Searchable Re-encryption Cloud Storage Method Based on Markov Chain

Wang Hui[a], Wang ZhongSheng[b]
School of Computer Science and Engineering,
Xi'an Technological University,
Xi'an, 710021, China
e-mail: [a]277019826@qq.com;
[b]59483672@qq.com

Li Jinguang
Department of Information Technology
Shaanxi Heavy Duty Automobile CO.LTD
Xi 'an, 710200, Shaanxi, China

***Abstract*—Cloud storage is an emerging paradigm that offers on-demand, flexible, and elastic computational and storage services for the terminal users. When the large amount of data increases dramatically, the storage efficiency of the system would be decreased seriously. In this paper a new method of SReCSM(Searchable Re-encryption Cloud Storage Method) based on Markov chain is proposed. It predicts periodically by using the steady Markov strategy in stages and easy to select the optimal storage node. The data is scheduled to store in the node with the lowest cost in real time, and the node is selected to implement cloud storage access on mobile terminal. By using searchable re-encryption method, SReCSM has increased the storage requirement flexibility and minimize cost and searching time. And then the reliability model of SReCSM is established. Simulation results show that SReCSM introduced in this paper has the ability to predict accurately when the size of the data is different. Moreover, the influence of storage efficiency is reduced effectively through SReCSM when different size of the data is stored in storage nodes regardless of the storage cost. It is verified that the SReCSM based on Markov chain has higher reliability.**

Keywords-Cloud Storage; Markov Chain; Re-encryption; Reliability Model

I. INTRODUCTION

In order to meet the various storage demands, cloud storage is designed to store data in cloud and is widely used in the Internet. Compared with traditional data storage, it greatly improves the efficiency of the mass data storage and utilization of network resource. However, access from a mobile device to data, stored in a cloud, leads to poor client quality experience [1-3]. It is essential to reduce user download wait time for a requested file from a network to enhance client quality experience. As a result, cloud storage techniques are quite challenging. On one hand, storage density of the cloud storage is not big and the comprehensive storage efficiency is low. On the other hand, the high latency limited the use of mobile cloud storage, especially for the applications with frequent random accesses to a large set of small files. Therefore, cloud storage faces serious security and efficient problems [4-6].

Traditional cloud storage systems do not adapt well to different application environment and does not guarantee the integrity and confidentiality of cloud data. In other words, the cloud storage service does not guarantee that the data and operation of mobile users will not be lost, damaged, leaked, or illegally exploited by malicious or nonmalicious. Therefore, it's very dangerous for sensitive data to be stored directly in the cloud. The reliability of the mobile cloud storage depends on the extent of the impact on system storage efficiency while the storage solution fails [7]. Therefore, storing sensitive data on untrusted server is a challenging issue [8]. Simple encryption techniques have key management issues and which can't support complex requirements such as query, parallel modification, and fine-grained authorization. To guarantee confidentiality and proper access control of sensitive data, classical encryption are used [9-10].

To solve the problems brought by the hysteretic and density of tranditional storage methods in the cloud storage system, in this paper, SReCSM, Searchable Re-encryption

DOI: 10.21307/ijanmc-2019-004

Cloud Storage Method is proposed using the idea of Markov chain. In this method, instead of using traditional storage, a proactive storage approach is adopted to avoid delayed storage. In addition, the SReCSM predicts periodically to increase the effiency of cloud storage. The major contribution of our work includes:

- Searchable Re-encryption Cloud Storage Method based on Markov chain is proposed and built as a SReCSM model. In order to find out the optimal storage policy in the cloud, an optimization problem is formulated as a Markov chain. The SReCSM model is designed to predict periodically by using the steady Markov strategy in stages and determine the lowest storage cost by which the storage benefit can be maximized.
- A value iteration algorithm of SReCSM is presented for computing the stationary deterministic policy. The data is scheduled to be stored in the node with the lowest storage cost in real time. Therefore, the optimal storage node must be selected to implement cloud storage access on mobile terminal.
- The central idea of the searchable re-encryption is proposed, which is to generate a re-encryption key and decrypt while the keyword matched. The objective of the proposed searchable re-encryption method is to increase the storage requirement, flexibility and reduce the security issues, overhead ratio and minimize the cost and searching time.
- The reliability of SReCSM model is proposed and analyzed. Because of the large scale of the cloud storage system, method of Monte Carlo simulation is adopted in the reliability evaluation of the cloud storage system.
- We conduct a series of experiment to validate the effectiveness, performance and robustness of SReCSM. Results show that the present approach can store mass data in cloud system effectively and securitily.

The rest of the paper is organized as follows. In section 2, related work is discussed. Section 3, Searchable Re-encrypption Cloud Storage Method based on Markov Chain is proposed. Section 4 describes the searchable re-encryption method in SReCSM. Section 5 presents the basic prototype system of SReCSM and simulation experiments. In addition, the security, performance and robustness of SReCSM are analyzed. Section 6 concludes the paper.

II. RELATED WORK

Efforts have been taken by researchers, developers, practitioners, and educators to identify and discuss the technical challenges and recent advances related to cloud storage. Han et al. proposed [11] multi-path data prefetching in mobile cloud storage. Multi-Path prefetching tend to prefetch more successors to ensure high prefetching hit rate and efficiency, and achieve a higher overall throughput performance of accessing cloud storage services via mobile network. Based on cloud storage, Lee et al. [12] designed an efficient delta synchronization (EDS) algorithm for mobile cloud storage applications, which can aggregates the updated data to reduce the synchronization traffic and synchronizes the aggregate done periodically to satisfy the consistency. Wang et al. [13] presented an Optimized Replica Distribution Method (ORDM) in cloud storage system. Zhang et al. [14] conducts modeling analysis of a cloud storage system, and proposes a Markov decision process modeling framework to analyze the reliability of the storage system. Chen et al. [15] proposed a new metric called joint response time, which not only considers the waiting time when the requested data are unavailable but also the queuing delay and service time when data become available. These methods mentioned above achieve cloud storage, but the cloud storage according to data size based on Markov is not considered. Aiming at the problem that the cloud storage according to data size based on Markov, SReCSM is proposed in this paper to satisfy the cloud storage periodically.

Most of the existing security schemes that are designed for mobile cloud storage environment are based on traditional cryptographic methods or pairing-based cryptography. The further deployment of cloud storage is limited by its security risks. The schemes presented in

[17-19], and [20, 25] provided the security features for mobile user in the cloud environment using the traditional cryptography methods. The rest of the security schemes discussed in [21-23], and [24] are based on pairing based cryptography for secure offloading of the data access operations on the cloud in a trusted mode.

Few of the schemes presented in the classification execute the entire security operations on the mobile device. The rest of the schemes delegate the security management operations on the cloud, trusted entity under the control of client organization, or trusted entity ender the control of the third party. In the category of local execution, a small number of schemes focus on the reduction of computational complexity of cryptography algorithms for reducing the processing burden from the mobile device.

Zhao et al. [17] proposed a method for trusted data sharing on untrused cloud servers using Progressive Elliptic Curve Encryption(PECE) scheme. The PECE allows the encryption of message multiple times with different keys that can be decrypted in a single run with a single key. Ren et al. [20] provide the security features for mobile user in the cloud environment using the traditional cryptography methods. The focus of the proposed schemes is to reduce the computational complexity of the cryptography operations instead of offloading the computationally intensive operations on the cloud. The size of the ciphertext grows linearly with increase in number of ciphertext attributes. [26] that involves more pairing evaluation exponential operations while decrypting the ciphertext. However, the data owner has to transform the plaintext into ciphertext, and vice versa that involves execution of computationally intensive multiplication and exponential operations of large numbers on resouce constraion mobile device.

