ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

# Privacy and Security of Distributed Accountability in Cloud

Gokulakannan R , Manjunath G, R Arulselvi

ME-First Year, Department of CSE, Muthayammal Engineering College, India.

ME-First Year, Department of CSE, Muthayammal Engineering College, India.

Assistant professor, Department of CSE, Muthayammal Engineering College, India.

**ABSTRACT:** Cloud computing ensures Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Privacy protection technologies which are built on the hide-it-or-lose-it perspective, information accountability focuses on keeping the data usage transparent and tractable. 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. Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is deobfuscated by the privacy manager to reveal the correct result.

**KEYWORDS:** Cloud accountability using JAR files for secure authentication and ease of access.

## I. INTRODUCTION

Cloud computing, as an emerging computing paradigm, enables users to remotely store their data into a cloud so as to enjoy scalable services on-demand. Especially for small and medium-sized enterprises with limited budgets, they can achieve cost savings and productivity enhancement by using cloud-based services to manage projects, to make collaborations, and the like. However, allowing cloud service providers (CSPs), which are not in the same trusted domains as enterprise users, to take care of confidential data, may raise potential security and privacy issues. To keep the sensitive user data confidential against untrusted CSPs, a natural way is to apply cryptographic approaches, by disclosing decryption keys only to authorized users. However, when enterprise users outsource confidential data for sharing on cloud servers, the adopted encryption system should not only support fine-grained access control, but also provide high performance, full delegation, and scalability, so as to best serve the needs of accessing data anytime and anywhere, delegating within enterprises, and achieving a dynamic set of users. Achieve flexible and fine-grained access control but those schemes are only applicable to systems in which data owners and the service providers are within the same trusted domain. Normally data owners and service providers are not in the same trusted domain in cloud computing. Enterprises usually store data in internal storage and install firewalls to protect against intruders to access the data. They also standardize data access procedures to prevent insiders to disclose the information without permission. In cloud computing, the data will be stored in storage provided by service providers. Service providers should not be a trusted one anyhow they are all third party. 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. 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. Verifying the authenticity of data has emerged as a critical issue in storing data on untrusted servers. It arises in peer- to-peer storage systems network file systems, long-term archives, web-service object stores. . Privacy protection technologies which are built on the hide-it-or-lose-it perspective, information accountability focuses on keeping the data usage transparent and tractable. However, archival storage requires guarantees about the authenticity of data on storage, namely that storage servers possess data. It have been modified or deleted when accessing the data, because it may be too late to recover lost or damaged data. Archival storage servers retain tremendous amounts of data, little of which are accessed. Furthermore, I/O incurred to establish data possession interferes with on-demand band- width to store and retrieve data. Normally data owners and service providers are not

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

in the same trusted domain in cloud computing .It is insufficient to detect that data have been modified or deleted when accessing the data, because it may be too late to recover lost or damaged data. Enterprises usually store data in internal storage and install firewalls to protect against intruders to access the data. Conclude that clients need to be able to verify that a server has retained file data without retrieving the data from the server and without having the server access the entire file.

## II.AUTHENTICATION OF JAR FILE

Generally many security schemes have been implemented to address the security issues. This is based on the restricted access to all portable applications to a system of authentication in which different levels of permission will be given based on whether the application can be authenticated as having come from the trusted source. In existing scheme they have used X.509 certificate to denote the authentication instead here we are using the signed ADF.APPLICATION DESCRIPTOR FILE which is used to authenticate a portable application code. This allows the developer to develop powerful applications. This work is relates in general to portable code transfer, such as JAVA technology, and more particularly to provide security and authentication of portable code.

### 2.1 Advantages

It can able to distribute applications to different *mobile devices*The information-gathering capability is high. Typically, a regular user-level process is not allowed to modify this table, while a legitimate installation of a system component (e.g., anti-virus software) may need to modify the system call table, thus the address space of the table needs to be updated.

With traditional access control model such as DAC, any hijacked root process such as a rootkit can change the system call table. Existing MAC policies such as SE Linux and LOMAC can prevent this but lack the capability to update object attributes (e.g., system call table function pointers) after a legitimate access. As another example, during a process's sensitive operation such as reading a sensitive data file, the process should be in a "good" running state such that whenever the process's integrity is changed, its access should be revoked immediately since a compromised process can maliciously release data to other entities.

## III.CLOUD PRIVACY AND SECURITY

Cloud computing has raised a range of important privacy and security issues .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 third-party. Concerns arise 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. To date, little work has been done in this space, in particular with respect to accountability.Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is deobfuscated by the privacy manager to reveal the correct result. However, the privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed.

