R. Reni Hena Helan *et al*, International Journal of Computer Science and Mobile Applications,
Vol.2 Issue. 4, April- 2014, pg. 23-30 **ISSN: 2321-8363**

Design of a Framework for Trusted Data Monitoring in the Cloud

R. Reni Hena Helan[1], S. Arunkumar[2]

[1]M.Tech Student, Department of CSE, SRM University, Ramapuram, renihenahelan@gmail.com
[2]Assistant Professor, Department of CSE, SRM University, Ramapuram, arunatr85@gmail.com

Abstract

Cloud Computing is a technology that provides network based services on-demand. Data owners host their private data in the cloud and worry about unauthorized access of their data. They feel uncomfortable about any user misusing their private data. This insecure feeling of data owners holds them back from using cloud services. Any unauthorized users accessing the owner's private data leads to accountability issues. To solve the accountability issue, a mechanism to monitor the actual data usage is proposed. This approach grants access rights to users based on their role and also monitors every access to the owner's data, verifying that the service level agreements have been violated or not.

Keywords: Cloud Computing, Accountability, Service level agreement

1. Introduction

Cloud Computing technology provides advantages to end users and business organizations. Few notable advantages are cost efficiency, increased storage capacity, backup and recovery, continuous resource availability and location independence. Inspite of these advantages, the biggest issue with the cloud is the 'Security'. Though there are several advantages with cloud, it also imposes several security threats related to outsourced user's data. Since private data is hosted in the cloud and they are being processed at remote machines and are administered by the cloud service providers (CSP), the users are worried about loss of data control in the cloud. There are various reasons for the CSP to involve in unfaithful disclosure or leakage of user's data to any external entity that may turn out to be a serious privacy and security concern for any user towards his/her data. These security issues may lead to the decline of cloud usage among users. So, there should be a specific mechanism to track every data access in the cloud and should solve the accountability issue. Accountability means checking violation of user authorization policies.

Cloud Users do not know the machines where their data are processed, so they start bothering about losing control over their data. There are no specific mechanisms to check if the service level agreements made between the data owner and the end users have been preserved or not. Data is often being outsourced in cloud, leading to accountability issues and manipulation of personally identifiable information. The monitoring mechanism involves third party services. The third party is an external entity who can behave unfaithfully while the data is disclosed during the auditing process. Moreover, simply relying on a third party auditor without applying any cryptographic technique on the user's data may turn the situation even more worse. So, downloading the user's data alone for auditing will not help for verifying the integrity of user's data since downloading the entire data is expensive because of I/O and transmission cost through the network.

To solve these security problems with the cloud, users are assigned access rights based on their role and if any user tries to violate the assigned access rights the data owner is notified with a log report stating about the service level agreement violation.

The rest of the paper is arranged as follows: Section 2 discusses about the problem statement and about the Cloud Data Accountability(CDA) framework. This section also explains the automated logging mechanism. The algorithm and the flowchart for the auditing modes are described in the same section. Section 3 describes the security issues of the CDA framework and the possible solutions. At last Section 4 concludes the paper and gives an overview on the future work.

2. Problem statement

Considering the security problems as mentioned above, a decentralized framework called the Cloud Data Accountability (CDA) framework is proposed. This CDA framework integrates the aspects of access control, usage control and authentication of users. The usage control is implemented by means of a role based approach, which assigns access control rights of any data in the cloud based on the user's role. This mechanism supports accountability in the cloud environment. The main components of the CDA framework are the log generator and the log converger. The log generator encrypts the log records using the public key of the data owner and it is sent to the log converger. The log converger creates a master key for decrypting the logs to detect and correct errors.

The data owner creates a JAR file (JAVA ARCHIVE file) which contains the access control policies and log generation policies, this JAR file is sent to the Cloud Service Provider(CSP). The authenticity of the CSP is verified based on the SSL certificates. For every JAR access, logs are being generated by the log generator and they are encrypted using the public key of the data owner and these logs are sent to the log converger for integrity verification by decrypting the logs. Many logs for the same JAR access is merged and sent to the data owner in the form of log record using two modes namely, the automatic mode and the on request mode. These modes help the data owner in monitoring his/her data usage in the cloud.

The main aim of this paper is to provide a mechanism that achieves cloud data accountability. Access rights such as read, write, copy are granted to the users using conventional access control approaches. The cloud service provider grants data access to users based on these specified access rights. Every data access is tracked using this mechanism. The following are the main requirements satisfied by our approach,

1. Every data access is correctly and automatically logged.
2. There are recovery mechanisms to recover the log files.
3. Log files are retrievable anytime from anywhere.