[27-31] propose proxy re-encryption algorithm to confirm the security of the data in the cloud, which can alleviate the client's burden, and enhance the confidentiality of cloud data. However, there are two major disadvantages with these techniques. First, for re-encryotion, the data owner must obtain user's public key before uploading. Second, because the same plaintext is used with different keys generated by proxy, therefore, the storage overhead becomes excessive.

A manager-based re-encryption scheme (MReS) defined in [16] achieves the data confidentiality for the mobile users by using the proxy re-encryption strategy. The proxy re-encryption helps the mobile user to delegate the data access operation on third party. Another cryptographic scheme defined by Tysowski and Hasan in [16] is cloud-based re-encryption scheme (CReS). The proposed scheme covers the limitations of the existing manager-based re-encryption scheme and the variations of manager-based re-encryption based on user-managed key ciphertext fetch by user. But these two schemes are more complex and need more time to re-encrypt. To optimize the cloud storage, safety transmission, minimize the cost and computational complexity, here we have proposed a new scheme as searchable re-encrypted data in SReCSM. This technique is more simple and faster when sorting the arrays using the most significant radix sort. This technique will reduce the cost of the data owners up to O(Nt*3) and the time complexity will be reduced up to the O (B), where B denotes the Bucket size of the data base.

III. CONSTRUCTION OF SRECSM MODEL

The transfer probability of Markov chain is introduced into the cloud storage. The stored procedure in the cloud system is similar to the Markov process. Therefore, a Searchable Re-encryption Cloud Storage Method based on Markov Chain is proposed in this paper, which can realize reliable and safe storage on mobile terminal.

From the probability point of view, the data is scheduled to be stored in the node with the lowest storage cost in real time. Firstly, the storage state of SReCSM is divided and three state models are obtained in the paper. According to the storage cost of the nodes for storing different size file, the state matrix in current time is got by using the steady state of Markov strategy. The state transfer probability matrix is calculated with the support of a sufficient number of samples. The storage state probability of SReCSM can be predicted quickly in the future. Therefore, the optimal

storage node is selected to implement cloud storage access on mobile terminal.

When a request is made by a mobile terminal, the storage node which has the lowest cost is selected for storing. The SReCSM can save time and improve storage efficiency effectively. And at the same time the load balancing of the system is also guaranteed.

A. The finite state space and the state distribution

The Markov chain is a discrete random process with Markov properties. The next state of the random process is only related to the current state, not its historical state. Therefore, the distribution of the future state of the random process depends only on the present, not the past. The SReCSM based on Markov chains consists of the following three parts $\{S, R, P\}$. In which S is the finite state space; R stands for the state distribution; P is the probability of state conversion.

The server in the cloud storage group is named storage node. Each node can store large files, medium files, and small files. Files stored over 100MB is called a large file, files no more than10MB are defined as small files, files between 10MB and 100MB are defined as secondary files. The storage nodes are different in hardware performance, load situation, network link state and so on. Therefore, each storage node has different storage cost for storing large data, secondary data, and small data. That is, the storage cost for storing different size of files in each node is different. In the cloud system, depending on the size of the storage data file, each node has different storage costs for large, secondary and small files, and then the optimal storage node is selected accordingly.

Based on the analysis above, the SReCSM based on Markov Chain is proposed in this paper. The processes of the storage that store three different sizes of data in a storage cluster are treated as space S which is discrete state. Moreover, in the procedures of data storing, the selection of the next storage node is only related to the storage cost of the current storage node. Therefore, the data stored procedure of cloud system conforms to the characteristics of Markov chain. In the SReCSM proposed in this paper, the state space S is expressed as the following three types.

State 1 (Small file status): In the state of small file status, the optimal storage node for storing small files in a storage group cluster is selected to implement data storage.

State 2 (Secondary file status): In the state of secondary file status, the optimal storage node for storing secondary files in a storage group cluster is selected to implement data storage. Secondary files are between a small file and a large file.

State 3 (Large file status): In the state of large file status, the optimal storage node for storing large files in a storage group cluster is selected to implement data storage.

The state space of the cloud storage based on Markov chain is defined in formula (1).

$$S = \{s_1, s_2, s_3\} \tag{1}$$

In formula (1), s_1 denote the storage cost for storing small file on each storage node in the cluster, s_2 denote the storage cost for storing secondary file on each storage node in the cluster, and s_3 denote the storage cost for storing large file storage on each storage node in the cluster.

Moreover, according to the probability rule, the more detailed the storage state of the cloud storage system is, the more explicit the guidance of the data storage solution prediction will be. However, the greater the burden of computing resources needed for partitioning the state of the cloud system storage scheme. As we can see, in implementing the choice of cloud storage solutions, the degree of partitioning the storage state in cloud system storage scheme is a compromise decision problem. The more detailed the storage state is the more simplification is needed to reduce the computational burden. And the more storage states are, the greater the redundancy of the storage scheme. Therefore, the approach of partitioning the storage state not only complicates the analysis, but also reduces

storage speed. In fact, storage status can't be considered completely, and it is difficult for the SReCSM to achieve multiple storage state ultimately. Therefore, in the study of the SReCSM, the partitioning of the storage state is appropriate. Obviously, the storage state defined in this paper can be divided into three state, those are large file status, middle file status and small file status. Whether for the actual data of the project or the simulation analysis, these three states are relatively simple and can better satisfy the requirement of data storage in cloud system.

The storage group has an irregular connection to the network. The distribution matrix of the storage state in the cloud system is defined as R. And each data in the matrix is the storage cost on a certain storage node for data storing. Therefore, according to the storage cost of the nodes in large, medium and small three different states, the distribution matrix $R(i)$ of the storage status at time t_i can be obtained. The matrix $R(i)$ is to be defined in formula (2).

$$R(i)=\begin{bmatrix} s_1(i) \\ s_2(i) \\ s_3(i) \end{bmatrix}^T = \begin{bmatrix} R[X(t_i)=s_1] \\ R[X(t_i)=s_2] \\ R[X(t_i)=s_3] \end{bmatrix}^T \tag{2}$$

B. The probability matrix of the state transfer

The number of states in the state space is denoted with n. Therefore, the transfer probability can be represented in the form of matrix, which is defined as the probability matrix of state transfer. The probability matrix P of the state transfer always keeps the same. Moreover, and P is the matrix of order n. If the interval among $t_1, t_2, \cdots, t_n$ are always the same, according to homogeneity, P is defined in (3).

$$P=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \tag{3}$$

Obviously, the transfer probability matrix has two properties, which are shown in formula (4).

$$\begin{cases} p_{ij}(\Delta t)\geq 0 & i,j=1,2,\cdots,n \\ \sum_{j=1}^{n} p_{ij}(\Delta t)=1 & i=1,2,\cdots,n \end{cases} \tag{4}$$

In equation (4), p_{ij} refers to the probability of transition from the state s_i to the other state s_j. The sum of each row in the probability matrix is one. Moreover, the sum of the probabilities from the storage state s_i to all the other storage states s_j is one too. According to the analysis previous, each storage node in a storage group has different storage costs for storing three different size files. Therefore, in the cloud system, if a large, medium, and small files are need to be stored in the storage node, the higher the cost of integrated storage, the less likely this node is used to store data. On the contrary, the smaller the storage cost of the node for storing large, medium and small files, the more likely the node will be selected. The state space of SReCSM is represented in (1). The storage status transfer can be shown in figure 1.