A layered architecture for addressing the end-to-end trust management and accountability problem in federated systems,in that they mainly leverage trust relationships for accountability, along with authentication and anomaly detection. Researchers have investigated accountability mostly as approvable property through cryptographic mechanisms, particularly in the context of electronic Distributed jobs, along with the resource consumption at local machines are tracked by static software agents. The notion of accountability policies in is related to ours, but it is mainly focused on resource consumption and on tracking of sub jobs processed at multiple computing nodes, rather than access control.

## IV.MODULE DESCRIPTION

### 4.1 Developing a cloud environment

Initially the basic network model for the cloud data storage is developed in this module. Four different network entities can be identified as follows, Client(Data Owner): an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations; Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the clients' data; Certificate Authority: an entity, which has expertise and capabilities

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 7, July 2014

that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

Its main tasks include automatically logging access and 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.The logger will control the data access even after it is downloaded by user X.The logger requires only minimal support from the server in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement. which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

Furthermore, 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 correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement. The log harmonizer is responsible for 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 he log file is pushed back to the data owner.

The pull mode is an on-demand approach, the data owner as often as requested. These two modes allow us to satisfy the aforementioned fourth design requirement. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner. 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.

### 4.2 Proposing a CIA framework

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. 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.

### 4.3 Data Access in CIA

We develop the JAR file includes a set of simple access control rules specifying whether and how the cloud servers and possibly other data owners 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, we use CA wherein a trusted certificate authority certifies the CSP[3].Once the authentication succeeds, 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.

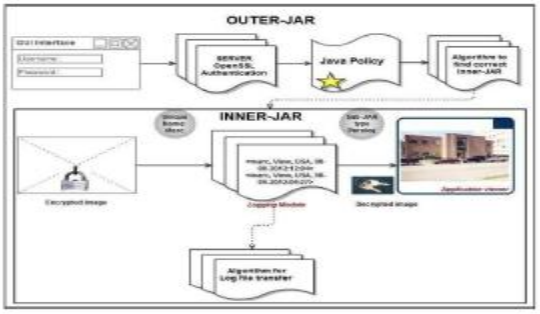


**Fig4.1 Structure of JAR file**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

**4.4 Develop Automated Logging Mechanism**

In the automated logging mechanism, the main responsibility of the 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. Each JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item. The log files are used for pure auditing purpose.It has two functions: logging actions and enforcing access control. In case an access request is denied, the JAR will record the time when the request is made. If the access request is granted, the JAR will additionally record the access information along with the duration for which the access is allowed. In the current system, we support four types of actions, i.e., Act has one of the following four values: view, download, timed-access, and Location-based access. For each action, we propose a specific method to correctly record or enforce it depending on the type of the logging module. And finally we design the end to end mechanism for auditing using push or pull mode configure for log files send to the data owner.Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item.

.

**4.5 Performance evaluation**

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. We can evaluate our performance by the following para meters they are Log Creation Time, Time Taken to Perform Logging, and Log Merging Time Size of the Data JAR Files. If the access request is granted, the JAR will additionally record the access information along with the duration for which the access is allowed. Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item.

## V.AUTOMATED LOGGING MECHANISM

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

**5.1 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..Hence, authentication is specified according to the server's functionality (which we assume to be known through a lookup service), rather than the server's URL or identity. In order for a system to scale, trust management must be decentralized to allow trust to be flexibly delegated between principals Accountable delegation is key in managing trust relationships and to trace misuse of the system .

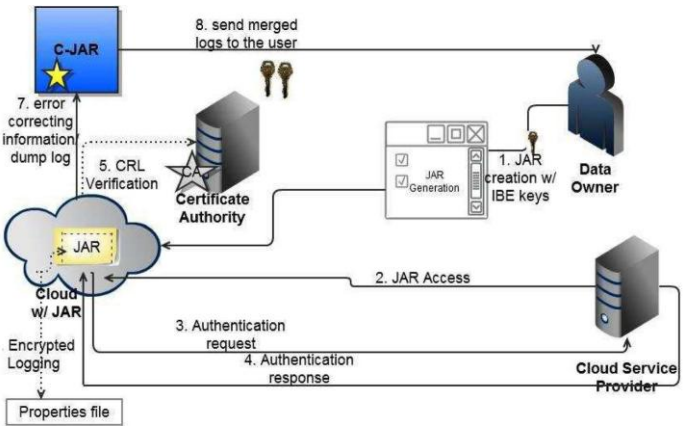


**Fig. 5.1 Cloud accountability framework.**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

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.

## VI.ALGORITHMS

Pushing or pulling strategies have interesting tradeoffs. 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 the cost of operations like copying data .The pushing mode may be preferred by data owners who are organizations and need to keep track of the data usage consistently over time. For such data owners, receiving the logs automatically can lighten the load of the data analyzers. The maximum size at which logs are pushed out is a parameter which can be easily configured while creating the logger component. The pull strategy is most needed when the data owner suspects some misuse of his data.

