Journal of Software Engineering and Applications, 2013, 6, 29-32
http://dx.doi.org/10.4236/jsea.2013.61004 Published Online January 2013 (http://www.scirp.org/journal/jsea)

K-Gram Based Fuzzy Keyword Search over Encrypted Cloud Computing

Wei Zhou[1,2*], Lixi Liu[1], He Jing[1,2], Chi Zhang[1], Shaowen Yao[1,2], Shipu Wang[1,2*]

[1]National Software School of Yunnan University, Kunming, China; [2]Key Laboratory in Software Engineering of Yunnan Province, Kunming, China.
Email: [*]wz.weizhou@gmail.com, [*]spwang@ynu.edu.cn

Received October 21st, 2012; revised November 23rd, 2012; accepted November 30th, 2012

ABSTRACT

With recent significant development in the portable device market, cloud computing is getting more and more utilized. Many sensitive data are stored in cloud central servers. To ensure privacy, these data are usually encrypted before being uploaded—making file searching complicated. Although previous cloud computing searchable encryption schemes allow users to search encrypted data by keywords securely, these techniques only support exact keyword search and will fail if there are some spelling errors or if some morphological variants of words are used. In this paper, we provide the solution for fuzzy keyword search over encrypted cloud data. K-grams is used to produce fuzzy results. For security reasons, we use two separate servers that cannot communicate with each other. Our experiment result shows that our system is effective and scalable to handle large number of encrypted files.

Keywords: K-Gram; Fuzzy Keyword; Encrypted Cloud Computing

1. Introduction

Over the past few years, many people have started to use cloud computing services for their works. With cloud computing, people can store, access and share their information anywhere anytime. Therefore, there are more and more sensitive information being stored in the cloud. In such open environment, users expect not only efficient operations, but also guaranteed privacy and security from the cloud service provider.

Cloud storage is an online storage model where people upload their files and their data will be stored on multiple virtual servers, which are generally hosted by third parties, instead of on dedicated servers. Only authorized users, such as the data owners, can access the stored information. To further protect their data, people usually encrypt not only their file content, but also their file names, before uploading them to the cloud—which makes it difficult for the cloud storage provider to search through the data.

In recent years, searchable encryption techniques have been developed to solve these problems [1-10]. However, these methods are too slow to be used on a large dataset, *i.e.* they are not scalable. Furthermore, users often have spelling errors or use morphological variants of a word. Hence, cloud storage search service should support fuzzy searching.

To solve these problems, in this paper, we propose a k-gram based fuzzy keyword ranked search system over encrypted cloud data. To increase the security of our system, we use two separate servers: search server and storage server. When the search server is compromised, the attacker will not be able to use the file access pattern to deduce the corresponding document that is stored in the storage server. We use k-grams to construct fuzzy keyword sets and Jaccard coefficient to calculate the keywords similarity. To avoid enumerating all fuzzy keywords, and thus reducing the search space, we eliminate keywords with Jaccard coefficient smaller than our threshold value. This threshold value is decided through experimentations described in Section 5. Search results are ranked according to our proposed weighted ranking function. With our system, a user can do fuzzy keyword search in encrypted environment effectively and securely.

This paper is organized as follows: Section 2 presents some researches that have been done on encrypted keyword search; Section 3 introduces our system architecture; Section 4 describes our fuzzy keyword search algorithm; Section 5 shows the simulation results and analysis, and Section 6 concludes this paper.

2. Related Work

Boneh *et al.* [1] introduced the first searchable encryp-

[*]Corresponding authors.

tion system, where anyone with public key can write to the data stored on server but only authorized users with private key can search. However, it takes too much time to calculate public key and the server may decrypted the uploaded file by using the received trapdoor and the keywords encrypted by public key.

Many researchers then tried to improve the efficiency and security definition of a single keyword search system [2-5]. Golle *et al.* [6] defined the first secured model for conducting conjunctive keyword search. Jin Wook Byun *et al.* [7] proposed a more efficient conjunctive keyword search that reduces the data storage and communication cost. Next, Ning Caoy [9] presented multi-keywords ranked search over encrypted cloud data and established variety of privacy requirements. To reduce the user's computational overhead, Qin Liu *et al.* [10] utilized the cloud provider to participate in the partial decipherment of the files. However, all of those schemes only support exact keywords search.

Jin Li [8] is the first to describe the scheme for fuzzy keyword search over encrypted data in the cloud. In this paper, we are introducing a system for performing multiple fuzzy keywords search in encrypted environment.

