ICIC Express Letters
Part B: Applications
Volume **7**, Number **7**, July **2016**

ICIC International ©2016 ISSN 2185-2766
pp. 1629–1636

ENABLING SECURE AND EFFICIENT MULTI-KEYWORD FUZZY RETRIEVAL OVER ENCRYPTED DATA IN CLOUD

Yanguang Shen[1], Meng Zhang[1] and Chenghua Shi[2,*]

[1]School of Information and Electrical Engineering
[2]School of Economics and Management
Hebei University of Engineering
No. 199, Guangming South Street, Handan 056038, P. R. China
shenyanguang@aliyun.com; 3981157573@qq.com
*Corresponding author: chenghuashi@hebeu.edu.cn

Received December 2015; accepted March 2016

Abstract. *In cloud computing, for the privacy-preserving of cloud data, sensitive data often has to be encrypted before outsourcing, which makes data utilization very complex work. Though many solutions have been proposed to support the keyword retrieval, they cannot solve inadequacies of fuzzy retrieval or trapdoor linkability effectively. In this paper, we proposed a secure and efficient Multi-Keyword Fuzzy Retrieval (MKFR) over encrypted cloud data. Firstly, to build plaintext indexes, we select Locality-Sensitive Hashing (LSH) functions and the Bloom filter to hash binary coding combination of each keyword. And then MKFR adopts the public key to encrypt the index which is extended by constant and normal random number. With the similar process, we generate the trapdoor and the retrieval result ranking is measured by the inner product computing. The experiments implemented on the Enron Email Dataset further show efficiency and validity of the MKFR scheme.*

Keywords: Cloud computing, Privacy-preserving, Multi-keyword fuzzy retrieval, Binary coding combination, Locality-Sensitive Hashing (LSH), Bloom filter

1. **Introduction.** With the prevalence of cloud computer technology, data owners could remotely store and fetch data through cloud service so as to enjoy the on-demand high quality applications and services [1]. In cloud environment, customers authorized by cloud server could search required data and download them. However, the cloud server is usually considered as "honest-but-curious", which could not acquire the trust from data owners [2]. To protect the sensitive cloud data, it follows that sensitive data should be encrypted before outsourcing them to the cloud. Though data encryption ensures the confidentiality of cloud data, it makes some effective retrieval schemes (e.g., Equivalent Search and Beam Search) be complicated work under a large number of outsourced data [3].

In the literature, some effective searchable encryption schemes were proposed to solve the above problems. Li et al. exploited edit distance to quantify keywords similarity and improved constructing fuzzy keyword sets [4]. Li et al. took user access and different weights into account when generating the results [5]. Sun et al. proposed a tree-based index based on the vector space model and term frequency [6]. Cheng et al. designed a hierarchical index based on Bloom filters and improved searchable public key encryption [7]. Cao et al. [8] adopted "the inner product similarity" to calculate the similarity and then based on the security K-Nearest Neighbor [9] (KNN) technology to make correlation ranking. However, these schemes cannot effectively solve the multi-keyword fuzzy retrieval and keyword spelling errors in edit distance. Moreover, the problems of trapdoor linkability and index similarity may happen to searching process.

In this paper, we concentrate on an efficient yet secure multi-keyword fuzzy retrieval over encrypted cloud data. To our knowledge, fuzzy retrieval largely enhances system availability when uses' request retrieval is ambiguous. More specifically, we use LSH to replace the standard hash functions of Bloom filter. Based on the fuzzy keyword sets, feature vectors of keywords are used to denote as input parameters. Through rigorous analysis and simulation experiment, we show the MKFR scheme is secure and efficient.

