Ensuring Distributed Accountability for Data Sharing in the Cloud

Pallavi D. Rakh[1] Vishwanath T. Kamble[2] Chinmay S. Shetty[3] Chetan Goje[4]prof. Sulakshana Mane[5]
[1, 2,3,4,5,] Computer Engineering, Bharati Vidyapeeth College of Engineering, Navi Mumbai, India

Abstract---Cloud computing is one of the latest technology which helps providing scalable service on the internet as per the needs of the users. All the data of the user are usually processed on a remote computer which is not known to the users. Because of this reason users fear of losing information as they do not have full control and access to their data This becomes a big barrier to the complete adoption of these features. To control this problem, in this paper we present a decentralized information accountability framework to keep track of the user's data in the cloud. Here we enclose our logging information together with the user's data and policies. We use the JAR technology to provide a trigger able authentication and automatic logging locals to the JARs, which will be dynamic and traveling objects. In this project, we will also be providing a distributed auditing mechanism. The experimental studies will show how efficient and effective the proposed system can be in the field of cloud computing.

Keywords: Cloud computing, accountability, data sharing.

I. INTRODUCTION

Cloud computing gives a very good alternative way to replace the tradition style of IT Service, by providing a platform over the Internet, providing a dynamic, scalable and completely virtual resources of service over the Internet. Many companies provide cloud computing services which are being run successfully which include Amazon, Google, Microsoft, Yahoo and Sales force [1]. Details of the services provided are abstracted from the users who no longer need to be experts of technology infrastructure. In cloud computing, users don't know the machines which are storing their data. And this is the major reason why the users worry about losing their 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 leads to significant barrier in the wide adoption of this service as a basic application in may field [2].

And that is why, to get the trust of the users, it is really essential to provide a very effective mechanism for users to monitor how their data is being used 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 which used domains such as databases and operating systems, or techniquest like centralized server in a distributed fashion are not suitable, since this contain different methods characterizing cloud environment. Initially, data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and theses entities can also request its tasks to others, and so on. Then other entities, which are low to the initial one leaves the cloud system in a flexible manner. And in this way, the data handing in the cloud follows as dynamic and complex hierarchical service chain which is not usually found in the traditional environments.

CIA

Here in this paper, we provide an approach, viz. Cloud Information Accountability (CIA) framework, based on the working of accountability of information [3]. Information accountability provides a method which keeps the data usage of the user's transparent and trackable unlike the traditional way which uses the hide it or lose it approach. 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 lightweight and powerful account-ability maintaining that combines aspects of access control, usage control and authentication mechanisms. 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. Associated with the accountability feature, we also develop two distinct modes for auditing: push mode and pull mode. Push mode means the information which is sent to the data owners and the stakeholder while any change is being made in the log, 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 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.

II. PROPOSED SYSTEM

A. *Cloud Information Accountability*

In this section, we present an overview of thev Cloud Information Accountability framework and discuss how the

CIA framework meets the design requirements discussed in the previous section. Here in this system, we propose the CIA method in which we conduct automated logging and distributed auditing of entities relevant access to the system, which is done at any point of time at the cloud service provider. The system also maintains how logging has been done and notifies the users and the owners of the system about the changes being made.
It has two major components: logger and log harmonizer.

B. *Major Components*

The two major components of the CIA are the logger, which is the first and the second most important component is the log harmonizer. The logger keeps track of when the data is being accessed and when it is being copied by getting downloaded every time the data is accessed. Similarly tracking of the copying of the data is done by copying of the logger information. 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 is the central component which allows the user access into the log files edited by the user.

The logger is attached with the user's data which is either the single or multiple data items. The major task of the logger include giving logging access automatically to data items that it contains, log entries and data encryption using the public key of the content owner, and then periodically sending them to the log harmonizer. The configuration can also be done in such a way that it ensures access and control policies usage associated with the data are honoured. For instance, a data owner can specify that user X is only allowed to view but not to modify the data. The data access of the logger will be controlled even after it has been downloaded by any user. For this very less support is required i.e. virtual Java machine installed in the server in order to be deployed correctly. The coupling between the data and the logger, results in a highly distributed logging system, therefore meeting our first design requirement. Adding to it, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, the logger is also responsible for generating the error correction information for each log record and sends the same to the log harmonizer.

The error correcting information together with the encryption and authentication designs gives a robust and robust, reliable and recoverable mechanism, therefore meeting the third most requirement. The log harmonizer is the main reason behind auditing. Being 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. In this case, the harmonizer sends the key to the client in a secure key exchange. It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically and sequentially as per the needs in an automated fashion. The pull mode is an on-demand approach, in which the log files are obtained by the data owner as often as ordered or needed. These two techniques give us to satisfy the previously mentioned fourth design requirement. In case there are already multiple loggers who are accessing same set of data items, the log harmonizer will mix up log records from them before being sent back to the data owner.

