Verifiable Multi-Keyword Fuzzy Search over Encrypted Data in the Cloud

Xue Wei[1, a], Hua Zhang[2, b]

[1] school of science ,Beijing University of Posts and Telecommunications, Beijing 100876, China;

[2] state key laboratory of networking and switching technology ,Beijing University of Posts and Telecommunications, Beijing 100876, China.

[a]weixue0612@163.com, [b]zhanghua 288@bupt.edu.cn

Keywords: Multi-keyword, Fuzzy, Verify, Cloud computing, Search.

Abstract. Cloud computing has been developing rapidly as an emerging technology, it supports the data owner to outsource the data files to the cloud server, then the user can get the ideal results by searching algorithms. Now, the existing searching algorithms cannot support fuzzy multi-keyword searching and result verification very well. So in this paper, we propose a searching algorithm to support fuzzy multi-keyword searching and result verification. Specially, our scheme does not need the dictionary any more, and we use LSH function and bloom filter, together with binary tree to improve the speed of searching. Our experiment has shown that our algorithm has high efficiency for the fuzzy multi-keyword searching in the cloud based on IEEE INFOCOM publications.

Introduction

The cloud computing has been developing rapidly and popular, it has been emerging as a new infrastructure. It brings a lot of convenience to us. Such as the cloud allows company and organization to enjoy the storage savings [4, 5, 6], the data owner can outsource the data files to the cloud server, and at the same time ,the data user can search the data files to get the corresponding results in the cloud. This means the cloud is a more scalable, low-cost, high efficiency, and therefore it is favorable to us.

But in the real scenario, we can not completely trust the cloud server, because it may leakage the user's sensitive data, such as the health records, financial transactions and so on. So in order to protect our privacy, we need to encrypt the data files before we transmitted to the cloud server. This in turn will make the data utilization a challenging problem. The data user can download all the data files and decrypt them to get the results they want, but it apparently impractical. So as the result, mechanisms that allow users to search directly on the encrypted data are of great interest in the cloud computing era.

Since now, a lot of work has been done to develop a effective searching mechanism which support the encrypted data searching [4, 5, 6, 7, 8, 9, 10]. Specially, fuzzy keyword search has made a lot of progress [1].But all these schemes only support single keyword search, with a common approach involving expanding the index file by covering possible combinations of keyword misspelling so that a certain degree of spelling error, measured by edit distance, can be tolerated. Although a wild-card approach can be adopted to minimize the expansion of the resulting index file, for a letter long keyword to tolerate an error up to an edit distance , the index has to be expanded by O(ld) times. Recently, Wang et al. [14] has proposed an scheme that can support fuzzy multi-keyword searching, moreover, it do not need dictionary anymore. Sun et al. [15] proposed a verifiable privacy-preserving scheme that can support multi-keyword text search in the cloud. But the scheme only can support exact word matching.

Today, an efficient multi-keyword search is still a problem .In this paper, I perform fuzzy multi-keyword search over encrypted data in the cloud and result verification. So in this paper, to meet fuzzy multi-keyword search with high efficiency, we firstly exploit the LSH functions and bloom filter [14] with binary tree . With this approach, the keywords and the query will be described in bigram vector and we uses LSH function to support typo errors and format errors. To validate the efficiency of our approach, we conducted extensive experiment using IEEE INFOCOM publications.

The results show that the our approach has high efficiency for the fuzzy multi-keyword search in the cloud. Our contributions can be summarized as follows:

(1) To the best of our knowledge, this is first work that propose fuzzy multi- keyword search result verification.

(2) In contrast to previous keyword search solution [12, 13, 14], our scheme exploits binary tree to improve the searching efficiency.

(3) We implemented our scheme and performed an evaluation using IEEE INFOCOM publications. The result shows that our scheme has high efficiency.

The remainder of this paper is organized as follows. In Section 2, we will state the system model, threat model, design goals and some Preliminaries. In Section 3, we will propose the construction of multi-keyword fuzzy search scheme. And in section 4, we will outline the search result verification scheme. Then we will carry out the experimental performance evaluation and analysis in Section 5. Section 6 will offer some related works and Section 7 concludes the paper publications.

