Secure Data Processing in Cloud Computing

Snehal P. Suryawanshi
PG student of Department of Information Technology Pune Institute of Computer Technology
Sr. No 27, Dhankawadi, Pune Satara Road, Pune – 411043

Anant M. Bagade
Associate Professor of Department of Information Technology Pune Institute of Computer Technology
Sr. No 27, Dhankawadi, Pune Satara Road, Pune – 411043

ABSTRACT

Cloud computing is type of system that uses sharing resource, it cannot uses local sever or personal devices for application, it uses single network to share resources. This services are provided to user's computer and laptops using internet. Main feature of the cloud services is that user’s data are usually process on machines that users does not know. It can become a main problem for growing cloud services. This work giving a highly decentralized framework to keep track of the actual usage of the data owner’s data in the cloud. The Cloud Information Accountability framework given in this paper conducts automated logging and distributed auditing of access of data by any user, carried out at any time . It has two major components: logger and log harmonizer. Data owner will get confirmation that his data is handled according to his desire.

Keywords

Cloud Computing, Accountability, Security, Privacy, ICIA Framework .

1. INTRODUCTION

In cloud computing we can share our data and application in common center. It is a technology which uses internet and share resources to maintain data. Security is important issue because cloud having many benefits so, it have many users. For security purpose this paper providing ICIA framework The Improve Cloud Information Accountability framework given in this work performs automatic login and auditing. Using this mechanism data owner get updates that his data is safe on cloud. This auditing is carried out at any time at any CSP. It has two major components: logger and log harmonizer. Logger is a JAR file with user's data as data access it creates log record of each data access. The JAR file contains set of access control rules specifying who(company and users),where and when will use data. Apart from that going to check the integrity of the JRE on the systems on which the logger components is initiated. This integrity checks are carried out by using oblivious hashing. Depending on the configuration policies defined at the time of creation, the JAR will provide usage control policies with data owner's data. During each time access of data, Jar will automatically create log record.

In this ICIA framework data owner will send his data to cloud service provider with data and access control policies in encrypted form with keys(private and public key),whenuser will use data, he will decrypt data using public key, and when he will use particular data then jar will automatically create corresponding log records using logger component, logger will send log records to log harmonizer. Then log harmonizer will push this records to data owner so owner can see logs when he wants, he get confirmation that his data is handled according to service level agreement and his data is safe on cloud using this framework[1].

2. RELATED WORK

In this section we first review related works addressing security in cloud. Security issue is very important in cloud there are many techniques available so here is review of all these.

Q. Wang et al describes Third party auditor for verification, they describes three network entities i.e. client which is user, cloud storage server which is handled by cloud service provider and Third party auditor which is verifier.TPA having public key, it is act with only trusted server, they are not focuses on data privacy [2].In [3], the author presents effective usage control model for security of kernel integrity. UCONKI model with properties of continue the decision for OS kernel integrity protection, they proposes virtual machine monitor (VMM) based architecture for preventing attacks inside virtual machine . Ryan K L Ko et al in their paper presents accountability for cloud computing, accountability is verification of access control policies. This paper discuss main challenges for achieving cloud computing services, this problem focuses on accountability in cloud computing [4]. In [5], author presents author presents three layer architecture for preventing information leakage from indexing in cloud .The three layers are storage protection, medium protection, low protection, in this paper according to user's security requirement the request analyzer will select one of the layer and send output to the service provider. R. corin et al in their paper presents language in which data owner will send data and policies to agent, responsibility of agent to check all authentication and authorization policies of users and action of users. but there is problem of Continuous monitoring of agent[6].Jia Xu et al in their paper gives the proofs of retrievability i.e. POR model to ensure security of data storage in cloud. This is cryptographic function for remotely auditing purpose[7].

3. SYSTEM FUNCTIONALITY

Proposed system contains following scenarios:

3.1 JAR Creation

Jar file contains set of access control rules specifying who(company and users), where and when use particular data i.e., data owner’s data. Depending on the settings defined at the time of creation, the JAR will provide usage control rules with logging. Data owner will send access control policies and data to cloud service provider in JAR file.JAR file will placed at chosen location.

3.2 Functioning of Outer and Inner Jar

This paper is using programmable capability of JARs to conduct automatic logging. A logger component is a 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 this 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 assume to be known through a lookup service), rather than the server's URL or identity. The data owner specify the rules that is who will going to handle particular data. Outer JAR contains access control policies and Java authentication policies. One Outer JAR contain one or many inner JAR. Inner JARs contain data of data owner. Outer JAR will decide correct inner JAR[1].