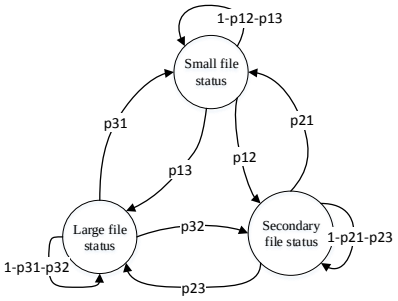

Figure 1. Storage status transition diagram

According to the probability values of the transformation in three different states of the storage node, the probability matrix of the state transfer in the SReCSM is shown in formula (5).

$$P=\begin{bmatrix} 1-p_{12}-p_{13} & p_{12} & p_{13} \\ p_{21} & 1-p_{21}-p_{23} & p_{23} \\ p_{31} & p_{32} & 1-p_{31}-p_{32} \end{bmatrix} \quad (5)$$

C. Algorithm of SReCSM

Users access the network through a mobile terminal, and the mobile terminal will choose to communicate with the node which has the lowest cost in the storage group. Moreover, irregular connection is used in the storage group. Therefore, in order to improve the reliability of the cloud storage system, a multi-replica mechanism is introduced. The storage efficiency of the cloud system can be effectively improved by using Multi-replica mechanism. When a mobile terminal makes a request, a copy with the lowest cost can be chosen. However, in the traditional cloud storage system, the master copy is selected first only when the master copy is wrong. The request will then be sent to the backup copy regardless of the region. Therefore, this kind of storage will affect the speed of the cloud system. Based on this idea, the transfer probability of Markov chain is used in the SReCSM. The storage cost of each storage node is measured from the perspective of probability. The state matrix of one-step transfer will then be obtained by means of the probability matrix of the state transfer. Finally, the state matrix of multi-step transfer will be obtained to predict the probability of the storage cost at a certain time in the SReCSM. Therefore, the optimal storage node will be chosen to realize the data storage at the end.

The main idea of the SReCSM algorithm in the cloud system will be introduced in the following chapter. The states of the storage in the strategy are proposed for the implementation of the data storage. Moreover, in the SReCSM algorithm of the cloud system, the standard of storage status division is proposed, which is convenient for engineering implementation. At the same time, the transfer expression of the state corresponding to the storage state is obtained. Then, on the base of the historical data stored in the cloud system, the probability matrix of state transfer is obtained. The probability and reliability of the storage status in the cloud system at some time in the future are predicted by using the probability matrix of the state transfer and the matrix of the storage state at the moment.

The steps of the algorithm of the SReCSM based on Markov chain are described as follows:

Step 1

The distribution matrix $R(0)$ of the SReCSM at the initial time t_0 is obtained.

Step 2

According to the distribution matrix of the storage state and the probability matrix P of the state transition at t_0 time, the state distribution matrix of one-step transfer $R(1)$ at next time will be got after Δt time. The state distribution matrix of one-step transfer $R(1)$ is shown in formula (6).

$$R(1)=R(0)P \quad (6)$$

Step 3

Obviously, the storage distribution of the cloud storage system after Δt time is shown in formula (7).

$$R(2)=R(1)P=R(0)P^2 \quad (7)$$

After a period of time, about m times Δt time, the storage status of cloud system is shown in formula (8).

$$R(m)=R(m-1)P=R(0)P^m \quad (8)$$

Through the procedure above, $R(0)$ and P are known, the steady-state value of the cloud storage distribution in the system can be predicted quickly in the future after every Δt time interval.

According to the storage cost of data stored in different nodes, the storage cost matrix of the current time of the system is obtained, which is the state distribution matrix of the system. The state distribution matrix is shown in formula (9).

$$R(i)=\begin{bmatrix} s_1(i) \\ s_2(i) \\ s_3(i) \end{bmatrix}^T = \begin{bmatrix} R[X(t_i)=s_1] \\ R[X(t_i)=s_2] \\ R[X(t_i)=s_3] \end{bmatrix}^T \quad (9)$$

In formula (9), s_i represents the storage cost needed to select a storage node at time t_i. We can know from formula (9) that s_i is contained within the range of 0 to 1, which satisfies the inclusion relationship $s_i \in [0,1]$. And s_i is obtained by normalizing the correlation property, which is stored in the node at t_i time. There are three variables in related properties at time t. Three variables are represented by symbols q, d and l, which respectively represent the length, latency and packet loss of stored data respectively. The storage cost of the node will be got in formula (10).

$$s = \beta_q s_q + \beta_d s_d + \beta_l s_l \quad (10)$$

In formula (10), three variables q, d and l satisfies the relationship in formula (11).

$$\sum_{k=\{q,d,l\}} \beta_k = 1 \quad (11)$$

Because parameters including q, d and l are all cost indicators. Therefore, the smaller the value, the better the quality of the data stored in this node. The storage cost will be got in formula (12) in this way.

$$s_k = \frac{k_{\max} - k}{k_{\max} - k_{\min}} \left(k = \{q, d, l\}\right) \quad (12)$$

In formula (11), q is obtained by the length of the stored data block. The parameters d and l are obtained by timing first, and then calculated through the time. Start the timer when each data block is stored at the beginning of the storage till the response is received. Therefore, the evaluation of the storage quality in each storage node is carried out through the above method. Moreover, and the distribution matrix of the storage status is updated in stages. The pseudo-code of the algorithm in SReCSM is described in figure 2.

Algorithm 1 Cloude Storage Strategy

```
procedure CloudStorage(a, c)
    S = {1, 2, 3}                                  ▷ Initialize the state space
    R = {1, 2, 3, ..., N}                          ▷ Initialize the state distribution
    Tdsf = T0                                      ▷ Initialize storage time
    Tsto = T1                                      ▷ Initialize Markov time
    while (Tdsf > 0) do
        if ( ! IsEmpty ( NC ) ) then               ▷ Node cost is not zero
            {
              GetDataFromNode(&Data)
              GetCost(&m)                          ▷ Get cost of node m
              StoreData( m , Data)                 ▷ Store data in node m
            if (Node.Cost > 1) then
              {
                StoreDataToOtherNode(&Data.Cost − 1)
              }
            end if
            Refresh(NC)
            }
        end if
        Tdsf − = 1
    end while
    //Markov Phase
    Build TM(&P)                                   ▷ Build State transfer matrix
    Build RV(&R)                                   ▷ Build State distribution matrix
    while (! bQuit) do
        {
        //Getthebeststrategy Phase
        while (Tsto > 0) do
            if ( ! IsEmpty ( NC ) ) then           ▷ Node cost is not zero
                {
                GetCostFromNode(&Data)             ▷ Get next time cost of node
                GetNextStrategy(Policy, &d)        ▷ Get strategy
                StoreData( d , Data)
                if (Node.Cost > 1) then
                  {
                    StoreDataToOtherNode(&Data.Cost − 1)
                  }
                end if
                Refresh(NC)                        ▷ Update node cost
                }
            end if
            Tsto − = 1
        end while
        Tsto = T1
        Refresh TM(&P)                             ▷ Initialize the state transfer matrix
        Refresh RV(&R)                             ▷ Initialize the state transfer distribution
    end while                                      1
    return c
end procedure
```

Figure 2. Pseudo-code in SReCSM

Therefore, in the SReCSM based on Markov chains, the distribution of the state in the future will be got by means of the probability matrix P of the state transfer and the distribution matrix $R(0)$ of the state at the initial time. The distribution of the storage state at time t_n will be calculated and the choice of SReCSM will be realized through matrix operation.

IV. SEARCHABLE RE-ENCRYPTION METHOD

The central idea of the searchable re-encryption is to generate a re-encryption key and decrypted while the keyword matched. By using searchable re-encryption method, SReCSM has increased the storage requirement due to the storing of all encrypted data. The objective of the proposed searchable re-encryption method is to increase the storage requirement, flexibility and reduce the security issues, overhead ratio and minimize the cost and searching time. This technique will not provide the effective data utilization. This technique is more simple and faster when sorting the arrays using the most significant radix sort. This technique will reduce the cost of the data owners and the time complexity will be reduced. This technique is more efficient and requires more storage space for the data.

