International Journal of Security and Its Applications
Vol. 11, No. 10 (2017) pp.1-14
http://dx.doi.org/10.14257/ijsia.2017.11.10.01

A Secure Distributed Cloud Storage System Achieving Data Secrecy and Load Balance

Shin-Yan Chiou

Department of Electrical Engineering, College of Engineering, Chang Gung University; Department of Nuclear Medicine, Linkou Chang Gung Memorial Hospital, Tao-Yuan, Taiwan
ansel@mail.cgu.edu.tw

Abstract

Cloud storage is widely used and has become prevalent over the past decade. It has very desirable properties such as scalability, fault tolerance, robustness, and data availability and accessibility. Several Cloud index structures have been proposed for equality queries, range queries, or other purposes. However, Cloud system is fraught with security risks and many security issues still exist in Cloud system. For solving these security issues, in our paper, we propose a secure and balanced storage system for Cloud system. The system offers load balance, secrecy, integrity, and robustness for data protection. It is efficient, reliant, resilient and scalable.

Keywords*: Cloud networks, storage system, P2P, load balance, security*

1. Introduction

Cloud and peer-to-peer (P2P) technology is widely used for file sharing. In the past decade a number of prototypes about P2P information retrieval systems have been developed. The appearance of data-sharing applications, such as Napster [1] and Gnutella [2], has made P2P systems popular for widespread exchange of resources and voluminous information between millions of users. P2P systems have very desirable properties such as scalability (from resource-sharing among cooperating peers), fault tolerance (as the symmetrical nature of peers), and robustness (due to self-reorganization after failures.)

Although P2P or cloud computing move the application and databases to the large data centres, the management of the data and services are not trustworthy. As the cloud services become popular [3], attackers may use cloud services to establish botnet and launch attacks. This attribute poses many security challenges [4,5].

In 2012, S.H. Lee and I.Y. Lee [6,7] proposed a keyword searchable re-encryption scheme that let user share data with others safely by generating searchable encryption index. However, P2P system is still fraught with security risks [8] and many security issues still exist in P2P system. These issues [9] include (but not limited to) accessibility vulnerabilities, physical access issues, privacy and control issues arising from third parties having physical control of data, and issues related to data verification, tampering, integrity, confidentiality, data loss and theft.

For solving these security issues, in our paper, we propose a secure and balanced storage system for cloud P2P system. The system provide balance ring, which lets data stored in each nodes averagely. In addition, for data protection, it offers security properties including secrecy, integrity, and robustness. It prevents files from stealing, modifying, or destroying. Moreover, the system is efficient, reliant, scalable.

Received (February 11, 2017), Review Result (August 29, 2017), Accepted (September 4, 2017)

ISSN: 1738-9976 IJSIA

2. Related Works

This section reviews Chiou's scheme [10]. Chiou [10] proposed a secure saving system for cloud storage system including six parts. We briefly introduce two important parts: (1) file preprocessing and (2) data block indexing.

2.1. File Preprocessing

The file preprocess can be divided into two parts, (1) file encryption and segment, and (2) file composition, decryption and verification.

(1) File encryption and segment

As shown in Figure 1, the steps of file encryption and segment are as follows.

Step 1: Compute the hash value of message M and get $h(M)$, where $h(\cdot)$ is one way hash function.

Step 2: Encrypt $M \| h(M)$ and get cipher text C.

Step 3: Segment cipher text C, with size S for one block, into k blocks $c_1, c_2, ..., c_k$, where $k = \lceil |C|/S \rceil$.

(2) File composition, decryption and verification

The steps of file composition, decryption and verification are opposite to the steps of file encryption and segment. Their steps are shown as follows.

Step 1: Compose k' blocks $c_1', c_2', ..., c_{k'}'$ to get cipher text C'.

Step 2: Decrypt cipher text C' and get $M' \| h(M)'$.