2. MKFR Scheme for Encrypted Cloud Data.

2.1. Basic theories.

1) **Bloom filter**

A Bloom filter represents a set $S = \{a_1, a_2, \ldots, a_n\}$ containing n elements by an array of m bits [10], and initially all bits are set to 0. The Bloom filter selects l independent hash functions from the hash family $H = \{h_i | h_i : S \to [1, m], 1 \le i \le l\}$. Each item a_j is mapped to the $h_i(a_j)$-th position by hashing functions $\{h_1, h_2, \ldots, h_l\}$ and the corresponding slot is set to 1. Obviously, the Bloom filter possibly introduces the false positive, indicating that an item a_j belongs to the set S although it in fact is not. The formula for false positive is shown by

$$f = (1-p)^l \approx \left(1 - \left(1 - \tfrac{1}{m}\right)^{nl}\right)^l \approx \left(1 - e^{\frac{nl}{m}}\right)^l \tag{1}$$

2) **Locality-Sensitive Hashing (LSH) scheme based on *p*-stable distributions**

The LSH based on p-stable distribution is an improved method which can only be used for Euclidean norm. It is known stable distributions exist for any $p \in (0, 2]$. The hash function of LSH can be defined as:

$$h_{a,b}(v) = \left\lfloor \frac{a \bullet v + b}{r} \right\rfloor \tag{2}$$

where a is a d-dimensional random vector with chosen entries following an s-stable distribution and r is a large constant where b is a real number selected from $[0, r)$ [11].

2.2. Fuzzy keyword matching scheme based on improved Bloom filter.

1) **Binary encoding of the bigram vector**

An extracted keyword is transformed into a bigram vector set, which contains all the contiguous 2 letters showed in the keyword. For instance, the set of keyword "cloud" is {cl, lo, ou, ud}, which ignores the case. We initialize a 26^2-bit length array to store the bigram vector set and each bit corresponds to a 10-bit binary encoding. Finally, we concatenate all the encodings whose position is set to 1. The process is shown as Figure 1.

2) **Index construction**

We replace the standard functions of the Bloom filter with independent LSH functions, which can hash the similar string to the same bucket with a high probability for similarity retrieval [11]. The process is shown in Figure 2.

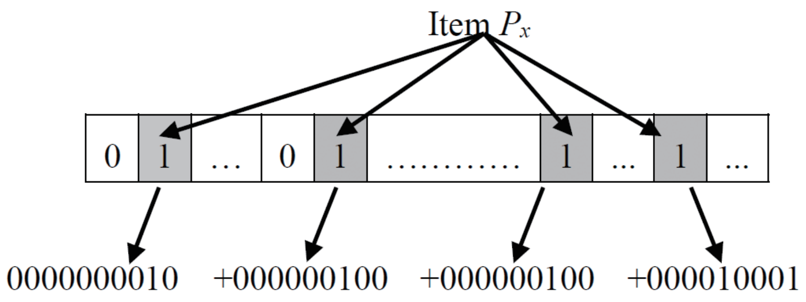


FIGURE 1. The binary encoding of bigram vector set

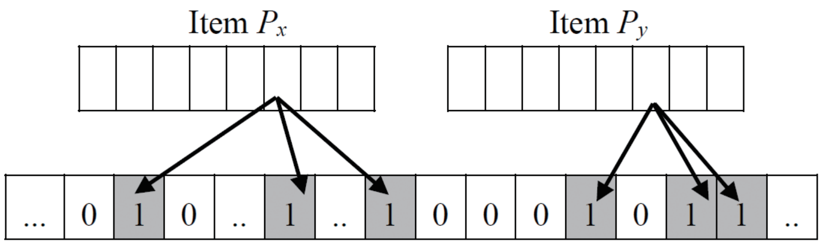


FIGURE 2. The hashing process of building index

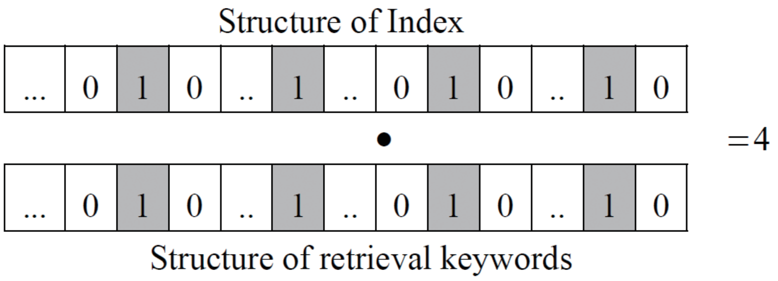


FIGURE 3. The inner product scores

3) **Inner product**

The client should adopt the same LSH functions to hash retrieval keywords when the users submit request keywords. As shown in Figure 3, if any document contains the keyword(s) during the retrieval process, the corresponding bits in both vectors will be 1; hence the inner product could get a higher value. The result of the inner product is a good method of the number of matching keywords.

