Decentralized Information Accountability Framework for Information Sharing In Cloud Environment

Deepthi Srinivas

Assistant Professor, BNM Institute of technology, Bangalore, India, srinivasdeepthi29@gmail.com

Rajeev RK

Assistant Professor, Rai Technology University, Dollaballapur, India, rajeev.rk72@gmail.com

Shylaja BR

Assistant Professor, Rai Technology University, Dollaballapur, India, shylaja.b.r@gmail.com

Muruli R

Assistant Professor, Rai Technology University, Dollaballapur, India, murulir088@gmail.com

Abstract — Cloud computing is an alternative to a dedicated IT infrastructure. However, cloud computing has several challenges for both the customer and the cloud service provider including trust, security, and accountability. In the event of data leaks, or unauthorized data access, it is difficult to resolve the disputes between service provider and customer. This paper suggests an effective mechanism using accountability frame work to keep track of the actual usage of the users' data in the cloud to solve this problem. It provides an object-centered approach. This enables logging mechanism together with users' data based on proposed policies. Accountability is checking of authorization of policies for transparent data access. An automatic logging mechanism is provided using JAR programming, which improves security and privacy of data in cloud. Distributed auditing mechanism is explored to improve user data security and better control.

Index terms *- cloud computing, logging, auditing, accountability, data sharing*

---***---

1. INTRODUCTION

Cloud computing is a technology which uses internet and remote servers to store data and application. In cloud there is no need to install particular hardware, software on user machine, so user can get the required infrastructure on his machine in cheap charges/rates. Cloud computing is an infrastructure which provides useful, on demand network services to use various resources with less effort. Features of Cloud computing are, huge access of data, application, resources and hardware without installation of any software, user can access the data from any machine or anywhere in the world, business can get resource in one place, that means cloud computing provides scalability in on demand services to the business users.

Data usage in cloud is very large by users and businesses, so data security in cloud is very important issue to solve. Many users want to do business of their data through cloud, but users may not know the machines which actually process and host their data. While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data [1], [8].

Cloud provides three service models, which are; platform as a service, infrastructure as a service and software as a service. Under the Database as a service, this is having four parts which are as per mentioned below,

- Encryption and Decryption - For security purpose of data stored in cloud, encryption seems to be perfect security solution.
- Key Management - If encryption is necessary to store data in the cloud, encryption keys can't be stored there, so user requires key management.
- Authentication - For accessing stored data in cloud by authorized users.
- Authorization – Rights given to user as well as cloud provider.

To solve the security issues in cloud; other user can't read the respective users data without having access. Data owner should not bother about security of his data, and should not fear about damage done to his data by hacker; there is need for a security mechanism which will track usage of data in the cloud. Accountability is necessary for monitoring data usage, all actions of users like sending of file are cryptographically linked to the server, that performs them and server maintains secured record of all the actions of past and server can use the past records to know the correctness of action. It also provides reliable information about usage of data and it observes all the records, so it helps in creating trust, relationship and reputation. Hence accountability will aid for verification of authentication and authorization. It is powerful tool to check the authorization policies [9].Accountability describes authorization requirement for data usage policies. Accountability mechanisms, which rely on verification, are an attractive means to enforce authorization policies [7].

There are 7 phases of accountability:

1. Policy setting with data

2. Use of data by users

3. Logging

4. Merge logs

5. Error correctness in log

6. Auditing

7. Rectify and improvement.
These phases may change as per framework.

First the data owner will set the policies with data and send it to cloud service provider (CSP), data will be used by users and log of each record will be created and these logs will be further merged and error correction in the log record will be done. Auditing of log records will be done in the last phase.

