International Journal of Computer Applications Technology and Research
Volume 2– Issue 4, 431 - 435, 2013, ISSN: 2319–8656

Accountability in Distributed Environment For Data Sharing in the Cloud

K .Neeraja
JNTU Hyderabad
Hyderabad, India

B.Savitha Reddy
JNTU Hyderabad
Hyderabad, India

D. Rajani
JNTU Hyderabad
Hyderabad, India

Abstract—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, in this paper, 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-centred 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 travelling 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.

Keywords—Cloud computing, accountability, data sharing.

1. INTRODUCTION

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. To date, 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

2. EXISTING SYSTEM

To allay users' concerns, 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.

Problems on existing system:

First, 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 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.

3. PROPOSED SYSTEM

We propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, information accountability focuses on keeping the data usage transparent and track able. Our proposed 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 honoured, but also enforce access and usage control rules as needed. Associated with the accountability feature, we also develop two distinct modes for auditing: push mode and pull mode. The push mode refers to logs being periodically sent to the data owner or stakeholder while

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

Our main contributions are as follows:

- We propose a novel automatic and enforceable logging mechanism in the cloud.
- Our proposed architecture is platform independent and highly decentralized, in that it does not require any dedicated authentication or storage system in place.
- We go beyond traditional access control in that we provide a certain degree of usage control for the protected data after these are delivered to the receiver.

4. MODULES

4.1 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 honoured, but also enforce access and usage control rules as needed.

4.2 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.

4.3 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.

4.4 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 honoured. 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.

We conduct experiments on a real cloud test bed. The results demonstrate the efficiency, scalability, and granularity of our approach. We also provide a detailed security analysis and discuss the reliability and strength of our architecture.

4.4.1 Overview of CIA

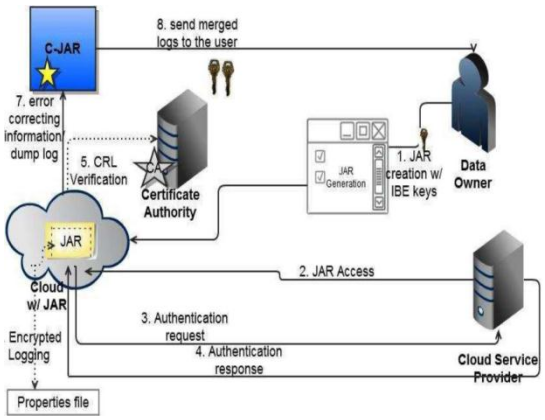

Fig 1. Architecture of CIA

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 (step 1 inFig.1). This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture. 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, where in a trusted identity provider issues certificates verifying the user's identity based on his username.

Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data (step 6 in Fig. 1). The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption (step 7 in Fig. 1). To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity verified. 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 (step 8 in Fig. 1).

5. EXPERIMENTAL RESULTS

In the experiments, we first examine the time taken to create a log file and then measure the overhead in the system. With respect to time, the overhead can occur at three points: during the authentication, during encryption of a log record, and during the merging of the logs. Also, with respect to storage overhead, we notice that our architecture is very lightweight, in that the only data to be stored are given by the actual files and the associated logs. Further, JAR act as a compressor of the files that it handles. Multiple files can be handled by the same logger component. To this extent, we investigate whether a single logger component, used to handle more than one file, results in storage overhead.

5.1 Log Creation Time

In the first round of experiments, we are interested in finding out the time taken to create a log file when there are entities continuously accessing the data, causing continuous logging. Results are shown in Fig. 2. It is not surprising to see that the time to create a log file increases linearly with the size of the log file. Specifically, the time to create a 100 Kb file is about 114.5 ms while the time to create a 1 MB file averages at 731 ms. With this experiment as the baseline, one can decide the amount of time to be specified between dumps, keeping other variables like space constraints or network traffic in mind.

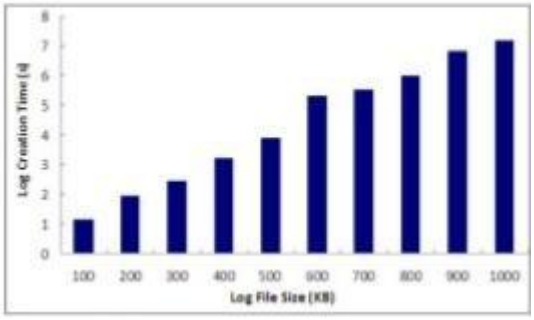

Fig 2. Time to create log files of different sizes

5.2 Authentication Time

The next point that the overhead can occur is during the authentication of a CSP. If the time taken for this authentication is too long, it may become a bottleneck for accessing the enclosed data. To evaluate this, the head node issued OpenSSL certificates for the computing nodes and we measured the total time for the OpenSSL authentication to be completed and the certificate revocation to be checked.