Step 3: Compare whether $h(M)'$ is equal to $h(M')$. If it is not, there is some problem with these blocks. Try to find the backup blocks in each backup nodes and compare whether there is any difference between each blocks.

2.2. Data Block Indexing

The data block index is divided into two parts, (1) get indexes from blocks and (2) get blocks from indexes.

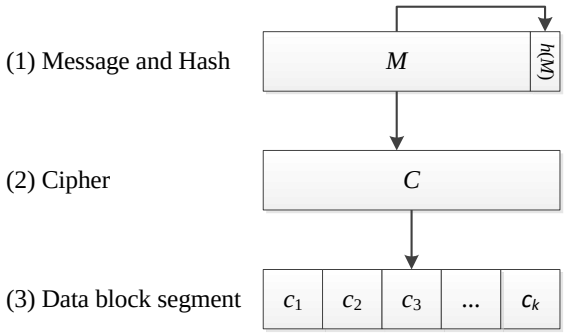

Figure 1. File Preprocess

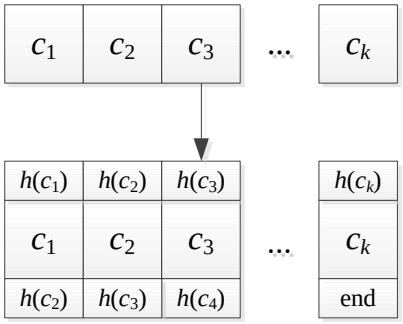

Figure 2. Data Block Index

Table 1. File Index Table

File Name	Time	Index
FN_1	t_1	$h(c_1^{(1)})$
FN_2	t_2	$h(c_1^{(2)})$
FN_3	t_3	$h(c_1^{(3)})$
FN_4	t_4	$h(c_1^{(4)})$

(1) Get indexes from blocks

The steps to get indexes from blocks are shown as follows.

Step 1: Compute the hash value of c_i and get $h(c_i)$, where $i = 1, 2, ..., k$.

Step 2: As shown in Figure 2, let $\{h(c_i) \| c_i \| h(c_{i+1})\}$ be a block and $h(c_i)$ be the block index.

Step 3: Make File Index Table (as shown in Table 1), which includes file names FN_i of message M_i, time t_i, and the index of the first block $h(c_1^{(i)})$. Where time t_i can be create time, modify time, and/or access time.

(2) Get blocks from indexes

The steps to get blocks from indexes are shown as follows.

Step 1: Find file names FN_j and its file index $h(c_j)$ from File Index Table.

Step 2: Let $h(c_j)$ be the index of the block $\{h(c_j) \| c_j \| h(c_{j+1})\}$. Retrieval data block $\{h(c_j) \| c_j' \| h(c_{j+1})'\}$ and compare whether $h(c_j)$ is equal to $h(c_j')$. If it is not, there is some problem with this block and try to find the backup of this block.

Step 3: $j++$. Repeat step 2 until find $\{h(c_i) \| c_i' \| end\}$.

3. Proposed System

The proposed system can be divided into five parts: (1) file preprocessing, (2) data block indexing, (3) balance ring management, (4) block distribution and retrieval, and (5) peer changing. In the system, two important tables, file index and node tables, are suggested putting on both the Client node and the other place. The file index/node table can be put on the Successor and Backer of $ID(0) / ID(1)$. Table 2 defines the notations used in our proposed system. In the table, "||" denotes concatenation.

3.1. File Preprocessing

As shown in Figure 1, file preprocessing can be divided into two parts, (1) file encryption and segmentation, and (2) file composition, decryption and verification. The details are described in section 2.1.

3.2. Data Block Indexing

As shown in Figure 3, we proposed another index method: dual index. Dual index method let each block have one index and all the indexes are integrated as one (or more) new block, which also has one index I. Only the index I is recorded in one File Index Table. In this way if we get the index I from File Index Table, we can obtain all the indexes and then get the whole data blocks.