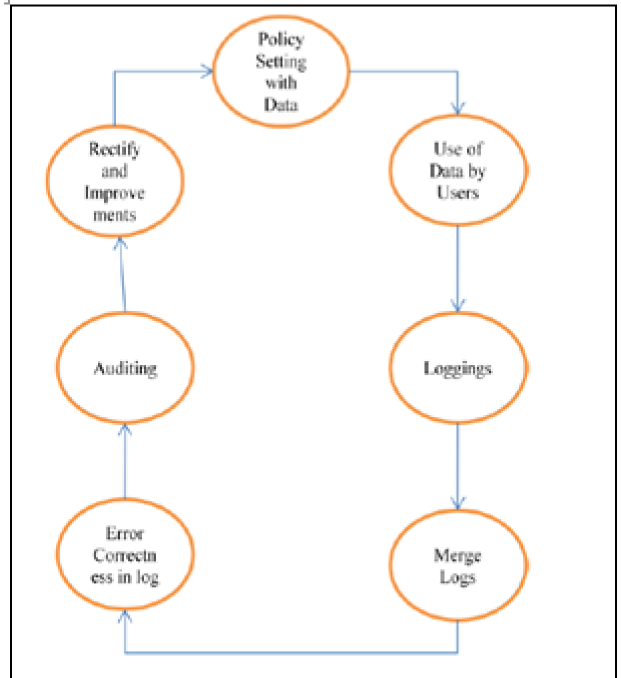


Fig 1:Phases of Accountability

In the Fig 1 Steps of accountability are given, these are 7 steps each step is important to perform next step, accountability is nothing but validation of user actions i.e to verify if user has rights for accessing the data or not. Log records are created for each action made by the user and they are merged together and checked for correctness and sent to the CSP where auditing is done.

This flow diagram depicts majorly the phases of how the accountability is established without disrupting the ongoing cloud services and providing data transparency to the owner of the data and also he can clearly track how his data is travelling and what actions are being carried out on his data and who is doing the action. These actions can be controlled by the owner as he would have previously mentioned the access control rights of his data for various users who would like to do business with him.

2. LITERATURE SURVEY

In this section review related works addressing security in cloud is given. Security issue is very important issue in cloud, there are many techniques available so here is review of all these.

S. Pearson et al describes privacy manager mechanism in which user's data is safe on cloud , in this technique the user's data is in encrypted form in cloud and evaluating is done on encrypted data, the privacy manager make readable data from result of evaluation manager to get the correct result. In obfuscation data is not present on Service provider's machine so there is no risk with data, so data is safe on cloud, but this solution is not suitable for all cloud application, when input data is large this method can still require a large amount of memory[2]. In [3], the authors present procedural and technical solution both are producing solution to accountability to solving security risk in cloud in this mechanism these policies are decided by the parties that use, store or share that data irrespective of the jurisdiction in which information is processed. But it has limitation that data processed on SP is in unencrypted at the point of processing .so there is a risk of data leakage. In [4], the author gives a language which permits to serve data with policies by agent; agent should prove their action and authorization to use particular data. In this logic data owner attach Policies with data, which contain a description of which actions are allowed with which data, but there is the problem of Continuous auditing of agent, but they provide solution that incorrect behavior should monitor and agent should give justification for their action, after that authority will check the justification. In [5], authors gives a three layer architecture which protect information leakage from cloud, it provides three layer to protect data, in first layer the service provider should not view confidential data in second layer service provider should not do the indexing of data, in third layer user specify use of his data and indexing in policies, so policies always travel with data. In [6], authors present accountability in federated system to achieve trust management. The trust towards use of resources is accomplished through accountability so to resolve problem for trust management in federated system they have given three layers architecture, in first layer is authentication and authorization in this authentication does using public key cryptography. Second layer is accountability which perform monitoring and logging. The third layer is anomaly detection which detects misuse of resources. This mechanism requires third party services to observe network resources.

3. PROPOSED WORK

Cloud computing is a large infrastructure which provides many services to user without installation of resources on their own machine. This is the ‘pay as you use’ model. Examples of the cloud services are Yahoo email, Google, Gmail and Hotmail. There are many users, businesses, government uses cloud, so data usage in cloud is large. So data maintenance in cloud is complex. Many data owners can do business of their data using cloud. For example an artist can sell his painting using cloud but he wants to be assured that his paintings on cloud are not misused.