2.3. **Encryption scheme for MKFR.** We follow the analogous schemes of previously proposed symmetric key to encrypt the outsourced indexes and files. Based on the similar scheme [8,9], we add two random numbers to extend the index and the one follows normal distribution. Our searchable encryption scheme includes four steps (KeyGen, BulidIndex, Trapdoor, Search). The specific implementation process is as follows.

• **KeyGen($\boldsymbol{P}$):** Taking a random parameter P to algorithm as input, it generates $S \in \{0,1\}^{m+2}$ and invertible matrix M_1, $M_2 \in R^{(m+2)\times(m+2)}$. The final public key consists of three parts denoted as $SK\{S, M_1, M_2\}$.

• **Bulid_EncIndex($\boldsymbol{SK}$, $\boldsymbol{I}$):** The parameter I is the Bloom filter of m bit length. Based on index I, we extend the $(m+1)$-th bit of the index with a random number ε following the normal distribution $N(\mu, \sigma^2)$ and the $(m+2)$-th bit was set as a constant 1. The extended index structure is denoted as $(I, \varepsilon, 1)$. And then the extended index is split into two random sub-indexes (I', I'') by the splitting indicator $S \in \{0,1\}^{m+2}$ from SK. Namely, if the i-th bit of S is $s_i = 0$, the sub-indexes equation $I'_i = I''_i = I_i$ is true, while the I'_i and I''_i depend on random number λ so that their sum equals I_i. It can be denoted as $I'_i = 0.5 \times I_i + \lambda$ and $I''_i = 0.5 \times I_i - \lambda$. Finally, the invertible matrix M_1, M_2 would be used to encrypt revised index denoted as $Index\_Enc(I) = \{M_1^T I', M_2^T I''\}$.

• **Trapdoor($\boldsymbol{SK}$, $\boldsymbol{Q}$):** Q represents query-keywords which are hashed into the Bloom filter denoted as m bit length. Firstly, Q is extended to $(m+1)$ which is set to 1 and then select two random numbers r and t. The extended Q is scaled by r and the $(m+2)$-th bit is set to t. Therefore, The final extended Q equals (rQ, r, t). After the same segmentation and encryption as extended index in BulidIndex, the trapdoor is denoted as $Query\_Enc(Q) = \{M_1^{-1} I', M_2^{-1} I''\}$.

• **Search($\boldsymbol{Index\_Enc(I)}$, $\boldsymbol{Query\_Enc(Q)}$):** With the Trapdoor $Query\_Enc(Q)$, the cloud server would compute the inner product similarity scores. In this part, we assume the parameter $r > 0$. After sorting all the scores, the cloud server returns files of the highest score.

The process of *Simil_Score* computer is as follows.

$$\begin{aligned}
&Index\_Enc(I) \bullet Query\_Enc(Q) \\
&= \{M_1^T I', M_2^T I''\} \bullet \{M_1^{-1}Q', M_2^{-1}Q''\} \\
&= M_1^T I' \bullet M_1^{-1}Q' + M_2^T I'' \bullet M_2^{-1}Q'' \\
&= M_1^T I' \bullet ((Q')^T(M_1^{-1})^T)^T + M_2^T I'' \bullet ((Q'')^T(M_2^{-1})^T)^T \\
&= (I')^T \bullet Q' + (I'')^T \bullet Q'' \\
&= (I, \varepsilon, 1) \bullet (rQ, r, t) \\
&= r(I \bullet Q + \varepsilon) + t
\end{aligned}$$

Note that in the secure KNN scheme [8], the formula rIQ for two approximate query keywords would produce scores with proportional relationship. Based on background knowledge the attackers may infer some encryption keywords and then speculate the encryption cloud data through the word frequency or other information. However, the formula $r(I \bullet Q + \varepsilon) + t$ of MKFR scheme eliminates the proportional relationship between the query scores in approximate query.