3.3 Log Records

Log records are generated by the logger component. Logging occurs at any access to the data in the JAR, and new log entries generates, in order of creation ,Lr = r1, r2, r3, r4... rk. Each record rk is encrypted individually and placed to the log file. When there is a view-only access request, the inner JAR will decrypt the data create a temporary decrypted file and create log record of view. The decrypted file will then be displayed to user using particular data of data owner. The java application disable copying function using right click.

4. PRAPOSED WORK

4.1 Logging Mechanism

The data owner specify the rules that is who will going to handle particular data. Outer JAR contains access control policies and Java authentication policies. One Outer JAR contain one or many inner JAR. Inner JARs contain data of data owner. Outer JAR will decide correct inner JAR. Data owner does not know on which server data will placed.

Authentication is done according to server's URL.

4.1.1 Log record generation

Log records are generated by the logger component. Logging occurs at any access to the data in the JAR, and new log entries are placed sequentially, in order of creation

Lr = r1, r2, r3, r4... rk. Each record rk is encrypted individually and placed to the log file. In particular, a log record takes the following form:

rk = (id, action, T, loc,h((id, action, T, loc)|ri-1|…r1), sig)

Where,

rk = log record

id = user identification

action = action perform on user's data

T = Time at location loc

loc = Location

h((id, action, T, loc)|ri-1|…r1) = checksum component

sig = Signature of record by server[1].

4.2 Push and Pull Mode

To allow users to be timely and accurately informed about their data usage, This distributed logging mechanism is providing auditing mechanism i.e. 1.push mode 2.pull mode

4.2.1Push Mode

In this mode, the logs are automatically send to the data owner by the Harmonizer ,it is important when usage of data is so large. Using this mode owner get knowledge of timely access of his data in CSP. If log records are not send to owner periodically then JAR file become very large in size. This mode is very important for owner who want update of timely access of his data. By this mode size of log file does not exceed and also achieve timely detection and correction of log records. Auditor also detect any wrong entries to log file, using checksum added to each log record. This happen at two times when JAR file exceed by the data of data owner and timer in JAR file over after decided time. As logs are dumped periodically then free space is use for future logs[1].

4.2.2 Pull Mode

This mode allows auditors to check the logs anytime when they want to check the recent access to their own data. The request will be sent to the harmonizer, and the user will be informed of the data's locations and obtain Copy of encrypted log file .This mode is important when data owner want to know his data access immediate[1].

4.3 Data Flow

The overall framework shown in fig 1, It shows data, users, logger and harmonizer . At the start, each data owner creates a pair of public and private keys based on Identity-Based Encryption. Using the generated key, the data owner will create a JAR file(logger component), to store its data items. The JAR file contains simple access control rules specifying who (company and users), where and when use particular data i.e. data owner's data. Then, he sends the JAR file to the cloud service provider that he subscribes to. For authentication of the CSP to the JAR , we use Open SSL-based certificates. Once the authentication succeeds, the service provider or the user will be allowed to access the data in the JAR. As access (i.e. view, modify, copy) to particular data the log records will generate and they are store in jar file with data in encrypted form , So attackers cannot make changes to log record[1].

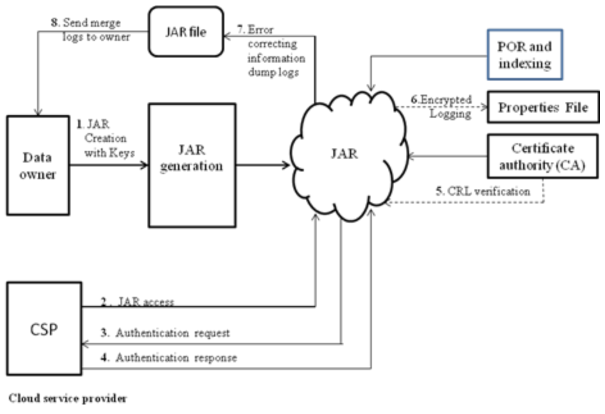
8. Send merge logs to owner
JAR file
7. Error correcting information dump logs
POR and indexing
6.Encrypted Logging
Properties File
Data owner
1. JAR Creation with Keys
JAR generation
JAR
Certificate authority (CA)
5. CRL verification
CSP
2. JAR access
3. Authentication request
4. Authentication response
Cloud service provider

Fig 1. ICIA Framework

4.4 Proof-of-retrievability (POR) Model

proof-of-retrievability is a cryptographic function for remotely auditing purpose. It verify the integrity of files stored in cloud server, no need of storage of original files in local storage.POR model consist of setup phase and verification phase. Efficiency is calculated by using some factors 1.No of communication bits 2.storage overhead 3.computation time for verification.

