International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-2 Issue-1, March 2013

Enabling Data Security for Collective Records in the Cloud

Muthulakshmi V, Ahamed Yaseen A, Santhoshkumar D, Vivek M

Abstract-: Cloud computing is a computing paradigm in which tasks are assigned to a combination of connections, software and services that can be accessed over internet. A major advantage of a cloud computing is enabling the distributed or remote access from known or unknown machines at any time. While Accessing the Cloud Services, the data owner faces a lot of issues related to security services (while sharing their data). To overcome these security issues, we provide automated – decentralized mechanism to capture and monitor the every usage of the users from various location. In this paper, we proposed a logging mechanism to keep track of the actual usage of the system. We leverage the jar file mechanism to ensure any data access will trigger authentication and automated logging mechanism. To consolidate user's control, we provide distributed auditing mechanisms, we also provide comprehensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches.

Index Terms: Cloud computing, data sharing, data security.

I. INTRODUCTION

Cloud computing presents a new way to enhance the current consumption and delivery model for IT services based on the Internet, by indulge for dynamically scalable and often virtualized resources as a service over the Internet. To date, there are a number of notable retailing and individual cloud computing relevance, including Amazon, Google, Microsoft, Yahoo, and Salesforce. Moreover, users may not know the machines which actually process and host their data. While accessing the convenience brought by this new technology, users also start worrying about exhaust control of their own data. The data processed on clouds are frequently outsourced, leading to a number of issues disclose to accountability, including the handling of subjectively identifiable information. Such fears are becoming a indicative barrier to the wide adoption of cloud services. To allay user's 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 assure 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 advance developed for closed domains such as databases using a centralized server in distributed environments, are not suitable, due to the pursue features characterizing cloud environments. First, information handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and theses entities can also delegate the errand to others, and so on. Second, entities are allowed to join and leave the cloud in a flexible manner. As a result, information handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments. To overcome the above problems, we propose a innovative approach, namely Cloud Information Accountability (CIA) framework. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, data accountability focuses on keeping the data usage transparent and trackable. Our proposed CIA framework provides end-to-end accountability in a highly distributed environment. 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. Associated with the accountability feature, we also develop two apparent modes for auditing: push mode and pull mode. The push mode refers to logs being systematically sent to the data owner while the pull mode refers to an alternative approach whereby the user(Or another authorized party) can retrieve the logs as needed. The design of the CIA framework presents substantial challenges, including solitary identifying CSPs, ensuring the reliability of the log, accommodate to a highly decentralized infrastructure. Our basic approach toward addressing these issues is to leverage and comprehensive the programmable capability of JAR (Java ARchives) files to automatically log the usage of the user's data by any entity in the cloud. Users will send their data along with policies such as access control policies and logging policies that they want to enclosed in JAR files, to cloud service providers. Any ingress to the data will trigger an automated and authenticated logging mechanism local to the JARs. We refer to this type of impulse as "strong binding" since the policies and the logging mechanism travel with the data. This strong binding abide even when copies of the JARs are created, thus, the user will have control past his data at any location. Such decentralized logging mechanism conflicts the dynamic nature of the cloud but also imposes challenges on ensuring the integrity of the logging.It records the error correction information sent by the JARs, which allows it to auditor the loss of any logs from any of the JARs. In addition, our approach can handle personal identifiable information provided they are stored as image files (they contain an image of any textual content, for example, the SSN stored as a .jpg file). In addition, we also provide a detailed security analysis and discuss the reliability and strength of our architecture in the face of various nontrivial attacks, dispatch by malicious users or due to compromised Java Running Environment(JRE).

Revised Manuscript Received on 30 March 2013.
* Correspondence Author
Ass.Prof. Muthulakshmi V*, IT Department Name, Kumaraguru College of Technology, Coimbatore, India.
Mr. Ahamed Yaseen A, IT Department Name, Kumaraguru College of Technology, Coimbatore, India.
Mr.SanthoshKumar D, IT Department Name, Kumaraguru College of Technology, Coimbatore, India.
Mr.Vivek M, IT Department Name, Kumaraguru College of Technology, Coimbatore, India.