3. Architecture

The architecture of our system is shown in the **Figure 1**. Our system is divided into three parts: user clients, storage servers and search servers. When a user sends a request to the search server, search sever will search through its local index and send the result to the user. User can then downloads, modifies any of these files from the storage service using the file identifier.

In the real-word environment, there are always threats from adversaries. Some adversaries may have no knowledge about the contents stored by the client. However, some may know the user's query records and use it to generate the list of keywords associated with some of the files. Thus, we use two separate servers (**Figure 2**), one for search and another for storage.

Cloud of Storage Servers: Users upload tuples of file and file identifier to the storage servers. These servers are supplied by the cloud service provider. All the files stored are encrypted. Authorized users use the file identifier to manage and retrieve the file on the storage servers.

Cloud of Search Servers: Search servers use the information uploaded by the user to build index. People can perform k-grams based fuzzy keyword search on the search servers.

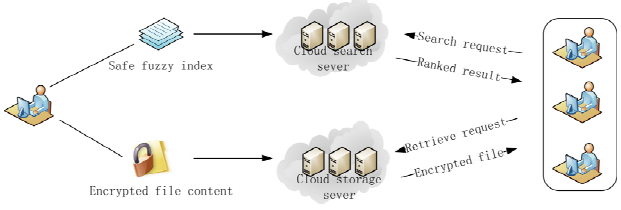

Figure 1. System architecture.

By performing all the searches in a server that cannot communicate directly to the server that stores the actual data, an attacker who knows the content of the search servers will not be able to find the files stored in the storage servers.

Now we briefly describe the ways users can interact with our system.

Figure 3 shows the procedure of uploading files. A user uses a web client to upload an encrypted file with identifier D_i to the storage service and generate the fuzzy keyword trapdoor index I_i. The tuple $< I_i, D_i >$ is uploaded to the search server. The search server inserts the tuple into the Bloom filter and build safe index.

When a user wants to search through their collection, he sends a fuzzy query request to the search server and a list of file identifiers will be returned and sorted based on their rank. He then decides which document to get, modify or delete. Finally he sends a request to the storage service to complete his operation. **Figure 4** shows the procedure of searching files.

4. Fuzzy Keyword Search Algorithm

To generate the fuzzy keyword set, we use the concept of k-grams index, which is used to perform wildcard queries on plain-text files. K-grams is a sequence of k characters. For example, "cou", "our", "urs" and "rse" are all the 3-grams of the word "course". We use the character $ to denote the beginning or the end of a word. Thus, the set of 3-grams generated is: "$co", "cou", "our", "urs", "rse"

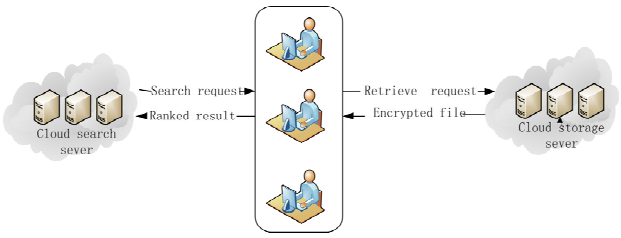

Figure 2. Separated storage service and searching service.

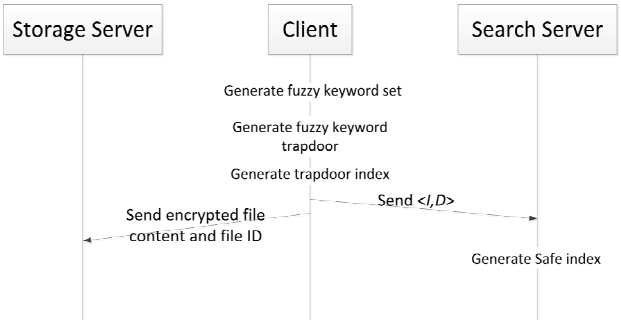

Figure 3. Procedure of uploading file.

and "se$". In a k-grams index, our dictionary contains all the k-grams of every word in the collection. For each k-grams, we create a posting list of all the words in the collection that contain all the characters in the gram. For instance, in **Figure 5** the 3-gram "emp" would point to all the words such as employable and employee.

During the indexing process, our system first constructs the dictionary of all the k-grams in the collection. Posting list for each k-grams are then generated. All of these posting lists compose the k-gram index, that we called safe index. This predefined index will be used to generate fuzzy keyword set.