Problem Formulation

System Model. As the figure 1 we can see, to support the multi-keyword fuzzy search over the encrypted data on the cloud, the data owner have to build an secure index tree for the data files, and outsource the index tree to the cloud server together with the encrypted data. To search the encrypted data files, the data user also should give the trapdoor to the server, we assume that the data owner and the data user already have mutual authentication. When the cloud server receives the trapdoor, it will search the index tree, then return corresponding the encrypted data files to the data user, and the authenticated data user can decrypt data files**.**

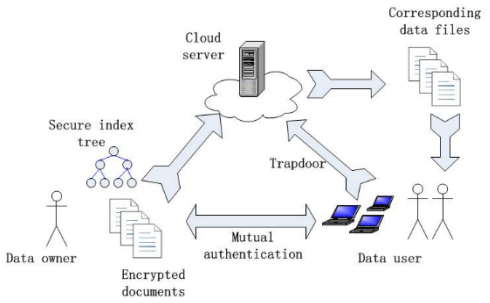


Fig. 1 System model of search over cloud encrypted data

Threat Model. In this scheme, we believe that the cloud server is “honest-but-curious”. We assume the cloud server will honest follow the design procedure to do his job, but he can analyze the information he can get from the secure index and the encrypted data to deduce other information of the data user.

Known Ciphertext Model: In this model, only encrypted document collection and index collection can be known to the cloud server, and all the information are outsourced from the data owner.

Design Goal. In this scheme, we can achieve the following performance and security guarantee:

(1) Multi-Keyword Fuzzy Search: In this scheme, we can solve the fuzzy multiple keyword search, the scheme can

tolerate simple typo errors or format errors, it can also return the ideal results.

(2) Privacy Guarantee: In this scheme, we can provide privacy guarantees by not leaking the information about the data files or the query keywords beyond the search results to the cloud server.

(3) Result Verification： In this scheme, the user can verify whether the result come from the cloud server is correctness and completeness.

(4) High Efficiency: In this scheme, we adopt binary tree to improve the searching efficiency.

Preliminaries. LSH function

Locality-Sensitive Hashing: Given a distance metric d, e.g. Euclidean distance, a LSH function hashes close items to the same hash value with higher probability than the items that are far apart. A hash function family H is (r1, r2, p1, p2)-sensitive if any two points s, t and h ∈ H satisfy:

if d(s, t) ≤ r1 : P r[h(s) = h(t)] ≥ p1 (1)

if d(s, t) ≥ r2 : P r[h(s) = h(t)] ≤ p2 (2)

where d(s, t) is the distance between the point s and the point t

Bloom Filter

Initially, a Bloom filter is a bit array of m bits, all of which are set to 0. Given a set S = {},a Bloom filter uses l independent hash functions from H ={hi |hi : S → [1, m], 1 ≤ i ≤ l} to insert an element ∈ S into the Bloom filter by setting the bits at all the hi (a)-the positions in the array to 1. To test whether an element q is in S, feed it to each of the l hash functions to get l array positions. If the bit at any position is 0, then q ∉ S; otherwise, either q belongs to S or q yields a false positive. The false positive rate of a m-bit Bloom filter is approximately rate of a m-bit Bloom filter is approximately. The optimal false positive rate is when

The Construction of Multi-Keyword Fuzzy Search

Keyword Extraction. The first step is to extract keywords for every document in the data files.

We denote as:

{,, … , }

{,, … , }

…

Keyword and Query Transformation. Keyword(Query) → bigram set → bigram vector

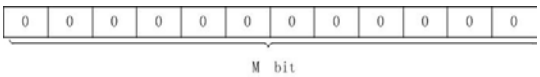
We need to make the keyword transformation in a bigram set, which contains all the contiguous 2 letters appeared in the keyword.