In summary, our main contributions are as follows:

- We propose a novel automatic and enforceable logging mechanism in the cloud. To our knowledge a systematic approach to data accountability through the novel usage of JAR files is proposed.
- Our proposed architecture is a 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. We also provide a detailed security analysis and discuss the reliability and strength of our architecture.

II. RELATED WORK

In this section, we first review related works addressing the privacy and security issues in the cloud. Then, we concisely discuss works which adopt similar techniques as our approach but serve for different purposes.

A. Cloud Privacy and Security

Cloud computing has raised a range of important privacy and security issues [9], [10] .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 emerge 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.

Researchers have inquired accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of electronic commerce [12].

III. PROBLEM STATEMENT

We begin this section by deliberate an illustrative example which serves as the basis of our problem statement and will be used throughout the paper to demonstrate the main features of our system.

Example 1.Alice, a professional photographer, aims to sell her photographs by using the SkyHigh Cloud Services. For her persuit in the cloud, she has the business requirements:

- Her photographs are downloaded only by authorized users who have paid for her services.
- Implied buyers are allowed to view her pictures first before they make the payment to obtain the download right.
- Due to the nature of her works, only users from certain countries can view or download some sets of photographs.
- For some of her works, authorized users are allowed to only view them for a limited time, so that the users cannot reproduce her work easily.

A user who subscribed to a certain cloud service, usually needs to send thier data as well as associated access control policies to the service provider. After the information are received by the cloud service provider, the service provider will have authorize access privileges, such as read, write, and copy, on the data. In order to track the actual usage of the data, we aim to develop innovative logging and auditing techniques which satisfy the following requirements:

1. The logging should be decentralized in order to adapt to the dynamic nature of the cloud.
2. Every access to the user's data should be correctly and automatically logged. This requires compact techniques to authenticate the entity who accesses the information, 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 awful parties.
4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data.

IV. CLOUD INFORMATION ACCOUNTABILITY

In this area, 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, relay out at any point of time at any cloud service provider. It has major components such as: logger and log harmonizer.

A. Major Components

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

The logger is strongly coupled with user's data .It's main tasks include automatically logging access to data items that it contains, it encrypts the log record using the public key of the content owner, and frequently sending them to the log harmonizer.

The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The clasped coupling between data and logger, results in a powerful distributed logging system, therefore meeting our design requirement. Finally, the logger is also responsible for generating the error correction information for each log record and send the same to the log harmonizer. The error correction data combined with the encryption and authentication mechanism provides a robust and reliable recovery structured, therefore meeting the third requirement.

The log harmonizer is answerable for auditing. It supports two auditing strategies: push and pull. Under the push procedure, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand access, whereby the log file is acquired by the data owner as often as requested. In case there exist multiple loggers for the same set of data items, the log harmonizer will combine log records from them before sending back to the data owner.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

B. Major Components

The overall CIA framework, merging data, users, logger and harmonizer is sketched in Fig. 1. At the beginning, each authorized user creates a pair of public and private keys based on Identity-Based Encryption [4] (step 1 in Fig. 1). This IBE is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture as described in Section 7. Using the generated automate key, the user will create a logger component which is a JAR file, to store its user information.

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 created JAR file to the cloud service provider that he contribute to. To authenticate the cloud service provider to the JAR (steps 3-5 in Fig. 1), we use OpenSSLbased certificates, in that trusted certificate authority certifies the CSP. In the event the data access is requested by a user, once the authentication succeeds, the service provider will be allowed to access the data enclosed with JAR. Depending on the configuration settings of the time of creation, the JAR will provide usage control associated with logging, or will provide only logging serviceable.

