Research on Search Method of Mixed Cloud Storage Segmentation Ciphertext

Wu Xiang¹, Yangliu Ou², Zhang Quan¹, Xiaoming Ju¹

¹East China Normal University, School Computer Science and Software Engineering, Shanghai, China.
²StateGrid Zhejiang Electric Power Company Information and Communication Branch, Hangzhou, China.

Abstract: Aiming at the security protection and effective search requirements of files in hybrid cloud environment and overcoming the untrustworthiness of third-party cloud services, a secure ciphertext search method for hybrid cloud environment based on file-sensitive data segmentation is proposed. Using file-sensitive words to split the file into several sub-files. The sub-files are sensitively encrypted and stored in different cloud environments, allowing file to have better flexibility and security, and construct a ciphertext index of segmented ciphertext. Experiments show that this method can complete the secure file search and restoration work in a complex mixed cloud environment.

1. Introduction

Cloud computing combines related technologies such as network and virtualization to provide services to users [1]. The cost of storage in cloud environments is low, and storage can be allocated according to the needs of users. However, cloud storage currently faces a problem: the security of data stored in the cloud, the data stored in the cloud by the enterprise or users, and the untrustworthiness of cloud service providers. This indirectly means that sensitive information which is stored in the cloud may be leaked. In order to ensure the security of the data, the data is stored in the form of ciphertext in the cloud. However, the ciphertext cannot be searched like a plaintext. Ciphertext search has become a research issue. In 2013, Kamara[2] et al. introduced the red-black tree structure as an index structure, using dynamic SSE, capable of supporting multiple processors. In the same year, Kamara et al. proposed a parallel homomorphic encryption scheme and studied various MapReduce operations including keyword retrieval under the MapReduce parallel computing model. In 2013, Wu Qi [3] aimed at Yang and other multi-keyword combination search solutions, based on the user must give full keywords. In 2014, Chen Hefeng [4] proposed a ciphertext method-based Chinese keyword fuzzy search scheme to study the improvement of the existing ciphertext search based on the similarity of Pinyin. In 2016, Xiaowei Su[5] proposed a parallel ciphertext search scheme, which reduced the index construction time and improved the performance of searching large files.

This paper relies on the background of hybrid cloud environment, proposes file segmentation based on file sensitive words, encrypts child files to store cloud, builds ciphertext index, and establishes ciphertext search method in hybrid cloud environment.
2. File Splitting and Indexing

2.1. File Segmentation
The split encryption [6] scheme divides the data uploaded by the client and uploads it to the cloud storage server, and stores the directory information on the local server, effectively preventing the disclosure of cloud file information.

The file-sensitive word segmentation[7] strategy is adopted to segment the data uploaded by the cloud storage. The small data blocks are stored in the private cloud, and the big data is stored in the public cloud according to a certain storage strategy. Considering the keyword-sensitive factors, let users define a set of sensitive words when uploading files to the cloud. For an uploaded cloud file D, file D has a sensitive word set $Sen = \{W_1, W_2, ..., W_n\}$ and non-sensitive subfile collection $UnSen = \{C_1, C_2, ..., C_m\}$. They make up file D:

$$D = (K_1, L_1) \cup (K_2, L_2) \cup ... (K_n, L_n)$$

The use of (1) sensitive word location to split the file, at the same time, each read a block, block numbering, get the split subfile, stored in $SenD$ and $UnSenD$. In the two file collections, $SenD$ stores the sensitive information data blocks contained in file D, while $UnSenD$ stores the non-sensitive information data blocks in the file. The above method is processed to obtain two sets of data blocks, one is a data block set containing only sensitive words and the other is a set of files containing no sensitive word files. Segmentation diagram is shown in Figure 1.

2.2. Document Key Table
This part is mainly to build a keyword table for the decomposed sub-files. There is no word in the keyword table, the establishment of the index will not be realized. The keyword table of the file is the core data for indexing.

Table 1 Keyword table structure schematic

| keywords | keywords index |
|----------|----------------|
| resource | one            |
| finance  | two            |
| bonus    | three          |
| ...      | ...            |

Table 1 is a simple description of the keyword table. The keyword table has a special word segmentation method. In the cloud environment, there are many kinds of data files to upload, and the data volume is large. If a fixed keyword table cannot be used to deal with new words, a variable keyword table must be used.

