Data Sharing: Efficient Distributed Accountability in Cloud using Third Party Auditor

M.Vanitha, R.Raju

Abstract- We propose a Third party auditor(TPA) between data owner and cloud service provider(CSP) which reduce the burden of data owner to audit the data in the cloud and it also make the data owner free from worrying about the data lose in cloud storage . To highlight the security purpose we introduce an novel highly decentralized information accountability framework and object-centered approach. we enclosed the data and set of policies for the user access which make the data to be secured from the malicious action made in the cloud. The JAR programmable capability which is used to create both dynamic and traveling object. When any access is made to the user's data will be trigger the authentication and automated logging control to JARs. A distributed auditing mechanism is used to control the users.

Keywords- cloud service provider, Third party auditor, accountability, data sharing.

I. INTRODUCTION

Cloud computing is the delivery of computing and storage capacity as a service to a heterogeneous community of end-recipients. The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts services with a user's data, software and computation over a network.

There are three types of cloud computing:

- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS), and
- Software as a Service (SaaS).

Using Infrastructure as a Service, users rent use of servers (as many as needed during the rental period) provided by one or more cloud providers. Using Platform as a Service, users rent use of servers and the system software to use in them. Using Software as a Service, users also rent application software and databases. The cloud providers manage the infrastructure and platforms on which the applications run.

Cloud architecture, the systems architecture of the software systems involved in the delivery of cloud computing, typically involves multiple cloud components communicating with each other over a loose coupling mechanism such as a messaging queue. Elastic provision implies intelligence in the use of tight or loose coupling as applied to mechanisms such as these and others.

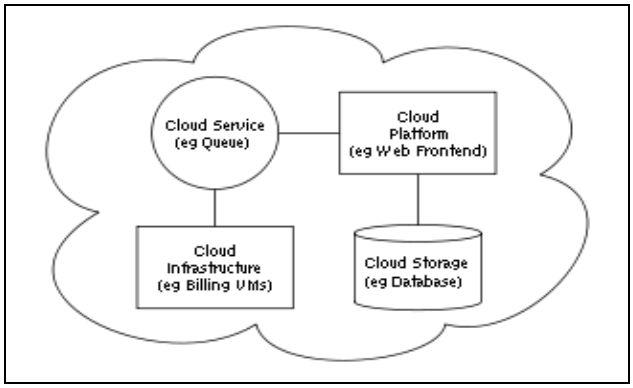


Fig 1: Cloud computing sample architecture

While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data. Such fears are becoming a barrier to cloud services provider. two issues are there

- Data handling can be done by the cloud service provider (CSP) to other cloud users (i.e. entities) and these cloud users may hand over to the other users so on.
- Cloud made the users whenever they can join and leave the cloud in the flexible manner.

These issues make the data handling as more complex and tedious task in the cloud. To overcome these we introduce Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Third Party Auditor is used to reduce the burden for data owner to audit the data. IBE technique is used for encrypting purpose. Third party auditor is placed in between the data owner and the cloud service provider.

II. CLOUD INFORMATION ACCOUNTABILITY

Cloud information accountability framework is introduced in order to proceed the automated logging and distributed Auditing mechanism which consists of two component namely logger and log harmonizer.

1 Logger componen:

Using this key, the data owner will create the logger component in JAR file along with store the data items. The JAR file contains outer JAR and Inner JAR. The major accountability of JAR is to hold the authentication of entities and it requires accessing the data that are stored in the JAR file. Every Inner JAR consists of encrypted data and class files to recover the log file, the log file for every encrypted item. It offers Pure Log and Access Log. The PureLog, records only the general information about every access. Access Log, Records general information and also the time duration.

Manuscript received on April, 2013.

M.Vanitha, M.Tech. Student, Department of Information Technology, Sri Manukula Vinayagar Engineering College, Puducherry, Puducherry, India

R.Raju, M.Tech. Student, Department of Information Technology, Sri Manukula Vinayagar Engineering College, Puducherry, Puducherry, India

It supports four types of actions, i.e., perform has one of the following four values: view, download, timed access, and Location-based access.

- PureLog. Its main task is to record every access to the data. The log files are used for pure auditing purpose.
- AccessLog. It has two functions: logging actions and enforcing access control. In case an access request is denied, the JAR will record the time when the request is made. If the access request is granted, the JAR will additionally record the access information along with the duration for which the access is allowed.