Take “apple” as an example, the bigram set is {#a,ap,pp,pl,le,e# }, then we use -bit long vector to represent the set. In the vector, each element represents one bigram .If the corresponding bigram exists, we set the element is 1;if the corresponding bigram does not exist, we then set the element is 0.So the keyword “apple” can be denoted as an -bit long vector.

To preserve the keyword privacy, the data user and the data owner have mutual authentication, so only the authentic user know that the i-th bit denote which bigram set in the bigram vector. The cloud server can not know, so that the scheme can protect the keyword privacy.

Build Index. In this scheme, each document has its own index. And each document index generated by its corresponding keywords.

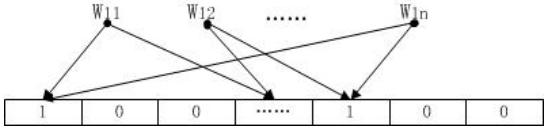
In the initial phase, every index is an m-bit bloom filter which every bit is set 0, denoted as:



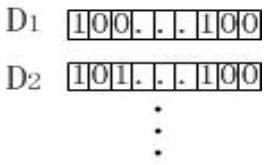
The primitive bloom filter uses traditional hash functions, in the scheme, we use locality sensitively hash (LSH) function to instead traditional hash function. Because the scheme support fuzzy multiple keyword search, the traditional hash function will let two similar words have completely different output, so we now use LSH function to instead. LSH function can let two similar words have the same output with high probability , so in this paper we uses it to solve the fuzzy keyword search. The detailed procedure shows as follows.

We take the {,, … , } as an example.

We insert all the keywords in into the empty bloom filter by using two LSH functions, as



Rules：In order to prevent collisions, in this scheme ,we use two LSH functions, if the i-th bit has been set 1,then no matter how many time the bit be mapped, it always be 1.
Every data file will be done following this rules, then we can have the index for each file. We donated as:



Build Index Tree. (1) For each document , we can generate a leaf node, which stores the document index, the identifier, and the hash value for the document.（The hash value is for user to verify the corresponding data files which the cloud server give, we will show you later）
(2) Then the searching binary tree can be built following a post order traversal with all the nodes stores an m-bit bloom filter generated in step (1). All the internal leaves store an m-bit bloom filter generated by his two own child leave (each bit of the bloom filter has the same order with his child leaves). If the value of corresponding bit in the two child leaves at least have 1,then we set the corresponding bit of the internal leave is 1; otherwise we set it is 0.
(3) We repeat the step (2) to generate all the leaves until the root node is generated.
For example, as the figure 2, the index tree for a document collection of #=8 document with n keywords.

