Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol. 2 No. 1, 2013, pp. 22-27
© The Research Publication, www.trp.org.in
DOI: https://doi.org/10.51983/ajcst-2013.2.1.1712

Indexing Policies in Cloud Information Accountability for Data Sharing

G.Venkatesan[1] and K. Ambika[2]
[1]PG Scholar and [2]Assistant Professor,
Department of Computer Science and Engineering, Anna University, Trichy, Tamil Nadu, India
E-mail : ajkksapt@gmail.com

(Received on 05 February 2013 and accepted on 08 April 2013)

***Abstract* - Indexing Policies in Cloud Information Accountability for Data Sharing presents a new way to supplement the current consumption and delivery model for IT services based on the Internet, by providing for dynamically scalable and often virtualized resources as a service over the Internet. Here, User Control Policies files and Access control policies files are indexed, and then only we can search the particular file easily. Cloud Information Accountability (CIA) framework, based on the notion of information accountability. One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed.. Access control policies and user control policies are spread across the cloud within the control of Cloud Service provider, the new user can enter their authorized login and fetch needed files, when these files are indexed. The integrity checks and oblivious hashing (OH) technique to our system in order to strengthen the dependability of our system in case of compromised JRE and also I updated to log records structure to provide additional guarantees of integrity and authenticity.**

***Keywords* - Indexing, CIA, JAR, Security**

I. Introduction

Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. CLOUD computing presents a new way to supplement the current consumption and delivery model for IT services based on the Internet, by providing for dynamically scalable and often virtualized resources as a service over the Internet. There are a number of notable commercial and individual cloud computing services, including Amazon, Google, Microsoft, Yahoo, and Sales force Details of the services provided are abstracted from the users who no longer need to be experts of technology infrastructure. Moreover, 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.

The data processed on clouds are often outsourced, leading to a number of issues related to accountability, including the handling of personally identifiable information. Such fears are becoming a significant barrier to the wide adoption of cloud services.

It is essential to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example, users need to be able to ensure that their data are handled according to the service level agreements made at the time they sign on for services in the cloud. Conventional access control approaches developed for closed domains such as databases and operating systems, or approaches using a centralized server in distributed environments, are not suitable, due to the following features characterizing cloud environments.

Data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and theses entities can also delegate the tasks to others, and soon. Entities are allowed to join and leave the cloud in a flexible manner. As a result, data handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments. Information accountability focuses on keeping the data usage transparent and track able. CIA framework provides end-to end accountability in a highly distributed fashion.

One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. Data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed.

The design of the CIA framework presents substantial challenges, including uniquely identifying CSPs, ensuring the reliability of the log, adapting to a highly decentralized infrastructure, etc. Our basic approach toward addressing these issues is to leverage and extend the programmable capability of JAR (Java ARchives) files to automatically log the usage of the users' data by any entity in the cloud. Users will send their data along with any policies such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Any access to the data will trigger an automated and authenticated logging mechanism local to the JARs. We refer to this type of enforcement as "strong binding" since the policies and the logging mechanism travel with the data. This strong binding exists even when copies of the JARs are created; thus, the user will have control over his data at any location. Such decentralized logging mechanism meets the dynamic nature of the cloud but also imposes challenges on ensuring the integrity of the logging. To cope with this issue, we provide the JARs with a central point of contact which forms a link between them and the user. It records the error correction information sent by the JARs, which allows it to monitor the loss of any logs from any of the JARs. Moreover, if a JAR is not able to contact its central point, any access to its enclosed data will be denied.

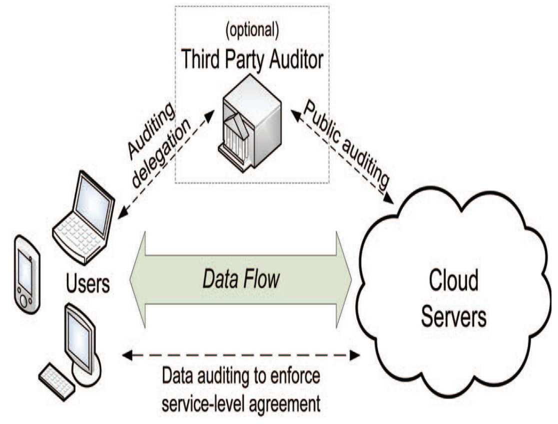

Fig.1 Cloud Structure

Push mode and pull mode:

1. The push mode refers to logs being periodically sent to the data owner or stakeholder.
2. Pull mode refers to an alternative approach whereby the user (or another authorized party) can retrieve the logs as needed.

