A Full-Text Retrieval Algorithm for Encrypted Data in Cloud Storage Applications

Wei Song[1,2], Yihui Cui[2], and Zhiyong Peng[1,2(✉)]

[1] State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China
{songwei,peng}@whu.edu.cn
[2] School of Computer, Wuhan University, Wuhan, China
cuiyihui@whu.edu.cn

Abstract. Nowadays, more and more Internet users use the cloud storage services to store their personal data, especially when the mobile devices which have limited storage capacity popularize. With the cloud storage services, the users can access their personal data at any time and anywhere without storing the data at local. However, the cloud storage service provider is not completely trusted. Therefore, the first concern of using cloud storage services is the data security. A straightforward method to address the security problem is to encrypt the data before uploading to the cloud server. The encryption method is able to keep the data secret from the cloud server, but cloud server also can not manipulate the data after encryption. It will greatly undermine the advantage of the cloud storage. For example, a user encrypts his personal data before uploading them to the cloud. When he wants to access some data at the cloud, he has to download all the data and decrypt them. Obviously, this service mode will incur the huge overheads of communication and computation. Several related works have been proposed to enable the search over the encrypted data, but all of them only support the encrypted keyword search. In this paper, we propose a new full-text retrieval algorithm over the encrypted data for the scenario of cloud storage, in which all the words in a document have been extracted and built a privacy-preserved full-text retrieval index. Based on the privacy-preserved full-text retrieval index, cloud server can execute full-text retrieval over the large scale encrypted documents. The numerical analysis and experimental results further validate the high efficiency and scalability of the proposed algorithm.

1 Introduction

Cloud computing attracts considerable attentions and interests from both industry and academia because of its scalability, flexibility, and cost-effective features. With cloud storage services, users can rent the cloud storage space to store their personal data and does not have to buy the storage hardware. This mechanism is particularly suitable for the mobile devices which have the limited storage spaces. However, due to the nature of the public, users usually prefer not to store their sensitive private information into the cloud in the plaintext form even if the data privacy is enforced by law.

J. Li et al. (Eds.): NLPCC 2015, LNAI 9362, pp. 229–241, 2015.
DOI: 10.1007/978-3-319-25207-0_20

A straightforward solution for users to protect their data privacy is to encrypt the data before outsourcing. This service mode has been adopted by Amazon IS service. By encryption, the data privacy is preserved, but the data utilization, i.e., search, becomes difficult over the encrypted data. A naive way requires the user to download and decrypt the data and execute the search over the plaintext. Obviously, such approach incurs tremendous overheads of communication and computation and greatly undermines the advantages of using cloud storage applications.

Searchable encryption schemes [1–6] have been developed in recent years for balancing the search efficiency and data privacy. However, the existing approaches mainly focus on the keyword-based search which is difficult to meet the requirements of the large-scale cloud storage systems. Full-text retrieval is a successful information retrieval technology for content search over the large scale data. Its effectiveness and efficiency have been verified by the success of Internet search engine systems. However, the full-text retrieval technology needs to extract all the words in the contents of documents, which makes the scale of index words is much larger than that in the keyword-based search. Therefore, it is far from practical to provide the full-text retrieval services for the cloud storage applications using the existing searchable encryption schemes.

To achieve the full-text retrieval over the large scale encrypted documents, we design a privacy-preserved full-text retrieval index based on which a full-text retrieval algorithm have been proposed. In our scheme, all the documents are encrypted, the search processes do not need to decrypt the data. We analyze the efficiency of the proposed scheme and prove its security. The main contributions of this paper can be summarized as follows:

- To the best of our knowledge, this is the first work that identifies the problem of privacy-preserved full-text retrieval over the encrypted data for a large-scale cloud storage system.
- To address this problem, we propose a secure index structure, based on which an efficient and secure full-text retrieval scheme over the encrypted data has been proposed.
- We analyze the security and efficiency of the proposed scheme. Moreover, we demonstrate the effectiveness and efficiency of the proposed scheme through extensive experimental evaluation.

The rest of our paper is organized as follows. In the next section, we discuss some related work. Then we introduce our full-text retrieval algorithm in Section 3. In Section 4, we analysis the efficiency of our scheme and prove the security. We evaluate the performance of our scheme by the experiments in Section 5. Finally, our paper is concluded in Section 6.