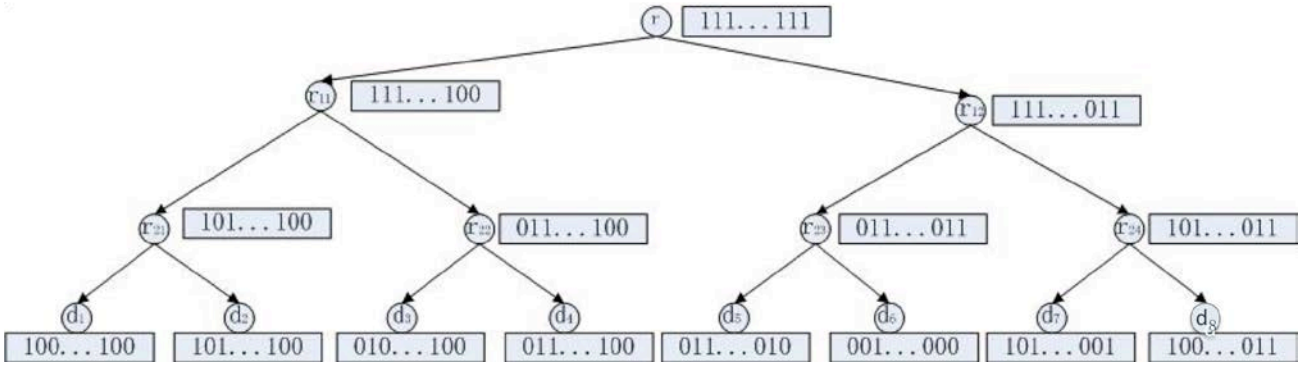


Fig. 2 The index tree for a document collection of #=8 documents with n keywords

Tree-based search algorithm. In this scheme, the tree-based searching algorithm starts from the root node, when the cloud server receive the trapdoor ,he begin to test whether the keywords in the bloom filter by the LSH, if the query all in the bloom filter ,then we begin to search in his child leaves ,until arrives at the leaf node, the cloud server can return the corresponding the data files. But if query is not in the bloom filter, we stop to search in his child leaves.
We take the figure 2 as an example. We assume that the bigram vector of the query is (0,1,0,…),Then we use the vector to test whether the query is in the root node by LSH function. We can find that the root node contains the query , then we continue to test in his child node. We can find the node contains the query, and the node don't. So, the next step is to test the child node of node , and stop to test the child node of node . At last, we will find the node is the results we need. So the cloud server will return documents in to the user.

Search Result Verification
In this scheme, we assume the cloud server is "honest-but-curious", so the data user should verify the search results he received whether is false or contain errors. Our verification mechanism can achieve following goals:
(1) Correctness: it can test the search results whether is true and remain unmodified;
(2) Completeness: it can test the search results whether is completeness.

The detailed procedures show as follows:

In this scheme, the data owner first need to compute the hash value of every document , then he store the hash value in the corresponding leaf node. Then the data owner can generate the hash value, which summary its child nodes:).Every nodes do as this until we get the hash value of the root node, then the data owner signs the hash value , outsource to the cloud server.

To let the data user to verify the search results, the cloud server should return not only the search results, but also the minimum secure index tree, the hash value and the index of the unreturned documents in the minimum index tree, he hash values and the index of the necessary non-leaf nodes outside the sub-tree, which are not accessed in the search process, and the root signature . For example, if the cloud server return the data to the search user, together with the sub-tree which included , ,.

When the data user receive the information, the data owner can search the tree again to verify the return results, and he can compute the root signature by the hash value of the returned nodes.

Experimental Results

In order to test the efficiency of the scheme, we use more than one million data files, which come from the IEEE INFOCOM publication to do the experiment. We implement our schemes on a notebook equipped with Intel Core2 T6670 processor with 2.2GHz and 6G 1600 MHz DDR3 memory. To improve the results precision, the value of the experiment is an average of 100 times experiment values. In order to prove the high efficiency of our scheme, we also implement Wang et al[14]scheme on the same desktop PC using the same data files.

(1) Trapdoor generation: The trapdoor generation process consists only one step: the bigram vector. So it consumes little time. The following figure can show the trapdoor generation time for the query which the query consists of different numbers of keywords. We can see that the time consuming is very small.

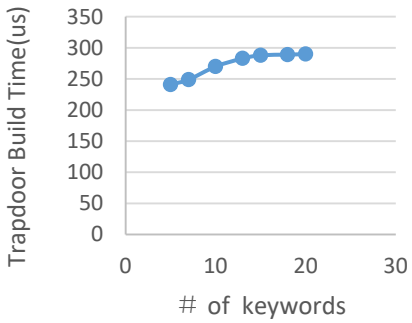


Fig. 3The Trapdoor build time for a single file with # of the keywords

(2) Search over the encrypted data files: The search operation executed at the cloud server which need to check the query whether is in the datasets by bloom filter and LSH functions. The following figure 4 shows that the searching speed of our scheme and Wang's scheme .We implement the experiment on 100 thousand datasets. We can see that when the number of data files is 100 thousand, our searching speed is faster than Wang. In addition, when the number of keyword is more and more, our searching speed will be faster due to the binary tree. The figure 5 shows that in our scheme ,as the number of the documents is growing ,our searching speed is growing slower than the Wang. And when the number of documents is 100 thousand, our searching is apparently faster than Wang. In short, our scheme has high efficiency, especially when the number of document is large.