4.1. Fuzzy Keyword Set Generation

Let's assume that a user queries the keyword *K*. First, we generate the k-grams for keyword *K*, called $G(K)$. For every gram $g_i \in G_k(K)$, the system looks for g_i in the k-gram index introduced above and returns the list of words containing the gram g_i. To reduce our search space, we only want to retrieve vocabularies that are closely similar to the user's query.

If *W* is one of the words in our k-gram index that contain the gram g_i, we use the Jaccard coefficient $(|A \cap B| / |A \cup B|)$ to measure the similarity of the word *K* and the word *W*. Sets *A* and *B* represent the set of k-grams for *K* and *W*, respectively. If *W* is equal to *K*, *W* will have the highest Jaccard coefficient value compare to the other words in the index. If the Jaccard coefficient of *W*, λ_W, is bigger than our threshold value, $\lambda_{\min}$, *i.e.*

$$\lambda_w = |A_W \cap B_K| / |A_W \cup B_K| > \lambda_{\min}$$

we add *W* to our fuzzy keyword set F_k. As explained in Section 5, in our system, $\lambda_{\min}$ is 0.18.

Because each word in the fuzzy keyword set we generated for the word *K* has its own Jaccard coefficient (λ), our fuzzy keyword set is sorted in descending order based on the words' λ values.

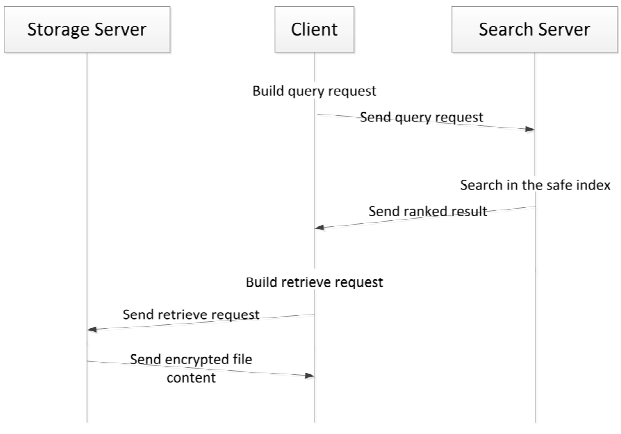

Figure 4. Procedure of searching file.

Figure 5. A 3-gram posting list beginning with emp.

4.2. Weighted Ranking Algorithm

Once users entered their search query, our system will generate the fuzzy keyword set for all the fuzzy words in the query and calculate the weight of each word in the set. The weight of word *W* in fuzzy keyword set F_k is the multiplication of the predefined weight of the word *K* with the Jaccard coefficient of word *W*. We then find the weight of each file in our collection. The weight of a file is defined as the total weight of all the words in the file that are inside the fuzzy keyword set. Thus, we can sort all of the files in our collection in ascending order based on their weights.

4.3. Fuzzy Keyword Search Sumary

To describe how our multiple fuzzy keyword search works, suppose the user types a query $Q = \{K_1, K_2, \cdots, K_n\}$, where n is the number of keywords entered by the user.

As **Figure 6** shows, for each $K_i \in Q$, we compute the fuzzy keyword set F_{K_i} by using the method explained in Section 4.1. Thus, the complete fuzzy keyword set for *Q* is $F_Q = \{F_{K_1}, F_{K_2}, \cdots, F_{K_n}\}$.

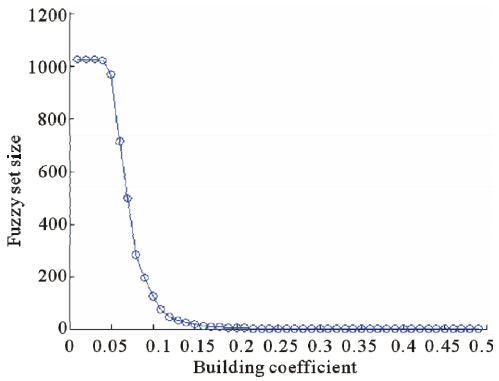

Figure 6. Relation between coefficient and fuzzy sizes.

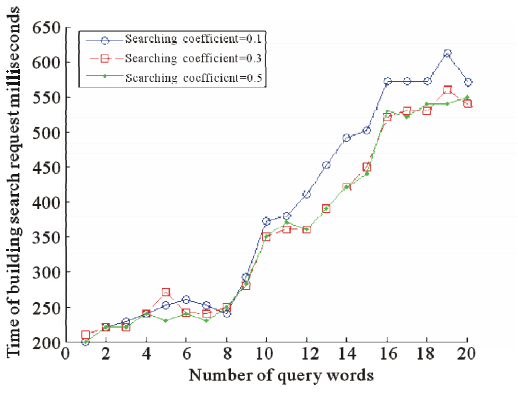

Figure 7. Time of building query request with different query coefficient.

