Federated Cloud Storage Management Through Provable Data Possession using Dynamic Audit Protocol

Sermakani A.M. Paulraj.D

ABSTRACT ***Cloud computing is a technology for sharing the resources for on demand request and for processing the data. It facilitates cloud storage for adopting cloud users with the help of cloud service providers. It enhances need of enterprises by adhering large volume of data to store and owned privately through third party auditors via data centres. The proposed system analyse cloud storage and provide free data storage for computing the data and maintain variety of cloud storage in one place. This scenario promotes storage of files in one system, so the user doesn't require various accounts like GoogleDrive, Microsoft Onedrive and Dropbox. This application enhances multiple cloud storage for accessing all files in one particular storage area. The proposed system eradicates visiting of multiple sites for downloading the apps and reduces installing of multiple apps for downloading all the files. The work mainly focuses on the SaaS that permits users to upload data and share the resources from the cloud to post in the Web browser. Our work designed for creating single level of Application programming interface which is for all the cloud service providers. This adopts external applications that leverage the service of platform which is easier to build scalable, and automated cloud based applications. The final API promotes multiple cloud storage in one place and leads to provision Federated Cloud.***

Keywords: Cloud Computing, Multi Cloud Data Storage, Provable data possession, Dynamic Audit Protocol, Federated Cloud

I. INTRODUCTION

Cloud storage are very important to store and retrieve the files so that we can get the files from wherever we need. So, we create accounts on google drive, Microsoft one drive and also in some other drives. The main purpose of creating accounts in these drives is to enhance free cloud spaces. Each of these application have a particular amount of space that is provided for free. The extra space can be provided for cost. A new account can be created in similar application for gaining free spaces. On using number of applications, we will find difficulty in switching between accounts for retrieves of our files. The main aim of creating the multi cloud storage management application is to make authentication for every individual application of our account for the first time. Then, we can get authorization of the original application and can be viewed from our single application hub.

In recent years, enterprise requires highly automated data to be host in their cloud servers, and makes their users to access the data from cloud storage. The dependence of cloud storage requires highly secured auditing service and leads to many security challenges. The system also needs to check the integrity of each data in cloud storage. Previously security analysis will not confirm about integrity of data and cannot able to provide experimental results for error condition. This requires efficient and secure dynamic auditing services for dealing of improper authentication and request has to be reject as a result for cloud consumers. The system has to device remote data authentication for data collection and to process statistical analysis as a result of validation. In proposed system, the framework has been designed for auditing cloud storage and then extend to proceed with provable data possession for dynamic auditing. Our work supports dynamic data operations such as insert, delete and update about users data remotely and secure mechanism is followed in oracle model implementation. The support of proposed work further extends to build statistical analysis for bidirectional authentication.

II. EXISTING SYSTEM

The cloud storage application that are available have the facilities to store our files, picture, videos, backups and so on. These cloud stored files can be retrieved or viewed from any systems. We can also upload and download the files from wherever we want. And also it can be shared to number of accounts.

III. RELATED WORK

a. PUBLIC VERIFICATION

The process of verifying integrity of the data proportionally related to third party verification system in cloud. The storage system works with previous schemes such as Provable Data Possession(PDP) that highly depends on third party auditing. This scenario needs three participating parties such as owner of the data, third party auditor and cloud service provider to perform PDP schemes. The main aim of this scheme is to determine the authentication party as CSP is not privileged for finding it. In many situation CSP provides only the data and it is pretend to be work with TPA for making semi trusted scenario by Data owner. The functionality of CSP becomes potential if it works with Remote Data Auditing(RDA) Protocol. Being semi trusted, the CSP cannot build the three

Revised Manuscript Received on July 10, 2019.

Sermakani A.M. Research scholar, Anna University, Guindy, Chennai-600 025, Tamil Nadu, India. (sermakani@gmail.com)

Paulraj.D Professor, Department of IT, R.M.D College of Engineering, Chennai-601 206. Tamil Nadu, India. (kingrajpaul@gmail.com)

level entity structure for auditing. The CSP model makes the user to send the data and it cannot authenticate the security challenge of the sender. In the previous schemes, CSP permits sender to send the data and it wont reflect information about data owner of the related file. And it may adopts information about owner through statistical analysis. With all this analysis PDP models support public verifiability in order to meet auditing as a service for overcome the security threats.

b) COMPUTATIONAL OVERHEAD ALLOCATION

The considerable amount of computational load will be detected through RDA schemes for ensuring security and accuracy level for data verification. This requires study about remote networks when it is changed in topology level and all routing protocols have to indulge in same mechanism. The scheme utilizes levels of auditing protocol and the algorithm adopted for implement it. Through the operational unit the statistical analysis will be determine to verify the computational overhead. It depends on the operations on cloud data storage and specific about selection of routing protocol. Previous schemes such as RIP, EIGRP, and OSPF are utilized for generating operations in remote data and access easily through CSP. The following summary depicts the details about the operations of a dynamic routing protocol that adopted previously:

1. The main task of router is to send the data and receives messages through the routing protocol with its interfaces.
2. The information passed from router to other routers through sharing messages in the same routing protocol.
3. After learning remote networks, routers deal the routing information to other routers.
4. Every changes in the topology indicated to other routers through routing protocol.