A. Symbol Definitions

Symbols in the Searchable Re-encryption method are defined in the in Table I.

TABLE I. SYMBOL DEFINITION

Symbol	Definition
PR(K)	Private key enabled by Data owners
PU(K)	Public key enabled by data owners
Kw	Keyword
Db	Database {all the data passed to the cloud servers will be stored here}
Ek	Editing Keyword
Ed	Encrypted Data
PR(K)Re	Re-encryptedPrivate key
PU(K) Re	Re-encrypted Public key
Ed Re	Re-encryptedEncrypted Data
Dd	Decrypted Data
Kw(pu)	Keyword assigned with the public key
Kw(pr)	Keyword assigned with the private key
Do	Data Owners (Sender)
Du	Data Users (Receiver)
Cs	Cloud Server (Third Party User)

B. Keyword Editing

By searching the keyword we can edit the keyword with the help of three notations. While editing the keyword it requires, insertion, deletion or substitution. Here assuming the notation as n and editing the keyword as Ek.

Case 1 If $Ek \sim= n(i)$

Inserting the character into the first place

For example:

Character string (old) = ‘‘BOK’’

If $i=3$; $Ek \sim= n(3)$

In third place have to insert new character

Character String (new) = ‘‘BOOK’’

Case 2 If $Ek < n(i)$

Deleting the character into the first place

For example:

Character string (old) = ‘‘BOK’’

If $i=3$; $Ek < n(3)$

In third place have to delete the letter

Character String (new) = ‘‘BO’’

Case 3 If $Ek > n(i)$

Substitute the character into the first place

For example:

Character string (old) = ‘‘BOK’’

If $i=3$; $Ek > n(3)$

In third place have to substitute a new character

Character String (new) = ‘‘BOK’’

These three different notations help in editing the keyword easily. If the data users apply the keyword with some mistakes, they can edit the keyword to recover their respective data by using these three different cases.

C. Searchable Re-encryotion

The central idea of the searchable re-encryption is to generate a re-encryption key and decrypte while the keyword matched. Re-encryption operates over two groups G_1 and G_2 of prime order q with a bilinear map e: $G_1 \times G_1 \to G_2$. The system parameters are random generators

$g \in G_1$ and $Z = e(g, g) \in G_2$. The Searchable Re-encryption can be defined in the following algorithms.

Algorithm 2 follows that the Searchable Re-Encrypted Keyword and it will search the keyword to decrypt the files. If the keyword matched, the data can be decrypted and can bereceived and accessed by the users.

Algorithm 2 Keyword Searching

Input: Keyword for $PR(K)$

Output: Keyword Matched

If $PR(K) = Kw$

Goto $Db(i)$

Else If $Ek > n(i)$ \\ case-3

Goto $Db(i)$

Else "Not Matching"

End If $Ek < n(i)$ \\ case-2

"Keyword Matched"

Initially private key contains the keyword and that keyword will be searched in the database. Keyword with the private key should be verified in the database of data cloud storage. Data owners will transmit the data to the cloud servers and the cloud server will store the data in the database. The data users will receive the data from the database using the keyword. If the keyword did not match, data user use the editing scheme for insertion, deletion and substitution. If the keyword matched, data users will receive the data from the cloud servers which is explained in the Algorithm 2.

Algorithm 3 Data sharing

Generate $PU(K)$ $PR(K)$

$PU(K) = \{Ed, Kw(pu)\}$

$PR(K) = \{Kw(pr)\}$

Do shares $PU(K)$ to Cs

Do shares $PR(K)$ to Du

Send Ed to Cs

Du sends $Kw(pr)$ to Cs

Cs verify $Kw(pr)$

If $Kw(pu)$ = $Kw(pr)$

Cs sends $\{Dd, Kw(pr)\}$ to Du

Algorithm 3 states that the data file sharing between the data owners and data users through the cloud server. Initially data owners will create the public key and private key. The public key will be shared to the cloud servers and the private key will be shared to the data users. Every private and public key has its own keyword based on that keyword the data users will retrieve the data. If the keyword of public key and keyword of private key is matched, cloud server will transmit the data to the data users.

Algorithm 4 Data encryption

Consider two prime numbers as x and y

Assign $z = x * y$, where z will be used for the modulo of private and public keys

Assign Euler's function as $E(z) = (x-1)(y-1)$

Consider an integer as i such that $1 < i < E(z)$ for all $\{i, E(z) = 1\}$

Where i and $E(z)$ are co-prime

Assign $D = \{f_1, f_2, \cdots, f_n\}$; f as files, D as data and n as number of files.

$D(z) = \{f_1(z), f_2(z), \cdots, f_n(z)\}$

Encrypted data,

$Ed = \{D(z) \, mod \, E_z; Kw, PU(K)\}$

Algorithm 4 states that the data encryption scheme and this follows the RSA algorithm to encrypt the data. Initially the module multiplication of the two prime numbers will be calculated and this states that the Euler's function and this integer value is co-prime. Each data will be encrypted with modulo of E (z). The encrypted data contains the public key and the private keyword and the each data will be modulo with the Euler's function.

Algorithm 5 Data Re-encryption

Generate $PR(K)_{Re}$, $PU(K)_{Re}$

Assign $D=\{f_1, f_2, \cdots, f_n\}$, $g \in G_1$; f as files, D as data and n as number of files.

$$D(z)=\{f_1(z), f_2(z), \cdots, f_n(z)\}$$

Re-Encrypted data,

$$Ed(C_A)_{\mathrm{Re}} = e(PU(K), \quad Ed(C_A))$$

Algorithm 5 states that the data re-encryption scheme and this follows the AES algorithm to encrypt the data. The mobile user 'A' generates the re-encryption keys for authorized user's list 'U' using users' public key and personal private key as shown below:

$$PU(K)_{\mathrm{Re}} = PR(K)_{\mathrm{Re}} = (g^{x_i})^{\frac{1}{x_A}}, \forall u_i \in U$$

The CSReSM generates the $r_I \in Z_q^*$ randomly and encrypts the message using the following procedure:

$$Z_{new}^{r_I} = \frac{Z^{r_I}}{PU(K)}$$

$$(Ed \cdot Z_{new}^{r_I}) = (Z^{r_I} \cdot M)$$

$$Ed(C) = Z^{r_I} \cdot M, \quad Ed(CA) = g^{r_I x_A}$$

The CSReSM uploads the encrypted message ' $Ed(C)$ ' and ' $Ed(C_A)$ ' on behalf of the mobile user 'A'. The CSReSM transforms ' $Ed(C_A)$ ' into ' $Ed(C_A)_{Re}$ ' using the re-encryption key $PU(K)_{\mathrm{Re}}$ and $Ed(C_A)$ as shown in the following equation:

$$Ed(C_A)_{\mathrm{Re}} = e(PU(K), \quad Ed(C_A))$$

$$= e(g^{\frac{x_i}{x_A}}, \quad g^{x_A r_I})$$

$$= e(g, \ g)^{x_i r_I}$$

Algorithm 6 Data decryption

Consider D key = $\{PR(K), Kw\}$

For $f = 1$ to N //Data files

$$Dd = \{Ed \ mod \ PR(K)\}$$

End for f

loop

Goto Dd

Algorithm 6 states that the data decryption algorithm to decrypt the data using the above steps. Initially data users uses the decryption key and that decryption key contains private key with keyword. Select the number of files and then decrypt the data using modulo function. Each data will be decrypted using the decryption key.

