Research Paper Engineering

# Reorganized Data Auditing and Security Administration for Cloud Environment

*Prasanth SP ** Ranjith K

* IEEE student member, Department of Information Technology, V.S.B. Engineering College, Karur, TN, India

** IEEE Student member, Department of Information Technology, V.S.B. Engineering College, Karur, TN, India.

**ABSTRACT**

*Cloud computing enables highly scalable services to be consumed over the Internet. Cloud services are provided on user request basis. In cloud environment users' data are usually processed remotely in unknown machines that users do not own or operate. User data control is reduced on data sharing under remote machines. Cloud Information Accountability (CIA) framework is a highly decentralized information accountability model. CIA framework provides end-to- end accountability in a highly distributed fashion. CIA framework combines aspects of access control, usage control and authentication. Two distinct modes are developed for auditing push mode and pull mode. The push mode refers to logs being periodically sent to the data owner or stakeholder. The pull mode refers to the user or another authorized party can retrieve the logs as needed. JAR (Java ARchives) files are used to automatically log the usage of the users' data by any entity in the cloud. Distributed auditing mechanisms are also used to strengthen user's control. The data are sending along with access control policies and logging policies 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. The Push and Pull mode log retrieval algorithm is used for the log management process. Integrity test and attack test mechanisms are used to manage the logs. The Cloud Information Accountability (CIA) framework is improved to provide authentication scheme for JAR files. The system combines the data and runtime integrity verification process. Log data analysis is provided with indexing and aggregation functions. The system includes data and executable access control mode.*

**Keywords : Cloud Information Accountability (CIA), Cloud Logging model, Data Auditing Schemes,Decentralized Data Auditing and Security Management.**

## *1. Introduction:*

Cloud computing is a recent trends in ITThat moves computing and data away from desktop and portable PCs into large data centers. It refers to applications delivered as services over the Internet as well as to the actual cloud infrastructure namely, the hardware and systems software in data centers that provide these services. The key driving forces behind cloud computing are the ubiquity of broadband and wireless networking, falling storage costs, and progressive improvements in Internet computing software. Cloud-service clients will be able to add more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity, whereas service providers will increase utilization via multiplexing, and allow for larger investments in software and hardware. Currently, the main technical underpinnings of cloud computing infrastructures and services include virtualization, service-oriented software, grid computing technologies, management of large facilities, and power efficiency. Consumers purchase such services in the form of infrastructure-as-a-service (IaaS), platform-as- aservice (PaaS), or software-as-a-service (SaaS) and sell value-added services to users. Within the cloud, the laws of probability give service providers great leverage through statistical multiplexing of varying workloads and easier management — a single software installation can cover many users' needs [10].

## 2. Related Work:

With respect to Java-based techniques for security, our methods are related to self-defending objects (SDO). Self-defending objects are an extension of the object-oriented programming paradigm, where software objects that offer sensitive functions or hold sensitive data are responsible for protecting those functions/data. Similarly, we also extend the concepts of object- oriented programming. The key difference in our implementations is that the authors still rely on a centralized database to maintain the access records, while the items being protected are held as separate files. In previous work, we provided a Java-based approach to prevent privacy leakage from indexing [9], which could be integrated with the CIA framework proposed in this work since they build on related architectures. In terms of authentication techniques, Appel and Felten proposed the Proof- Carrying authentication (PCA) framework. The PCA includes a high order logic language that allows quantification over predicates, and focuses on access control for web services. While related to ours to the extent that it helps maintaining safe, high-performance, and mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring content. Another work is by Mont etal. who proposed an approach for strongly coupling content with access control, using Identity-Based Encryption (IBE). We also leverage IBE techniques, but in a very different way. We do not rely on IBE to bind the content with the rules. Instead, we use it to provide strong guarantees for the encrypted content and the log files, such as protection against chosen plaintext and cipher text In addition, our work may look similar to works on secure data provenance [5], but in fact greatly differs from them in terms of goals, techniques, and application domains. Works on data provenance aim to guarantee data integrity by securing

the data provenance. They ensure that no one can add or remove entries in the middle of a provenance chain without detection, so that data are correctly delivered to the receiver. Differently, our work is to provide data accountability, to monitor the usage of the data and ensure that any access to the data is tracked. Since it is in a distributed environment, we also log where the data go. However, this is not for verifying data integrity, but rather for auditing whether data receivers use the data following specified policies.Along the lines of extended content protection, usage control [3] is being investigated as an extension of current access control mechanisms. Current efforts on usage control are primarily focused on conceptual analysis of usage control requirements and on languages to express constraints at various level of granularity. While some notable results have been achieved in this respect [4], thus far, there is no concrete contribution addressing the problem of usage constraints enforcement, especially in distributed settings. The few existing solutions are partial, restricted to a single domain, and often specialized [7]. Finally, general outsourcing techniques have been investigated over the past few years [2]. Although only [8] is specific to the cloud, some of the outsourcing protocols may also be applied in this realm. In this work, we do not cover issues of data storage security which are a complementary aspect of the privacy issues.

**3. Problem Statement**

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.

Currently, we focus on image files since images represent a very common content type for end users and organizations and are increasingly hosted in the cloud as part of the storage services offered by the utility computing paradigm featured by cloud computing [1]. Further, images often reveal social and personal habits of users, or are used for archiving important files from organizations. In addition, our approach can handle personal identifiable information provided they are stored as image files.

We have made the following new contributions.

**4. Cloud Information Accountability**

The Cloud Information Accountability framework proposed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It has two major components: logger and log harmonizer.

There are two major components of the CIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy.

**5. Cloud Logging Model**

In this section, we first elaborate on the automated logging mechanism and then present techniques to guarantee dependability.