In support of previous schemes, we propose the optimized allocation where cloud server will analyse the cloud data and storage capacity with computational overhead. Meanwhile the cloud server can able to manage the data storage and transfer the computation overhead to the third party dynamic auditing schemes for verification purpose. This scheme adopts more for producing high response in data storage and in need of verification of data owners.

c) ERROR HANDLING

In recent years, research has been improved to find out more appropriate error handling mechanisms. This is the next level of federated cloud to check the files from cloud storage if they got corrupted by other cloud user's data. The recovery process with these documents need full potential to allot them to end users. The cloud user can delete the file and can switch to other files for processing. The arbitrary failures may arise due to some sensitive information in deleted files. The existing system doesn't cares about protecting sensitive data and errors in huge volume of data. The file operations invokes important data sets and failure case may arise if entire file deleted from the storage. It creates failure attempt and there will be rise huge loss for the cloud user. The proposed system involves finding of errors as well as to determine sensitive data after commit operation over the files. Thus these cases are worth to recover in cloud data storage through provable data possession using dynamic audit protocol.

IV. PROPOSED SYSTEM

Our proposed work has following goals:

- ***Data Storage through public verifiability***: This system allows any cloud user to store the file on cloud and build the application. To allow anyone, not just the clients who originally stored the file on cloud drive applications, that gets verified for the login account verified through the database and stored in the cloud.
- ***Dynamic audit protocol***: This makes the user to sync their cloud application accounts with the multi cloud storage management account and the data is possessed to get the files to have a token generated. The generated token is stored in the database and the same token is used for further login process.
- ***Secured Provable Data Possession:*** To enhance TPA for auditing with low communication cost and computation overhead complexity, the secure Provable Data Possession scheme is followed. The server transfers the load for verification proactively leads to calculation process and it can transfer a part of PDP scheme.

This paper enhances us to group all the drive accounts in a single application. The secure cloud storage should need scalable platform that builds applications and enhances the functionality. This adopts external applications that leverage the service of platform which is easier to build scalable, and automated cloud based applications. Hence, there is no need of switching between different accounts of cloud account on search of some files that have been stored.

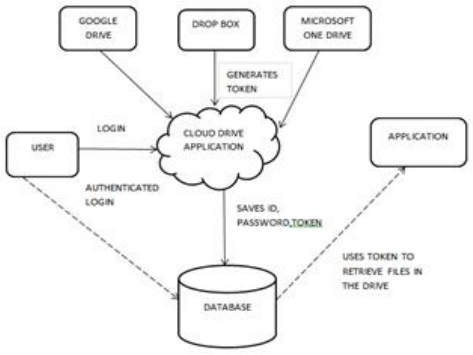

Fig 1: System Architecture for Storage

V. GENERATING AN API

This attempts API that invokes accessing of files and explicitly builds cloud storage to access the object based storage. This can be marketed with generating API as explained in Fig 1 that stores multiple application under one storage place which is meant to be federated cloud. are built

on an object-based storage platform. This enhances high levels of scalability factors that offers easy access via Hypertext Transfer Protocol (HTTP). In some organization implementation of protocol level is highly preferable to make computation overhead in low complex level.

All the files are accessed through object-based storage and is performed through a web services application programming interface (API). The attempt of increasing web based file access is done through on either the Simple Object Access Protocol (SOAP) or through Representational State Transfer (REST) protocol. Though the enterprise organizations prefers their file access still they need industry-standard Common Internet File System (CIFS) or Network File System (NFS) protocols has to be implement.

The deployment in proposed scheme for cloud storage in the least disruptive manner, and there is no further gateway required for accessing the platform. As shown in Figure 1, a cloud storage gateway adopts a local file system packages to act as remote object-based storage platform. Enterprise organizations prefers their file access through industry-standard Common Internet File System (CIFS) or Network File System (NFS) protocols. The unique job of gateway is to translate the file that access to the appropriate web services API. It enhances platform enability and retrieves the file from the cloud, and to place them in the local file system for users to access.

The main aim of the file system is to check the your application and sends to the Drive API. This includes an authorization token and issued to the cloud user for authentication. The token plays the unique role to identify your application to Google.

This is authorized using **OAuth2.0** protocol.

AUTHORIZATION PROTOCOLS:

Our proposed system uses One-Cloud to authorize requests. This protocol doesn't need additional authorization protocols which is supported for cloud storage support. If your application uses Google Sign-In, some aspects of authorization are handled for you. It authorizes requests with One-Cloud. The requests to the Drive API must be authorized by an authenticated user. The details of the authorization process, or "flow," for one-cloud vary somewhat depending on what kind of application you're writing. The following are the steps to adopt general process to all application types:

1. When the application got created, you register it using the Google API Console. Then account user from Google receives information regarding client ID and a client secret.
2. This will follow up by activation of Drive API in the Google API Console. (If the API isn't listed in the API Console, then skip this step.)
3. The submission process need access to your application when it requires for user data, it asks Google for a particular scope of access.
4. The authorization of user data is done through consent request in Google page to get permission from the user.
5. After user provides the details it approves the user for accessing the files in the one-cloud drive.
6. Once user got approval, then Google submits your request and sanction your application a short-lived **access token**.
7. Automatically the processing of your application requests get sanctioned and further process of attaching the access token to the request will be continue.
8. If Google determines that your request is authenticated and the token are valid, it returns the requested data.