To carry out these functions, the inner JAR contains a class file for writing the log records, another class file which corresponds with the log harmonizer, the encrypted data, a third class file for displaying or downloading the data (based on whether to have a PureLog, or an AccessLog), and the public key of the IBE key pair that is necessary for encrypting the log records. No secret keys are ever stored in the system. The outer JAR may contain one or more inner JARs, in addition to a class file for authenticating the servers or the users, another class file finding the correct inner JAR, a third class file which checks the JVM's validity using oblivious hashing.

2. Log harmonizer

The encryption of the log file avoids the unauthorized change to the file by attackers. The log harmonizer is to hold the log file corruption and the logger send the error correction information in to the log harmonizer. To guarantee trustworthiness of the logs, every record is signed by the entity accessing the content. After that the entity records should be hashed together and to generate the chain structure, so it can easily detect the errors and missing records. To verify the integrity, the encrypted log files can be converted in to the decrypted form. Every log harmonizer is in charge of copies of the logger components contains the similar set of data items.

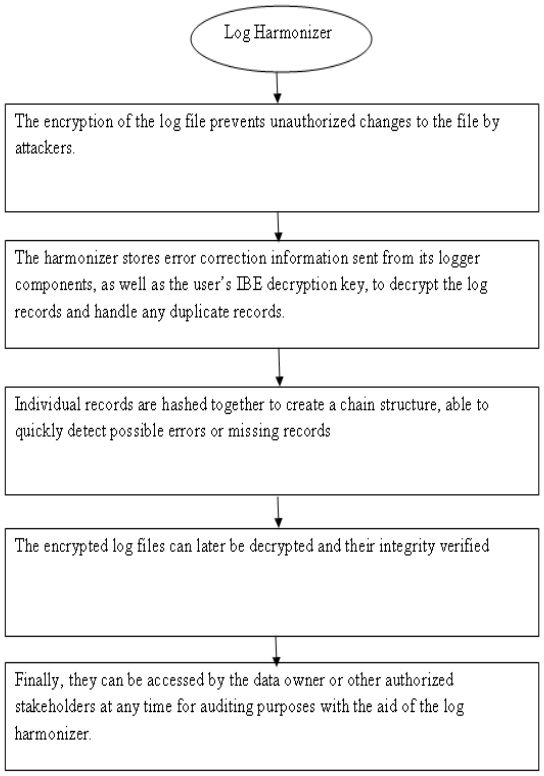


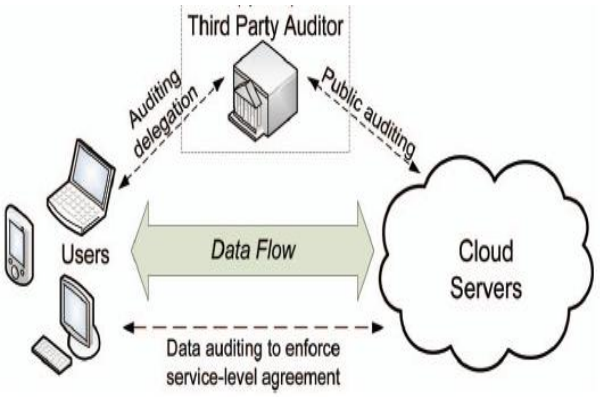
Fig 2 Log Harmonizer

3 Third party auditor

The data owners having large amount of outsourced data and the task of auditing the data correctness in a cloud environment can be difficult and expensive for data owners. So, the communication between the data owners and cloud servers through third party auditing. Third party auditing provides a transparent yet **cost-effective** method for establishing trust between data owner and cloud server.

In order to save the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors (TPA) and be worry-free to use the cloud storage services. whenever data corruption has been detected during the storage correctness verification,

This scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s). dynamic data support to ensure the correctness and availability of users' data in the cloud.



In fact, based on the audit result from a TPA, the released audit report would not only help owners to evaluate the risk of their subscribed cloud data services, but also be beneficial for the cloud service provider to improve their cloud based service platform. The third party auditor performs the every auditing operation. The set of policies can be implemented i.e., the normal users can be communicate with cloud servers through the third party auditing, the particular data can be downloaded at one time and the next data can be download at a particular time interval.

III. LOGGING MECHANISM

1 Log structure

A logger component is a nested Java JAR file which stores a user's data items and corresponding log files. Each File consists of two JARs.

they are

- Inner JAR and
- Outer JAR

Inner JAR consists of data and outer JAR consists of set of policies regarding the data access.