The variable keyword table needs to be implemented with TFIDF[8] algorithm. TF denotes keyword frequency, and DF denotes keyword frequency. In practice, TF and DF are a set of words. Under normal circumstances, TFIDF is performed. After analysis, if the values of TF and DF are larger, the frequency of use of the keyword is greater, then it should be more likely to join the keyword table,
and the weight size is defined as W:

$$W = \log(TF) \times \log(DF)$$  \hspace{1cm} (2)

The value of TF and DF is less than 1 floating point number, because the above file split will produce two $SenD$ and $UnSenD$ sub file collection. $SenD$ is a user-defined sensitive keyword, directly into the keyword table. The $UnSenD$ collection needs to use TFIDF to count different keywords for each subfile. According to W weight size to determine whether to join the keyword table. In summary, the establishment of the keyword table for uploading the text is completed.

### 2.3. Split Ciphertext Indexing Scheme

This section discusses the process of creating an index by inserting relevant information from the processing file into the index after completing the relevant processing of the file. The traditional inverted file uses the keyword as the access structure of the index key. The file with inverted index [9] structure is called inverted file. The structure of a typical inverted list is shown in Figure 2:

![Inverted List Structure](image)

The inverted file in Figure 2 is consisted by a collection $k = \{k1,k2,\ldots,kn\}$ and a collection of files $Doc =\{D11,D21,\ldots,Dn1\}$. A pointer is used to connect between the two collections. Each entry $K_i$ has its own file storage information.

The index structure in the hybrid cloud environment is an inverted index built on it. The above file $D$ is divided into two sets of data, $SenD$ and $UnSenD$. For the $SenD$ set, it is a keyword. In the process, It is likely to contain duplicate sensitive keywords, so, as $SenD = \{p1,p2,\ldots,pn\}$, Now define a set $S$ that does not contain duplicate elements

$$S = \text{select}(p1,p2,\ldots,pn)$$ \hspace{1cm} (3)

The select function in Equation 3 implements extraction of a set of unique words in $SenD$. Now, the set of non-repeating elements $S$ in the $SenD$ collection has been obtained. Then, the frequency of occurrence of each element in $S$ in $SenD$ is set to $f$, and for any element in $S$, the position of original file $D$ is set to $d_i$:

$$d_i = \left\lfloor \frac{\log(f[s])}{\log(2)} \right\rfloor$$ \hspace{1cm} (4)

When the frequency of $s_i$ occurs only once, there is only one position, and the record is $k_i$. When the frequency $f_i$ is large, there are multiple places, and the set $Li$ is used for recording.

As $UnSenD = \{q1,q2,\ldots,qn\}$ set, in the file preprocessing, the TFIDF algorithm has been used to obtain the keyword table of the sub-file blocks. When the sub-file data blocks are encrypted and uploaded to create an index, the sub-file is divided by the file $D$, and the location information itself is clear. The inverted list mainly consists of five parts. For an undivided upload file $D$, a unique file number is defined for it. The second part of the index is the file name, which is the name of the uploaded file $D$, facilitating the auxiliary search. The results are presented. The third part is the sensitive level. This is for segmented and stored in the $SenD$ and $UnSenD$ sets. The sensitivity level of the subfile $x$ in any two sets is set as:

$$Entry_A \quad x \in SenD$$

$$Entry_B \quad x \in UnSenD$$

(5)

The files stored in the private cloud are labeled A at the sensitive level and B is stored at the public cloud. The fourth part is where the subtext containing the keyword appears in the original file. Now
the files in UsenD and SenD are discussed separately. The position of the subfile $i$ in the USenD set is $k_i$, and for the element subfile of SenD, it is known as $L_i$ or $k_i$ by Equation (4) The fifth part is the file upload to the cloud address, select a prime $q$ through the authorization center to generate a bilinear group. $f, x, y \in _{p}Z_q^*$. The public parameter has two

$$PK = \{G, f, h = f^\beta, y = g^\beta, \epsilon(f, f)\} \quad (6)$$

$$MK = \{\beta, f^\alpha\} \quad (7)$$

Creating PK, SK, Setting $P = \{p_1, p_2, ... , p_n\}$. For all attribute collections, Then a single user attribute I is a nonempty subset of $P$, $I \subseteq \{p_1, p_2, ... , p_n\}$, $SK = KeyGen(PK, MK, I)$. Generates an access tree Tree using all attributes. The ciphertext $CT = Encrypt(PK, policy, m)$ encrypts the plaintext $m$ using the PK and access $I \subseteq P$ policies, and encrypts the encrypted ciphertext as $CT$. The uploaded ciphertext $CT$ is uploaded to the private cloud. Sensitive subfiles are uploaded to the public cloud via AES encryption and returned to the cloud address. Each information is added to the index information table after a series of processing.

The structure diagram of the inverted cipher text inverted index obtained after a series of processing in the information is added to the index information table is shown in Figure 3.