There is need to provide technique which will audit data in cloud. On the basis of accountability, a mechanism is proposed which makes data usage transparent i.e. data owner should get information about usage of his data. This mechanism supports accountability in distributed environment, data owner should not bother about safety of his data, he can be assured that his data is handled according to service level agreement and his data is safe on cloud. Data owner will decide the access rules and policies and user will handle data using this rule and logs of each data access will be created. In this mechanism there are two main components i.e. logger and log harmonizer.

There is need to provide technique which will audit data in cloud. On the basis of accountability, a mechanism is proposed which makes data usage transparent i.e. data owner should get information about usage of his data. This mechanism supports accountability in distributed environment, data owner should not bother about safety of his data, he can be assured that his data is handled according to service level agreement and his data is safe on cloud. Data owner will decide the access rules and policies and user will handle data using this rule and logs of each data access will be created. In this mechanism there are two main components i.e. logger and log harmonizer.

The logger is with the data owner's data, it provides logging access to data and encrypts log record by using public key which is given by data owner and sends it to log harmonizer. The log harmonizer is performing the monitoring and rectifying, it generates the master key it holds decryption key decrypting the logs at client side, it sends key to client. In this mechanism data owner will create private key and public key it includes his policies like access policies and logging policies sent with data. cloud service provider Authentication has been done using open SSL based certificates after authentication of cloud service provider user will be able to access data in JAR, log of each data usage is created and encrypted using public key and it automatically sends to log harmonizer, for integrity, log records are signed by entity which is using the data and log records are decrypted and accessed by owner. In push mode logs are automatically sent to data owner and in pull mode owner can demand logs, so he can see access of his data at anytime, anywhere and he can do monitoring of his data [1].

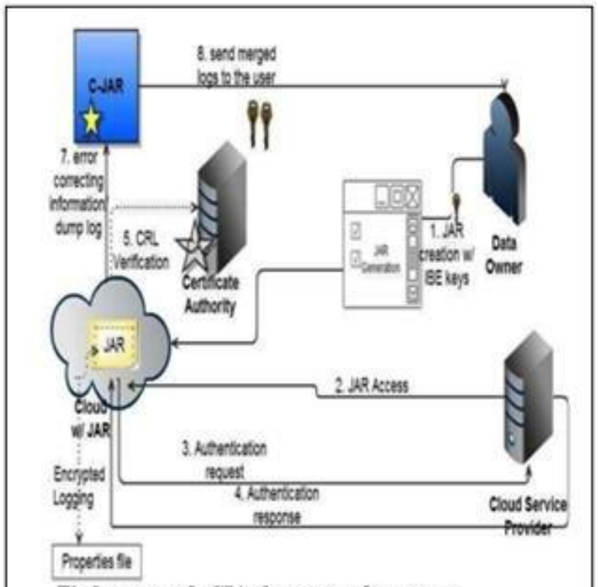


Fig2.proposed CIA frame work system architecture:

Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data (in Fig. 2). The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner can opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption (Fig. 2).

To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity is verified. They can be accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer (Fig. 2). The proposed framework prevents various attacks such as detecting illegal copies of users' data. This work is different from traditional logging methods which use encryption to protect log files. With only encryption, their logging mechanisms are neither automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud.

