

Distributed Auditing Mechanism for Data Sharing In the Cloud

H. Tejeswini^{*1}, and S.Sravani^{#2}

^{*}Student, Dept of CSE, St.Mark Educational Institution Society Group Of Institution, Affiliated to JNTUA University, Anantapur

[#] Associate Professor, Dept of CSE, St.Mark Educational Institution Society Group Of Institution , Affiliated to JNTUA University, Anantapur

¹tejeswini09@gmail.com

²sravani2989@gmail.com

Abstract— Cloud computing has great potential of providing robust computational power to the society at reduced cost. It enables customers with limited computational resources to outsource their large computation workloads to the cloud, and economically enjoy the massive computational power, bandwidth, storage, and even appropriate software that can be shared in a pay-per-use manner. 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. 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.

Keywords: Cloud Computing, CIA Framework, Authentication, Accountability.

I. INTRODUCTION

Cloud computing enables highly scalable services to be easily consumed over the Internet on an as-needed basis. A major feature of the cloud services is that users' data are usually processed remotely in unknown machines that users do not own or operate. While enjoying the convenience brought by this new emerging technology, users' fears of losing control of their own data (particularly, financial and health data) can become a significant barrier to the wide adoption of cloud services. To address this problem, here, we propose a novel highly decentralized information accountability framework to keep track of the actual usage of the users' data in the cloud. In particular, we propose an object-centered approach that enables enclosing our logging mechanism together with users' data and policies. We leverage the JAR programmable capabilities to both create a dynamic and traveling object, and to ensure that any access to

users' data will trigger authentication and automated logging local to the JARs. To strengthen user's control, we also provide distributed auditing mechanisms. We provide extensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches. First, data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and these entities can also delegate the tasks to others, and so on. Second, 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.

II. IMPLEMENTATION

Cloud Information Accountability (CIA) Framework: 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.

Distinct mode for auditing:

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

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

Logging and auditing Techniques:

1. The logging should be decentralized in order to adapt to the dynamic nature of the cloud. More specifically, log files should be tightly bounded with the corresponding data being

controlled, and require minimal infrastructural support from any server.

2. Every access to the user's data should be correctly and automatically logged. This requires integrated techniques to authenticate the entity that accesses the data, verify, and record the actual operations on the data as well as the time that the data have been accessed.

3. Log files should be reliable and tamper proof to avoid illegal insertion, deletion, and modification by malicious parties. Recovery mechanisms are also desirable to restore damaged log files caused by technical problems.

4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data. More importantly, log files should be retrievable anytime by their data owners when needed regardless the location where the files are stored,

5. The proposed technique should not intrusively monitor data recipients' systems, nor it should introduce heavy communication and computation overhead, which otherwise will hinder its feasibility and adoption in practice.

Major components of CIA:

There are two major components of the CIA, the first being the logger, and the second being the log harmonizer. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X. The log harmonizer forms the central component which allows the user access to the log files. The log harmonizer is responsible for auditing.

III. CLOUD INFORMATION ACCOUNTABILITY

In this section, we present an overview of the Cloud Information Accountability framework and discuss how the CIA framework meets the design requirements discussed in the previous section. 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.

Major Components:

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. The log harmonizer forms the central component which allows the user access to the log files. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X.

The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable JAR files. The JAR file implementation provides automatic logging functions, which meets the second design requirement.

Data Flow

The overall CIA framework, combining data, users, logger and harmonizer is sketched in Fig. 1. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption [4] (step 1 in Fig. 1). This IBE scheme is a Weil- attack to our architecture as described below. Using the generated key, the user will create a logger component which is a JAR file, to store its data items. The JAR file includes a set of simple access control rules specifying whether and how the cloud servers, and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. To authenticate the CSP to the JAR (steps 3-5 in Fig. 1), we use Open SSL based certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication [8], wherein a trusted identity provider issues certificates verifying the user's identity based on his username.

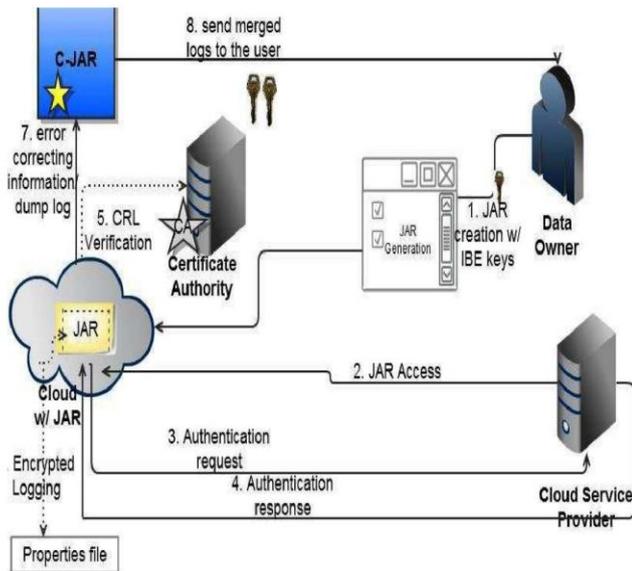


Fig. 1. Overview of the cloud information accountability framework

IV. AUTOMATED LOGGING MECHANISM

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

The Logger Structure:

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. For example, a policy may state that Server X is allowed to download the data if it is a storage server. As discussed below, the outer JAR may also have the access control functionality to enforce the data owner's requirements, specified as Java policies, on the usage of the data. 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.

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 indexing policies for text files, usage control for executables, and generic accountability and provenance controls

REFERENCES

- [1] P. Ammann and S. Jajodia, "Distributed Timestamp Generation in Planar Lattice Networks," *ACM Trans. Computer Systems*, vol. 11, pp. 205-225, Aug. 1993.
- [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] E. Barka and A. Lakas, "Integrating Usage Control with SIP-Based Communications," *J. Computer Systems, Networks, and Comm.*, vol. 2008, pp. 1-8, 2008.
- [4] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Int'l Cryptology Conf. Advances in Cryptology*, pp. 213-229, 2001.
- [5] R. Bose and J. Frew, "Lineage Retrieval for Scientific Data Processing: A Survey," *ACM Computing Surveys*, vol. 37, pp. 1-28, Mar. 2005.
- [6] P. Buneman, A. Chapman, and J. Cheney, "Provenance Management in Curated Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06)*, pp. 539-550, 2006.
- [7] B. Chun and A.C. Bavier, "Decentralized Trust Management and Accountability in Federated Systems," *Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS)*, 2004.
- [8] 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.
- [9] 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.