3. Security Analysis for MKFR Scheme.

1) Analysis of encrypted index structure

In the MKFR scheme, we replace the standard hash functions of the Bloom filter with LSH functions, which can hide some keywords information and defend the typical attack such as keywords statistical attack. Especially when the LSH functions have extraordinary random, different files with the analogous keywords can get larger different indexes. In addition, we select two random numbers ε and t to weaken the correlation between similar encrypt indexes.

2) Analysis of retrieval process

In this section, we introduce the secure KNN algorithm and scale attack to verify MKFR. The scale attack is that CSP could analyze proportional relationship or correlation between the scores calculated by similar trapdoors, and then speculates some specific information of keywords such as document frequency via retrieval scale. Given two related trapdoor for $Q_1\{w_1, w_2, w_3\}$ and $Q_2\{w_1, w_2\}$, the scr_i and scr'_i represent the scores of inner product. According to the analysis in literature [8], we can get that the scores meet Equation (3) as follows:

$$\frac{scr_1 - scr_2}{scr'_1 - scr'_2} = \frac{scr_2 - scr_3}{scr'_2 - scr'_3} = \frac{scr_3 - scr_1}{scr'_3 - scr'_1} \tag{3}$$

And then the CSP based on the known background model and Equation (3) could speculate privacy information. Because in the formula $r(I \bullet Q + \varepsilon) + t$ of secure KNN algorithm, $I \bullet Q$ contains the matched information for keywords. However, in the MKFR, the keywords firstly are changed into binary coding and then are hashed into the Bloom filter. It hides the privacy information of matched keywords. Thus this scheme can resist the scale analysis attack.

4. Performance Analysis. The whole experiment system is implemented by Java and NLPIR toolkit on the Windows7 32-bit operating system. We demonstrate the performance of proposed scheme on a real-world dataset: the beck-s folder of the Enron Email Dataset. The performance of our scheme is compared with two proposed MRSE [8]. Moreover, the experiments just focus on preset keywords and others would be ignored. We adopt the precision rate and recall rate [12] as criteria, and make comparison to precision rate of the fuzzy retrieval and precise retrieval to verify the validity.

4.1. **False positive rate and precision.**

1) **False positive rate**

In this scheme, false positive rate is affected by the bit length m of the Bloom filter and the number of LSH functions l. As shown in Figure 4, given the same scale of dataset where $M = 10^4$, the false positive rate could be kept about 0% if we adjust appropriately the parameters m and l. In next experiments, we should keep the rate less than 1.5%.

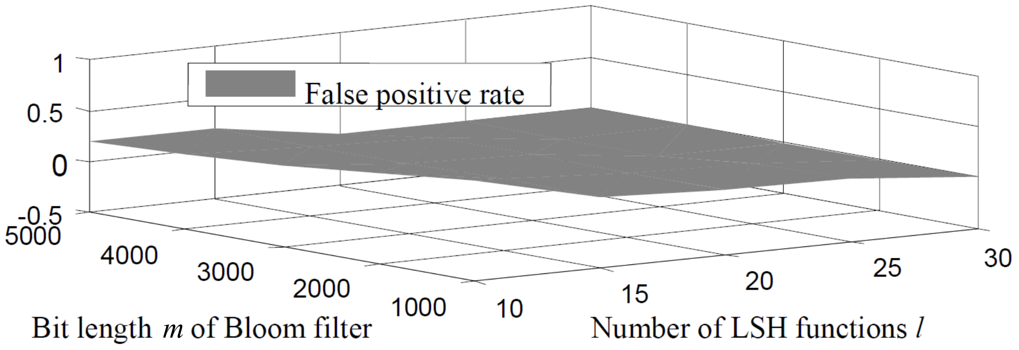
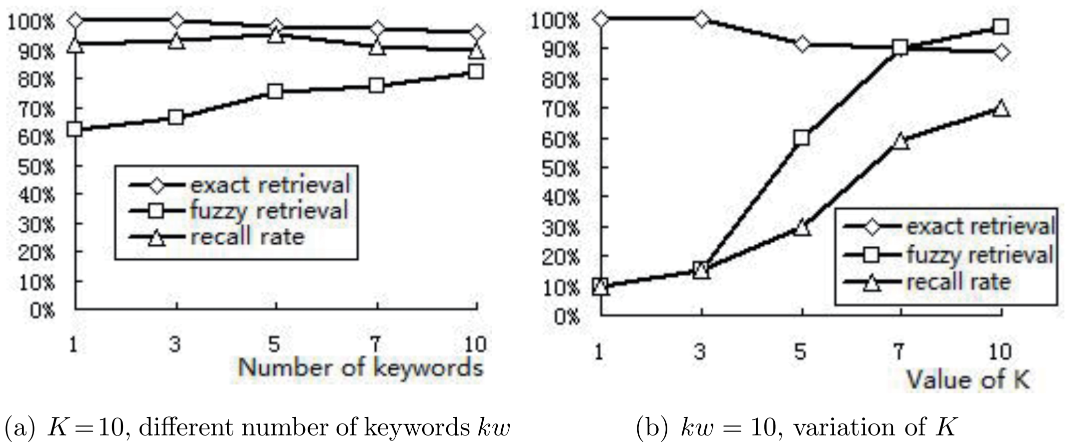