**5.1. Construction of Logger**

We leverage the programmablecapability 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. Our proposed JAR file consists of one outer JAR enclosing one or more inner JARs. The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. 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:

- **Pure Log**. Its main task is to record every access to the data. The log files are used for pure auditing purpose.
- **Access Log**. 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.

**5.2. Log Dependability Issues.**

In this section, we discuss how weensure the dependability of logs. In particular,we aim to prevent the following two types ofattacks. First, an attacker may try to evade theauditing mechanism by storing the JARsremotely, corrupting the JAR, or trying toprevent them from communicating with the user. Second, the attacker may try to compromise theJRE used to run the JAR files.

**6. Data Auditing Schemes**

In this section, we describe ourdistributed auditing mechanism including thealgorithms for data owners to query the logsregarding their data.

**6.1. Push and Pull Mode**

Push mode logs are periodically pushedto the data owner by the harmonizer. The pushaction will be triggered by either type of thefollowing two events: one is that the timeelapses for a certain period according to thetemporal timer inserted as part of the JAR file;the other is that the JAR file exceeds the sizestipulated by the content owner at the time ofcreation. After the logs are sent to the dataowner, the log files will be dumped, so as to freethe space for future access logs. Along with thelog files, the error correcting information forThose logs is also dumped.This push mode is the basic mode whichcan be adopted by both the Pure Log and theAccess Log, regardless of whether there is arequest from the data owner for the log files.This mode serves two essential functions in thelogging architecture: 1) it ensures that the size ofthe log files does not explode and 2) it enablestimely detection and correction of any loss ordamage to the log files.Pull mode allows auditors to retrieve thelogs anytime when they want to check the recentaccess to their own data. The pull messageconsists simply of an FTP pull command, whichcan be issues from the command line. For naïve users, a wizard comprising a batch file can be easily built. The request will be sent to theharmonizer, and the user will be informed of thedata's locations and obtain an integrated copy ofthe authentic and sealed log file.

**6.2 Log Retrieval Algorithm**

Pushing or pulling strategies haveinteresting tradeoffs. The pushing strategy isbeneficial when there are a large number ofaccesses to the data within a short period oftime. In this

case, if the data are not pushed outfrequently enough, the log file may become verylarge, which may increase cost of operations likecopying data. The pushing mode may bepreferred by data owners who are organizationsand need to keep track of the data usageconsistently over time. For such data owners, receiving the logs automatically can lighten theload of the data analyzers. The maximum size atwhich logs are pushed out is a parameter whichcan be easily configured while creating thelogger component. The pull strategy is mostneeded when the data owner suspects somemisuse of his data; the pull mode allows him tomonitor the usage of his content immediately. Ahybrid strategy can actually be implemented tobenefit of the consistent information offered bypushing mode and the convenience of the pullmode. Supporting both pushing and pullingmodes helps protecting from some nontrivialattacks.

**7. Decentralized Data Auditing and Security Management**

The system is designed to perform datacenter management and access control activities. Decentralized access control monitoring is provided in the system. Object based access monitoring is performed for the data owners.The system is divided into six major modules. They are data owner, cloud data center, client,JAR authentication, security and access controland attack verification.Data owner share the data files under thecloud environment. Data center maintains theshared data for the data owner. Cloud clientdownloads and access the shared data from thedata centers. JAR(Java ARchieve) files are usedto monitor the data access under the clients.Code and data privilege mechanism are used forthe security process. Attack properties andshared data files are protected from attackers.

**7.1. Data Owner**

The data owner shares the data files tothe clients. Data files are provided withdifferent access permissions. Access permissionsare assigned by the data owner based on the usergroup. The system is designed with multipledata owners.

**7.2. Cloud Data Center**

The cloud data center provides storagespaces for the cloud users. Shared data filesprovided by data owners are uploaded to thecloud data centers. Client requests are processedby the data centers. Access logs are maintainedunder the cloud data centers.

**7.3. Client**

The client application is designed toaccess the data files under the cloudenvironment. The data owner assigns the client access levels.Data files are provided withreference to he access levels. Client collects thedata files from the data centers.

**7.4. JAR Authentication**

The JAR files are distributed from thedata centers with the data files. The classes inthe JAR components are authenticated by thedata centers. The JAR execution is initiated afterthe access verification process. Authenticationmethods are used to control anonymous JARcomponent access.

**7.5. Security and Access Control**

The security and access control methodsare used to verify the JAR components. Dataaccess levels are monitored and verified withclient permissions. Client monitoring codes areprovided with different access levels. Accesslevel based functions are integrated in themonitoring component.

**7.6. Attack Verification**

The attack verification is carried outwith integrity checking methods. Data andruntime integrity checking methods are used inthe system. The data integrity verification isused to check the data transmission process. Theruntime verification is performed to verify thecode execution process.

**8. Conclusions**

The data centers are used to share thedata around cloud nodes Cloud InformationAccountability (CIA) framework is used to perform data access monitoring process. TheCIA model is enhanced with authentication andintegrity analysis models. The system security isensured with data and executable access controlmechanism. The CIA framework providesdecentralized auditing model. Accountabilitymonitoring is carried out under the usageenvironment. Policy based model integrates Security and accounting process. Platformindependent accountability management model.

**REFERENCES**

[1] SmithaSundareswaran, Anna C.Squicciarini, and Dan Lin, “Ensuring DistributedAccountability for Data Sharing in the Cloud”.IEEE Transactions on Dependable And SecureComputing, Vol. 9, No. | 4, July/August 2012. | [2] 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. | [3] A. Pretschner, M. Hilty, F. Schuo¨ tz, C.Schaefer, and T. Walter, “Usage ControlEnforcement: Present and Future,” IEEESecurity & Privacy, vol. 6, no. 4, pp. 44- | 53,July/Aug. 2008.