2.1 Cloud Data Accountability Framework

2.1.1 Major components

The major components of the CDA framework are the log generator and the log converger. The log generator is responsible for the generation of log records for every JAR access and the log converger is responsible for decrypting logs for error correction, merging and sending logs to the data owner. The logs are encrypted using the public key generated by the data owner and the logs are decrypted by the data converger using the master key for integrity verification of logs. The major components of the CDA framework are outlined in Figure.1.

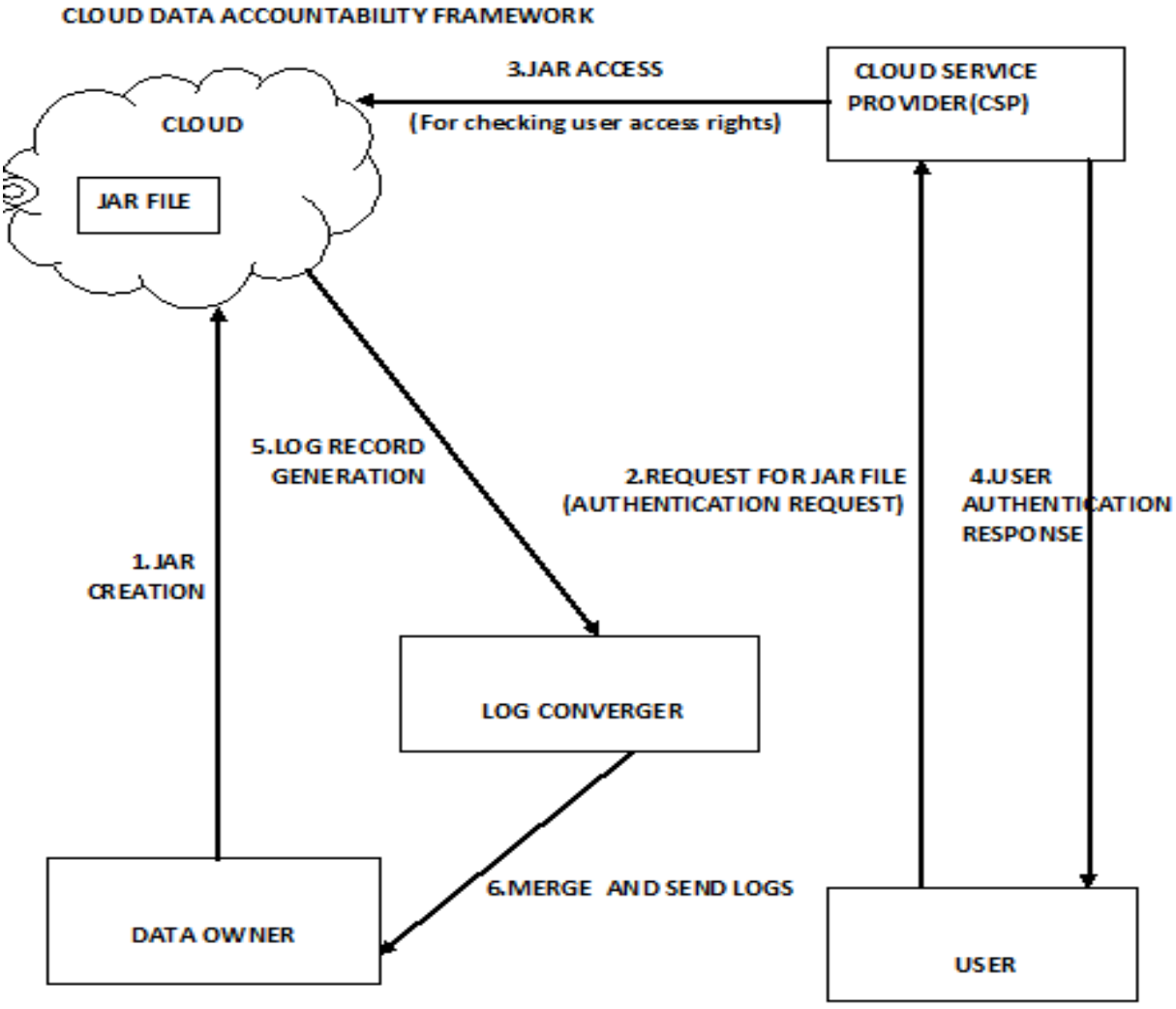

Figure. 1. Cloud Data Accountability Framework

2.1.2 Jar file creation

The data owner creates a JAR file that encloses the original encrypted data, the access control policies and the logging policies. These access control policies help the CSP in authenticating the user and also in granting access rights to the users. These access control policies are fixed by the data owners based on the user's role to be played in accessing the cloud data. The structure of a JAR File showing the outer Jar and the Inner Jar components are shown in Figure.2.