![Figure 3: Inverted Index Structure](image)

According to the file collection information to form a keyword table, the logical pointer of the keyword points to the logical address of the inverted list. The records in the inverted list mainly consist of five parts, the unique index number of the file, the file name when uploading the file, and the Entry file. To determine whether the record file is from the public cloud or the private cloud, the $loc$ information record corresponds to the location information of the subfile in the original file, and finally records the ciphertext file address information. This completes the construction of the index.

### 3. search methods

#### 3.1. Search Model Research

This section discusses the search work based on the index. From the first part of the process of building the index shows that the original plaintext file is divided and the data is encrypted using different encryption methods. The file's subfiles are stored in two cloud environments. [9-10] Next, private and public clouds. Therefore, during the search, the search of the file address in different cloud environments is performed according to the index. Here, since the file is stored according to the file block, after decrypting the ciphertext, the file block is also restored according to the index information. The search model for the above document search process is shown in Figure 4:
In the search model of Figure 4, the user uploads the keyword information of the file to be searched to the server where the index is located. The server analyzes the file cloud address satisfying the query condition in the inverted index, and downloads the file ciphertext in the cloud environment to the server. The downloaded file is the ciphertext data block decrypts the ciphertext and obtains the plaintext subfile. The splicing of the sub-file is completed through the location information of the sub-file, and the complete plain-text file is obtained and returned to the user. The details are described in the next section.

3.2. File Restoration Strategy

Now the query keyword set is \( A = \{a_1, a_2, \ldots, a_s\} \). When using the keyword collection to query the file, the first is needed to find any element \( a_i (1 \leq i \leq s) \) in the collection \( A \) on the inverted index. The index record of the keyword corresponding to the inverted index (no corresponding is invalid keyword).

Through the index query, get the query record, separate all the records by the bar, set the query record set \( G = \{g_1, g_2, \ldots, g_n\} \). Each record \( g_i (1 \leq i \leq n) \) is grouped according to the same index information on the elements in \( G \) to obtain a number of sub-records grouped. At this point \( G = \{p_1, p_2, \ldots, p_l\} (1 \leq y \leq n) \). any \( P_i \) is a file record set, and the correlation is determined by comparing the keyword corresponding to the record in \( P_i \) with the keyword of the original input set \( A \):

\[
\rho_i = \frac{P_i}{A} \quad 1 \leq i \leq y
\]

Obviously, the higher the value \( \rho \in [0,1] \), the more consistent with the requirements of the query, according to the \( \rho \) to select of the set of addresses that need to be downloaded in the cloud. It is assumed that for a certain file \( P_i \) with some sub-file, the cloud terminal files are searched, and the plaintext collections \( M = \{m_1, m_2, \ldots, m_l\} \) obtained through decryption are recorded. Each of \( m_i (1 \leq i \leq l) \) has a record in \( P_i \). The \( loc \) recording information in the file helps to achieve the splicing of the files, and the file splicing diagram is shown. As shown in Figure 5:

![Figure 5 file mosaic diagram](image)

As shown in figure 5, when the file is spliced and restored, the plaintext set \( M \) is composed of a
non-repeated sensitive set $s$ and a divided non-sensitive word sub-file, and the original file $D$ is composed of $UnSenD$ and $SenD$. The multi-loc information of the index records the repetitive appearance of sensitive words and position information. Therefore, if there is more than one loc in a certain $m_i$, it means that it must belong to the sensitive word set, and it will appear repeatedly, then the plaintext $m_i$ will be reused in the splicing. The schematic diagram above shows that in the end, the complete recovery of the plaintext can be completed.

4. Performance Test

4.1. Security Analysis

Theorem 1 This method can resist collusion attacks

Proof: Assuming that the index table is obtained by the internal method, there are multiple attributes of the access structure tree that satisfy the CP-ABE [12]. When the user does not perform system-authorized collusion access, each user is different when files are encrypted and uploaded. The users $D_i = g^{H(\lambda) i}$ may be different. The user cannot decrypt the sensitive information cryptogram $CT_{SenD}$. In addition, if the $CT_{UnSenD}$ does not have an AES key, it is difficult to decrypt the type of subfile information. Therefore, if there is multiuser conspiracy, even if the file is known in the cloud, the index information, the conspiracy attack, is also unable to get the plaintext $M$, so this method can resist collusion attacks.

Theorem 2: This method can protect data privacy

Proof: The attacker is difficult to access the index information table stored on the local server through the server without the authorization of the system, and it is more difficult to download and obtain the address information stored in the cloud ciphertext and decrypt the subfile. Splicing to restore the original file (2) When an attacker bypasses authentication, but the attacker does not have access to the property tree of the consultant tree Tree, it can't calculate $\epsilon(f_i)^\nu$, and $SS^-$ tree. Therefore, the plaintext $M$ cannot be obtained. Therefore, this method has the privacy protection of data.