D. Security analysis

Assuming that the reliability of the cloud storage system is identified by symbol A. The time of encryption through different encryption algorithms is A_t, the encryption time A_t is reversed first, and after the normalization processing, A_j is got from A_t. According to the Markov chain, the storage cost of the different node is normalized to be the value A_k. The number of the storage states for the cloud storage is the value n the reliability model of the system is shown in formula (13).

$$A = [1-(1-A_j)^n][1-(1-A_k)^n] \tag{13}$$

It can be concluded from the analysis of the reliability model. When the valet tends to be infinite, which means $t \to \infty$, the storage cost of the node in the cloud system also tends to a certain stable value, and the state of storage strategy tends to be stable. When the value of A_j and A_k are more closer to 1, and the value of n is more large, the cloud storage system will be more reliable and with higher security.

V. PROTOTYPE SYSTEM AND EXPERIMENT

A. Prototype system

HDFS and Dynamo are reliable solutions that are commonly used in the cloud storage system. HDFS is a distributed file system that is suitable for running on common hardware. Moreover, HDFS has good fault tolerance and can be used for inexpensive hardware. The program in HDFS has a lot of data sets. The file size in the HDFS is typically gigabyte to terabyte. As a result, terabytes of large files can be supported in HDFS through higher aggregated data bandwidth. Therefore, hundreds of nodal devices can be contained in a cluster, which allowing the terabytes of large files to be supported in it. The consistency hash algorithm is used by Dynamo. At that time, it's not the exact hash value, but a range of hash values. When the hash value of the key is in this range, it will be searched clockwise along the loop, and the first node encountered is what we need. The consistency hash algorithm is improved by Dynamo, and in the ring，a set of devices are acted as a node rather than only one device is acted as a node. The synchronization mechanism is used to achieve the consistency of the data.

In HDFS, numbers of the copies are set to be three. Whether the data would be stored in the node or not depends on the capacity of the node. The greater the capacity of the node is, the greater the probability that the data will be stored in this node. Therefore, when the capacity of the node is very different, the node with large capacity in the system would be overloaded. According to the analysis above, the storage cost of each storage node in the cloud system is measured from the perspective of probability. Moreover, the dynamic copy mechanism is adopted to adjust the number of copies in the storage strategy and their placement in real time according to the system performance requirement, load and so on. With the probability of passing the Markov chain, the dynamic replica mechanism can be used to complete data storage in time. The reliability and availability of SReCSM are improved effectively.

The storage group is used as the hardware architecture of the cloud storage system. Moreover, the storage group system is connected in the irregular network. The architecture of the storage group is shown in figure 3. The PC is used as a storage medium in the storage group. However, the reliability of the PC is not high, and it will even fail when the data are stored. Therefore, a copy is required to ensure that the data is reliable. According to the reliability criteria of the system, the storage state of the cloud system is divided into three types: small file status, large file status and secondary file status between the two.

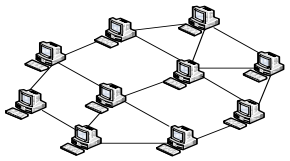

Figure 3. The system architecture diagram of the storage group

In the storage group system, all information about adjacent nodes is stored in every PC. The storage nodes needed can be found quickly through querying the information stored in the nodes. According to the transfer probability matrix and state distribution matrix of the SReCSM described precious, the storage strategy of a certain time is predicted.

The structure of storage space in the storage group is ring, and at the same time, the method of the unified addressing is adopted. In the storage group, the difference in performance of the PC can be offset by the virtual contiguous storage space. First, the hash algorithm message-digest is used to implement system address conversion. The actual physical address is processed and converted to 32-bit information string through the MD5 algorithm. And then these information strings are stored in the virtual continuous

address. Thus, the differences in performance between devices will be offset.

The converted address is mapped to the virtual storage space loop of the storage group through the MD5 algorithm. The device is found in the clockwise direction, and then the data is stored in the first PC mapped. Therefore, the data is backed up to two adjacent PC. The larger the amount of data in the system, the more uniform the spatial distribution will be. The data are stored when the routing of the corresponding PC and adjacent PC are updated. The routing information table is shown in Table II.

TABLE II. ROUTING TABLE

Field	Type	Length	Note
ID	int		Serial number
fname	varchar	255	Filename
fsize	int		File size
IP	varchar	15	IP address

The IP address of the PC device where the file replica located is stored in the IP field in the routing information table. The IP field is the routing information for the adjacent PC. However, once a node fails, all the information stored in the node are backed up and the routing information of the adjacent node is modified in time. According to the principle of the consistency hash algorithm, the storage space of the new PC device will be mapped to the new virtual address space when a new device needs to be added to the storage group. The existing space on the ring will not be changed, and this method can be very effective in avoiding the vibration of the address space. Meanwhile, the routing information on the adjacent PC is updated. The process of adding a PC is as shown in figure 4.

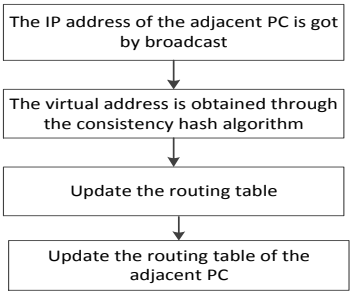

Figure 4. The process of adding a PC

B. Experiments and result analysis

The proposed scheme SReCSM Searchable Re-encryption Cloud Storage Method based on Markov Chain was developed by using the Java coding in the Linux platform. Finally evaluation of the overall performance of the proposed scheme in the real time data applications is done. This proposed scheme involved data owners and data users. Data owners create the encrypted data and then re-encrypted data will be forwarded to the cloud storage servers. Data users will decrypt the data using the keyword and private key.

In this section, we analyzed the prediction, the searching time, searching efficiency and storage space while performing moving, copying, encryption, decryption, and re-encryption operations. The final results are compared with the existing techniques such as CReS Cloud-based Re-encryption Scheme, MReS Searchable Encrypted Data File Sharing scheme.

1) Prediction of SReCSM

SReCSM based on Markov chain is processed by class Dynamo system model. On the basis of the thought and model mentioned above，the algorithm is implemented in python and verified by example. The simulation experiment is only to verify the Markov characteristics of distributed storage strategy.

At first, the Monte Carlo method is used to simulate 8000 times and 300 intervals each time. The distribution of storage state in these intervals and the change of storage status between adjacent intervals are counted. The state transfer probability matrix of the stored strategy is shown in formula (14).

$$P = \begin{bmatrix} 0.986 & 0.013 & 0.001 \\ 0.008 & 0.986 & 0.006 \\ 0.002 & 0.185 & 0.813 \end{bmatrix} \tag{14}$$

The meaning of the state transfer probability matrix P is described as follows. Suppose the last time the cloud system is in a small file storage state. At the next moment, the probability of 0.986 is kept in a small storage state; the probability of 0.013 is transferred to the secondary storage

state; and the probability of 0.001 is transferred to the large storage state.