2 Related Work

Our work mainly focuses on addressing the problem of the secure and efficient full-text retrieval over the encrypted data at the cloud. To the best of our knowledge, no existing research has addressed this issue. The existing researches that

are similar to ours can be found in the areas of keyword searchable encryption, rich functional encrypted data search, and ranked search over encrypted data.

2.1 Keyword Searchable Encryption

The existing keyword searchable encryption schemes [1–6] usually build an encrypted searchable index such that content of sensitive documents is hidden to the cloud server. The client gives appreciate query trapdoors through the secret key to retrieve the interested documents. It is first studied by Song et al. [4] with the symmetric surroundings. Some improvements and advanced security definitions, including Goh [1], Chang et al. [2], Curtmola et al. [5], and Kamara et al. [8] are given. In the public key setting, Boneh et al. [6] propose the first public key searchable encryption construction (PEKS), where anyone encrypts the data using public key but only authorized users with private key can create query trapdoors to search. Public key solutions usually have a high computation overhead. The Bloom filter is an effectively searchable index structure by which some researches [2,3,6] have been given to implement encrypted data query. However, these researches put keywords of a document into a Bloom filter, this document-based index pattern needs to match Bloom filter one by one during query, so the query efficiency will be reduced with document scale increasing. Our work, which builds the hierarchical Bloom filter tree index based on the word, can keep the high query efficiency with a large scale of documents.

2.2 Rich Functional Encrypted Data Search

To achieve authorized encrypted data search, the public-key searchable encryption schemes [9–12] have been proposed. A common approach of them is to use a paring based encryption to construct the searchable indexes. However, these solutions are inefficient to support the cloud storage application because paring operations are expensive. To enrich the search functionalities, conjunctive keyword search, fuzzy keyword search, and subset query [9,13–16] over encrypted data have been proposed. These schemes result in large overhead for their functional computations, such as bilinear computation cost in [13], communication cost for secret sharing in [14]. And a more general search schemes, predicate encryption schemes [17–19] are recently proposed to support both conjunctive and disjunctive search. The disjunctive search is similar to our full-text retrieval which returns every document that contains a subset of the query keywords. Moreover, our full-text retrieval scheme designs the effective index structure contains all words of a document, so it can describe document contents more accurately and comprehensively and make the cloud with high availability.

2.3 Ranked Search Over Encrypted Data

Ranked searchable encryption enables users to retrieve the most relevant documents from the cloud in the case that both the user queries and data are in the

encrypted form to protect user privacy. The work in [20] is the first research for ranked search encryption. It only supports single-keyword search, and encrypts documents and queries with a one-to-many OPSE (Order Preserving Symmetric Encryption) scheme [21] and utilizes keyword frequency to rank query results. Their following work [22], which supports multi-keyword searches, uses the secure KNN scheme [23] to rank the results based on inner product value.

3 Full-Text Retrieval Algorithm Over Encrypted Data

3.1 Full-Text Retrieval Model and Its Security Problem

The full-text retrieval model is a successful technology in the scenario of information retrieval applications, its efficiency has been verified in the popular search engine applications. The straightforward method to achieve the full-text retrieval over encrypted data is to improve the index structure in the existing full-text retrieval model. To protect the privacy of user's personal data, we should encrypt the full-text retrieval index. According to the encryption granularity, the encrypted index can be divided into two main categories: '*index level*' and '*token level*'. The '*index level*' mode is to encrypt the whole index. During the search processes, cloud server has to decrypt the entire index or partial index and execute the query over the plaintext index. The searches in the '*index level*' mode will lead to a great deal of encryption/decryption operations, so it is not suitable for the large-scale cloud data. Moreover, this service mode makes the cloud server be able to decrypt the index on the cloud, which makes it be not able to resist to the internal threatens.

The '*token level*' mode encrypts the tokens and builds the secure full-text retrieval index to enable the searches over the encrypted data. But, the existing various full-text retrieval algorithms are based on the token offset position and the token frequency, which will leak the user's privacy. During the search processes, the server needs to compute the offset position and the token frequency, so we can not directly encrypt the offset position and the token frequency. We design a novel full-text retrieval algorithm over the encrypted data without using the offset position and the token frequency.

To introduce our scheme, we first define the full-text retrieval model.

Definition 1: A full-text retrieval system R can be defined as $\{D, Q, F\}$, where D is the set of documents in R, and Q is the expression of the user's queries, and F denotes the framework of document expression and content extraction.