The data block index can be broken into two parts, (1) obtain the indexes from the blocks, and (2) obtain the blocks from the indexes.

(1) Obtain indexes from blocks

The steps to get indexes from blocks are shown as follows.

(A) Compute the hash value of $c_i^{(x)}$ to obtain $h(c_i^{(x)})$, where $i = 1, 2, ..., k^{(x)}$.

(B) Let $I^{(x)} = h(indx^{(x)})$, where $\lceil \log_2 N \rceil$.

Table 2. Notations

C	Ciphertext
$h(x)$	The hash value of x
I	$h(indx)$
$indx$	$h(c_1) \| h(c_2) \| ... \| h(c_k)$
l	The bit length of ID
M	Plaintext
N	The number of Peer nodes
N_m	The estimate maximum number of peer node
$\lceil \cdot \rceil$	Ceiling function
$left_l(x)$	The left l bits of x
$SID(node_i)$	The successor ID of $node_i$

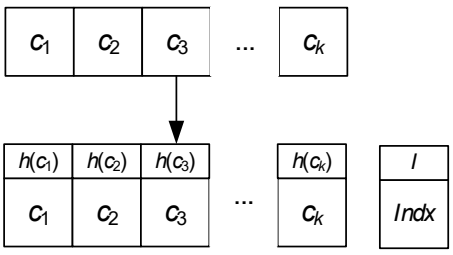

Figure 3. Data Block Index (Dual Index)

Table 3. File Index Table

File Name	Time	Index
$FN^{(1)}$	$t^{(1)}$	$I^{(1)}$
$FN^{(2)}$	$t^{(2)}$	$I^{(2)}$
$FN^{(3)}$	$t^{(3)}$	$I^{(3)}$
$FN^{(4)}$	$t^{(4)}$	$I^{(4)}$

(C) Let $\lceil \log_2 N_m \rceil$ and $\lceil \log_2 N \rceil$ be blocks and $\lceil \log_2 N \rceil = 4$ and $ID(0) = 0000$ be the block indexes.

(D) Make the File Index Table (as shown in Table 3), which includes the filenames $IP(n)$ of message $n = 1,2,3,...$, time $IP(n)$, and the index $IP(n)$.

(2) Obtain blocks from indexes

(A) Find file name $IP(n)$ and its file index $n = 5 \sim 12$ from the File Index Table.

(B) Let $ID(n)$ be the index of the block $IP(n)$. Retrieve data block $SID(node_i) = \begin{cases} ID_i(n) & ,\text{if } IP_i(n) \neq \text{Null} \\ SID(\text{node}_{i-1}), & \text{if } IP_i(n) = \text{Null} \end{cases}$ and determine whether $BID(node_i) = \begin{cases} SID(\text{node}_{i-1}), & \text{if } IP_i(n) \neq \text{Null} \\ BID(\text{node}_{i-1}), & \text{if } IP_i(n) = \text{Null} \end{cases}$ is equal to $ID(1)$. If not, there is a problem with this block; find the block backup. Otherwise, let $j = 1$.

(C) Let $h(c_j^{(y)})$ of $indx^{(y)}$ be the data block $h(c_j) \| c_j$. Retrieve data block $h(c_j^{(y)}) \| c_j'^{(y)}$ and determine whether $h(c_j^{(y)})$ is equal to $h(c_j'^{(y)})$. If not, there is a problem with this block; find the block backup.

(D) $j \leftarrow j+1$. Repeat step 3 until $h(c_j^{(y)}) \| c_j'^{(y)} \| end$ is found.

3.3. Balance Ring Management

In this paper, we proposed a peer distribution method, called Balance Ring, where the most important point is the building of a Node Table, which records the information of each node's (1)ID, (2) IP or Domain Name, (3) Successor ID and (4) Backer ID. Each data will be stored in the Successor of each node. If needed, important data will be backup in the Backer of each node.

A physical peer occupies at least one peer node and is responsible for storing the blocks which are distributed to the peer node. The steps of building node table are shown as follows.