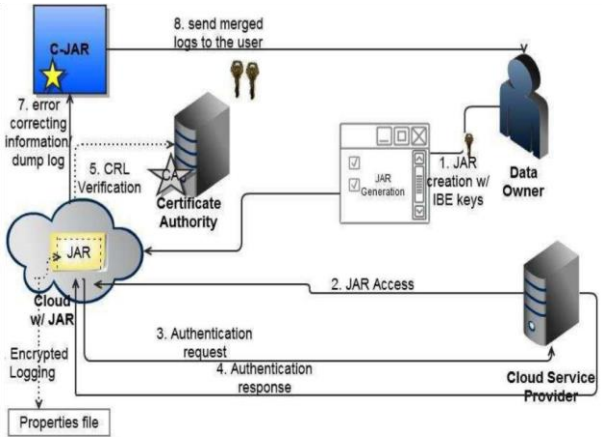

Fig.1 Overview of the cloud information accountability framework

As for the logging, each time there is an approach to the data, the JAR will certainly generate a log record, encrypt it using the public key diffuse by the data owner, and keep it along with the data (step 6 in Fig. 1). The encryption of the log file avert unauthorized changes to the file by aggressor. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for detached JARs. Using separate keys can enhance the security (detailed discussion is in Section 7) without introducing any overhead except in the computerize 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 certify trustworthiness of the logs, each data is signed by the entity approaching the content. Further, single datas are inspect together to create a chain structure, able to quickly detect possible errors or blended records. The encrypted log files can later be decrypted and their integrity certified. They can be approached 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).

Example 2. Considering Example 1, Alice can enclose her photographs and access control policies in a JAR file and send the JAR file to the cloud service provider. With the aid of control associated logging .User's will be able to enforce the first four requirements and record the actual data access. On a regular basis, the push-mode auditing mechanism will inform to the User'A about the activity on each of her photographs as this allows her to keep track of her clients demographics and the usage of her data by the cloud service provider. In the event of some dispute with her clients, Alice can rely on the pull-mode auditing mechanism to obtain log records.

V. AUTOMATED LOGGING MECHANISM

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

A. 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 hormonize log files. As shown in Fig. 2, 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.The three groups of photos are stored in three inner JAR J1, J2, and J3 frequently, combined with unique access control policies.

If some article are permitted to approach only one group of the photos, say J1, the outer JAR will just render the corresponding inner JAR to the entity based on the policy evaluation result.

Each inner JAR includes the encrypted data, class records to expedite gathering of log files and display enclosed data in a matched format, and a log file for each encrypted item. We include two options:

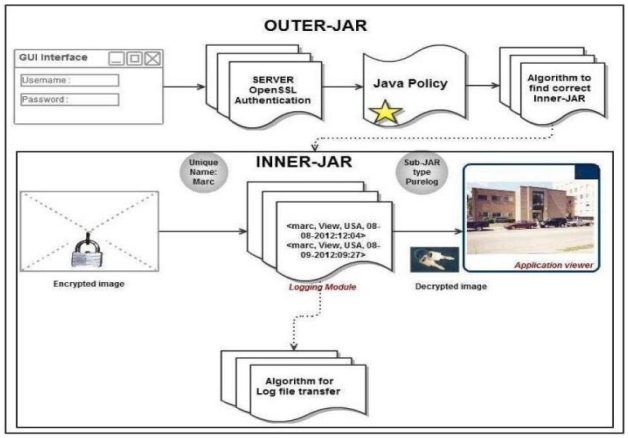

Fig. 2 The structure of the JAR file.

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

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

To carry out these functions, the inner JAR contains a class file for writing the log records, another class file which complement with the log harmonizer, the encrypted data, a third class file for displaying or downloading the data, and the public key of the IBE key pair that is necessary for encrypting **the** log records. No private keys are ever keep in the system.
Example 3. Consider Example 1. Suppose that User'A photographs are classified into three categories according to the locations where the photos were taken.

B. Log Record Generation

Log records are generated by the logger component. Logging occurs at any approach to the data in the JAR, and new log entries are added frequently, in order of creation each record $r_i$is encrypted individually and appended to the log file. In specifically, a log record handle the following form:

r_i=(ID,Act,T,Loc,h(ID,Act,T,Loc)|r_i - 1|...|r_1),sig).