Based on the above definition, we introduce our privacy-preserved full-text retrieval algorithm as below. For a document d_i, it is mapped into a set of tokens $T_i((t_1, p_1), (t_2, p_2), \ldots, (t_k, p_k))$ under the framework F, in which t_j represents a token in d_i's contents, and p_j represents the t_j's offset position in d_i. After extracting all the tokens in d_i, client uses a one-way hash function H to process every plaintext token t_j by its private key key_{pri} and outputs the encrypted token e_j as Equation 1.

$$e_j = H(t_j | key_{pri}) \tag{1}$$

Once the client gets the set of encrypted tokens $ET_i = (e_i, p_i)_{1 \le i \le k}$ as in Equation 2, it builds the encrypted full-text retrieval index $EIndex$ for d_i and uploads to the cloud server.

$$d_i \xrightarrow{F} T_i((t_1, p_1), \ldots, (t_k, p_k)) \xrightarrow{Encrypt} ET_i((e_1, p_1), \ldots, (e_k, p_k)) \xrightarrow{Index} EIndex \quad (2)$$

3.2 System Model

We first present the overview of the proposed privacy-preserved full-text retrieval framework in this subsection, which is shown in Fig. 1.

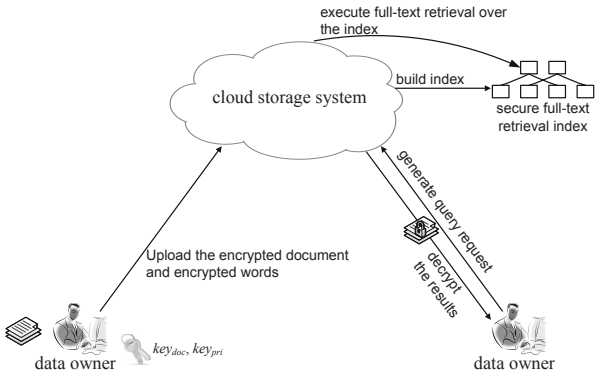

Fig. 1. The full-text retrieval framework of our scheme

The processing flow of the full-text retrieval over the cloud encrypted data is as follows. While a new user joins the cloud storage system, he first chooses a private key key_{pri} and a document encryption key key_{doc} and stores them at local. And the cloud server initializes a hash function H for the new user. H represents a one-way hash function which maps arbitrary string to an integer between 1 and 2^m. m is a system global parameter decided by the cloud server. To protect data privacy, the data owner utilizes the symmetric encryption algorithm to encrypt the documents before outsourcing. Besides encrypting the documents, the data owner extracts all the words from the contents of the document and encrypts these words by the hash function H with the key key_{pri}. Finally, the data owner uploads the encrypted documents and the encrypted index words to the cloud. Once the cloud server receives the files uploaded from the data owner, it inserts them into the full-text retrieval index to provide the secure and efficient full-text retrieval services.

When the user wants to query the data in the cloud with certain query words, he generates encrypted query words using the key key_{pri} and the hash function H. After the cloud server receives the query request, it executes the full-text retrieval over the index. Finally, the authorized user decrypts the results returned from cloud with key_{doc} to finish the query processes.

3.3 Privacy-Preserved Full-Text Retrieval Index Based on B+ Tree

But the token's offset position will leak the user's privacy, so we design a privacy-preserved full-text retrieval index based on B+ tree without token's offset position. The index structure is shown in Figure 2.