(1) Decide the ID value

Before building node table, we have to obtain the number of peer node N or estimate the maximum number of peer node Nm. Take the value N or Nm to the pseudo-code in Figure 2 to decide the ID value of each node, where the bit length of node ID is $\lceil \log_2 N \rceil$ or $\lceil \log_2 N_m \rceil$. After that, fill in the value in the ID field of node table by the sequence of ID.

For example, assume $\lceil \log_2 N \rceil$ is 4. Take $\lceil \log_2 N \rceil = 4$ to the pseudo-code in Figure 4 and get the balance ring of Figure 5, where 0000(1) means that the node is the first node and its ID is 0000. It is recorded as $ID(0) = 0000$.

(2) Fill in IP or Domain Name

Fill in IP or Domain Name in the field $IP(n)$ in node table by following the peer node number of each physical peer and the sequence $n = 1, 2, 3, \ldots$. Fill in "Null" in the field $IP(n)$ which does not fill in values.

For example (Table 4), assume there are 6 physical peers to store files. The first physical peer has four peer nodes and its ID is 163.97.25.33. Then Fill in 163.97.25.33 in the filed $IP(n)$, where n = 1 ~ 4. Assume another five physical peer have (3, 2, 1, 1, 1) peer node separately and their IP are (61.219.38.220, 140.96.111.39, 239.27.74.104, 196.190.112.77, 209.59.137.56). We can fill in the corresponding IP value in the field $IP(n)$, where $n = 5 \sim 12$.

```
get_every_ID(N)
    ID(1) = 0;
    ID(2) = 1;
    for (j=1; j ≤ ⌈logN⌉-1; j++){
        for (i=2^j+1; i ≤ 2^(j+1); i++) {
            ID(i-2^j) = ID(i-2^j) *2;
            ID(i) = ID(i-2^j)+1;
        }
    }
```

Figure 4. Obtaining the Pseudo-Code of ID

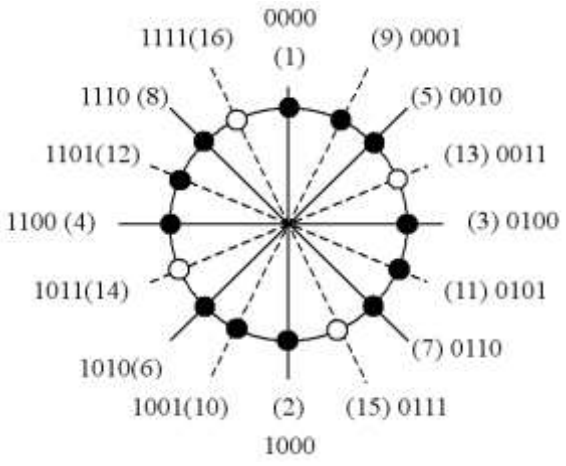

Figure 5. Illustration of Balance Ring

Table 4. Node Table

n	$ID(n)$	$IP(n)$ **(IP or DN)**	**Successor ID**	**Backer ID**
1	0000	163.97.25.33	0000	1110
9	0001	140.96.111.39	0001	0000
5	0010	61.219.38.220	0010	0001
13	0011	Null	0010	0001
3	0100	163.97.25.33	0100	0010
11	0101	196.190.112.77	0101	0100
7	0110	61.219.38.220	0110	0101
15	0111	Null	0110	0101
2	1000	163.97.25.33	1000	0110
10	1001	239.27.74.104	1001	1000
6	1010	61.219.38.220	1010	1001
14	1011	Null	1010	1001
4	1100	163.97.25.33	1100	1010
12	1101	209.59.137.56	1101	1100
8	1110	140.96.111.39	1110	1101
16	1111	Null	1110	1101