We then calculate the weight of each fuzzy word in the set. Let's W_K be the pre-defined weight of K_i. The weight of the word $K_j \in F_{K_i}$ is $W_{K_j} = \lambda_{K_i} \cdot W_K$.

For each file, its weight is calculated by adding the weight of all the words in the file that are inside F_Q. Once we finished calculating the weight of each file, we sort them in ascending order based on their weights and return it to the user.

5. Experiments

According to our algorithm, the building coefficient, $\lambda_{\min}$, controls the size of the fuzzy keyword set. It describes the minimum similarity level between the predefined keywords (or query keywords) and the fuzzy keywords we generated. To find the best building coefficient value, we scan all the words in the Oxford dictionary and generate the fuzzy sets by using various building coefficient values from 0 to 0.5. As shown in **Figure 6**, when the building coefficient is bigger than 0.2, the demanded similarity level will be too high and the size of the fuzzy keyword set is almost 0. Hence, the result of the fuzzy operation will not be obvious. For example, consider the query word "quick", the fuzzy set will only contain the word "quick" when the coefficient is set to 0.5. **Figure 6** shows that the size of the fuzzy keyword set will be larger than 20 if the coefficient is less than 0.15. Fuzzy keyword set that contains more than 20 words will waste computer space and operation time. Furthermore, those words may have nothing to do with the query word. Through experimentations, we decide that we can get a reasonable fuzzy keyword set size when we set the coefficient to 0.18.

In our scheme, building query request includes 3 procedures: generating fuzzy keyword set according to the building coefficient; generating trap doors; generating index. In procedure 1, we use a predefined k-gram index. To provide more security, two hash functions are used in procedure 2 to generate trap doors. In procedure 3, we use 10 hash functions to generate index. This makes the keyword invisible and safer, and increases the speed to process the query. **Figure 7** shows that as the number of query words increases, the time required to build the query request is growing linearly which means that the time complexity of our building algorithm is O(n). We also find that the time we spent to build the query request is nearly the same even for different fuzzy keyword set.

6. Conclusion

In this paper, we proposed a novel way for performing multiple fuzzy keyword ranked search over encrypted cloud data by utilizing some advanced techniques (such as k-gram) are used to generate a search-efficient index.

7. Acknowledgements

This work is supported by the National Natural Science Foundation of Yunnan Province, China (Grant No. 2008CD084), and the Key Discipline foundation of School of Software of Yunnan university (Grant No. 2010KS05).

REFERENCES

[1] D. Boneh, G. D. Crescenzo, R. Ostrovsky and G. Persiano, "Public Key Encryption with Keyword Search," *Proceedings of EUROCRYPT*, Interlaken, 2-6 May 2004.

[2] E.-J. Goh, "Secure Indexes," Cryptology ePrint Archive, 2003. http://eprint.iacr.org/2003/216

[3] Y.-C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data," *Lecture Notes in Computer Science*, *Applied Cryptography and Network Security*, Vol. 3531, 2005, pp. 391-421.

[4] R. Curtmola, J. A. Garay, S. Kamara and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proceedings of ACM CCS*, Alexandria, 30 October-3 November 2006.

[5] M. Bellare, A. Boldyreva and A. ONeill, "Deterministic and Efficiently Searchable Encryption," *Lecture Notes in Computer Science*, *Advances in Cryptology*, Vol. 4622, 2007, pp. 535-552.

[6] P. Golle, J. Staddon and B. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," *Lecture Notes in Computer Science*, *Applied Cryptography and Network Security*, Vol. 3089, 2004, pp. 31-45.

[7] J. Byun, D. Lee and J. Lim, "Efficient Conjunctive Keyword Search on Encrypted Data Storage System," *Lecture Notes in Computer Science*, *Public Key Infrastructure*, Vol. 4043, 2006, pp. 184-196.

[8] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," *Proceedings of IEEE INFOCOM*'10 *Mini-Conference*, San Diego, March 2010.

[9] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," 2011 *Proceedings IEEE INFOCOM*, Shanghai, 10-15 April 2011, pp. 829-837.

[10] Q. Liu, G.-J. Wang and J. Wu, "An Efficient Privacy Preserving Keyword Search Scheme in Cloud Computing," *International Conference on Computational Science and Engineering*, Vancouver, 29-31 August 2009, pp. 715-720.