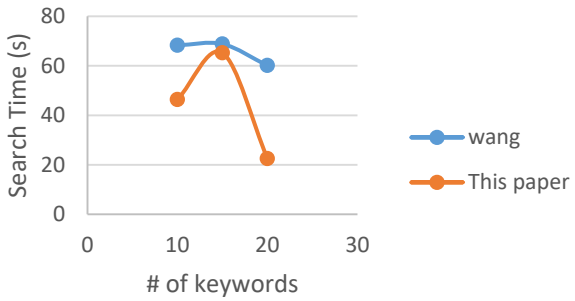


Fig. 4 The search time with # of the keywords

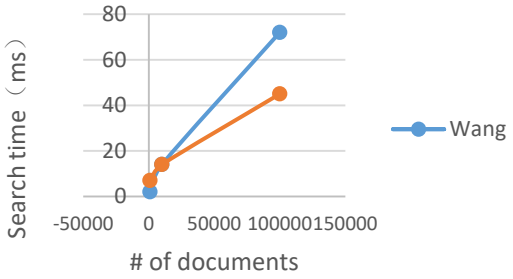


Fig. 5 The search time with # of the documents

Related Works

A. Single Keyword searchable Encryption

To solve the single keyword searching problem, a lot of efforts has been done. Song et al.[4] has proposed the symmetric key setting for the first time. Then Goh [5], Chang et al. [6] and Curtmola et al. [7] has improved the security definitions. Also, Wang et al has done a lot of works in ranking returned results. But all the related works only solved the single keyword searching problem.

B. Boolean Keyword Searchable Encryption

To support more functionalities for the keyword search over the encrypted data in the cloud, some works has done to solve conjunctive search.[12,13]. Later, encryption schemes has proposed to support the conjunctive keyword search and disjunctive search. However, all the related works can not support multi-keyword searching over the encrypted data in the cloud. So Wang et al has done a lot of work to solve the problem. Recently, Wang et al[14] has proposed an effective solution to solve the fuzzy multi-keyword searching on the encrypted cloud data. They firstly use LSH functions and bloom filter to solve the problem

Acknowledgements

This work is supported by NSFC (Grant Nos.61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No.2015RC23).

References

[1] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, “An smdp- based service model for interdomain resource allocation in mobile cloud networks,” *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2222–2232, 2012.

[2] M. M. Mahmoud and X. Shen, “A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1805–1818, 2012.

[3] Q. Shen, X. Liang, X. Shen, X. Lin, and H. Luo, “Exploiting geo- distributed clouds for e-health monitoring system with minimum service delay and privacy preservation,” *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 2, pp. 430–439, 2014.

[4] D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in *Proc. of S&P*, 2000.

[5] E.-J. Goh, “Secure indexes,” Cryptology ePrint Archive, 2003, http:// eprint.iacr.org/2003/216.

[6] Y.-C. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote encrypted data,” in *Proc. of ACNS*, 2005.

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

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *Proc. of EUROCRYPT*, 2004.

[9] J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, “Searchable encryp- tion revisited: Consistency properties, relation to anonymous ibe, and extensions,” *J. Cryptol.*, vol. 21, no. 3, pp. 350–391, 2008.

[10] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, “Fuzzy keyword search over encrypted data in cloud computing,” in *Proc. of IEEE INFOCOM’10 Mini-Conference*, San Diego, CA, USA, March 2010.

[11] L. Ballard, S. Kamara, and F. Monrose, “Achieving efficient conjunctive keyword searches over encrypted data,” in *Proc. of ICICS*, 2005.

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

[13] R. Brinkman, “Searching in encrypted data,” in *University of Twente, PhD thesis*, 2007.

[14] Bing Wang,“Privacy-Preserving Multi-Keyword Fuzzy Search over Encrypted Data in the Cloud ” in *Proc. of IEEE INFOCOM’ 2014*

[15] Wenhai Sun,“Verifiable Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Rank ” in *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 11, NOVEMBER 2014*