(3) Successor ID

If a block is distributed to one node, the successor of the node is responsible for the block. The Successor ID of the node decides the storage address of the block. Of course, if there exist a peer node in one node and it is ready to store blocks, then the Successor ID of the node is the node itself. If there is not a peer node exist in one node, which means the filed IP of the node is Null, then the Successor ID of the node is the previous ID in which the field ID is not Null. (If we cannot find such node through the top of the node, we can find if from the bottom of the table.)

$$SID(node_i) = \begin{cases} ID_i(n) & \text{,if } IP_i(n) \neq \text{Null} \\ SID(\text{node}_{i-1}) & \text{,if } IP_i(n) = \text{Null} \end{cases} \quad (1)$$

For example, in Table 4, the Successor ID of a node, in which the field IP is not Null, equals to the ID itself. The Successor ID of a node, in which the field IP is Null, equals to the previous ID. Also in Figure 5, the Successor ID of a black node, which stand for there is a peer in the node, equals to the ID itself. The Successor ID of a node, which stand for there is not a peer in the node, equals to the first black ID counterclockwise.

(4) Backer ID

If a block is very important, it is needed to be duplicated to prevent the damage of the Physical Peer or the block. Then the block is needed to be duplicated to the Backer of the node. The Backer ID of a node equals to last ID of which the filed IP is neither Null nor the ID of its Successor IP.

$$BID(node_i) = \begin{cases} SID(\text{node}_{i-1}) & \text{,if } IP_i(n) \neq \text{Null} \\ BID(\text{node}_{i-1}) & \text{,if } IP_i(n) = \text{Null} \end{cases} \quad (2)$$

For example, in Table 4, the Backer ID of the ID (0101, 1100, 1111) is (0100, 1010, 1110). As shown in Figure 5, the Backer of a black node A is the first counterclockwise black node B of which the IP is not that of node A.

The Note Table is very important and is suggested to be put on not only the Client node but also the Successor and Backer of $ID(1)$. It can be encrypted if needed.

3.4. Block Distribution and Retrieval

If a file is going to be stored on our system, data blocks have to be distributed and stored in the Peer node after file encryption and segmentation. On the contrary, if a file is going to be retried from our system, data blocks have to be found from peer nodes and the file has to be synchronized, decrypted and verified after getting data blocks. The methods of block distribution and retrieval are described as follows.

(1) Block distribution and storage

After file encryption and segmentation, we will get data blocks such as $\{h(c_i) \| c_i\}$ and $\{I \| indx\}$, where $h(c_i)$ and I are the index of blocks. Next, let $SID(index_i) = left_l(index_i)$ and $SID(index_{file}) = left_l(I)$, where $index_i = h(c_i)$. We take the former l bits of each block to map the ID of node table, where l is the bit length of an ID. Then, store each block on the Successor of the ID. If the block is important, it is duplicated on the Backer of the ID.

(2) Block search and retrieval

If a file block is going to be retrieved, the file name and the file Index I have to be found form File Index Table. Next, take the former l bits of I to map the ID of node table. Then, use the Successor of the ID to fine data block $\{I \| indx\}$. After that, use indx = $h(c_1) \| h(c_2) \| ... \| h(c_k)$ to get $h(c_i)$ and then use former l bits of $h(c_i)$ to map the ID in the Node Table. From the Successor the ID, $\{h(c_i) \| c_i\}$ can be found. If a data block can not be found or an error is occurred, we can try to find the Backer of the mapped ID and check whether there is a duplicated block.

Besides, if a file is going to be deleted, the steps are the same as block search. After the block is found, delete the block directly and delete the file data from the File Index Table. If a file is going to be renewed, the action of "block search and retrieval" can be executed first. After the decision of file renewed, delete the old file and make the new file to execute the action of "block distribution and storage."

3.5. Peer Changing

After the decision of the number of Physical Peer (larger then 2) and the number of Peer Node, the system can be started to build. The peer node address of the system is not changed basically. However, the addition, departure and damage of peers will affect the peer node address. Then the node table has to be adjusted.