Considering one access at the time, we find that the authentication time averages around 920 ms which proves that not too much overhead is added during this phase. As of present, the authentication takes place each time the CSP needs to access the data. The performance can be further improved by caching the certificates. The time for authenticating an end user is about the same when we consider only the actions required by the JAR, viz. obtaining a SAML certificate and then evaluating it. This is because both the OpenSSL and the SAML certificates are handled in a similar fashion by the JAR. When we consider the user actions (i.e., submitting his username to the JAR), it averages at 1.2 minutes.

5.3 Time Taken to Perform Logging

This set of experiments studies the effect of log file size on the logging performance. We measure the average time taken to grant an access plus the time to write the corresponding log record. The time for granting any access to the data items in a JAR file includes the time to evaluate and enforce the applicable policies and to locate the requested data items.

In the experiment, we let multiple servers continuously access the same data JAR file for a minute and recorded the number of log records generated. Each access is just a view request and hence the time for executing the action is negligible. As a result, the average time to log an action is about 10 seconds, which includes the time taken by a user to double click the JAR or by a server to run the script to open the JAR. We also measured the log encryption time which is about 300 ms (per record) and is seemingly unrelated from the log size.

5.4 Log Merging Time

To check if the log harmonizer can be a bottleneck, we measure the amount of time required to merge log files. In this experiment, we ensured that each of the log files had 10 to 25 percent of the records in common with one other. The exact number of records in common was random for each repetition of the experiment. The time was averaged over 10 repetitions. We tested the time to merge up to 70 log files of 100 KB, 300 KB, 500 KB, 700 KB, 900 KB, and 1 MB each. The results are shown in Fig. 6. We can observe that the time increases almost linearly to the number of files and size of files, with the least time being taken for merging two 100 KB log files at 59 ms, while the time to merge 70 1 MB files was 2.35 minutes.

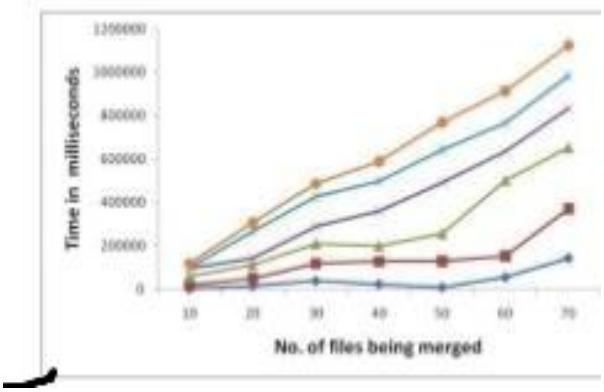

Fig 3. Time to merge log files

5.5 Size of the Data JAR Files

Finally, we investigate whether a single logger, used to handle more than one file, results in storage overhead. We measure the size of the loggers (JARs) by varying the number and size of data items held by them. We tested the increase in size of the logger containing 10 content files (i.e., images) of the same size as the file size increases. Intuitively, in case of larger size of data items held by a logger, the overall logger also increases in size. The size of logger grows from 3,500 to 4,035 KB when the size of content items changes from 200 KB to 1 MB. Overall, due to the compression provided by JAR files, the size of the logger is dictated by the size of the largest files it contains.

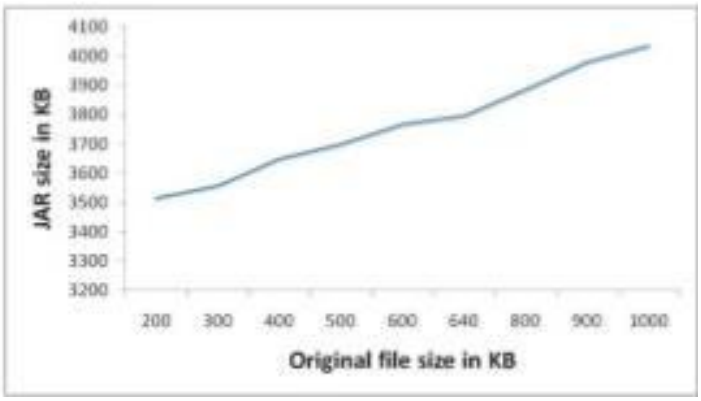

Fig 4. Size of the logger component

6. CONCLUSION AND 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 travelling 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.

7. 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.