The log harmonizer is also the reason for handling log file corruption and failure. In addition, the log harmonizer can itself carry out logging and logging out in addition to auditing. Separating the logging and auditing functions improves the outcome of the system. The logger and the log harmonizer are both implemented as lightweight, inexpensive, without much pressure on the system and portable JAR files. The JAR file implementation provides automatic access into the system functions, which meets the second design requirement.

C. *Data Flow*

The overall CIA framework, combining data, users, logging user to the system, logger and harmonizer working is sketched in Fig. 1. At the beginning of this process, each user creates a pair of public and private keys based on Identity-Based Encryption (as shown in step 1 in Fig. 1). This IBE scheme is a Weil-pairing-based IBE scheme, which is used to secure us against one of the most prevalent threats to our architecture. Using the generated key, the user will create a logger module which is a JAR file, to store its data items and make further proceedings.

The JAR file has a combination of simple access control rules which specifies whether and how the cloud servers, and possibly all the other data stakeholders (users, companies) are authorized to make changes to the content itself. Then, he sends the JAR file to the cloud service provider that he makes subscription to the system. To authenticate the CSP to the JAR (steps 3-5 in Fig. 1), we use Open SSL based certificates and authority, wherein the CSP certification is done by a trusted certificate authority which duly signs the certificates. In the event that the access is requested by a user, we employ SAML-based authentication mechanism, wherein a trusted identity provider assigns certificates which verifies the user's identity based on his authentication data.

Once the verification is done, the service provider (or the user) will be allowed to open and make modifications to the data enclosed in the JAR. The JAR will give prevention to excessive usage related with logging, or will provide only logging functionality and mechanism, which depends on the configuration settings which has been predefined or decided at the time of creation. As for the logging, each time there is an access made to the data, the JAR will automatically create a log record, encrypt it using the public key distributed by the data owner, and save it with the data between them (as shown in step 6 in Fig. 1). To ensure in what way the logs are trustworthy, each record is signed by the entity which is accessing the content. Furthermore, each individual record is hashed and merged together to create a chain structure, which will be able to quickly detect possible mistakes or missing records. The encrypted log files can also be later decrypted and their integrity checked. They can be given access by the data owner or other authorized people who hold the stakes at any

time for auditing usage details with the aid of the log harmonizer (step 8 in Fig. 1).

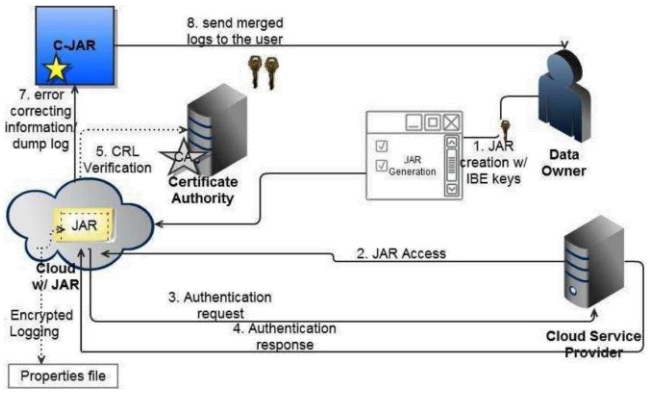

Fig. 1: Overview of the cloud information accountability framework

As discussed, our proposed framework also prevents various attacks such as finding out illegal copies of user's data. Note that our work is different from traditional logging methods which use encryption to protect log files. With only encryption, their logging mechanisms are neither automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud.

III. End-to-End Mechanism

In this section, we describe our distributed auditing mechanism including the algorithms for data owners to query the logs regarding their data.

A. *Push and Pull Mechanism*

To allow users to be timely and exactly informed about their data usage, our distributed logging mechanism is complemented by an innovative auditing mechanism. We support two complementing auditing modes: 1) push mode; 2) pull mode.