POR model consist of four algorithm

1.keyGen(1^k)→(pk, sk) . This algorithm is run by client, it takes k as input and return public and private keys. The client stores private key and send public key to server.

2.DEncode(sk,F) → (idF, ^F).

To encoder provide private key sk and data file F, the encoding algorithm produces a unique identifier idF and encoded file F.

3.Prove (pk, idF, ^F,q)→Ψ

Given public key pk,an identifier idF, an encoded file ^F, and challenge query q,prover algorithm wil produce a proof Ψ.

4.Verify(sk, idF, q, Ψ)→ accept or reject

Given private key sk, an identifier idF,challenge query q, a proof Ψ, output of verify is either yes or no[7].

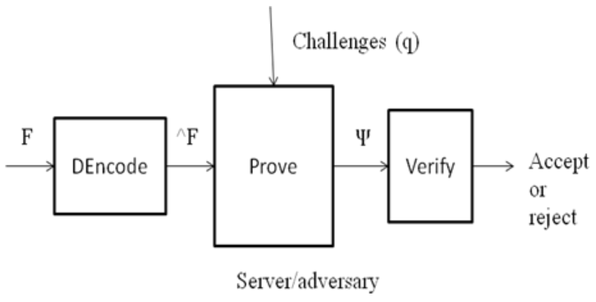

Fig 2. POR Model Working

5. OVERCOME ON ATTACKS

5.1. Copying Attack

The most important attack is that the attacker copies entire JAR files. The hacker may assume that doing this, can access data in JAR file. However, such attack will be detected by our auditing mechanism. Recall that every JAR file is required to send log records to the harmonizer.

In particular, with the push mode, the harmonizer will send the logs to data owners periodically. That is, even if the data owner is not aware of the existence of the additional copies of its JAR files, he will still be able to receive log files from all existing copies. If attackers move copies of JARs to places where the harmonizer cannot connect, the copies of JARs will soon become inaccessible[1].

5.2. Disassembling Attack

Another possible attack is to disassemble the JAR file of the logger and then attempt to extract useful information out of it or spoil the log records in it.

Once the JAR files are disassembled, the attacker is in possession of the public IBE key used forencrypting the log files, the encrypted log file itself, and the *.class files. Therefore, the attacker has to rely on learning the private key or subverting the encryption to read the log records. To compromise the confidentiality of the log files, the attacker may try to identify which encrypted log records correspond to his actions by mounting a chosen plaintext attack to obtain some pairs of encrypted log records and plain texts. If the attacker wants to infer access control policies, the only possible way is through analyzing the log file. This is, however, very hard to accomplish since, as mentioned earlier, log records are encrypted and breaking the encryption is computationally hard. Also, the attacker cannot modify the log files extracted from a disassembled JAR[1].

5.3 Man-In-Middle Attack

an attacker may intercept messages during the authentication of a service provider with the certificate authority, and reply the messages in order to masquerade as a legitimate service provider. There are two points in time that the attacker can replay the messages. One is after the actual service provider has completely disconnected and ended a session with the certificate authority. The other is when the actual service provider is disconnected but the session is not over, so the attacker may try to renegotiate the connection. The first type of attack will not succeed since the certificate typically has a time stamp. The second type of attack is also not possible because we are using Open SSL certificate [1].

6. RESULT

6.1 Log file creation

In Fig 3 time to create log file of different size has given, 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 3

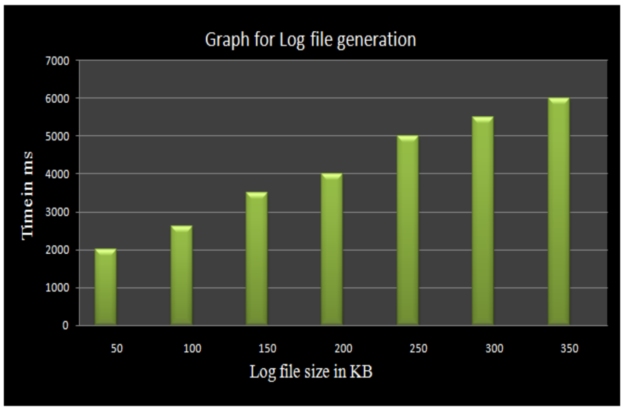

Fig 3. Time to create log file of different size

The time to create a log file increases linearly with the size of the log file. Time to create log file size of 350kb is 6000ms nearly. 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.