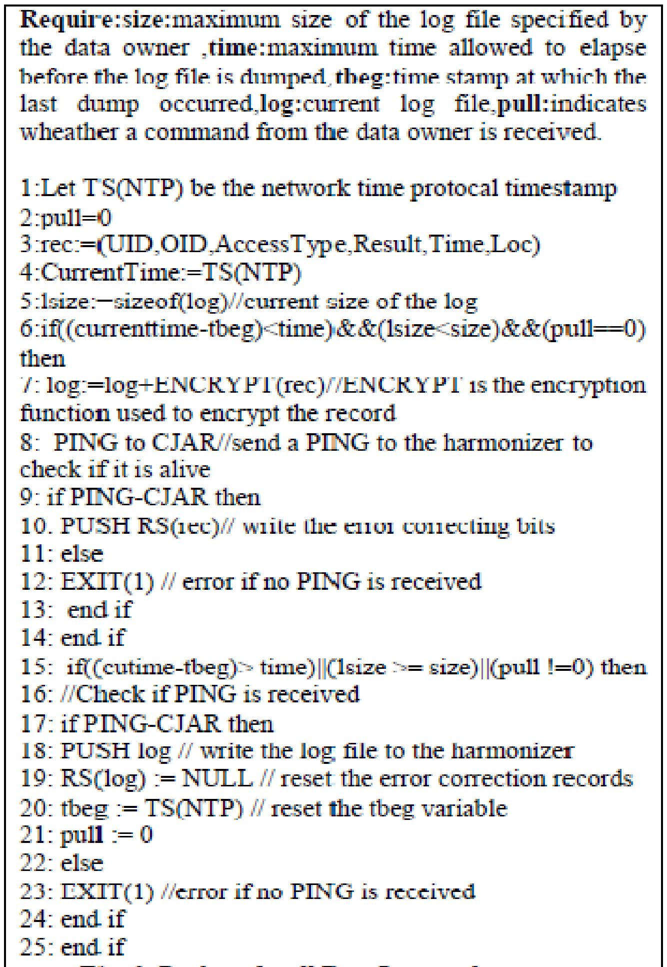
3.1. Push and Pull Mode

To allow users to be timely and accurately informed about their data usage, the distributed logging mechanism is complemented by an innovative auditing mechanism. This approach supports two auditing mode:

1) Push mode: In this mode, the logs are periodically pushed to the data owner (or auditor) by the harmonizer. The push action will be triggered by either type of the following two events: one is when the time elapses for a certain period according to the temporal timer inserted as part of the JAR file; the other is when the JAR file exceeds the size stipulated by the content owner at the time of creation. After the logs are sent to the data owner, the log files will be dumped, so as to free the space for future access logs. Along with the log files, the error correcting information for those logs is also dumped. This push mode is the basic mode which can be adopted by both the Pure Log and the Access Log, regardless of whether there is a request from the data owner for the log files. This mode serves two essential functions in the logging architecture: 1) it ensures that the size of the log files does not explode and 2) it enables timely detection and correction of any loss or damage to the log files. Concerning the latter function, we notice that the auditor, upon receiving the log file, will verify its cryptographic guarantees, by checking the records' integrity and authenticity. By construction of the records, the auditor will be able to quickly detect forgery of entries, using the checksum added to each and every record.
2) Pull mode. This mode allows auditors to retrieve the logs anytime when they want to check the recent access to their own data. The pull message consists simply of an FTP pull command, which can be issues from the command line. For naive users, a wizard comprising a batch file can be easily built. The request will be sent to the harmonizer, and the user will be informed of the data locations and obtain a integrated copy of the authentic and sealed log file.

The log retrieval algorithm for the Push and Pull modes is outlined in Fig. 3. The algorithm presents logging and synchronization steps with the harmonizer in case of Pure Log. First, the algorithm checks whether the size of the JAR has exceeded a stipulated size or the normal time between two consecutive dumps has elapsed. The size and time threshold for a dump are specified by the data owner at the time of creation of the JAR. The algorithm also checks whether the data owner has requested a dump of the log files. If none of these events has occurred, it proceeds to encrypt the record and write the error correction information to the harmonizer. The communication with the harmonizer begins with a simple handshake. If no response is received, the log file records an error. The data owner is then alerted through e-mails, if the JAR is configured to send error notifications. Once the handshake is completed, the communication with the harmonizer proceeds, using a TCP/IP protocol.