4.2. Performance Analysis and Simulation

In this experiment, we use that $SS^-$-search tree and the hybrid cloud-based ciphertext search proposed in this paper have a compared experiments. The experimental environment is in an AMD A8 1.9GHz CPU, 4GB of memory, operating system is windows8 64-bit operating system, using matplotlib to achieve data simulation.

The experiment selects 2500 economic articles from the web, and obtains key words from the article's keyword extraction technology, and sets sensitive words as part of the extracted keywords. Comparing the reverse index time between the $SS^-$ index.
Figure 6 compares the efficiency of indexing building time under the same situation. When it is found that when processing small data sets, the index of this solution has no SS’ tree efficiency, but when the data set volume becomes larger, about 1300 or so texts are constructed, the processing level of the two is similar. Thus, with the increase in the number of texts, the construction efficiency of the inverted index in this paper is higher than that of SS+, and the gap is more and more obvious.

![Figure 6: Efficiency Comparison](image)

Figure 7 Query Time Comparison Experiment

Figure 7 compares the text search efficiency under the same experimental environment. Even though the method involves the time complexity of text splicing and restoration, the overall consumption is still better than the index structure based on SS+-tree, which is also the inverted index. Effectiveness performance.

5. Conclusion

This article discusses a search method[13] using segmented ciphertext storage in a hybrid cloud environment. It uses file sensitive words for file segmentation and constructs a ciphertext index information table for subfiles stored in the cloud. It involves the sensitivity and repetitiveness of stored subfiles. Tag and address information tags discuss index construction. The cipher text search scheme is built on the index table, and the search completes the original text sub-file splicing and restoring operation. At the same time, the paper proves the security of the method in text cloud storage. The results of simulation experiments show that the search method has a good index construction and search efficiency. At the same time, the security and efficiency of cloud storage are both taken into account.

References

[1] VAQUERO L M, RODFRO-MERINO L, CACERES J, et al. A break in the clouds: towards a cloud definition[J]. ACM SIGCOMM Computer Communication Review, 2009(1): 50-55.

[2] Seny, Kamara. Encrypted Search[J]. XRDS, 2015, 21(2015): 30-34.

[3] WU Qi; WAN Chang-xuan. Multi-user Conjunctive Keyword Search Scheme over Ciphertext [J]. Computer Science, 2013, 40(9): 147-152.

[4] WU Yang; LIN Bogang; YANG Yang; CHEN Hefeng. Fuzzy keyword search over encrypted cloud data [J]. Computer Engineering and Applications, 2015, 51(24): 90-96.

[5] SHU Xiaowei; YANG Geng; NA Haiyang. Research on parallel crypt inverted index [J]. Computer Engineering and Applications, 2016, 52(20): 14-19.

[6] XU Xiao-long; ZHOU Jing-lan; YANG Geng. Data Privacy Protection Mechanism for Cloud Storage Based on Data Partition and Classification [J]. Computer Science, 2013, 40(2): 98-102.

[7] Yan YinZhen. Implementation of Text File Segmentation Method[J]. Computer Programming Skills & Maintenance, 2015, (5): 12-16.

[8] ZHANG Yufang, PENG Shiming, LV Jia. Improvement and Application of TFIDF Method Based on Text Classification [J]. Computer Engineering, 2006, 32(19): 76-78.
[9] ZHOU Rui; WANG Xiao-ming. Integrity Verifying Algorithm for Cloud Data Based on Homomorphic Hash Function [J]. Computer Engineering, 2014, 40(6): 64-69.

[10] XIANG Fei; LIU Chuan-yi; FANG Bin-xing; WANG Chun-lu; ZHONG Rui-ming. Research on ciphertext search for the cloud environment [J]. Journal on Communications, 2013, 34(7): 144-153.

[11] ZHANG Lihong; CHEN Jing; DU Ruiying; HE Kun; CHEN Jiong. Data Integrity Verification Method with Secure Deduplication in Cloud Storage [J]. Computer Engineering, 2017, 43(1): 32-36.

[12] ZHANG Qiang; LIU Xueyan; WANG Weizhou; NIU Shufen. Research on Access Control in Smart Grid Based on CP-ABE [J]. Computer Engineering, 2014, 40(12): 83-88.

[13] Tao Chuanyu, Li Weichun. Research on Ciphertext Database Search Method Based on Inverted Index[J]. Electronic Manufacturing, 2015, (11): 46-47.