6.2 Log file merging

In Fig 4 time to merge log file of different size has given, interested in finding out time to merge log file of different size. Observe that the time increases almost linearly to the number of files and size of files.

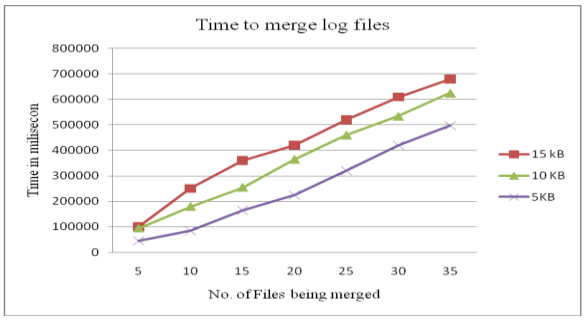

Fig 4. Time to merge log files

6.3 Size of Data JAR File

Finally, investigate whether a single logger, used to handle more than one file, results in storage overhead. The size of the loggers (JARs) by varying the number and size of data items held by them. in case of larger size of data items held by alogger, the overall logger also increases in size. Suppose data present in data file is large then corresponding log records also large. This is shown in fig 5.

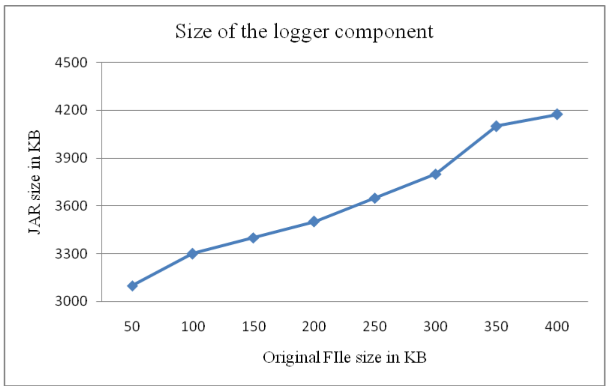

Fig. 5 Size of the logger component

7. CONCLUSION

This work proposes a novel highly decentralized information accountability framework to keep track of the actual usage of the user's data in the cloud. In particular, this proposes an object-centered approach that enables enclosing logging mechanism together with user's data and policies. This 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, also provide distributed auditing mechanisms.

This is innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. This 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 work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

Apart from that enclosed POR and indexing methodology to enhance the integrity of owner's data. In the future , plan to refine approach to verify the integrity of the JRE and the authentication of JARs. For example, investigate whether it is possible to leverage the notion of a secure JVM being developed by IBM.

8. REFERENCE

[1] SmithaSundareswaran, Anna C. Squicciarini and Dan Lin, "Ensuring Distributed Accountability for Data Sharing in the Cloud," IEEE Transaction on dependable a secure computing, *VOL. 9, NO. 4, pg 556-568, 2012*.

[2] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. EuropeanConf. Research in Computer Security (ESORICS), pp. 355-370, 2009.

[3] M. Xu, X. Jiang, R. Sandhu, and X. Zhang, "Towards a VMM Based Usage Control Framework for OS Kernel Integrity Protection," SACMAT '07: Proc. 12th ACM Symp. Access Control Models and Technologies, pp. 71-80, 2007.

[4] Ryan K L Ko, Peter Jagadpramana, Miranda Mowbray, Siani Pearson, Markus Kirchberg, Qianhui Liang and Bu Sung Lee,"TrustCloud: A Framework for Accountability and Trust in Cloud Computing," 2nd IEEE Cloud Forum for Practitioners (IEEE ICFP 2011), Washington DC, USA,pp 1-8.

[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] R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, "ALogic for Auditing Accountability in Decentralized Systems,"Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005.

[7] Jia Xu and Ee-Chien Chang," Towards Efficient Proofs of Retrievability", ASIACCS '12, May 2–4, 2012.

[8] Trusted Java Virtual Machine IBM, http://www.almaden.ibm. com/cs/projects/jvm/, 2012.

[9] Eucalyptus Systems, http://www.eucalyptus.com/, 2012.

[10] Amazon Cloud, http://aws.amazon.com