International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 12 | Dec-2017 www.irjet.net p-ISSN: 2395-0072

Review and Analysis of Self Destruction of Data in Cloud Computing

Pradnya Harpale[1], Mohini Korde[2], Bhagyashree Pawar[3], Pritam Kore[4], S.H.Patil[5]

[1,2,3,4] Dept. of Computer Engineering, JSPM's Jayawantrao Sawant College of Engineering, Pune, Maharashtra ,India
[5] Prof. Dept. of Computer Engineering, JSPM's Jayawantrao Sawant College of Engineering, Pune, Maharashtra ,India

---

Abstract - *Cloud computing is the most emerging technology today which is used by most of the social media sites to store the data. The data stored on the cloud is private data of the user so it must not be tampered by other entities. We propose a system to enhance the security, the data uploaded by a user is shuffled between the number of directories within cloud after a particular interval of time to avoid the tracking of the data. The backup of the data will be taken timely into the backup directory. The proposed system enhances the security as well as the ease to use the cloud.*

***Key Words*: Centrality, Cloud security, Fragmentation, replication, performance, Secure Hash Algorithm(SHA), Advanced Encryption Standard(AES),Cloud Computing.**

1. INTRODUCTION

Cloud computing enables on-demand network access to a shared pool of configurable computing resources such as servers, storage, and applications. These shared resources can be rapidly provisioned to the consumers on the basis of paying only for whatever they use. Cloud storage refers to the delivery of storage resources to the consumers over the Internet. Private cloud storage is restricted to a particular organization and data security risks are less compared to the public cloud storage. Hence, private cloud storage is built by exploiting the commodity machines within the organization and the important data is stored in it. When the utilization of such private cloud storage increases, there will be an increase in the storage demand. It leads to the expansion of the cloud storage with additional storage nodes. During such expansion, storage nodes in the cloud storage need to be balanced in terms of load. In order to maintain the load across several storage nodes, the data need to be migrated across the storage nodes. This data migration consumes more network bandwidth. The key idea behind this Application is to develop a dynamic load balancing algorithm based on deduplication to balance the load across the storage nodes during the expansion of private cloud storage.

Cloud computing is the most emerging technology today which is used by most of the social media sites to store the data. The data stored on the cloud is private data of the user so it must not be tampered by other entities. We propose a system to enhance the security, the data uploaded by a user is shuffled between the numbers of directories within cloud after a particular interval of time to avoid the tracking of the data. The backup of the data will be taken timely into the backup directory. The proposed system enhances the security as well as the ease to use the cloud.

2. PROPOSE SYSTEM

In proposed system, we collectively approach the issue of security and performance as a secure data replication problem. The division of a file into fragments is performed based on a given user criteria. Divided File can store in different nodes. In such a scenario, the security mechanism must substantially increase an attacker's effort to retrieve a reasonable amount of data even after a successful intrusion in the cloud. Moreover, the probable amount of loss (as a result of data leakage) must also be minimized. Security is one of the most crucial aspects among those prohibiting the widespread adoption of cloud computing. Cloud security issues may stem due to the core technologies implementation (virtual machine).

The data outsourced to a public cloud must be secured. Unauthorized data access by other users and processes must be prevented. The main aim of our project is secure the files store on cloud. The division of a file into fragments is performed based on a given user criteria. A successful attack on a single node must not reveal the locations of other fragments within the cloud.

To keep an attacker uncertain about the locations of the file fragments and to further improve the security. We select the nodes in a manner that they are not adjacent and are at certain distance from each other. To improve data retrieval time, the nodes are selected based on the centrality measures that ensure an improved access time.

We develop a scheme for outsourced data that takes into account both the security and performance. The proposed scheme fragments and replicates the data file over cloud nodes. The proposed DROPS scheme ensures that even in the case of a successful attack, no meaningful information is revealed to the attacker. We do not rely on traditional cryptographic techniques for data security. The non-cryptographic nature of the proposed scheme makes it faster to perform the required operations (placement and retrieval) on the data. We ensure a controlled replication of

the file fragments, where each of the fragments is replicated only once for the purpose of improved security.