Figure 4. False positive rate (The m and l are variable, the scale of dataset $M = 10^4$)

2) **Precision rate and recall rate**

To evaluate the influence on accuracy of retrieval by the number of keywords and parameter K in LSH, we adopt the precision rate and recall rate as the measure norm. Moreover, we choose randomly two keywords and modify them to meet the requirements of fuzzy retrieval. Figure 5(a) shows that, the precision rate curve of exact retrieval drops but the fuzzy retrieval gradually rises with the number of keywords. Since the false positive rate accumulates in process of hashing, the impact of negative positive rate caused by LSH functions is more than false positive rate.

As shown in Figure 5(b), the recall rate and precision rate of fuzzy retrieval increas, but exact retrieval declines with parameter K. It is worth noticing that the parameter K determines the effect of LSH functions. If the value of K is relatively small, it is hard to distinguish different keywords through the K hash functions. However, when the K is too large, it would affect the efficiency of time and space. Thus, the K should be adjusted by the characteristic of dataset.



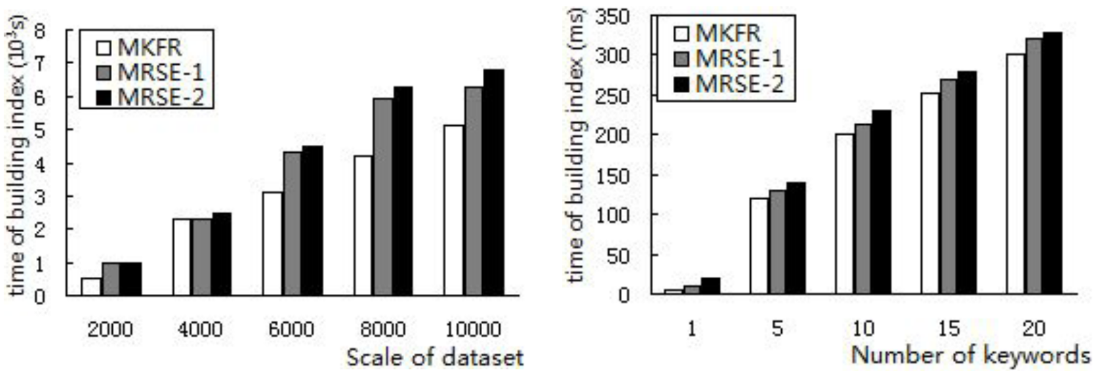
(a) $K = 10$, different number of keywords kw (b) $kw = 10$, variation of K