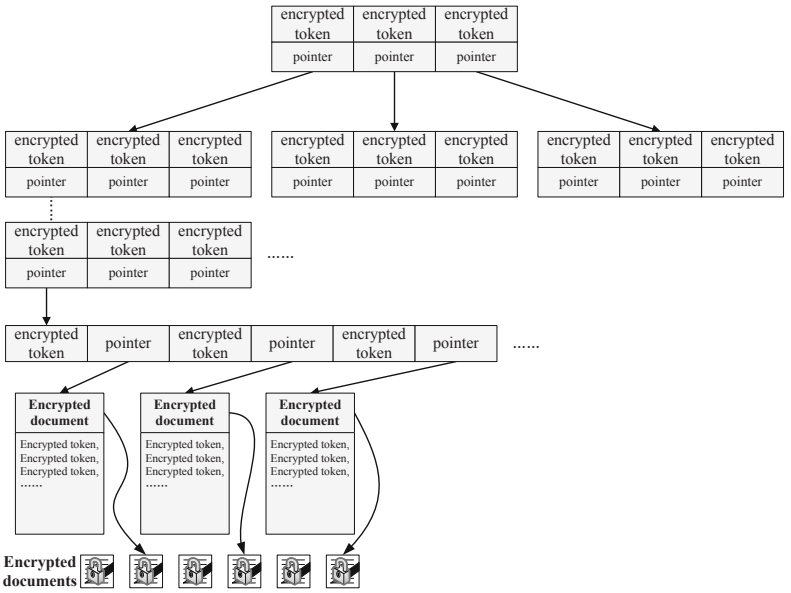

Fig. 2. The encrypted full-text retrieval index based on B+ tree

First, the client calls the document processing algorithm which we introduce in the next section to extract all the tokens $T_i(t_1, t_2, \ldots, t_k)$ for a document d_i. To protect user's privacy, the index in our scheme does not include the tokens offset position information. Then, we improve the index generation in Equation 2 as Equation 3

$$d_i \xrightarrow{F} T_i(t_1, t_2, \ldots, t_k) \xrightarrow{Encrypt} ET_i(e_1, e_2, \ldots, e_k) \xrightarrow{Index} EIndex \tag{3}$$

In our scheme, the search processes does not have the decryption operations. After the cloud server receives the query request from the client, it converts the request to the query condition formed by the encrypted query words.

3.4 Document Pre-processing

In the traditional plaintext full-text retrieval algorithm, the server determines whether a long compound word in a document's contents based on the word's offset position. For example, if a word *cloud*'s offset position in the document

d is x and a word *computing*'s offset position in the document d is $x+1$, then the server can tell that the compound word 'cloud computing' must be in d. To protect user's privacy, we do not store the tokens' offset position information in the full-text retrieval index, so we propose the privacy-preserved full-text retrieval algorithm without offset position supporting.

If we put all the possible compound words into the index, then we can achieve the full-text retrieval over the encrypted data without offset position. But, putting all the compound words of a document into the index will damage the search efficiency. Moreover, the compound word which has a long size has a small possibility for query hit. So, we design the maximal word length k extraction algorithm in Algorithm 1 to collect all the words for a document, which will extract all the single words and compound words, the length of which are no more than k.

Algorithm 1. Extract the index words for a document

Input: d, a document to be uploaded; k, maximal number of single words in a compound index word; SW, stop words list
Output: *words*, the index words for d

```
extract all the single words from d and put them into d_words;
for (i = 1; i ≤ d_words.size; i++) do
    t = d_words[i];
    if (t in SW) then
        continue;
    else
        add t into words;
    for (j = 1; j < k; j++) do
        if (the jth word after t is in SW) then
            break;
        else
            add the compound word from t to the (j)th word after t into words;
remove reduplicative compound words from words;
return words;
```

After the data owner collects all the index words for an uploading document d, he uses his private key key_{pri} and the hash function H to encrypt these index words as Equation 1. Then, the data owner uploads these encrypted index words with the encrypted document to the cloud server.

While the cloud server receives an encrypted document d and its corresponding encrypted index words $e_i (1 \leq i \leq k)$, it searches the full-text retrieval index at cloud and finds all the nodes which are equal to one encrypted index word e_i. At last, the cloud server creates the pointer links from these nodes to the encrypted document to finish the document insertion.

3.5 Full-Text Retrieval Algorithm Over the Encrypted Data

While a user wants to search his interested files at cloud, he first gives some query words like using the web search engine applications. The search processes are described by Equation 4. A user's query Q is composed by several query words $qw_i(1 \leq i \leq n)$. The user uses the same hash function H and his private key key_{pri} to encrypt these query words by $ew_i = H(qw_i|key_{pri})(1 \leq i \leq n)$ and submits them to the cloud server.

$$Q \rightarrow qw_1(\wedge, \vee)qw_2, \ldots, qw_n \xrightarrow{Encrypt} EQ(ew_1, ew_2, \ldots, ew_k) \xrightarrow{Search} Results \quad (4)$$

Once the cloud server receives the encrypted query words ew_i from the user, it executes the query based on the index in Fig. 2. Based on the characters of B+ tree, the cloud server finds all the nodes which equal to a encrypted word ew_i from the root node. Afterwards, the cloud server returns the documents which have pointer links to these nodes as the result of the query to the user.