In this system mainly consist of three modules:

- Owner
- User
- Cloud Admin

Owner-In these owner modules, owner add the multiple users in the cloud system. Owner also uploads the text files in the cloud storage. He can delete the files.

User-In user module, user completes the registration process and login the system. He can upload the text files on cloud storage. He can delete, update or modify the files

Cloud Admin-In this module, the views the all status, views user, view owner, and reports send back. The Cloud admin module file can fragments using the fragment placement and fragments replication algorithms.

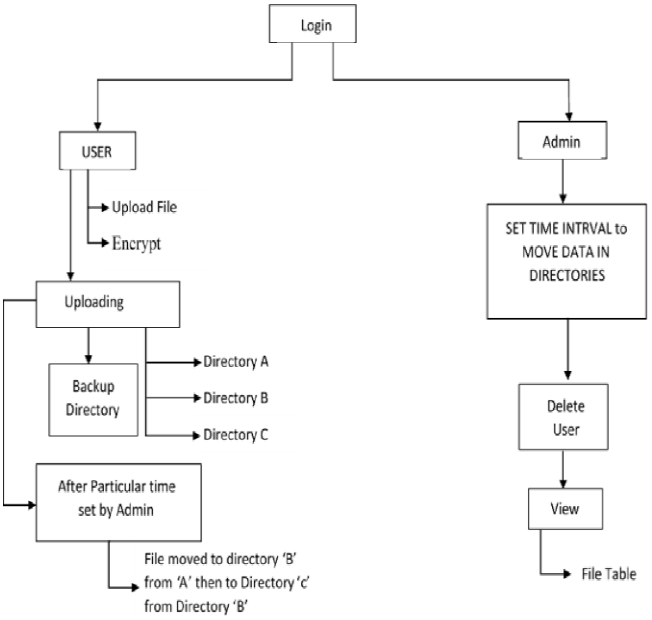


FIGURE 1 : SYSTEM ARCHITECTURE

3. ALGORITHM

3.1 T-COLORING ALGORITHM FOR FRAGMENTS ALLOCATION:

Using DROPS method file split into various fragment. For Security and Optimal Performance in cloud Division and Replication is used. When selecting the node in cloud keeping focus on performance and security. For enhanced access time we have to choose central nodes offered in cloud storage. DROPS technique uses centrality to reduce access time.

T-coloring technique selects nodes in cloud for fragment placement by keeping focus on performance and security. The central node in cloud network gives improved access time. The DROPS method utilize centrality concept to decrease the access time. Centrality concludes central node based on various measures.

T-coloring restricts node selection at hop distance. T-coloring gives more security performance in the cloud. The fragments are stored in different node so location of fragments can't able to determine.

3.2 FRAGMENT PLACEMENT

Algorithm 1 Algorithm for fragment placement

Inputs and initializations:

O = {O1;O2; :::;ON}
o = {sizeof(O1); sizeof(O2); :::::; sizeof(ON)}
col = {open color; close color}
cen = {cen1; cen2; :::; cenM}
col _ open color¦ i
cen _ ceni¦ i
Compute:
for each Ok > O **do**
select Si S Si _ indexof(max(ceni))
if colSi = open color and si >= ok then
Si _ Ok
si _ si − ok
colSi _ close color
Si'_ distance(Si; T) P /*returns all nodes at
distance T from Si and stores in temporary set Si'*/
colSi. _ close color
end if
end for

3.2 FRAGMENT'S REPLICATION

In addition to placing the fragments on the central nodes, we also perform a controlled replication to increase the data availability, reliability, and improve data retrieval time. We place the fragment on the node that provides the decreased access cost with an objective to improve retrieval time for accessing the fragments for reconstruction of original file. While replicating the fragment, the separation of fragments as explained in the placement technique through Tcoloring, is also taken care off. In case of a large number of fragments or small number of nodes, it is also possible that some of the fragments are left without being replicated because of the T-coloring. As discussed previously, T-coloring prohibits to store the fragment in neighborhood of a node storing a fragment, resulting in the elimination of a number of nodes to be used for storage. In such a case, only for the remaining fragments, the nodes that are not holding any fragment are selected for storage randomly. The replication strategy is presented in Algorithm 2. To handle the download request from user, the cloud manager collects all the fragments from

the nodes and re-assemble them into a single file. Afterwards, the file is sent to the user.