II. Related Work

Recently, much of growing interest has been pursued in the context of remotely stored data verification. Ateniese *et.al.* are the first to consider public auditability in their defined "provable data possession" (PDP) model for ensuring possession of files on untrusted storages. In their scheme, they utilize RSA based homomorphic tags for auditing outsourced data,thus public auditability is achieved. However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems. In their subsequent work Ateniese et al. propose a dynamic version of the prior PDP scheme. However, the system imposes a priori bound on the number of queries and does not support fully dynamic data operations, i.e., it only allows very basic block operations with limited functionality, and block insertions cannot be supported.

Although the existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud Computing.

Cloud computing has raised a range of important privacy and security issues. Such issues are due to the fact that, in the cloud, users' data and applications reside at least for a certain amount of time on the cloud cluster which is owned and maintained by a third party. Concerns arise since in the cloud it is not always clear to individuals why their personal information is requested or how it will be used or passed on to other parties. To date, little work has been done in this space, in particular with respect to accountability.

Pearson *et al.* have proposed accountability mechanisms to address privacy concerns of end users and then develop a privacy manager. Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. However, the privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed.

Further, their solution requires third-party services to complete the monitoring and focuses on lower level monitoring of system resources.Leverage and extend the programmable capability of JAR (Java ARchives) files to automatically log the usage of the users' data by any entity in the cloud. Users will send their data along with any policies such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Indexing policy control for the text files, usage control for executables, and generic accountability and provenance controls.

1. We integrated integrity checks and oblivious hashing (OH) technique to our system in order to strengthen the dependability of our system in case of compromised JRE and We also updated the log records structure to provide additional guarantees of integrity and authenticity.
2. We extended the security analysis to cover more possible attack scenarios.
3. We report the results of new experiments and provide a through evaluation of the system performance.

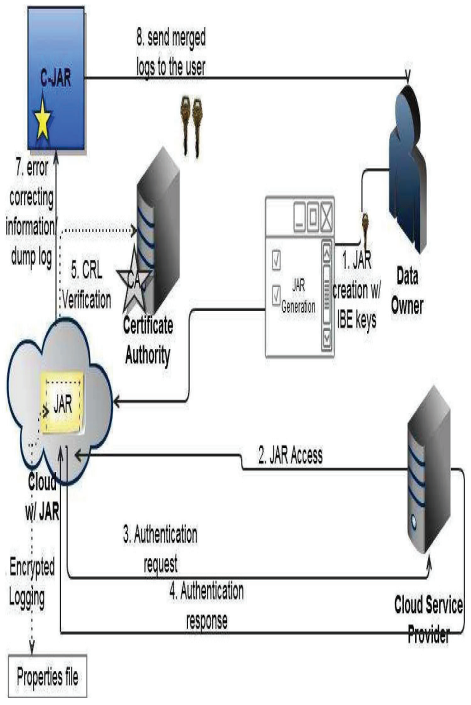

Fig.2 CIA Frame work Procers

III. Proposed Design

In Cloud Server, User Control Policies files and Access control policies files are stored in proper manner. User can access these files in server take much more time. So user control policies files and Access control policies files stored in servers are indexed. So user can search the particular file easily. Cloud Information Accountability (CIA) framework, based on the notion of information accountability.

One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed.

Access control policies and user control policies are spread across the cloud within the control of Cloud Service provider, the new user can enter their authorized login and fetch needed files, when these files are indexed. The integrity checks and oblivious hashing (OH) technique to our system in order to strengthen the dependability of our system in case of compromised JRE and also I updated to log records structure to provide additional guarantees of integrity and authenticity.

A. User Interface Design

The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centered design.

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design may be utilized to support its usability. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs. Interface design is involved in a wide range of projects from computer systems. All of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge.

B. Cloud Server

Cloud Storage is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. Hosting companies operate large data centers; and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs.

The data center operators, in the background, virtualizes the resources according to the requirements of the customer and expose them as virtual servers, which the customers can themselves manage. Physically, the resource may span across multiple servers.