If any of the aforementioned events (i.e., there is request of the log file, or the size or time exceeds the threshold) has occurred, the JAR simply dumps the log files and resets all the variables, to make space for new records. In case of Access Log is modified by adding an additional check . Precisely, the Access Log checks whether the CSP accessing the log satisfies all the conditions specified in the policies pertaining to it. If the conditions are satisfied, access is granted; otherwise, access is denied. Irrespective of the access control outcome, the attempted access to the data in the JAR file will be logged. The auditing mechanism has two main advantages. First, it guarantees a high level of availability of the logs. Second, the use of the harmonizer minimizes the amount of workload for human users in going through long log files sent by different copies of JAR file.



```
Require:size:maximum size of the log file specified by the data owner ,time:maximum time allowed to elapse before the log file is dumped,tbeg:time stamp at which the last dump occurred,log:current log file,pull:indicates wheather a command from the data owner is received.

1:Let TS(NTP) be the network time protocal timestamp
2:pull=0
3:rec:=(UID,OID,AccessType,Result,Time,Loc)
4:CurrentTime:=TS(NTP)
5:lsize:=sizeof(log)//current size of the log
6:if((currenttime-tbeg)<time)&&(lsize<size)&&(pull==0) then
7: log:=log+ENCRYPT(rec)//ENCRYPT is the encryption function used to encrypt the record
8:  PING to CJAR//send a PING to the harmonizer to check if it is alive
9: if PING-CJAR then
10. PUSH RS(rec)// write the error correcting bits
11: else
12: EXIT(1) // error if no PING is received
13:  end if
14: end if
15:  if((cutime-tbeg)> time)||(lsize >= size)||(pull !=0) then
16: //Check if PING is received
17: if PING-CJAR then
18: PUSH log // write the log file to the harmonizer
19: RS(log) := NULL // reset the error correction records
20: tbeg := TS(NTP) // reset the tbeg variable
21: pull := 0
22: else
23: EXIT(1) //error if no PING is received
24: end if
25: end if
```

Fig. 3. Push and pull PureLog mode.

4. CONCLUSION AND FUTURE RESEARCH

An innovative approach is proposed for automatically logging any access to the data in the cloud together with an auditing mechanism. This approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of this work is that it enables the data owner to audit even those copies of its data that were made without his known text files, usage control for executables, and generic accountability and provenance controls can be supported.

In the future, this approach can further be improved to verify the integrity of the JRE and the authentication of JARs [13]. In the long term, plan to design a comprehensive and more generic object-oriented approach to facilitate autonomous protection of traveling content can be made. A variety of security policies, like indexing policies for text files, usage control for executables, and generic accountability and provenance controls can be supported.

5. REFERENCES

[1] Smitha Sundareswaran, Anna C. Squicciarini and Dan Lin, "Ensuring Distributed Accountability for Data Sharing in the Cloud", IEEE Transaction on dependable a secure computing, VOL. 9, NO. 4, pg 556-568, 2012.

[2] S. Pearson , Y. Shen, and M. Mowbray," A privacy Manager for Cloud Computing," Proc. Int'l Conf. Cloud Computing (cloudcom), pp.90-106,2009.

[3] S. Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud," *Proc First Int'l conf. Cloud* Computing, 2009.

[4] R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu,- A Logic for Auditing Accountability in Decentralized Systems, Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005.

[5] A.Squicciarini, S. Sundareswaran and D. Lin, "Preventing Information Leakage from Indexing in the Cloud," *Proc. IEEE Int'l* Conf. Cloud Computing, 2010.

[6] B. Chun and A. C. Bavier ,"Decentralized Trust Management and Accountability in Federated System," *Proc. Ann. Hawaii Int'l Conf. System Science (HICSS), 2004.*

[7] B. Crispo and G. Ruffo, - Reasoning about Accountability within Delegation, Proc. Third Int'l Conf. Information and Comm. Security (ICICS), pp. 251-260, 2001.

[8] S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, "Promoting Distributed Accountability in the Cloud", *Proc. IEEE* Int'l Conf. Cloud Computing, 2011.

[9] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigen-baum, J. Hendler, and G.J. Sussman,"Information Accountability', Comm. ACM, vol. 51, no. 6, pp. 82-87, 2008.

[10] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1993.

[11] Praveen Gauravaram, John Kelesy, Lars Knudsen, and Soren Thomsen,"On Hash function using Checksums".