2.1.3 User Authentication

The users are authenticated by the cloud service provider and the requested JAR file access will be granted to the user based on the previously specified access control rights based on user's role.

2.1.4 Log generator structure

The automated logging mechanism is initiated for every data access and the log generator begins to generate log records for every data access in the cloud. The log generator is a nested JAR file storing user data and respective log files. The outer JAR contains access control policies for each user based on their role and is responsible for user authentication for a particular JAR file based on the specified access rights. The outer JAR may contain one or more inner JARs. The outer JAR helps in identifying the correct inner JAR based on user request.
The inner JAR consists of the data in encrypted form, class files initiating log generation, displaying data in required format and generation of log file for each encrypted data.

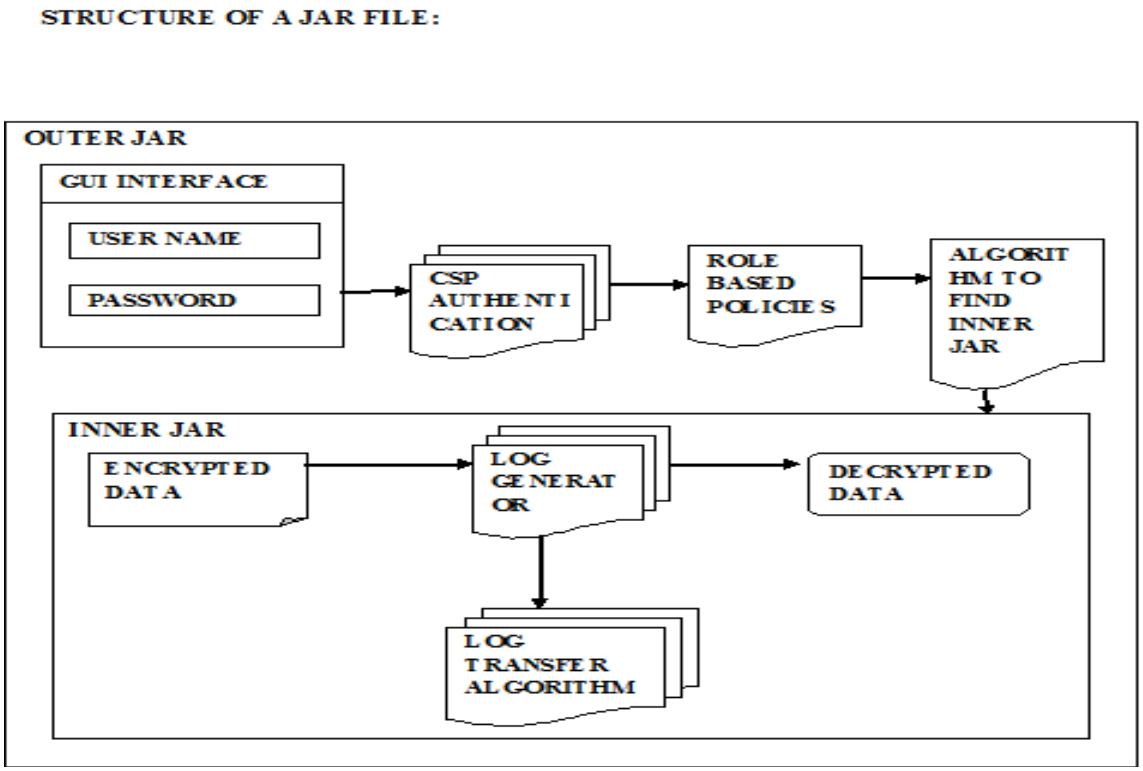

Figure. 2. Structure of a JAR File

2.1.5 Log record generation

The log generator creates log records for every data access and sents to the log converger. These logs are forwarded to the data owner using which necessary actions can be taken in the case of user violating the rights.
The log record contains user id, access type, access time, access location. The general form of the log record is,

r= (UID,UACTION,TIME,LOC,H((UID,UACTION,T,LOCATION)ri-1…r1),sig)
where,
r = log record
UID= user identifier
UACTION=action performed on user's data

T= Access time
LOCATION= access location
H((UID,UACTION,T,LOCATION)ri-1…r1) =checksum
sig = signature of record generated by server

The checksum is calculated for each record using the hash function
H[i] = f(H[i-1], m[i])

2.2 Auditing Modes

The two auditing modes help the data owner to monitor the usage of his/her data hosted in the cloud.