C. Cloud Information Accountability (CIA)

CIA framework provides end-to-end accountability in a highly distributed fashion. CIA framework powerful accountability that combines aspects of access control, usage control and authentication. Associated with the accountability feature, we also develop two distinct modes for auditing: push mode and pull mode.

Push mode and pull mode:

1. The push mode refers to logs being periodically sent to the data owner or stakeholder.

2. Pull mode refers to an alternative approach whereby the user (or another authorized party) can retrieve the logs as needed.

D. Logging Policies

We leverage the programmable capability of JARs to conduct automated logging. A logger component is a nested Java JAR file which stores a user's data items and corresponding log files.

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file.

In our context, the data owners may not know the exact CSPs that are going to handle the data. Hence, authentication is specified according to the servers' functionality (which we assume to be known through a lookup service), rather than the server's URL or identity.

E. Identity-Based Encryption Policy

The log harmonizer is responsible for auditing. The trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. 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 we have a Pure Log, or an Access Log), and the public key of the IBE key pair that is necessary for encrypting the log records.

F. Indexing Files

Index the user control files and Access control files and other text files for easy and effective access. For more security, Data are split into number of segment and stored in number of server and provide token to each segment of data.

IV. Automated Logging Mechanism

We leverage the programmable capability of JARs to conduct automated logging. A logger component is a nested Java JAR file which stores a user's data items and corresponding log files. As shown in Fig. Our proposed JAR file consists of one outer JAR enclosing one or more inner JARs.

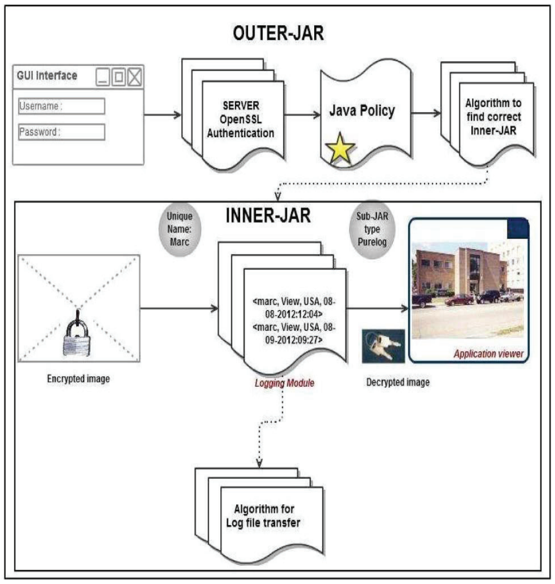
OUTER-JAR
GUI Interface
Username :
Password :
SERVER OpenSSL Authentication
Java Policy
Algorithm to find correct Inner-JAR
INNER-JAR
Unique Name: Marc
Sub-JAR type Purelog
<marc, View, USA, 08-08-2012:12:04>
<marc, View, USA, 08-09-2012:09:27>
Application viewer
Encrypted image
Logging Module
Decrypted image
Algorithm for Log file transfer

Fig.3 JAR Procers

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. In our context, the data owners may not know the exact CSPs that are going to handle the data.

Hence, authentication is specified according to the servers' functionality (which we assume to be known through a lookup service), rather than the server's URL or identity. A Java policy specifies which permissions are available for a particular piece of code in a Java application environment.

The permissions expressed in the Java policy are in terms of File System Permissions. However, the data owner can specify the permissions in user-centric terms as opposed to the usual code-centric security offered by Java. Using Java

Authentication and Authorization Services. Moreover, the outer JAR is also in charge of selecting the correct inner JAR according to the identity of the entity who requests the data

Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item. We support two options:

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.

The two kinds of logging modules allow the data owner to enforce certain access conditions either proactively (in case of AccessLogs) or reactively (in case of PureLogs). F o r example, services like billing may just need to use PureLogs. AccessLogs will be necessary for services which need to enforce service-level agreements such as limiting the visibility to some sensitive content at a given location.

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 we 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.