4 Performance and Security Analysis

In this paper, we propose a privacy-preserved full-text retrieval algorithm over the encrypted data. During the services, our scheme does not need the decryption operations. In this section, we analyze the efficiency and the security of our scheme.

4.1 Query Precision of Our Scheme

The query precision rate indicates the ratio of the exactly relevant documents to all the returned documents as illustrated in Equation 5, in which D_{match} represents the total documents which correctly match a query, and D_{return} represents the total documents returned from the cloud server.

$$Precision = \frac{D_{match}}{D_{return}} \times 100\% \quad (5)$$

The index of our scheme is built based on the Hash function H, so the query false positive rate brought by the confliction of Hash function has to be considered. We assume that the width of the index node is m, then the confliction of a node is $\frac{1}{2^m}$. Consider a document d' with k index words uploaded by a user u' whose private key is key'_{pri}, then the encrypted document d' has been mapped into k index nodes. While a user u whose private key is key_{pri} launches a query Q, the probability of a query word qw equal to an index node linked to d' but this index node is not the word qw is $1 - (1 - \frac{1}{2^m})^k$. For example, when $m = 16, k = 1000$ the false positive rate is 1.51%, and the false positive rate is 0.47% when $m = 20, k = 5000$.

4.2 Query Efficiency of Our Scheme

The query processes in our scheme include three main steps: 1) the user encrypts the query words and submits them to the cloud server. 2) the cloud server executes the query over the full-text retrieval index. 3) the user decrypts the encrypted documents returned from the cloud server by the key key_{doc}.

In the 1st query step, the cloud user first selects several original query words, then encrypts the query words by H. The computation cost of the first step is $c \times T_{hash}$, where c represents the number of original query words given by the user, and T_{hash} represents the computation cost of one hash operation.

For a privacy-preserved full-text retrieval index which depth is l, the cloud server needs to route to the nodes which are equal to the encrypted query word from the root node. So, the computation cost of the $2nd$ step is $c \times l$.

In the $3rd$ step, the user decrypts the encrypted documents returned from the cloud by his private key key_{doc}. Assuming that the cloud server returned d documents for a query, and then the computation cost for this step is $d \times T_{dec}$, in which T_{dec} represents the time cost of decrypting a document.

4.3 Security Analysis of Our Scheme

In our scheme, the cloud server stores and processes three types of data including the encrypted documents E, the full-text retrieval index I, and the query requests R from the user. In these information, E is encrypted by the data owner. Meanwhile, the key key_{doc} is grasped by the data owner. We assume that it is impossible for an adversary to stole key_{doc} from the data owner. Therefore, based on the security of encryption algorithms, the attacker is unable to break the data privacy through attacking E without key_{doc}.

For an external attacker, i.e., the hacker or the internal attacker, he masters the full-text retrieval index in the cloud. We assume that an adversary $\mathcal{A}$ tries to break the security of our scheme. First, $\mathcal{A}$ attempts to guess the word in the full-text retrieval index node and to guess the information of encrypted documents. To guess the word in an index node, $\mathcal{A}$ has to answer the one-way Hash function H. Based on the index structure, $\mathcal{A}$ can output the word in an index node with probability roughly $q_H/2^m$, where q_H is the total number of the queries on H. Take the index structure ($m = 24$) as the example, the probability that $\mathcal{A}$ can exactly guess the word in an index node is no more than $1/16,777,216$ in one time query. Therefore, through the analysis, we can think of the data privacy in our scheme is guaranteed.

5 Experiments

5.1 Experimental Setup

We use JAVA language to implement the proposed full-text retrieval scheme in this paper. We carry out the experiments on a PC machine running Windows 7 with a 64-bits, 3.0 GHz CPU and 4GB main memory. The parameters of our

scheme in the experiments are shown in Table 1. In the experiments, we use ICTCLAS[1] to extract the words from the contents of documents. Moreover, we run the experiments over the dataset: Chinese laws and regulations ceremony which size is 60,000.

Table 1. Experimental Parameters in the Experiments

Parameters	Parameter Descriptions	Values
m	the bit length of index node	24
k	the maximal length of the compound index word	$5, 6, 7$

5.2 Storage Overhead of Our Scheme

Usually, a cloud storage server stores large scale of documents. So, the storage overhead of secure full-text retrieval index is significant important for the performance of cloud storage system. We design the experiments to measure the storage space of the index in our scheme. We compare our scheme with 2-MCIS [7] which is multi-keyword search over the encrypted data. The experimental results are shown in Fig. 3.