Here, r_i indicates that an entity identified by ID has performed an action Act on the user's data at time T at location Loc. The component h ((ID,Act,T,Loc)|ri – 1|…|r1) corresponds to the checksum of the records preceding the newly appended one, combined with the main content of notify itself. The checksum is calculated using a collision-free hash function [6] .The component sig denotes the signature of the record created by the server.
If more than one file is handled by same logger, then additional ObjIDfield is added to each record.
Anexample of log record for a individual file is shown below.
Example 4. Suppose that a cloud service provider with ID Kronos, placed in USA, read the image in a JAR file (but did not download it) at 4:52 pm on May 20, 2011. The harmonizing log record is
hKronos, View, 2011-05-29 16:52:30 ,USA, 45rftT024g,r94gm30130ffi.

The location is changed from the IP address for improved readability.
To ensure the correctness of the log records, we notify the access time,places as well as actions. Similarly, if a trusted time stamp management infrastructure can be set up or influenced, it can be used to notify the time stamp in the accountability log .The most critical part is to log the actions on the users' data. In the present system, we include four types of actions, i.e., Act has one of the displace four values: view, download, timed_access, and Location-based_approached. For every action, we propose a paticular function to correctly record or enforce it depending on the type of the logging module, which are expatiate as follows:

- View: The entity (e.g., the cloud service provider) can only read the data but is not allowed to save a raw copy of it anywhere for always. For this type of action, the PureLog will simply write a log record about the approach, while the AccessLogs will admin the action through the enclosed access control module.
- Timed_access: This action is associated with the view-only approach, and it notifies 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 will admin that the request is allowed only within the particular period of time [8].

- Location-based_access: The PureLog will record the location of the articles. The AccessLog will check the places for every of such request. The access is permitted and the data are made available only to entities located at locations specified by the data owner.

C. Dependability of Logs

In this section, we discuss how we ensure the determination of logs. In particular, we aim to prevent the following two types of aggerssor. First, an aggressor may try to evade the authenticating appliance by keeping the JARs remotely, corrupting the JAR. Second, the aggressor may try to compromise the JRE used to run the JAR files.

D. JARs Availability

To prevent against attacks perpetrated on offline JARs, the CIA supports a log harmonizer which has two main responsibilities: to deal with copies of JARs and to recover corrupted logs. Each log harmonizer is in charge of copies of logger components containing the same set of data items. The harmonizer is achieved as a JAR file. It does not includes the user's data items being authenticated, but consists of class files for both a server and a client processes to allow it to interconnect with its logger components. The harmonizer keeps error correction information sent from its logger components. Consequently, the new copy of the logger contains the old log records with respect to the usage of data in the original data JAR file. Such old log records are repeated and irrelevant to the new copy of the data. To present the data owner a combined view, the harmonizer will merge log records from all copies of the data JARs by eliminating redundancy. For recovering purposes, logger components are required to send error correction information to the harmonizer after writing each log record. If the attacker took the data JAR offline after the harmonizer was clink, the harmonizer still has the error correction information about this access and will quickly notice the missing record.
In case of corruption of JAR files, the harmonizer will recover the logs with the aid of Reed-Solomon error correction code .Specifically, every single logging JAR, when created, includes a Reed-Solomon-based encoder. For every n symbols in the log file, n redundancy symbols are added to the log harmonizer in the design of bits. This creates an error correcting code of size 2n and allows the error correction to detect and correct n errors.
The log harmonizer is located at a known IP address. Typically, the harmonizer endure at the user's end as part of his local appliance, or frquently, it can either be keep in a user's desktop or in a proxy server.

E. Log Correctness

For the logs to be accurately recorded, it is essential that the JRE of the system on which the logger components are running stay alternated. To verify the integrity of the logger component, we carry on a two-step process: 1) we repair the JRE before the logger is launched and any kind of approach is given, so as to distribute guarantees of integrity of the JRE. 2) We include hash codes, which estimate the hash values of the program traces of the modules being executed by the logger component. This helps us identify alternations of the JRE once the logger component has been released, and are useful to check if the original code flow of execution is altered.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

These tasks are relay out by the log harmonizer and the logger components in tandem with each other.

```
//original code
1. Int uname = args [0]
2. if (permissions == “read”) //display image
3. Count = count + 1 //update counter
4. else
5. exit (0);
6. hash transform
// hashed code
7. INITIALIZE_HASH (hash1)
8. int uname = args [0]
9. UPDATE_HASH (hash1, uname)
10. if (permission == “read”)
11. UPDATE_HASH (hash1, BRANCH_ID_1) //display image
12. count = count + 1 // update counter
13. UPDATE_HASH (hash1, count);
14. else
15. UPDATE_HASH (hash1, BRANCH_ID_2)
16. exit (0)
17. VERIFY_HASH (hash1)
```

The log harmonizer is solely responsible for checking the integrity of the JRE on the systems on which the logger components exist before the execution of the logger components is started.