3.5.1. Peer Join

“Peer join” means the client can store or retrieve data in the duration of adding peers. If a peer just adds and leaves, then there is not any influence for the system. The action of peer join is not needed.

There are two kinds of peer join. One is that the number of added peer node is not exceed the number of Null IP (*i.e.* IP = Null) in Node Table. Then the added peer node can be put on the place of Null IP. The other is that the number of added peer node is more than the content of the remained null place in peer node. Then the space of Node Table has to be increased and the ID bit length l has to be added. Generally, if the estimated peer node number N_m is correct, then this case will not happen.

Case 1: The number of added Peer node is smaller than or equal to the number of Null IP.

In this case, the steps of peer addition are listed as follows. (Let SN_A denotes the added peer node.)

(1) Find the ID of added peer node

Let SN_A, IP_A and s denote the added peer node, the IP of SN_A and the added peer node number. Put s IP_A in the $IP(n)$ that the value n is small, where $n = 1, 2, 3, \ldots$. (For example in Table 4, put IP = 211.78.38.10 in $IP(13)$.) The pseudo code of adding peer node is shown as Figure 6.

(2) Adjust the Successor ID and Backer ID of Node Table

The steps are listed as follows.

(A) Change the Successor ID of SN_A

Change the Successor ID of SN_A to the ID itself. For instance, $SID(13) = 0011$.

(B) Change the Successor ID of other nodes

Change the Successor ID of the entire white node (*i.e.* ID = Null) between the node SN_A and its first clockwise black node (*i.e.* ID ≠ Null) to the ID of SN_A.

(C) Change the Backer ID of added peer nodes

Let BID_Z denotes the original Backer ID of SN_A. Change the Backer ID of SN_A to the ID of its first counterclockwise node of which the IP is neither Null nor the IP itself. For instance, Backer $BID(13) = 0010$.

(D) Change the Backer ID of other nodes

Find node SN_A's first clockwise black node of which the IP is not the IP itself. Let the node ID be ID_B. Change the Backer ID BID_B of ID_B to ID_A. Change the Backer ID BID_Z of the entire ID_v between the black node ID_B and the node SN_A to BID_A, where $v = 1, 2, 3, \ldots$.

(3) Store Successor blocks

After adding a physical peer and adjusting the Successor ID and Backer ID of Node Table, the entire added peer nodes execute the following steps to store the responsible data blocks.

(A) Let n_A, ID_A, IP_A, SID_A and BID_A denote the value n, ID, IP, Peer ID and Backer ID of node SN_A. Record the entire white node ID between node SN_A and its first clockwise black node. Let ID_u^w denote the ID of the white node, $u = 1, 2, 3, \ldots$.

(B) From the node SN_A's first counterclockwise black node of which the IP is not IP_A, find the entire blocks that the former l bit of Index is ID_A or ID_u^w. Move them to the address of IP_A.

```
for (i = 1, i ≤ s ,i++) {
        n = 1;
        While ( IP(n) ≠ Null) {n++;}
        IP(n) = IP_A;
}
```

Figure 6. The Pseudo-Code of Adding Peer Node

```
Add_ID(l) {
        l++;
        j = l – 1;
        for (i=2^j+1; i ≤ 2^(j+1); i++) {
                ID(i-2^j) = ID(i-2^j) *2;
                ID(i) = ID(i-2^j)+1;
        }
}
```

Figure 7. The Pseudo-Code for Increasing the Number of ID

(4) Store Backer blocks

After adding a physical peer and adjusting the Successor ID and Backer ID of Node Table, the entire added peer nodes execute the following steps to store the responsible data blocks.

(A) From the data base of BID_Z, move the entire blocks that the former l bit of Index is ID_A or ID_v to BID_A, where $v = 1,2,3,\ldots$.

(B) From the data base of BID_B, move the entire blocks that the former l bit of Index is ID_B to ID_A.

Case 2: The number of added Peer node is larger than the number of Null IP.