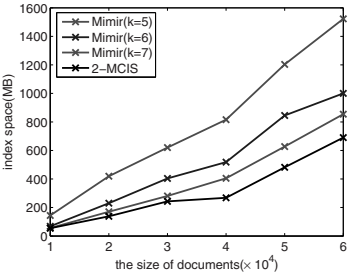

Fig. 3. Storage overhead of our scheme

From the experimental results, we can find that the storage overhead of our scheme is comparable with that in 2-MCIS. When $k = 5$ and $d = 60,000$, the index is no more than 830MB. So, our scheme achieves a satisfied storage performance which makes our scheme is able to efficiently support a large scale cloud storage system.

[1] NLPIR. http://ictclas.nlpir.org/

5.3 Query Efficiency of Our Scheme

For a cloud storage system, the query response time is one of the most important indicators. We design the experiments to evaluate the mean query response time with different k value. The experimental results are shown in Fig. 4.

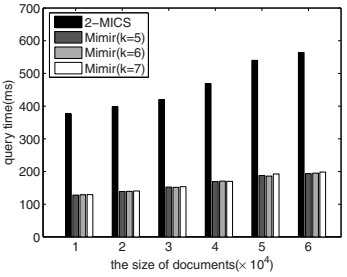

Fig. 4. query efficiency with various scale of documents

We can find from the experimental results in Fig. 4 that our scheme is more efficient than 2-MICS. For a query, our algorithm is able to make the cloud server execute the query in 200 ms. Moreover, the query efficiency will not reduce greatly with the increasing of the document scale. This interesting feature makes our scheme be able to support large scale cloud storage system.

6 Conclusion

In this paper, for the first time we define and address the problem of supporting efficient yet privacy-preserved full-text retrieval services to enrich the query function over the cloud encrypted data. We design a privacy-preserved full-text retrieval index structure to allow the authorized user to execute the full-text retrieval over the encrypted documents at the cloud. Through the rigorous security and performance analysis, we demonstrate that the proposed solution is secure and privacy preserving. Extensive experimental results further validate the effectiveness and efficiency of our solution.

Acknowledgments. This work is supported by National Natural Science Foundation of China No. 61202034 and 61232002, CCF Opening Project of Chinese Information Processing No. CCF2014-01-02, and the Program for Innovative Research Team of Wuhan No. 2014070504020237.

References

1. Goh, E.-J.: Secure indexes. IACR Cryptology ePrint Archive (2003)
2. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
3. Watanabe, C., Arai, Y.: Privacy-preserving queries for a DAS model using encrypted bloom filter. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 491–495. Springer, Heidelberg (2009)
4. Song, D., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. In: Proceedings of S&P, pp. 44–55 (2000)
5. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmertric encryption: improved definitions and efficient constructions. In: CCS (2006)
6. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
7. Wang, N., Zhao, W., Liu, G., Zhao, C.: K-mapping chipher index scheme as to character data in outsourced database. Journal of Yanshan University **33**(5), 438–443 (2009)
8. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic Searchable Symmetric Encryption. In: Proceedings of CCS, pp. 965–976 (2012)
9. Li, M., Yu, S., Cao, N., Lou, W.: Authorized Private Keyword Search over Encrypted Data in Cloud Computing. In: ICDCS, pp. 393–402 (2011)
10. Sun, W., Yu, S., Lou, W., Hou, Y., Li, H.: Protecting your Right: Attribute-based Keyword Search with Fine-grained Owner-enforced Search Authorization in the Cloud. In: Proceedings of INFOCOM, pp. 226–234 (2014)
11. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor Security in a Searchable Public-key Encryption Scheme with a Designated Tester. Journal of System and Software **83**(5), 763–771 (2010)
12. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
13. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM, pp. 441–445 (2010)
14. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)
15. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
16. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public key encryption that allows PIR queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
18. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (Hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)

19. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
20. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: Proceedings of ICDCS, pp. 253–262 (2010)
21. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009)
22. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of INFOCOM (2011)
23. Wong, W.K., Cheung, D.W., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: Proceedings of SIGMOD, pp. 139–152 (2009)