The logger and the log harmonizer work in tandem to carry out the integrity checks during runtime. The hash function is initialized at the beginning of the program, the hash value of the result variable is cleared and the hash value is updated every time there is a variable assignment, branching. An example of how the hashing transforms the code is shown in Fig. 3.

As shown, the hash code captures the computation results of each instruction and computes the oblivious-hash value as the computation proceeds. These hash codes are added to the logger components when they are created. They are present in both the inner and outer JARs.

VI. CONCLUSION AND FUTURE WORK

We proposed innovative requests for automatically logging any access to the data in the cloud together with an authenticating mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end assurance if needed. Moreover, one of the main characteristic 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 develop an application to protect the shared data in the cloud with more security using encryption.

REFREENCES

1. A. Squicciarini, S. Sundareswaran, and Dan Lin, “Ensuring Distributed Accountability for Data Sharing in the Cloud” IEEE Transactions on Dependable and Secure Computing,vol. 9, no. 4, July/Aug. 2012.
2. A. Squicciarini, S. Sundareswaran, and D. Lin, “Preventing Information Leakage from Indexing in the Cloud,” Proc. IEEE Int’l Conf. Cloud Computing, 2010.
3. S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, “Promoting Distributed Accountability in the Cloud,” Proc. IEEE Int’l Conf. Cloud Computing, 2011.
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. A. Squicciarini, S. Sundareswaran, and D. Lin, “Preventing Information Leakage from Indexing in the Cloud,” Proc. IEEE Int’l Conf. Cloud Computing, 2010.
6. B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1993.
7. D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigen-baum, J. Hendler, and G.J. Sussman, “Information Accountability,” Comm. ACM, vol. 51, no. 6, pp. 82-87, 2008.
8. Reed-Solomon Codes and Their Applications, S.B. Wicker and V.K. Bhargava, ed. John Wiley & Sons, 1999.
9. S. Pearson, Y. Shen, and M. Mowbray, “A Privacy Manager for Cloud Computing,” Proc. Int’l Conf. Cloud Computing (CloudCom), pp. 90-106, 2009.
10. S. Pearson and A. Charlesworth, “Accountability as a Way Forward for Privacy Protection in the Cloud,” Proc. First Int’l Conf. Cloud Computing, 2009.
11. J.H. Lin, R.L. Geiger, R.R. Smith, A.W. Chan, and S. Wanchoo, Method for Authenticating a Java Archive (jar) for Portable Devices, US Patent 6,766,353, July 2004.
12. 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.
13. P. Ammann and S. Jajodia, “Distributed Timestamp Generation in Planar Lattice Networks,” ACM Trans. Computer Systems, vol. 11 , pp. 205-225, Aug. 1993.

AUTHOR PROFILE

Muthulakshmi.V is the IEEE student branch counselor at Kumaraguru College of Technology. She received her UG at Anna University and continued her studies and achieved her Masters Network & Internet Engineering. Muthulakshmi has been in teaching for 5 years and Cisco Certified Trainer. Her dedication and commitment is to promote STEM education, and encourage students, especially young girls to pursue a career in the IEEE Fields of Interest. Her areas of interest includes networking, software development and testing and Assistive technology

Ahamed Yaseen A is pursuing his B.Tech degree with specialization in Information Technology in Kumaraguru College Of Technology, Coimbatore. He is an active member of CSI and ISTE.

Santhosh Kumar D is pursuing his B.Tech degree with specialization in Information Technology in Kumaraguru College Of Technology, Coimbatore. He is an active member of ISTE.

Vivek M is pursuing his B.Tech degree with specialization in Information Technology in Kumaraguru College Of Technology, Coimbatore. He is an active member of ISTE.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)