In this case, the steps of peer addition is the same as the steps in (1) except some steps of "note table extension and being renew." The steps are listed as follows.

(1) Increase the number and the bit number of ID

Take the bit length value l of the original ID to the pseudo code in Figure 7.

(2) Change the contents of Node Table

Double the value of Successor ID and Backer ID in Note Table. (*i.e.* Increase a bit "0" in the latter bit.) Let the IP value be Null in the added node and rearrange the Successor ID and Backer ID.

(3) Find the ID of added peer node (which is the same as (1)-(a))

(4) Adjust the Successor ID and Backer ID of Node Table (which is the same as (1)-(b))

(5) Store Successor blocks (which is the same as (1)-(c))

(6) Store Backer blocks (which is the same as (1)-(d))

3.5.2. Peer Leave

"Peer leave" means it will influence the host or handset of client to access data in the duration of adding departure. If a peer just leaves and joins, there is not any influence for the system and the action of peer leave is not needed. The steps of peer departure are listed as follows.

(1) Record the data of Successor ID and Backer ID

Let SN_L, $(n_L, ID_L, IP_L, SID_L, BID_L)$, and $ID_i^{S:L} / ID_j^{B:L}$ denote the departure peer node, (the value n, ID, IP, Peer ID, Backer ID) of SN_L, and the node of which the Successor/Backer ID value is ID_L, $i = 1,2,3,\ldots, j = 1,2,3,\ldots$.

(2) Renew the Node Table

Change IP_L to Null. Renew all the Successor ID and Backer ID in Node Table.

(3) Backup the data of old peer Backer to new Backer

In the file peer BID_L, backup the entire block of which the Index point to ID_L and $ID_i^{S:L}$, $i = 1,2,3,...$ to the new Backer of BID_L.

(4) Move the Successor data

In the file peer IP_L, move the entire block of which the Index point to ID_L and $ID_i^{S:L}$, $i = 1,2,3,...$ to BID_L.

(5) Move the Backer data

In the file peer IP_L, move the entire block of which the Index point to $ID_j^{B:L}$ to the new Backer of $ID_j^{B:L}$, $j = 1,2,3,....$

3.5.3. Peer Damage

"Peer damage" means a peer is unavailable or unrecoverable in the unexpected situation. If it can be known in advance or it is felt to be broken, the steps of Peer departure should be executed immediately. The steps of peer damage are listed as follows.

(1) Record the data of Successor ID and Backer ID

Let n_D, ID_D, IP_D, SID_D and BID_D denote the value n, ID, IP, Peer ID and Backer ID of the damaged peer node and $ID_i^{S:L} / ID_j^{B:L}$ stand for the node of which the Successor/ Backer ID value is ID_D, $i = 1,2,3,..., j = 1,2,3,....$

(2) Renew the Node Table

Change IP_D to Null. Renew all the Successor ID and Backer ID in Node Table.

(3) Backup the data of old peer Backer to new Backer.

In the file peer BID_D, backup the entire block of which the Index point to ID_D and $ID_i^{S:L}$, $i = 1,2,3,...$ to the new Backer of BID_D.

(4) Backup the Backer data

In the peer node $ID_j^{B:L}$ of which the IP is not Null, duplicate the entire (important) block of which the Index point to $ID_j^{B:L}$', $j' = 1,2,3,...$ to the new Backer of $ID_j^{B:L}$, $j = 1,2,3,....$

4. Analysis of System and Security

This section discusses some characteristics of the proposed system, including the scalability, the security, the efficiency, and the properties.

4.1. System Scalability

In the system of balance ring, the ID length of each peer node is $O(\log N)$ (*i.e.* $\lceil \log_2 N \rceil$ or $\lceil \log_2 N_m \rceil$). The ID length of peer node occupies the minimum resource theoretically. If the physical peer is added so that the value N or N_m is needed to add, the ID length is still $\lceil \log_2 N \rceil$ or $\lceil \log_2 N_m \rceil$, which is still the minimum source theoretically. Hence when the physical peer adds, the system will not get much overload. Therefore the system is scalable.

