinan, 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 have 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 only support Boolean search, without capturing any relevance of data files and rarely sort the search result. Recently, fuzzy keyword search over encrypted data techniques are introduced to resolve the problem of spelling errors and format inconsistencies. But these methods may incur large index size, search result inaccuracy and high search complexity, which greatly reduce the system usability and efficiency. This paper proposes the solution for privacy preserving ranked fuzzy keyword search over encrypted cloud data with small index. K-grams and Jaccard coefficient are utilized to construct fuzzy keyword set and produce fuzzy results, and an efficient relevance criteria is also provided to capture the relevance between data files and search requests. Extensive experimental results show the efficiency of our proposed method.**

Keywords—K-Gram, Fuzzy Keyword Search, Ranked Keyword Search, Searchable Encryption, Cloud Computing

I. INTRODUCTION

Recently, with the rapid development of cloud computing, more and more individuals and enterprises (collectively called data users) prefer to outsource their local data into the cloud server in order to save storage costs and enjoy the on-demand high quality applications. 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 only intend to search certain data files in a given occasion.

Although traditional searchable encryption techniques [1, 2, 3, 4] could achieve the security and efficiency by creating an index for each keyword and associate the keyword with the data files, 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[5], which greatly reduce the system usability. It is quite normal that users may input keywords that are not precisely match the pre-set keywords. Second, these schemes only support Boolean search, without capturing any relevance between data files and search request, and 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. Although the state-of-the-art information retrieval techniques have already utilized a variety of scoring mechanisms [6] to rank the relevance of data files, 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.

This paper proposes the solution for privacy preserving ranked fuzzy keyword search over encrypted cloud data. K-gram is used to construct fuzzy keyword set and Jaccard coefficient to quantify keyword similarity[11], avoiding enumerating all fuzzy keywords and reducing the index space and search space. We eliminate keywords with Jaccard coefficient smaller than our threshold value. After eliminating, it may greatly reduce the index size, storage and communication costs. In addition, an efficient relevance criteria (e.g., $TF \times IDF$) is also used to capture the relevance between data files and search requests, which named ad the criteria relevance score. For security consideration, One-to-many Order Preserving Mapping (OPM) algorithm is utilized to encrypt the score and Order-Preserving Symmetric Encryption (OPSE) scheme in [12, 8].

The rest of paper is organized as follows. Section II summarizes the features of related work. Section III introduces the system model, threat model, and our design. Section IV provides the construction and details of our proposed scheme. Section V presents the security and efficiency analysis. Finally, section VI concludes the paper.

II. RELATED WORKS

A. Searchable encryption

Existing searchable encryption schemes generally create an encrypted searchable index for each keyword and associate the keyword with the data files which contain the keyword. Boneh et al. [2] proposed a public key encryption (PKE) scheme, in which each file is encrypted using public key by data owners and 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. Paper [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 supported ranked or fuzzy keyword search.

B. Fuzzy keyword searchable encryption

Li et al. [5] 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 [7], in which each keyword is corresponding with the fuzzy keywords.

C. Ranked keyword searchable encryption

Ranked keyword search captured the relevance between data files and search request and ranks the search results. Boldyreva et al. [12] proposed a order preserving symmetric encryption (OPSE) scheme, it supports deterministic property, in which a random coin generator and sampling function was implemented. Wang et al. [8] introduced a more secure ranked keyword search scheme over encrypted cloud data, it used Order Preserving Mapping (OPM) algorithm to encrypt the score. Cao et al. [9] proposed several improvements such as multi-keyword search feature.

D. Multi-keywords searchable encryption

Many literatures tried to improve the efficiency and security of a single keyword search scheme. Golle et al. [14] first proposed the conjunctive keyword search technique, and Byun et al. [15] introduced a more efficient conjunctive keyword search scheme. Cao et al. [9] presented a muti-keyword search scheme over encrypted cloud data and established various privacy requirements. Liu et al. [16] allowed the cloud server to participate in the partial decryption of the data files.

III. PROBLEM FORMULATIONS

A. Abbreviations and Acronyms

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

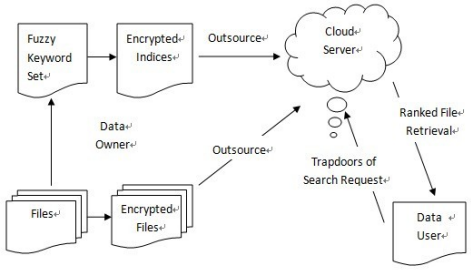

Fig. 1. Frame of keyword search over encrypted cloud data

In our system model, data owner has a collection of n files $F = (F_1, F_2, F_3, ..., F_n)$ and intends to outsource them to the cloud server in encrypted form C, and 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, data owner will first build a secure searchable index I for each keyword in W before outsourcing. Then outsourcing all the encrypted indices I_{Enc} and data files C to the cloud server.