Algorithm 2 Algorithm for fragment's replication
for each Ok in O **do**
select Si that has max(Ri
k +Wi
k)
if colSi = open color and si >= ok then
Si _ Ok
si _ si − ok
colSi _ close color
Si'_ distance(Si; T) P /*returns all nodes at
distance T from Si and stores in temporary set Si'*/
colSi. _ close color
end if
end for

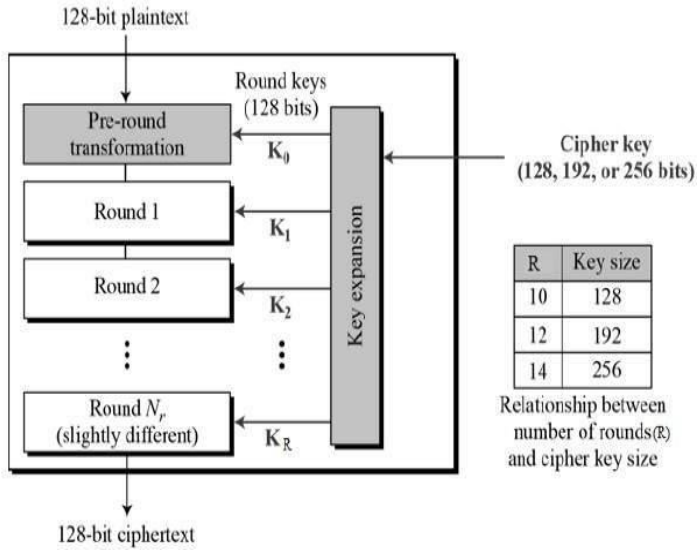
3.2 ADVANCE ENCRYPTION STANDARD (AES)

AES is an iterative rather than Feistel cipher. It is based on substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix –

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration –



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

FIGURE 2: AES STRUCTURE

3.2 SECURE HASH ALGORITHM

Hash functions are extremely useful and appear in almost all information security applications. A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.

Values returned by a hash function are called message digest or simply hash values. Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.

The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.

4. MATHMATICAL MODEL

Let Assume S be the system which execute Self Destruction of Data in Cloud Computing

S={s,e,X,Y,T,F_{main},NDD,DD,Success,Failure}

- **S(System)** = Is our proposed system which includes following tuple.
- **s (initial state at time T)** = GUI of search engine. The GUI provides space to enter a query/input for user.
- **X (input to system)** :- Input Query. The user has to first enter the query. The query may be ambiguous or not. The query also represents what user wants to search.
- **Y (output of system)** :- List of URLs with Snippets. User has to enter a query into search engine then search engine generates a result which contains relevant and irrelevant URL's and their snippets.
- **T (No. of steps to be performed)** :- **6**. These are the total number of steps required to process a query and generates results.
- **f_{main}(main algorithm)** :- It contains Process P. Process P contains Input ,Output and subordinates functions. It shows how the query will be processed into different modules and how the results are generated.
- **DD (deterministic data)**:- It contains Database data. Here we have considered MySQL, SQLite which contains number of queries. Such queries are user for showing results. Hence, SQLite is our DD.
- **NDD (non-deterministic data)**:- No. of input queries. In our system, user can enter numbers of

queries so that we cannot judge how many queries user enters into single session. Hence, Number of Input queries are our NDD.

- **Memory shared**: - MySQL. MySQL will store information like User Authentication, Performing Operations likeUser can upload the file. User can delete his own file by using secret pin.Since it is the only memory shared in our system, we have included it in the MYSQL.
- **CPU_{count}**: - 1. In our system, we require 1 CPU for server.
- **Success** = successfully recommended best system as per user's interest
- **Failure** = If application will not send the notification to user it will fail.