2.2.1 Automatic mode

The size of the log record and the maximum time that should elapse before dumping of log files are fixed by the data owner. If any one of these condition occurs, logs are being sent to the data owner.

2.2.2 Onrequest mode

The onrequest mode can be used by the data owners, if he/she suspects of data being misused. This mode helps the owner to monitor the data usage immediately.

2.3 The Log Retrieval Algorithm

The algorithm outlined here explains the log generation and log converging process. The size and time parameters are checked for dumping of logs to the data owners. When any of these parameters reach their fixed limit, automatic mode occurs and the onrequest mode works on the data owner's request.

Required Parameters :
size: maximum size of the log file specified by the data owner
time: maximum time allowed to elapse before the log file is dumped
tdump: timestamp at which the last dump occurred
log: Current log file
req: indicates whether a command from the data owner is received.

1.Find the ctime using TS (NTP) - Network time protocol timestamp, ctime := TS (NTP)
2. Initialize req = 0
3. r := {UID, OID UACTION, T, LOCATION}
4. Find the current size of the log,
Lsize := sizeof (log)
5. if ((ctime -tdump) < time) && (lsize < size) && (req = =0) then
6. log := log + E(r) // encryption of record
7. PING to LOG CONVERGER // to check if converger is alive
8. If PING- LOG CONVERGER then
9. send RS (r) // include error correcting bits
10. else EXIT(1) // error if no PING is received
11. end if
12. end if
13. if ((ctime - tdump) > time) || (lsize >= size) || (req != 0) then
14. If PING- LOG CONVERGER then

15. send log // send the log file to the converger
16. RS(log) := NULL // reset the error correction records
17 .tdump := TS(NTP) // reset the tdump variable
18. req := 0
19. else
20. EXIT(1) // error if no PING is received
21. end if
22. end if