Figure 5. Precision rate and recall rate

4.2. **Efficiency.**

1) **Time cost of the index construction and trapdoor generation**

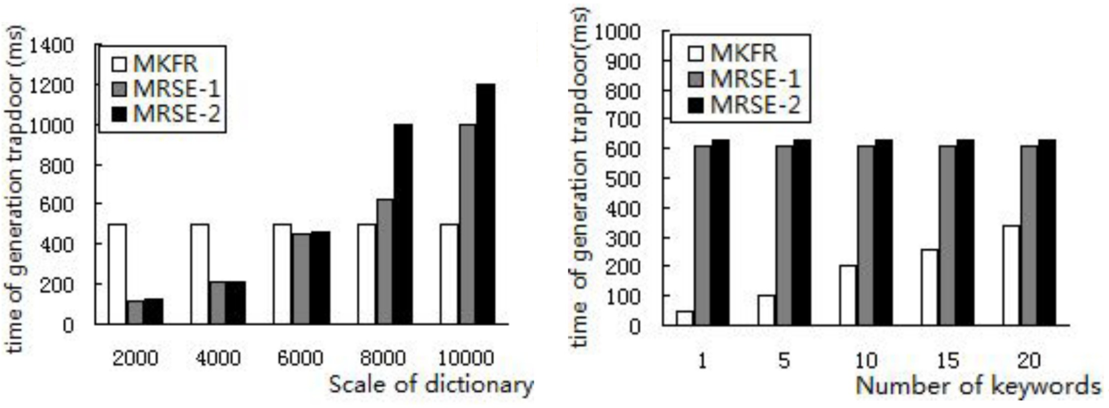
In the experiments for index construction, we choose the scale of dataset M and number of keywords kw as variable parameters. Given the same keywords dictionary $N = 4000$, simulation results are shown in Figure 6(a) and Figure 6(b). According to the strategic analysis of MKFR, the process of encryption index generating is a one-time calculation, which mainly contains two steps: building plaintext indexes and encryption. We use LSH functions to build plaintext indexes and the time complexity of algorithm is $O(n^2)$. Though the process of encryption is similar, the MRSE-1 and MRSE-2 enhanced matrix complexity to protect privacy. It may cost much more time than MKFR.



(a) $kw = 10$, different scale of dataset M (b) $M = 1000$, different number of keywords kw

FIGURE 6. Time cost of building index

As for trapdoor, we choose the keywords dictionary N and number of keywords kw as variable parameters. Given the same scale of dataset M, the simulation results are shown in Figure 7(a) and Figure 7(b). The trapdoor generation mechanism is similar to building index. Thus, the figure of MKFR is stable in Figure 7(a), but gradually it increases by kw in Figure 7(b). In the real retrieval process, the number of search keywords is at most twenty, which can fully express the user's intention. Although, the MRSE-1 and MRSE-2 are relatively stable in Figure 7(b), they may cost much more time to complete encryption for resisting trapdoor unlinkability.

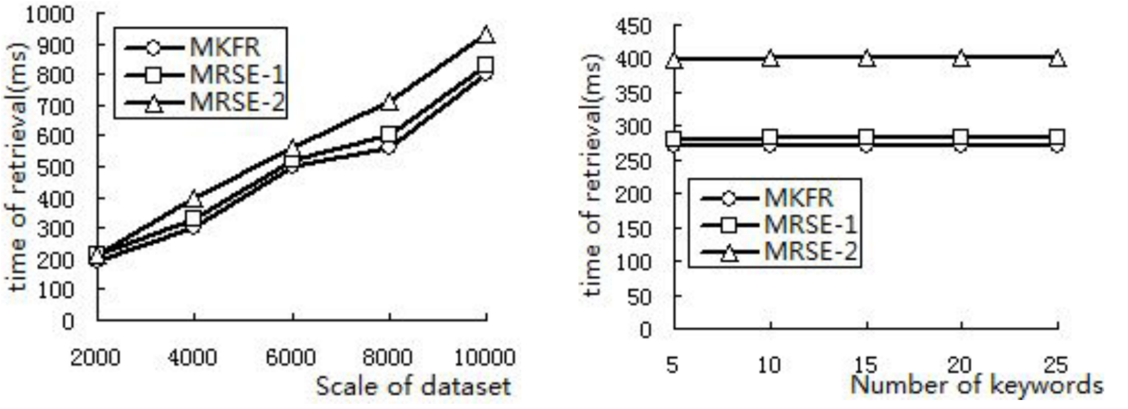


(a) $kw = 10$, different scale of dictionary N (b) $N = 4000$, different number of keywords kw

FIGURE 7. Time cost of generation trapdoor

2) **Time cost of retrieval**

According to the MKFR scheme and data characteristics in cloud environments, this experiment selects the scale of dataset M and number of keywords kw as variable parameters, given the same dictionary where $N = 4000$. Figure 8(a) shows that, the retrieval time of two comparison algorithms increases with M and the MKFR scheme is also affected by M because of introducing the inner product. In Figure 8(b), though the efficiency of MKFR is affected, the number of keywords about twenty can adequately express the user's requirements. Thus, MKFR scheme still keeps more efficient than MRSE-1 and MRSE-2.