The storage group in figure 2 contains 10 nodes. Because the hardware performance, load situation and network link state of the storage nodes in the cloud system are different, the storage cost of each node is different. Assuming at the initial moment t_0, the storage costs for large, medium, and small storage states are s_1, s_2 and s_3 respectively. The storage status distribution matrix $R(0)$ of the cloud storage system at time t_0 can be obtained through formula (2). The storage status distribution matrix $R(0)$ is shown in formula (15).

$$R(0)=\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}^T = \begin{bmatrix} 0.10 & 0.20 & 0.70 \\ 0.35 & 0.20 & 0.45 \\ 0.15 & 0.45 & 0.40 \\ 0.20 & 0.30 & 0.50 \\ 0.35 & 0.30 & 0.35 \\ 0.25 & 0.25 & 0.50 \\ 0.30 & 0.35 & 0.35 \\ 0.10 & 0.15 & 0.75 \\ 0.20 & 0.35 & 0.45 \\ 0.25 & 0.30 & 0.45 \end{bmatrix} \quad (15)$$

The formula (10) (11) is substituted into formula (6), and a one-step storage state distribution matrix is obtained, which is shown in formula (16).

$$R(1)=\begin{bmatrix} 0.10160 & 0.32800 & 0.57040 \\ 0.34760 & 0.28500 & 0.36740 \\ 0.15230 & 0.51965 & 0.32805 \\ 0.20060 & 0.39090 & 0.40850 \\ 0.34820 & 0.36510 & 0.28670 \\ 0.24950 & 0.34225 & 0.40825 \\ 0.29930 & 0.41370 & 0.28695 \\ 0.10130 & 0.28795 & 0.61075 \\ 0.20090 & 0.43095 & 0.36815 \\ 0.24980 & 0.38230 & 0.36790 \end{bmatrix} \quad (16)$$

After a multi-step transition, the storage state distribution matrix $R(i)$ at time t_i is predicted quickly, which is shown in formula (17).

$$R(i)=\begin{bmatrix} 0.1139180 & 0.6278466 & 0.2582362 \\ 0.3425064 & 0.4889122 & 0.1685814 \\ 0.1644820 & 0.6808290 & 0.1546892 \\ 0.2072915 & 0.6045806 & 0.1881279 \\ 0.3444450 & 0.5212299 & 0.1343350 \\ 0.2520400 & 0.5606398 & 0.1873201 \\ 0.2996965 & 0.5651607 & 0.1351428 \\ 0.1129479 & 0.6116927 & 0.2753594 \\ 0.2082608 & 0.6207354 & 0.1710047 \\ 0.2530094 & 0.5767937 & 0.1701969 \end{bmatrix} \quad (17)$$

The node with the lowest storage cost can be quickly selected by using the storage status distribution matrix introduced in equation (17), which can realize real-time data storage.

2) Uploading and Downloading of SReCSM

The data response tests are performed on file upload, file copy and file movement, for large files and small files respectively. Experimental results demonstrate that the page is properly displayed, and the response time of the login page is basically completed within two seconds. The percentage of the response time for the transaction is shown in figure 5.

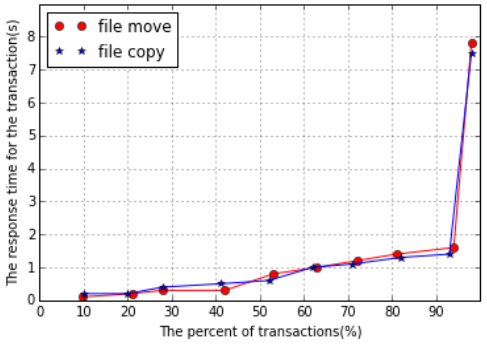

Figure 5. The percentage of the response time for the transaction

Through the analysis of figure 6, it can be known that 94 percent of transactions in a mobile cloud storage system can be implemented quickly within two seconds. The experimental results show that the system responds fast. The

average response time of the transaction is obtained from the diagram.

Although one of the response timing of transaction is longer, however the response time for most other transactions is acceptable. When this happens, it is thought that the performance of the mobile cloud storage system is better. Table III and Table IV are the test result.

TABLE III. THE PERFORMANCE OF THE MOBILE END UPLOAD THE DATA

type of test	utilization rate of Mobile CPU（%）	transmission rate （Mbps）	utilization rate of Mobile /transmission rate
Raw data	16.436	3.57020	4.603663
After the encryption	38.432	2.36015	16.283711

TABLE IV. THE PERFORMANCE OF THE MOBILE END DOWNLOAD THE DATA

type of test	utilization rate of Mobile CPU （%）	transmission rate （Mbps）	utilization rate of Mobile /transmission rate
Raw data	14.681	3.90135	3.763056
After the encryption	35.221	2.76147	12.754439

The ratio of CPU occupancy to upload speed is shown in table III, which the data on the mobile side are tested before the encryption and after the encryption respectively. The ratio of CPU occupancy to download speed is shown in table IV, which the data on the mobile side are tested before the decryption and after the decryption respectively. It can be known from the table III and the table IV, if the encryption and decryption mechanism are used for HDFS transmission, then the CPU utilization will be increased by an average of 22% ~ 25% and the overall file transfer rate will be reduced by 30% ~ 35%. As we can see，when the encryption and the decryption mechanism are used, more than three times the performance loss can be caused on the mobile end side.

3) Encryption and Re-Encryption

After applying the keywords, based on that proposed scheme it will search the keyword. If it matches, data users will receive their respective data. Based on the searching time and the storage space comparison graph has discussed below. Data users considered for the proposed scheme is 500 users and the available search keyword vary from 500 to 5000. Cloud servers storage space obtain more than 100 bytes and it uses the database to store all the encrypted data and the keywords.

There are two questions to be considered: One is the impact of encryption and decryption on file speed. The other is the impact of encryption and decryption on the performance of the client host. The experimental data are listed in Table V, which includes the time spent on encrypting the different sizes or different type files by using SReCSM and the time spent on transmitting the file in HDFS.

TABLE V. TIME COMPARISON ON ENCRYPTION AND DECRYPTION BY USING SRECSM

File size （M）	File type	SReCSM encryption （ms）	HDS upload （ms）	SReCSMdecryption（ms）	HDS download （ms）
3.07	pdf	1050	2685	370	2800
3.22	MP3	1178	2600	478	2830
23.8	mkv	3238	5930	2648	6290
25.8	doc	3140	5260	2163	6400
166.518	rmvb	23830	46400	16500	42460

It can be concluded from the above test data, the time spent on encryption or decryption by using SReCSM is regardless of the file type.

The time comparison on SReCSM encryption and re-encryption is shown in figure 6. We can see from figure 6 that there is little difference between encryption time and re-encryption time.

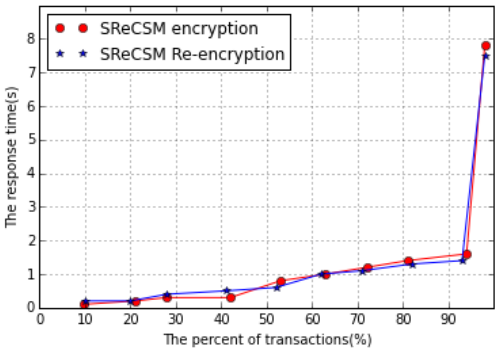

Figure 6. Time comparison on SReCSM encryption and Re-encryption

The same size files are encrypted by different CReS Cloud-based Re-encryption Scheme, MReS Manager-based Re-encryption Scheme, or SReCSM algorithms and then the re-encryption time is different, as shown in table VI and figure 7. The 167.58 MB file in table VI is the test case.

TABLE VI. TIME COMPARISON FOR DIFFERENT ALGORITHM ENCRYPTION

File size（M）	MReS Re-encrypt（ms）	CReSRe-encry pt（ms）	SReCSMRe-en crypt（ms）
3.04	1048	770	720
23.15	3230	2901	2600
80.35	12010	10230	8560
167.58	23820	23612	23598

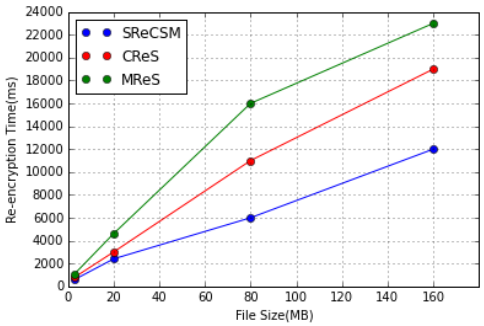

Figure 7. Comparison of Re-encryption time

In the SReCSM proposed in this paper，the time of encryption or decryption is relatively short. File transferring have little impact on total time loss and user experience. It may take a relatively long time to encrypt files by using the CReS, which cause a significant additional time overhead for HDFS. However, the encryption time that MReS encrypting the file was not significantly increased compared to SReCSM. Besides the impact on overall transmission rates, the impact of encryption and decryption on mobile performance is also important.