4.2. System Security

In our system, we consider the security problem including the data privacy and integrity to prevent data being stolen, modified or destroyed. If a data is stolen, the attacker cannot obtain any information about the plaintext sine the data is encrypted. If the data is modified, file owners can detect the modification from the inconsistent hash value and, in advance, use the backup files. If attacker destroy or even delete a block, we can find the duplicated block from the backer of the block to get the destroyed or deleted block. Thus the system affords secrecy, integrity and robustness.

4.3. System Efficiency

In this system, we use only symmetric crypto system and one-way hash function. Compared with CFS, the speed of the whole system is promoted since the asymmetric crypto system is not used. Therefore, the system is efficient.

4.4. System Properties

The system is reliant because it can retrieve and redistribute all important data after a single peer suddenly terminating. It is also resilient since we can use any computer device to save our invaluable data by getting the backup data and retrieving all the files from nodes if our computer devices, such as PC or mobile phone, destroyed unexpectedly. Moreover, the system is scalable since it considers peer's join, leave, and damage.

5. Conclusion

For solving security issues for P2P Cloud system, we have presented a secure and balanced storage system. The system offers load balance, secrecy, integrity, and robustness for data protection. It is efficient, reliant, resilient and scalable. The security analysis and data collision are also discussed.

Acknowledgments

This work is partially supported by the Ministry of Science and Technology under Grant MOST 104-2221-E-182-012 and by the CGMH project under Grant BMRPB46. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] Napster Inc., napster website, [Online]. Available: http://www.napster.com
[2] Gnutella, Gnutella website, [Online]. Available: http://www.gnutellaforums.com/
[3] D. Cenk Erdil, "Simulating peer-to-peer cloud resource scheduling", Peer-to-Peer Networking and Applications, vol. 5, no. 3, **(2012)**, pp. 219-230.
[4] C. Wang, Q. Wang and K. Ren, "Ensuring data storage security in cloud computing", Cryptology ePrintArchive, Report [Online]. Available: http://eprint.iacr.org/2009.
[5] H. Jiang and X. Shao, "Detecting P2P botnets by discovering flow dependency in C&C traffic", Peer-

to-Peer Networking and Applications, vol. 7, no. 4, **(2014)**, pp. 320-331.
[6] S.H. Lee and I.Y. Lee, "Secure Index Management Scheme on Cloud Storage Environment", International Journal of Security and Its Applications, vol. 6, no. 3, **(2012)**, pp. 75-82.
[7] S.H. Lee and I.Y. Lee, "Keyword Searchable Re-encryption Scheme Considering Cloud Storage-Service Environment", Information-An International Interdisciplinary Journal, vol. 15, no. 5, **(2012)**, pp. 2135-2146.
[8] G. Brunette and R. Mogull, "Security guidance for critical areas of focus in cloud computing v2.1", Cloud Security Alliance, **(2009)**, pp. 1-76.
[9] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", Journal of Network and Computer Applications, vol. 34, no. 1, **(2011)**, pp. 1-11.
[10] S. Y. Chiou, "A Secure Cloud Storage System with Privacy, Integrity and Authentication", ICIC express letters. Part B, Applications, vol. 5, no. 3, **(2014)**, pp. 843-849.

Author

Shin-Yan Chiou, he received the PhD degree in Electrical Engineering from National Cheng Kung University, Taiwan, in 2004. From 2004 to 2009, he worked at Industrial Technology Research Institute as a RD Engineer. Since 2009, he joined the faculty of the Department of Electrical Engineering, Chang Gung University, Taoyuan, Taiwan, where he is currently an Associate Professor. He has published a number of journal and conference papers in the areas of information security, social network security and mobile security. His research interests include information security, cryptography, social network security, and secure applications between mobile devices.