B. *Push Mode*

In this mode, the logs are periodically pushed to the data owner (or auditor) by the harmonizer for further proceedings. The push action is triggered by any of the following two types: one is that the time forwards for a certain period according to the temporal timing system which is led as part of the JAR CIA system; the other is that the JAR file exceeds the size certainly stipulated by the content owner at the time of creation. After the logs are sent to the data owner, the dumping of the log files is done, so as to free the space for future access logs. Along with the log files, the error correcting information for those logs is also garbage collected. This push mode is the basic mode which can be also be provided by both the Prelog and the Access Log, regardless of whether there is a request from the data owner for the log files. This mode serves two essential functions in the logging architecture: 1) it ensures that the size of the log files does not exceed to spill out and 2) it enables timely detection, prevention and correction of any loss or damage to the log files. Concerning the latter function, we notice that the auditor verifies its cryptographic guarantees when he receives the related log files by them, by checking the records integrity and authenticity. By construction of the records, the auditor detects any forgery of entries very fast, using the checksum added to each and every record.

C. *Pull Mode*

This mode allows auditors to receive the logs anytime when they want to check which recent access has been made. The pull message consists simply of an FTP pull command, which can be issued from the command line during the system implementation. For new users, a wizard help is provided comprising a batch file can be easily built and used. The request will be sent to the harmonizer, and the user will be informed of the data's locations and obtain an integrated copy of the authentic and protected log files.

D. *Algorithm used in this technique*

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** $((cutime - tbeg) < time) \&\& (lsize < size) \&\& (pull == 0)$ **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** $((cutime - tbeg) > time) || (lsize >= size) || (pull \neq 0)$ **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**

Fig.2: Push and pull Pure Log mode.

Pushing or pulling strategies have interesting trade-off. The pushing strategy is beneficial when there are a large number of accesses to the data within a short period of time. In this case, if the data are not pushed out frequently enough, the log file may become very large, which may increase cost of operations like copying data. The pull strategy is most needed when the data owner suspects some misuse of his data; the pull mode allows him to monitor the usage of his content immediately. A hybrid strategy can actually be implemented to benefit of the consistent information offered by pushing mode and the convenience of the pull mode. Further, as discussed, supporting both pushing and pulling modes helps protecting from some nontrivial attacks.

The log retrieval algorithm for the Push and Pull modes is outlined in Fig. 4. The algorithm presents logging and synchronization steps with the harmonizer in case of PureLog. First, the algorithm verifies whether the size of the JAR has exceeded a stipulated size or the normal time between two consecutive dumps has elapsed. If none of these events has occurred, it proceeds to encrypt the record and write the error-correction information to the harmonizer. The communication with the harmonizer begins with a simple handshake. If no response is received, the log file records an error. The data owner is then changed through e-mails, if the JAR is configured to send error notifications.

Once the handshake is completed, the communication with the harmonizer proceeds, using a TCP/IP protocol. If any of the aforementioned events (i.e., there is request of the log file, or the size or time exceeds the threshold) has occurred, the JAR simply dumps the log files and resets all the variables, to make space for new records. In case of AccessLog, the above algorithm is altered by adding an additional check after step 6. Precisely, the AccessLog checks whether the CSP accessing the log satisfies all the conditions given in the policies pertaining to it. If the conditions are fulfilled, access is granted; or else, access is denied. Irrespective of the access control outcome, the attempted access to the data in the JAR file will be logged. Our auditing mechanism has two main advantages. First, it guarantees a high level of availability of the logs. Second, the use of the harmonizer minimizes the amount of workload for human users in going through long log files sent by different copies of JAR files. For a better understanding of the auditing mechanism, we present the following example. Thus this mechanism provides a system to ensure that the data of the users are stored somewhere safe and secure whist being used for the cloud computing system.

IV. CONCLUSIONS

In this paper, we present an innovative method for logging any access to the database in the cloud automatically with an auditing mechanism. This approach gives the owner of the data, access to auditing his or her content but also to enforce a strong back-end protection. Also this system enables a user to audit even those copies of data which were not audited by him or were audited without his/her knowledge. Thus a better, secure and safe system is made available to the users in the cloud computing system.

ACKNOWLEDGEMENT

Our most sincere appreciation is to all the people who has helped and inspired us throughout the working of this project.

Firstly we are thankful to our principal Dr. M. Z. Shaikh for his help. We are extremely grateful for his friendly support and professionalism.

We express our heartfelt gratitude to our Head of Department Prof. D.R Ingale and our seminar coordinator Prof. B. W. Balkhande for their help and support. This task would not have been possible without the help and guidance of our project guide Prof. Mrs. Sulakshana Mane

We are also convening special thanks to all staff of Computer Engineering Department for their support and help. Last but not least, we are very much thankful to our friends who directly or indirectly helped us in completion of the project report.

REFERENCES

[1] 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.

[2] S. Pearson and A. Charles worth, "Accountability as a Way Forward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing, 2009

[3] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigen-baum, J. Hendler, and G.J. Susana, "Information Accountability," Comm. ACM, vol. 51, no. 6, pp. 82-87, 2008

.