Inner jar

This figure shows inner JAR data process, it has encrypted data and log record . the log record can be generated as pure log and access log where pure log have only the general information about every access made by users in the cloud and access log have general information and also the time duration about the access.

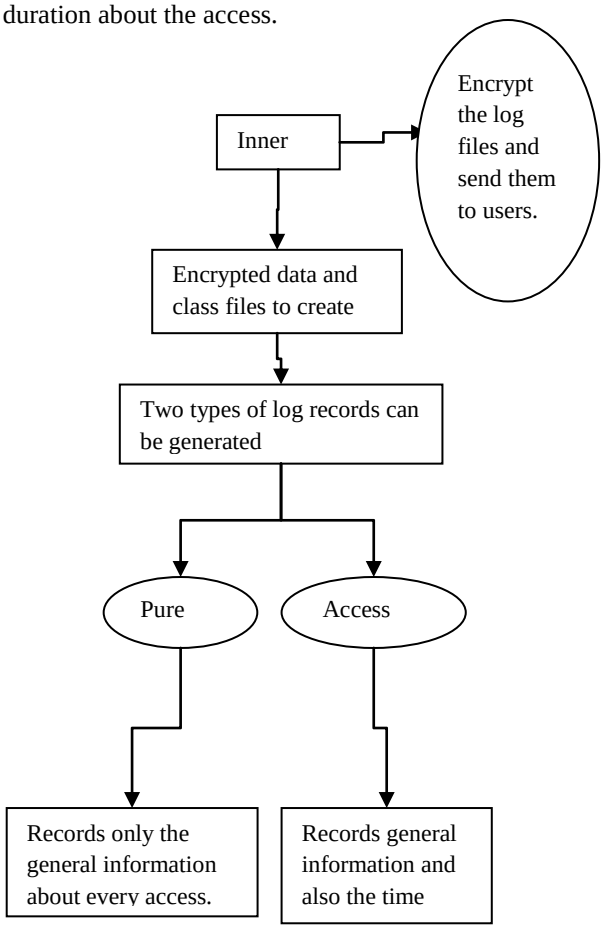


Fig 3.1 a. Inner JAR

Outer jar

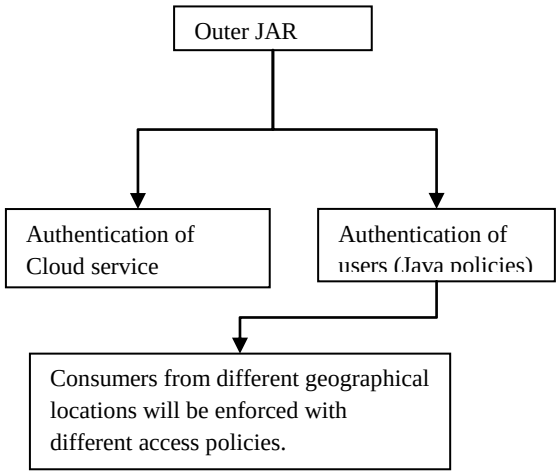


Fig 3.1(b) Outer JAR

This figure shows the authentication process made in the outer JAR. It will authenticate the cloud service provider and users by means of the policies whenever the user do any malicious action it will be automatically intimated to the data owner

2 Correctness of the log records

To verify the correctness of log record we presented the action such as view, download, timed access and location based access.

View: user can only read the data but is not allowed to save a copy of it anywhere permanently. For this type of action, the Pure Log will simply write a log record about the access, the data are encrypted and stored in the inner JAR. When there is a view-only access request, the inner JAR will decrypt the data on the fly and create a temporary decrypted file. The decrypted file will then be displayed to the users. to prevent the use of some screen capture software, the data will be hidden whenever the application viewer screen is out of focus. The content is displayed using the headless mode in Java on the command line when it is presented to a CSP.

Download: The cloud user is allowed to save a raw copy of the data. If PureLog is adopted, the user's data will be directly downloadable in a pure form using a link. When an entity clicks this download link, the JAR file associated with the data will decrypt the data and give it to the entity in raw form. In case of AccessLogs, the entire JAR file will be given to the entity. If the entity is a human user, he/she just needs to double click the JAR file to obtain the data. If the entity is a CSP, it can run a simple script to execute the JAR.