In this paper we assume that the authorization between data owner and data users have 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. 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 could not be fully trusted by users in our model, this is consistent with existing searchable encryption schemes[5, 8, 10]. 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 and revealing as little information as possible. We will follow the security definition proposed in the existing searchable encryption.

C. Design Goals

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

1) 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.

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

3) 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.

4) 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, the server is sure which data files should be return.

5) 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 [5, 7], they use edit distance technique 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 presented.

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 files 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 relevance 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, we choose the widely used relevance score defined as Equation (1).

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

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.

IV. Efficient ranked fuzzy keywords searchable symmetric encryption scheme

Before introducing our proposed scheme, first of all we separately discuss the previous fuzzy keyword search scheme, ranked keyword search scheme and our previous work on ranked fuzzy keyword search.

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 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 $O(m * (l - k + 3))$. 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 among finite sets, which is defined as the size of the intersection divided by the size of the union of the sets, i.e. Equation (2).

$$\lambda = J(A_w, B_K) = \frac{|A_w \cap B_K|}{|A_w \cup B_K|} \quad (2)$$

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, Jaccard coefficient is adopted 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 is 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 containing 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} , we add w_i to our fuzzy keyword set F_{fuzzy}.

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

B. K-gram based index

We assume that the user's search request contains multiple keywords, denoted as $Q = \{Q_1, Q_2, ..., Q_t\}$. In our scheme, for each $Q_i \epsilon Q$, we first compute its k-grams $G(Q_i)$, and then compute the fuzzy keywords set F_{Q_i} using the method mentioned in Section IV.A Thus all fuzzy keyword sets of Q are $F_{fuzzy} = \{F_{Q_i}\}$. So the 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 I.

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

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

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 [8]. An example of the posting list is shown in Table II.

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

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

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 function in [8], 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 $w_i \epsilon W$, build $F(w_i)$, $F(w_i)$ 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 I and Table II to build a k-gram index. The details are shown in Fig. 2.

Index (K, F)

1. Initialization:
   Scan F and extract the distinct words $W = (w_1, w_2, ..., w_m)$. 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_{ij}. Compute $OPM_{f_z(w_i)}(SC_{ij})$ according to Algorithm 1, and store it with F_j's identifier $<fid_j||OPM_{f_z(w_i)}(SC_{ij})>$ in the posting list I_i';
   c) For eachI_i', where$1 \leq i \leq m$, encrypted all N_i entries with l' padding$0'$, $<0^{l'}||fid_j||OPM_{f_z(w_i)}(SC_{ij})>$, 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 $I=(I_1, I_2, ..., I_{|KD|}) = (g_j || F(g_j))$ and corresponding inverted index $I' = (I_1', I_2', ..., I_m') = (< 0^{l'}||fid_j||OPM_{f_z(w_i)}(SC_{ij}) >)$

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

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. 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 $OPM_{f_z(w)}(S_{ij})$.

V. PERFORMANCE ANALYSIS

We conducted a thorough experiment of our proposed scheme which is implemented by Java language. In our experiment, 10714 real data files are selected from the

website [17].

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 shown in Fig.3, 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, and 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.3, we found that when the coefficient value is 0.18, we can get a reasonable fuzzy keyword size, which is consistent with [11].

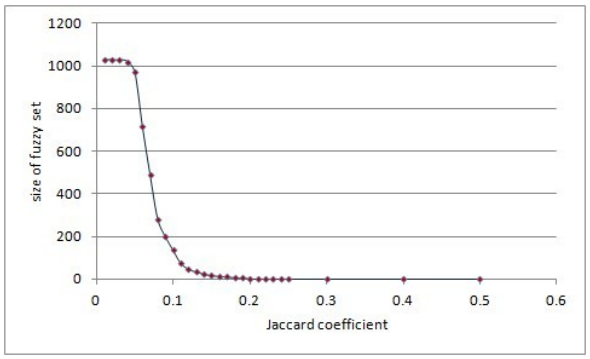

Fig. 3. Relationship between Jaccard coefficient value and fuzzy set size

Index construction: According to the complexity analysis Section IV.A 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 $O(m*(l-k+3))$. 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 I and Table II, when k is the same, the index size of our proposed scheme is smaller than DFSC scheme in [7]. We randomly select 10 words of length $5,6,7,8,9,10,11,12$ respectively to construct the fuzzy set, Fig.4 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 early RFKS scheme in [10]. Here we assign $d=2,k=3,\lambda_{min}=0.18$.

Search: The search time includes trapdoor building 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 I will not appear in this paper because of the k-gram based fuzzy set.