In the next experiment, we compared the searching time, searching efficiency and storage space while performing the encryption and re-encryption operations.

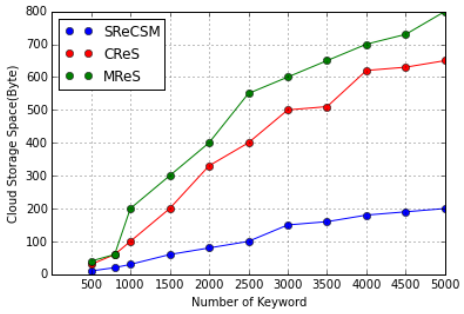

Figure 8. Storage spaceversus number of keyword

Figure 8 shows that the comparison graph of storage space in different algorithms. When there is increasing of the number of keywords, it requires more storage space. The existing algorithms CReS, MReS have require more storage space. But the proposed Searchable Re-encryption Cloud Storage Method (SReCSM) reduce the storage space requirement and utilize the data transfer effectively.

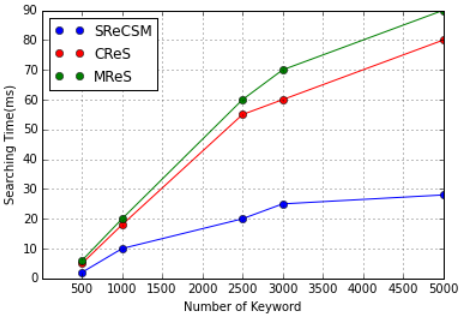

Figure 9. Searchingtimeversus number of keyword

Figure 9 shows that the comparison graph of the searching time and number of keywords. The number of keyword vary from 500 to 5000. When the number of keyword increases, searching time also increase. In previous techniques, CReS and MReS use the more searching time. However, SReCSM uses lesser time to search the data. If searching word is not matched, immediately the proposed SReCSM uses the editing values. Based on this different cases, the proposed SReCSM decreases the searching time.

MReS, CReS, and SReCSM offload the re-encryption operations on cloud. Therefore, in this experiment we examined the turnaround time and energy consumption on cloud while performing the re-encryption operations. The experimental results are shown in Fig. 10.

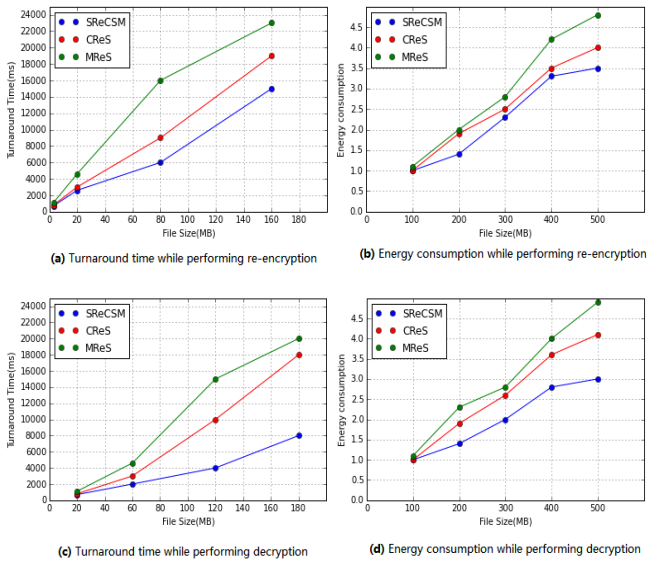

Figure 10. Comparison while performing re-encryption and decryption

It can be observed from the results presented in Figs. 10a and b that the increase in the size of file increases the turnaround time and energy consumption for completing the re-encryption operations on the mobile device. The increase in turnaround time and energy consumption is due to the increase in number of re-encryption operations while increasing the number of files. However, Figs. 10c and d show that the increase in the size of file increases the turnaround time and energy consumption for completing the decryption operations on the mobile device. The increase in turnaround time and energy consumption is due to the increase in number of decryption operations while increasing the size of files.

Using the reliability model formula (13) of the cloud storage system proposed in 4.4, combine the time required for processing the same size of file in table IV, when a different algorithm CReS, MReS and SReCSM is used, the encryption time required for encrypting file, after that the encryption time is reversed, A_j then be got after the encryption time is normalized. In the same way, after normalizing, storage cost A_k is got. If both A_j and A_k are closer to 1, and the number of storage state in the cloud storage system is larger, then the reliability of the system is higher. According to the above analysis，the data in one hour is sampled continuously, combined with the data in table IV and table V, the reliability contrast diagram for SReCSM is shown in figure 11.

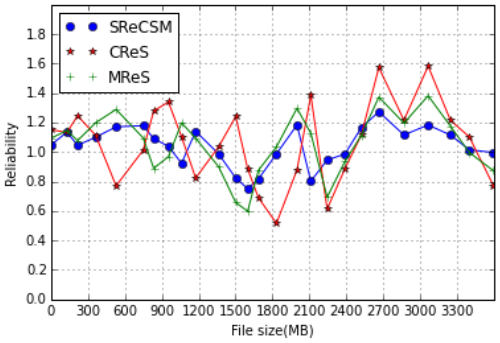

Figure 11. The reliability contrast diagram

It can be know the reliability of the system through different algorithm by comparing the data in figure 11. By using the CReS re-encryption， the reliability values are almost maintained at one. The reliability of the system is relatively high. That is， it has little impact on file transfer and user experience by using CReS re-encryption. It may take a relatively long time to re-encrypt files by using the MReS. And the reliability is very jitter. It is shown that the reliability is low with MReS re-encryption. However, the re-encryption time that MReS combined with CReS for re-encrypting the file was not significantly increased compared with CReS. The value of the reliability is consistent with the use of CReS, which can be maintained around one. It is concluded that the system is relatively reliable by using SReCSM encryption.

Through these simulation experiments, it is verified that SReCSM has a good user experience. It is also verified that the mechanism of SReCSM can effectively improve the efficiency of the cloud storage. When a mobile terminal makes a request, the optimal node is selected and then the time can be saved effectively.

In the SReCSM presented in this paper, the re-encryption and decryption has the following characteristics: transport security and storage security of the user data are guaranteed. The mobile finishes the re-encryption before calculating the checksum, so the re-encryption will not break the HDFS data integrity check mechanism. In the entire distributed file storage system, the re-encryption and decryption are

scattered to the various mobile devices. While this will cause some performance damage to the mobile, there is no additional performance penalty for name node and data node.

VI. CONCLUSIONS AND FUTURE WORK

CReS and MReS re-encrypts the keyword to transmit safety. But these two schemes are more complex and need more time to re-encrypt. To optimize the cloud storage, safety transmission, minimize the cost and searching time, here we have proposed a new scheme as searchable re-encrypted data in SReCSM. This proposed searchable re-encryption method supports the periodical and secure prediction by using the steady Markov strategy in stages and determine the lowest storage cost. SReCSM increases the storage requirement, flexibility and reduce the security issues, overhead ratio and minimize the cost and searching time.