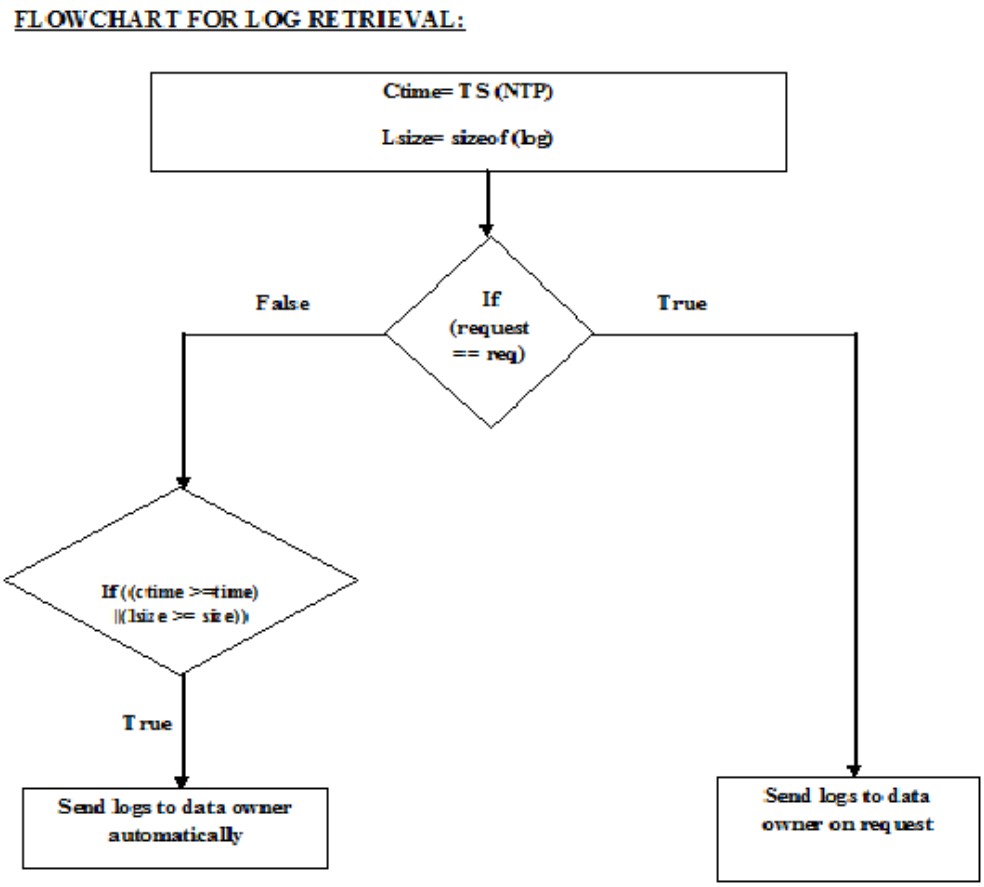

Figure. 3. Flowchart indicating automatic and onrequest mode

2.4 References and Citations

This section discusses related works on cloud security and privacy issues. Pearson et al. proposed mechanism to address accountability issues by developing a privacy manager[5].The encrypted cloud data is processed by the privacy manager to check the data accountability. The problem with this is that the data security cannot be preserved once the data is being revealed to the privacy manager. The privacy manager do not guarantee protection after the data being disclosed. In[8], B.Chun and A.C.Bavier presents a layered architecture that addresses trust and accountability issues in federated systems. They focus on trusted relationships for accountability issues and also depend on third party services for data monitoring. The authors of [7] proposed a work based on the usage of policies applied along with data to address accountability issues in distributed environment.

3. Security Analysis

Possible attacks to the CDA framework are discussed here. This framework encounters the following security attacks and the solution for the same is provided here.

3.1.1 Jar copy attack

The attack copies the JAR files and assumes that it will allow him to access the JAR file data without being noticed by the data owner. But every access to a JAR file creates log record and being sent to the data owner for auditing. Even for additional copies of the JAR, log records will be generated and sent to the data owner. If any copy of a JAR file is moved to a location that could not be accessed by the log converger, access to that particular JAR file is cancelled. The JAR file becomes inaccessible.

3.1.2 Disassembling attack

This is another attack possible in our scenario. Once the Jar files has been obtained by the hacker, he/she tries to disassemble the JAR file. But the JAR file contains data and logs in encrypted format. The hacker knows only the public key used for encrypting the logs, there are no possibilities to obtain the master key used for decrypting the logs.

The attacker cannot modify the log file content after disassembling the JAR. The integrity checking mechanism for logs will detect any modification in the log records since the integrity checks added to each record will not match at the time of verification by the log converger. The Reed-Solomon encoding is for this checking mechanism using which the log converger can easily detect corrupted logs.

4. Conclusion

This paper presents an effective mechanism to track and monitor the data usage in the cloud. This mechanism authenticates the user and grants the specified access rights based on the user's role then providing the JAR access. Existence of duplicate copies of data can also be detected by this mechanism since logs are being created for every copy of the same JAR file. Making the data usage transparent is the objective of this paper.

In future, improvement on this paper can be made by considering the areas like implementing various security policies, more usage control policies and also in reducing the log record generation time.

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

[3] Ryan K L Ko, Peter Jagadpramana, Miranda Mowbray, Siani Pearson, Markus Kirchberg, Qianhui, "TrustCloud: A Framework for Accountability and Trust in Cloud Computing" HP Laboratories, pp 1 – 7, HPL-2011

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

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

[6] S. Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud, " *Proc First Int'l conf. Cloud* Computing, 2009.
[7] 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.
[8] B. Chun and A. C. Bavier ,"Decentralized Trust Management and Accountability in Federated System," *Proc. Ann. Hawaii Int'l Conf. System Science (HICSS), 2004.*
[9] B. Crispo and G. Ruffo, "Reasoning about Accountability within Delegation," Proc. Third Int'l Conf. Information and Comm. Security (ICICS), pp. 251-260, 2001.
[10] 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.

A Brief Author Biography

R. Reni Hena Helan – Completed Bachelor of Engineering in Computer Science and Engineering and Presently doing M.TECH in Computer Science and Engineering at SRM University, Ramapuram.

S. Arun Kumar – Working as an Assistant Professor in the Department of Computer Science and Engineering at SRM Universtiy, Ramapuram.