Some flows include additional steps, such as using **refresh tokens** to acquire new access tokens. This will follow up the same procedure stated in authorization protocols.

VI. API CONNECTION & RESULTS

The proposed system involves easier qay to receive the API generation and Once you've generated your API Token, then it is easy to generate API for all other applications. The job of accessing files is done through HTTP request and it can be taken by sending request to api.lessannoyingcrm.com. You can use POST, GET, or whatever else you want (it's all the same to us).

The working principles of API involved from same URL that attached to it. This passes the information regarding the parameters and accessing rights to all the cloud users. All API requires parameter that passes for accessing the files. Once attempt got succeeded then four parameters can be followed for considering all types of file access from Google drive API call,

1. **UserCode** - the unique code identifying who you are (you get this when you generate your API Token)
2. **APIToken** - the security credentials which will give you access to the API
3. **Function**- the name of the function you want to call (like "CreateContact")
4. **Parameters** -a JSON-encoded array of parameters to pass to the function

```
<?php

function CallAPI($UserCode, $APIToken, $Function, $Pa
rameters){
   $APIResult = file_get_contents("http://api.lessa
nnoyingcrm.com?UserCode=$UserCode&
APIToken=".
            "$APIToken&Function=$Function&Parameters="
.urlencode(json_encode($Para
meters)));

   //decode the result into an associative array
   $APIResult = json_decode($APIResult, true);

   //check to see if the API call worked
   if(@$APIResult['Success'] === true){
      echo "Success!";
   }
   else{
      echo "API call failed. Error:".@$APIResult['
or'];
      exit;
   }
   return $APIResult;
}

?>
```

As you can see, the API returns a JSON encoded array. One of the items in that array is called "Success" and it will return true or false. If false, there will also be an "Error" so you can figure out what went wrong. If the API call was a success, there will normally be other data in the array such as the ID of the record you just created.

If you're comfortable reading PHP code, the easiest way to start using the API is to read through our example code. You can also see a complete list of all API functions in the Function Definition section.

LINKS OF USING AN API:

FILES:

For Files Resource details, see the resource representation page.

METHOD	HTTP REQUEST
Get	GET /files/fileId
Insert	POST /drive/v2/files POST /files
Patch	PATCH /files/fileId
Update	PUT /drive/v2/files/fileId PUT /files/fileId
Copy	POST /files/fileId/copy
Delete	DELETE /files/fileId
List	GET /files
Touch	POST /files/fileId/touch
Trash	POST /files/fileId/trash
Untrash	POST /files/fileId/untrash
Watch	POST /files/fileId/watch
emotvTrash	DELETE /files/trash
generateIds	GET /files/generateIds
Export	GET /files/fileId/export

VII. CONCLUSION AND FUTURE WORK:

In this paper, we proposed an easy way of storing and retrieving files from the cloud application with the single user verification. The dynamic audit protocol is used to verify the login system from the database. The data possession is done with for the authorization of the cloud application to have a legal way of developer's id. This enables the user to have a clear view on the verification and enhances to get the system to access files from their own drive of cloud applications.

As the future enhancement, a mobile application can be developed on the same process for invoking all file access through under single API.

REFERENCE

1. "An Efficient Protocol with Bidirectional Verification for Storage Security in Cloud Computing" , IEEE access, 2016.
2. D. Sudharsan, J. Adinarayana, S. Ninomiya, M. Hirafuji and T. Kiura "Dynamic Real Time Distributed Sensor Network Based Database Management System using Xml, Java and PHP Technologies" International Journal of Database Management Systems (IJDMS) Vol.4, No.1, February 2012.
3. "Secure and Efficient Attribute-Based Access Control for Multiauthority Cloud Storage",IEEE access,2015.
4. M.Armbrustetal.,''A view of cloud computing,''Commun. ACM,vol.53, no. 4, pp. 50–58, 2010.
5. M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, ''A survey on vehicular cloud computing,'' J.Netw.Comput.Appl.,vol.40,pp.325–344, Apr. 2014.
6. P. M. Mell and T. Grance, ''The NIST definition of cloud computing,'' Commun. ACM, vol. 53. no. 6, p. 50, 2011.
7. G. Han, A. Qian, J. Jiang, N. Sun, and L. Liu, ''A grid-based joint routing and charging algorithm for industrial wireless rechargeable sensor networks,'' Comput. Netw., vol. 101, no. 6, pp. 19–28, 2016.
8. T.Qiu,D.Luo,F.Xia,N.Deonauth,W.Si,andA.Tolba,''A greedy model with small world for improving the robustness of heterogeneous Internet of Things,'' Comput. Netw., vol. 101, no. 6, pp. 127–143, 2016.