The SReCSM based on Markov chain proposed in the paper has high reliability proved through a series of simulation experiments. The comparison graph evaluate the turnaround time and energy consumption with the different size of files. By increasing the size of files, proposed SReCSM can achieve accurate predictions, reduce the storage space requirement and the re-encrypting time. And then the data is scheduled to be stored in the node with the lowest storage cost. Finally conclude that the SEDFS proposed in this paper has better security and reliability. And this SReCSM reduces the storage space requirement, security issues, searching time and increases the searching efficiency.

ACKNOWLEDGMENT

Foundation item: The Industrial research project of Science and Technology Department of Shaanxi Province(Grant No. 2016KTZDGY4-09); Laboratory fund of Xi 'an Technological University (GSYSJ2017007)

REFERENCES

[1] Karel, Ferreira, Denzil,Goncalves, Jorge,Kostakos, Vassilis, De Moor, Katrien: Mobile cloud storage: A contextual experienceVandenbroucke.In: MobileHCI 2014 - Proceedings of the 16th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services, p 101-110, September 23, 2014

[2] Mitsutaka Kimura,Xufeng Zhao, Toshio Nakagawa: Using Markov Renewal Processes.Principles of Performance and Reliability Modeling and Evaluation Reliability Analysis of a Cloud Computing System with Replication, pp. 401-423(2016)

[3] Choo, Kim-Kwang Raymond: Mobile cloud storage users. In: IEEE Cloud Computing,v 1, n 3, p 20-23, September 1, 2014

[4] Iliadis, I., Sotnikov, D., Ta-Shma, P., Venkatesan, V.: Reliability of geo-replicated Cloud storage systems. In: 2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing, pp. 169–179 (2014)

[5] Jeyanthy, C.,Shaji, R.S.,Jayan, J.P., Symmetric key based cryptic scheme for mobile cloud storage.In: Global Conference on Communication Technologies, GCCT 2015, p 571-575, November 30, 2015

[6] Jung, Kye-Dong, Moon, Seok-Jae, Kim, Jin-Mook: Data access control method for multimedia content data sharing and security based onXMDR-DAI in mobile cloud storage.Multimedia Tools and Applications, v 76, n 19, p 19983-19999, October 1, 2017

[7] Chekam, T.T., Zhai, E., Li, Z., Cui, Y., Ren, K.: On the synchronization bottleneck of OpenStack Swift-like cloud storage systems. In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)

[8] Li, L., Li, D., Su, Z., Jin, L., Huang, G.: Performance analysis and framework optimization of open source cloud storage system. China Commun. 13(6), 110–122 (2016)

[9] Iliadis, I., Sotnikov, D., Ta-Shma, P., Venkatesan, V.: Reliability of geo-replicated Cloud storage systems. In: 2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing, pp. 169–179 (2014)

[10] Yu, Xiaojun, Wen, Qiaoyan: Design of security solution to mobile cloud storage. Advances in Intelligent and Soft Computing, v 135, p 255-263, 2012

[11] Han, Lin; Huang, Hao; Xie, Chang-Sheng: Multi-path data prefetching in mobile cloud storage. In: Proceedings - 2014 International Conference on Cloud Computing and Big Data, CCBD 2014, p 16-19, March17, 2014;

[12] Lee, Giwon; Ko, Haneul; Pack, Sangheon: An Efficient Delta Synchronization Algorithm for Mobile Cloud Storage Applications. IEEE Transactions on Services Computing, v 10, n 3, p 341-351, May-June 2017;

[13] SystemWang, Yan; Wang, Jinkuan: An Optimized Replica Distribution Method in Cloud Storage. Journal of Control Science and Engineering, v 2017

[14] Zhang, Rui; Lin, Chuang; Meng, Kun; Zhu, Lin: A modeling reliability analysis technique for cloud storage system. In: International Conference on Communication Technology Proceedings, ICCT, p 32-36, 2013, ICCT 2013 -Proceedings of 2013 15th IEEE International Conference on Communication Technology

[15] Chen, Ming-Hung; Tung, Yu-Chih; Hung, Shih-Hao; Lin, Kate Ching-Ju; Chou, Cheng-Fu: Availability Is Not Enough: Minimizing Joint Response Time in Peer-Assisted CloudStorage Systems. IEEE Systems Journal, v 10, n 4, p 1424-1434, December 2016

[16] Tysowski, P.K., Hasan, M.A.: Re-encryption-based keymanagement towards secure and scalable mobile applica-tions in clouds. IACR Cryptology ePrint Archive 668, 2011(2011)

[17] Zhao, G., Rong, C., Li, J., Zhang, F., Tang, Y.: Trusteddata sharing over untrusted cloud storage providers, pre-sented at the IEEE Second International Conference onCloud Computing Technology and Science (CloudCom’10), Washington, DC, USA (2010)

[18] Yang, J., Wang, H., Wang, J., Tan, C., Yu, D.: Provabledata possession of resource-constrained mobile devicesin cloud computing. Journal of Networks 6, 1033–1040(2011)

[19] Itani, W., Kayssi, A., Chehab, A.: Energy-efficient incre-mental integrity for securing storage in mobile cloudcomputing, presented at

the International Conference onEnergy Aware Computing (ICEAC ’10) Cairo, Egypt(2010)

[20] Ren, W., Yu, L., Gao, R., Xiong, F.: Lightweight andcompromise resilient storage outsourcing with distributedsecure accessibility in mobile cloud computing. TsinghuaScience & Technology 16, 520–528 (2011)

[21] Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud com-puting, presented atthe Proceedings IEEE(INFOCOM’10)NJ, USA (2010)

[22] Jia, W., Zhu, H., Cao, Z., Wei, L., Lin, X.: SDSM: Asecure data service mechanism in mobile cloud computing,presented at the IEEE Conference on Computer Commu-nications Workshops (INFOCOM ’11) Shanghai, China(2011)

[23] Zhou, Z., Huang, D.: Efficient and secure data storage operations for mobile cloud computing, presented at the 8thInternational Conference on Network and Service Management (CNSM ’12), AZ, USA (2012)

[24] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improvedproxy re-encryption schemes with applications to securedistributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC)9, 1–30 (2006)

[25] Zhang, Yuan; Xu, Chunxiang; Li, Hongwei; Liang, Xiaohui: Cryptographic Public Verification of Data Integrity for Cloud Storage Systems. IEEE Cloud Computing, v3, n5, p 44-52, 2016

[26] Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: Aciphertext-policy attribute-based encryption scheme withconstant ciphertext length. Inf. Secur. Practice Experience5451, 13–23 (2009)

[27] Purushothama, B.R; Shrinath, B.; Amberker, B.B. : Secure cloud storage service and limited proxy re-encryption for enforcing accesscontrol in public cloud. International Journal of Information and Communication Technology, v5, n2, p167-186, 2013

[28] Cui, Yihui; Peng, Zhiyong; Song, Wei; Li, Xiaojuan; Cheng, Fangquan; Ding, Luxiao: A time-based group key management algorithm based on proxy re-encryption for cloudstorage. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and LectureNotes in Bioinformatics), v8709 LNCS, p117-128, 2014

[29] Shao, Jun; Lu, Rongxing; Lin, Xiaodong; Liang, Kaitai: Secure bidirectional proxy re-encryption for cryptographic cloud storage. Pervasive and Mobile Computing, v28, p113-121, June 1, 2016

[30] Jiang, Linmei; Guo, Donghui: Dynamic Encrypted Data Sharing Scheme Based on Conditional Proxy Broadcast Re-Encryption for Cloud Storage. IEEE Access, v5, p13336-13345, July 13, 2017

[31] Wang, XuAn; Xhafa, Fatos; Hao, Wei; He, Wei: Non-transferable unidirectional proxy re-encryption scheme for secure social cloudstorage sharing. Proceedings - 2016 International Conference on Intelligent Networking and Collaborative Systems, IEEE INCoS 2016, p328-331, October 25, 2016