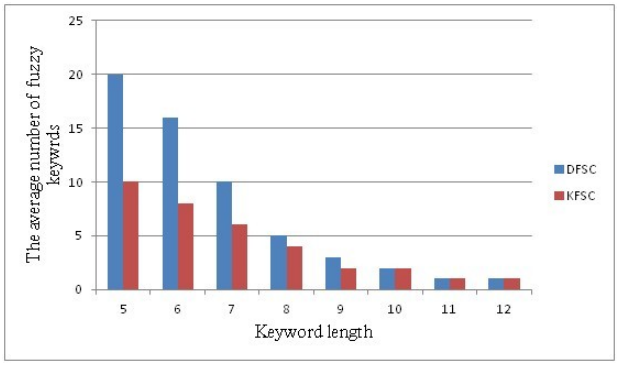

Fig. 4. The comparison of fuzzy keyword number between KFSC and DFSC

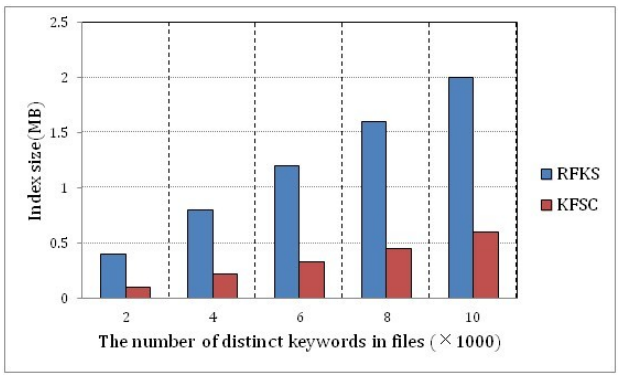

Fig. 5. The comparison of index size between KFSC and DFSC

Besides, as [8] 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 [8]. We will no longer prove it in this paper.

VI. Conlucsions

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

The future work 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.

References

[1] Bellare, M., Boldyreva, A., ONeill, A.: Deterministic and efficiently searchable encryption. In: Advances in Cryptology - CRYPTO 2007, vol. 4622, pp. 535-552. Springer Berlin / Heidelberg (2007).

[2] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology - EUROCRYPT 2004, vol. 3027, pp. 506-522. Springer Berlin / Heidelberg (2004).

[3] Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Theory of Cryptography, vol. 4392, pp. 535-554. Springer Berlin / Heidelberg (2007)

[4] Chang, Y.C., Mitzenmacher, M.: Privacy preserving keyword

searches on remote encrypted data. In: Applied Cryptography and Network Security, vol. 3531, pp. 391-421. Springer Berlin / Heidelberg (2005).

[5] Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM, 2010 Proceedings IEEE. pp. 1-5 (march 2010).

[6] Singhal, A.: Modern information retrieval: a brief overview. BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING 24, 2001 (2001).

[7] Liu, C.; Zhu, L.; Li, L. & Tan, Y. Fuzzy keyword search on encrypted cloud storage data with small index Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, 2011, 269 -273.

[8] Wang, C.; Cao, N.; Li, J.; Ren, K. & Lou, W. Secure Ranked Keyword Search over Encrypted Cloud Data Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, 2010, 253 -262.

[9] Cao, N.; Wang, C.; Li, M.; Ren, K. & Lou, W. Privacy-preserving multi-keyword ranked search over encrypted cloud data INFOCOM, 2011 Proceedings IEEE, 2011, 829 -837.

[10] Xu, Q.; Shen, H.; Sang, Y. & Tian, H. Privacy-Preserving Ranked Fuzzy Keyword Search over Encrypted Cloud Data PDCAT, 2013 Proceedings IEEE, 2013.

[11] Zhou, W.; Liu, L.; Jing, H.; Zhang, C.; Yao, S. & Wang, S. K-Gram Based Fuzzy Keyword Search over Encrypted Cloud Computing. Journal of Software Engineering & Applications, 2013, 6.

[12] Boldyreva, A.; Chenette, N.; Lee, Y. & O'Neill, A. Order-Preserving Symmetric Encryption Advances in Cryptology - EUROCRYPT 2009, Springer Berlin / Heidelberg, 2009, 5479, 224-241.

[13] Lewko, A.; Okamoto, T.; Sahai, A.; Takashima, K. & Waters, B. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption Advances in Cryptology – EUROCRYPT 2010, Springer Berlin / Heidelberg, 2010, 6110, 62-91.

[14] Golle, P.; Staddon, J. & Waters, B. Secure conjunctive keyword search over encrypted data Applied Cryptography and Network Security, 2004, 31-45.

[15] Byun, J. W.; Lee, D. H. & Lim, J. Efficient conjunctive keyword search on encrypted data storage system Public Key Infrastructure, Springer, 2006, 184-196.

[16] Liu, Q.; Wang, G. & Wu, J. An efficient privacy preserving keyword search scheme in cloud computing Computational Science and Engineering, 2009. CSE'09. International Conference on, 2009, 2, 715-720.

[17] http://www.ietf.org/rfc.html.