To allow users to be timely and accurately informed about their data usage, our distributed logging mechanism is complemented by an innovative auditing mechanism. We 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. The push action will be triggered by either type of the following two events: one is that the time elapses for a certain period according to the temporal timer inserted as part of the JAR file; the other is that 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

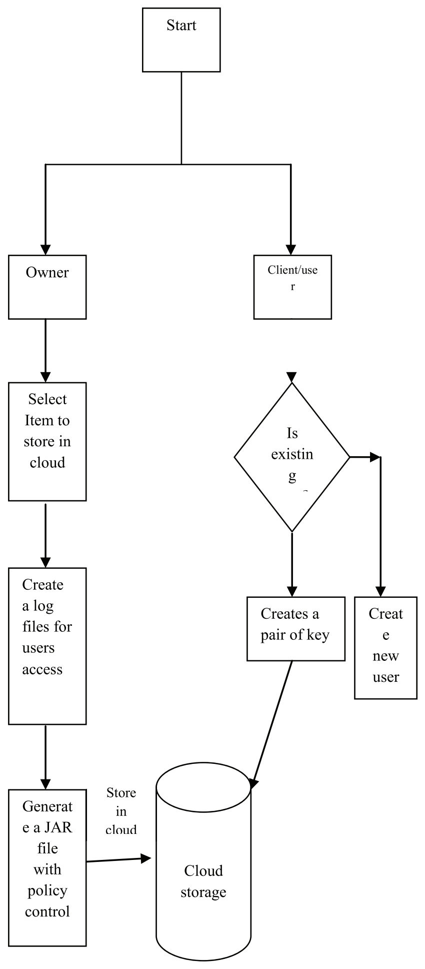
Start
Owner
Client/user
Select Item to store in cloud
Is existing
Create a log files for users access
Creates a pair of key
Create new user
Generate a JAR file with policy control
Store in cloud
Cloud storage

Fig. 4 Data Flow Diagram

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 PureLog and the AccessLog, regardless of whether there is a request from the data owner for 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 naive users, a wizard comprising a batch file can be easily built. The request will be sent to the harmonizer, and the

```
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: timestamp at which the last dump
    occurred, log: current log file, pull: indicates whether a command from the data owner is
    received.
1:  Let TS(NTP) be the network time protocol timestamp
2:  $pull = 0$
3:  $rec := \langle UID, OID, AccessType, Result, Time, Loc\rangle$
4:  $curtime := TS(NTP)$
5:  $lsize := sizeof(log)$ //current size of the log
6:  if (($curtime$ -$tbeg$) $< time$)&&
        ($lsize < size$)&&($pull == 0$) 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 (($curtime - tbeg$) $>$ $time$)||($lsize$ $>=$ $size$)
        ||($pull \neq 0$) 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
```

user will be informed of the data's locations and obtain an integrated copy of the authentic and sealed log file.

V. Conclusion and Future Research

We proposed innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our 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 our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

In the future, we plan to refine our approach to verify the integrity of the JRE and the authentication of JARs. For example, we will investigate whether it is possible to leverage the notion of a secure JVM being developed by IBM.

This research is aimed at providing software tamper resistance to Java applications. In the long term, we plan to design a comprehensive and more generic object-oriented approach to facilitate autonomous protection of traveling content. We would like to support a variety of security policies, like usage control for executables, and generic accountability and provenance controls.

Here Data owner's Utilization and the performance must be improved and also client access time are reduced. Overall cloud utilization and security performance is improved.

References

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

[2] OASIS Security Services Technical Committee, "Security Assertion Markup Language (saml) 2.0," http://www.oasis-open.org/ ommittees/ tc home.php?wg abbrev=security, 2012.

[3] 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] P.T. Jaeger, J. Lin, and J.M. Grimes, "Cloud Computing and Information Policy: Computing in a Policy Cloud?," J. *Information Technology and Politics,* Vol. 5, no. 3, pp. 269-283, 2009.

[6] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "Towards a Theory of Accountability and Audit," *Proc. 14th European Conf ResComputerSecurity* (ESORICS), pp. 152-167, 2009.

[7] T. Mather, S. Kumaraswamy, and S. Latif, Cloud Security and Privacy: An Enterprise Perspective on Risks and M.C. Mont, S. Pearson, and P. Bramhall, "Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services," Proc. Int'l Workshop Database and Expert Systems Applications (DEXA), pp. 377-382, 2003

[8] Park and R. Sandhu, "Towards Usage Control Models: Beyond Traditional Access Control," *SACMAT '02: Proc. Seventh ACM Symp. Access Control Models and Technologies,* pp. 57-64, 2002.

[9] A. Pretschner, M. Hilty, and D. Basin, "Distributed Usage Control," *Comm. ACM,* Vol. 49, No. 9, pp. 39-44, Sept. 2006.