Timed access: This action is combined with the view-only access, and it indicates that the data are made available only for a certain period of time. The Purelog will just record the access starting time and its duration, while the AccessLog willenforce that the access is allowed only within the specified period of time. The duration for which the access is allowed is calculated using the Network Time Protocol. To enforce the limit on the duration, the AccessLog records the start time using the NTP, and then uses a timer to stop the access. Naturally, this type of access can be enforced only when it is combined with the View access right and not when it is combined with the Download.

Location-based access: In this case, the PureLog will record the location of the user. The AccessLog will verify the location for each of such access. The access is granted and the data are made available only to cloud user located at locations specified by the data owner.

IV. LOG HORMONIZER

The log hormonizer help to encryption the log file avoids the unauthorized change to the file by attackers. The log harmonizer is to hold the log file corruption and the logger send the error correction information in to the log harmonizer. To guarantee trustworthiness of the logs, every record is signed by the entity accessing the content. After that the entity records should be hashed together and to generate the chain structure, so it can easily detect the errors and missing records. To verify the integrity, the encrypted log files can be converted in to the decrypted form. Every log harmonizer is in charge of copies of the logger components contains the similar set of data items. 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.

Algorithm

Distributed auditing mechanism including the algorithms for data owners to query the logs regarding their data.

Push and Pull Mode

To allow users to be timely and accurately informed about their data usage, Distributed logging mechanism is complemented by an innovative auditing mechanism. It support two complementary auditing modes:

1) push mode.
2) pull mode.

Push mode: In this mode, the logs are periodically pushed to the data owner (or auditor) by the harmonizer. 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.

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 native 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's locations and obtain an integrated copy of the authentic and sealed log file.

Algorithms

1) TS(NTP) be the Network time protocol timestamp
2) Pull=0
3) Rec:=‹UID, OID, TID, Access type, Result, Time, Loc›
4) Cutime:=TS(NTP)
5) Lsize:=size of log
6) If ((cutime-tbeg) < time) && (lsize < size) && (pull==0) then
7) Log := log + ENCRYPT(rec)
8) PING to CJAR
9) If PING-CJAR then
10) PUSH *RS*(rec)
11) Else
12) EXIT(1)
13) end if
14) end if
15) if ((cutime-tbeg) > Time) || (lsize >= size) || (pull ≠ 0)
16) // (check if PING is received)
17) If *PING*-CJAR then
18) PUSH log
19) *RS*(log) := *NULL*
20) *tbeg := TS(NTP)*
21) *pull* := 0
22) else
23) EXIT(1)
24) end if
25) end if

Pushing or pulling strategies have interesting tradeoffs. The pushing strategy is beneficial when there are a large number of accesses to the data within a short period of time. In this case, if the data are not pushed out frequently enough, the log file may become very large, which may increase cost of operations like copying data. The pushing mode may be preferred by data owners who are organizations and need to keep track of the data usage consistently over time. For such data owners, receiving the logs automatically can lighten the load of the data analyzers.

The maximum size at which logs are pushed out is a parameter which can be easily configured while creating the logger component. The pull strategy is most needed when the data owner suspects some misuse of his data; the pull mode allows him to monitor the usage of his content immediately. The algorithm presents logging and synchronization steps with the harmonizer in case of PureLog. 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.

V. CONCLUSION

Our approach allows the third party auditor to audit, not only audit the data but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the third party auditor to audit even those copies of its data that were made without his knowledge. We proposed innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. In the future, we plan to refine our approach to verify the integrity of the JRE and the authentication of JARs.

REFERENCE

1. Flickr, http://www.flickr.com/, 2012.
2. Trusted Java Virtual Machine IBM, http://www.almaden.ibm.com/cs/projects/jvm/, 2012.
3. OASIS Security Services Technical Committee, "Security Assertion Markup Language (saml) 2.0," http://www.oasis-open.org/committees/tc home.php?wg abbrev=security, 2012.
4. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. ACM Conf. Computer and Comm. Security, pp. 598-609, 2007.
5. K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.
6. B. Chun and A.C. Bavier, "Decentralized Trust Management and Accountability in Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS), 2004.
7. Sun Microsystems, Inc., "Building Customer Trust in Cloud Computing with Transparent Security," https://www.sun.com/offers/details/sun_transparency.xml, Nov. 2009.
8. C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), pp. 1-9, July 2009.
9. M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, http://eprint.iacr.org, 2008.
10. W. Lee, A. Cinzia Squicciarini, and E. Bertino, "The Design and Evaluation of Accountable Grid Computing System," Proc. 29th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '09), pp. 145-154, 2009.