(a) $kw = 10$, different scale of dataset M (b) $M = 4000$, different number of keywords kw

Figure 8. Time cost of retrieval

5. **Conclusions.** In this paper, to the demand of privacy, efficiency and accuracy for ciphertext retrieval in cloud computer, we proposed an efficient Multi-Keyword Fuzzy Retrieval (MKFR) over encrypted cloud data. This scheme replaces the uniform hash function of the Bloom filter with LSH for approximate keywords. To protect privacy of indexes and retrieval keywords, we adopt the invertible matrix to encrypt the keywords vector construction. Eventually, the result of retrieval is ranked by the inner product score. The theory analysis and experimental data have shown that MKFR scheme can not only satisfy the user's personalized retrieval in cloud computer, but also ensure the privacy and efficiency during the retrieval process. To achieve higher accuracy and efficiency, we would consider and explore the weight of each query keyword in file, and the correlation among the encrypted indexes in the future work.

Acknowledgment. This work is partially supported by Natural Science Foundation of Hebei Province (No. G2014402027), Project for Humanities and Social Sciences of Hebei Higher Education (No. 2014068), Scientific and Technical Project of Hebei Province (No. 15454707D) and Project for Handan Science and Technology Research and Development (No. 1221103082).

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, A break in the clouds: Towards a cloud definition, *ACM SIGCOMM Computer Communication Review*, vol.39, no.1, pp.50-55, 2009.
[2] P. Qian, M. Wu and Z. Liu, A mehod on homomorphic encryption privacy-preserving for cloud computing, *Journal of Chinese Computer Systems*, vol.36, no.4, pp.840-844, 2015.
[3] J. A. Jones, M. J. Harrold and J. Stasko, Visualization of test information to assist fault localization, *Proc. of the 24th International Conference on Software Engineering*, pp.467-477, 2012.
[4] J. Li, Q. Wang, C. Wang and N. Cao, Fuzzy keyword search over encrypted data in cloud computing, *Proc. of IEEE INFOCOM*, pp.1-5, 2010.

[5] R. Li, Z. Xu, W. Kang et al., Efficient multi-keyword ranked query over encrypted data in cloud computing, *Future Generation Computer Systems*, vol.30, no.3, pp.179-190, 2014.
[6] W. Sun, B. Wang, N. Cao et al., Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, *Proc. of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, pp.71-82, 2013.
[7] F. Q. Cheng, Z. Y. Peng, W. Song et al., An efficient privacy-preserving rank query over encrypted data in cloud computing, *Chinese Journal of Computers*, vol.35, no.11, pp.2215-2227, 2012.
[8] N. Cao, C. Wang, M. Li et al., Privacy-preserving multi-keyword ranked search over encrypted cloud data, *IEEE Trans. Parallel and Distributed Systems*, vol.25, no.1, pp.222-233, 2014.
[9] W. K. Wong, W. L. Cheung, B. Kao and N. Mamoulis, Secure kNN computation on encrypted databases, *Proc. of the 2009 ACM SIGMOD International Conference on Management of Data*, pp.139-152, 2009.
[10] A. Broder and M. Mitzenmacher, Network applications of Bloom filters: A survey, *Internet Mathematics*, vol.1, no.4, pp.485-509, 2004.
[11] Y. Hua, B. Xiao, B. Veeravalli et al., Locality-sensitive Bloom filter for approximate membership query, *IEEE Trans. Computers*, vol.61, no.6, pp.817-830, 2012.
[12] F. Xiang, C. Y. Liu et al., Research on ciphertext search for the cloud environment, *Journal on Communications*, vol.34, no.7, pp.143-152, 2013.