SUBORDINATE FUNCTIONS:

Let Assume S be the system which executeSelf Destruction of Data in Cloud Computing

$S=\{s,e,X,Y,F_{main},NDD,DD,Success,Failure\}$

Where

s=Start State

e=End State

X={Set Of Inputs}

X = {x1,x2,x3}

x1= Login credentials

x2= File that is to be uploaded

x3= Secret pin on download and delete request.

Y={Set of Outputs}

= {y1,y2,y3}

Where, y1= Encrypted file will be uploaded

y2= Hashcode of the file

y3= On download decrypted file will be downloaded.

F_{main} = {Set of procedure}

= {f1,f2,f3,f4,f5,f6}

Where

f1= Take x1 Input

f2= Give y1 Output

f3= Take x2 input

f4= Give y2 output

f5= Take x3 input

f6= Give y3 output

State Transition Diagram:

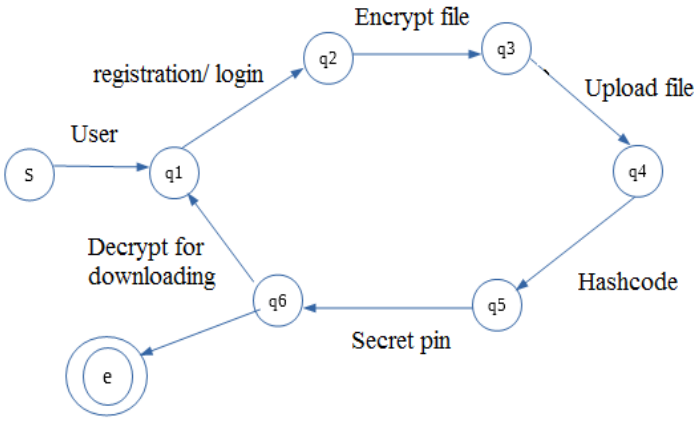


Fig: State Transition Diagram

Where,

s=input state

x=query

q1= Login credentials

q2= Encrypted file will be uploaded

q3= File that is to be uploaded

q4= Hashcode of the file

q5= Secret pin on download and delete request

q6= On download decrypted file will be downloaded

Explanation

- The q1 state accept the ambiguous query 'x' from the state 's' which is our initial state
- The q2 state is meant for storing Login credentialsof user,which stores the query x which is accept in state q1. The query stores in state q2 encrypted file will be uploaded and send to state q3
- The q3 state will be forFile that is to be uploaded
- State q4 for Hashcode of the file

- State q5 for inserting secret pin on download and delete request
- The q6 for downloading decrypted file.

3. CONCLUSIONS

This system is maintaining data theft. Reduce the data tracking. With the help of Hash code, algorithm data is divided into three different chunks and stored into different location so data load will be managed and provide the fast performance. This system is avoiding data duplication and reduce wastage of space.

REFERENCES

[1] X. Fu, Z. Wang, H. Wu, J. qi Yang, and Z. zhao Wang, "How to send a self-destructing email: A method of self-destructing email system," in Prof. of the IEEE International Congress on Big Data, 2014, pp.304–309.

[2] R. D. Binns, D. Millard, and L. Harris, "Data havens, or privacy sans frontieres?: a study of international personal data transfers," in Proc. Of the ACM conference on Web science (WebSci), 2014, pp. 273–274.

[3] R. Lu, H. Zhu, X. Liu, J. Liu, and J. Shao, "Toward efficient and privacy preserving computing in big data era," IEEE Network, vol. 28, no. 4,pp. 46–50.

[4] M. Arafati, G. G. Dagher, B. C. M. Fung, and P. C. K. Hung, "Dmash: A framework for privacy-preserving data-as-a-service mashups," in Proc. of the 8th IEEE International Conference on Cloud Computing (CLOUD), 2014.