## VII. OVERVIEW

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.When there is a discrepancy the sequence of instructions must be traced to determine the problem. Looking at the code itself, the tester might notice that there is a branch (an if-then) and might write a second test case to go down the path not executed by the first test casepages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text.

### 7.1 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. 5. It is not surprising to see that the time to create a log file increases linearly with the size of the log file. Some schemes provide a weaker guarantee by enforcing storage complexity: The server has to store an amount of data at least as large as the client's data, but not necessarily the same exact data. Moreover, all previous techniques require the server to access the entire file, which is not feasible when dealing with large amounts of data. We define a model for provable data possession (PDP) that provides probabilistic proof that a third party stores a 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. 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.

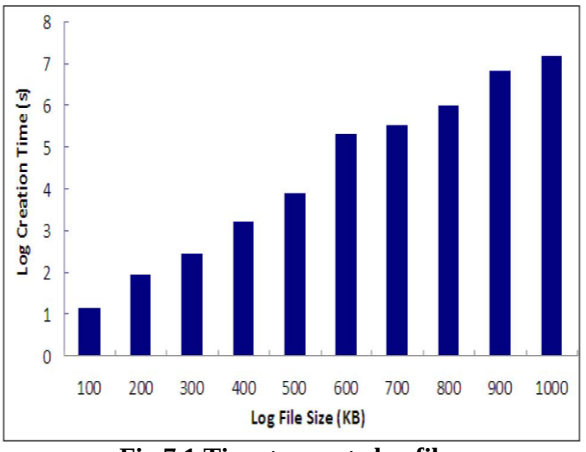


**Fig 7.1 Time to create log files**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

**7.2 Authentication Time**

The next point that the overhead can occur is during the authentication of a CSP[3]. 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 Open SSL certificates for the computing nodes and we measured the total time for the Open SSL authentication to be completed and the certificate revocation to be checked access at the time.

**7.3 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.

**7.4 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 repetitionsThe 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. You can use Service Manager to start, of these services.

Enterprise Manager is the main administrative console for SQL Server installations. It provides you with a graphical "birds-eye" view of all of the SQL Server installations on your network. You can perform high-level administrative functions that affect one or more servers, schedule common maintenance tasks or create and modify the structure of individual databases. It's a great way to quickly pull information out of a database in response to a user request, test queries before implementing them in other applications

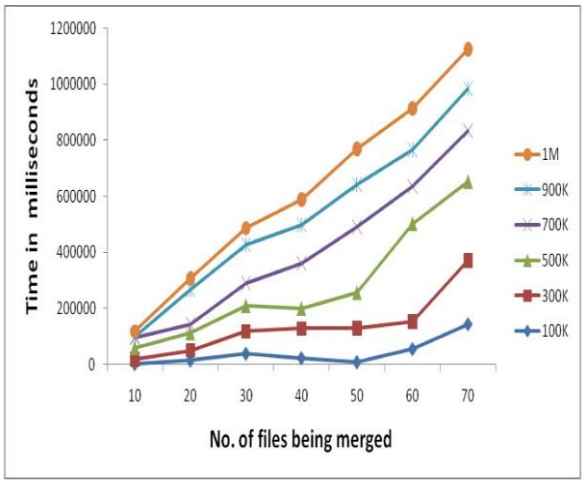


**Fig 6.4 Time to merge log files.**

**7.7.5 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

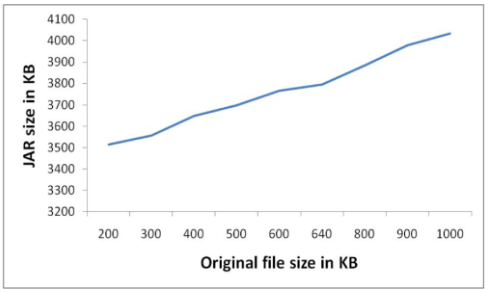


**Fig. 6.5 Size of the logger component**

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. Notice that we purposely did not include large log files (less than 5 KB), so as to focus on the overhead added by having multiple content files in a single JAR. Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at runtime.This makes it possible to dynamically link code in a safe and expedient manner for effective usage of data information.

**Using usage policies**

The proof system used to derive the actions on data that are allowed by the policies that a user posseses. We first give the inference rules followed by the notion provable. Note that each agents locally reasons about policies therefore the rules include the subject.

**Accountability**

As noticed in the example in the previous section, agents can potentially provide different justification proofs. We model an agent providing a proof of _ at time t as a function represents that the agent cannot provide a proof. We present two notions of accountability. The first notion, agent accountability, focuses on whether the actions of a given agent where authorized. The second notion, data accountability, expresses that a given piece of data was not misused.

**About Cloud Computing**

Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowchart s and diagrams.

**Re-Distribution**

To illustrate the problems related to policy distribution, we consider the example scenario of a movie provider. FairMovies produces all sorts of movies and distributes them through licensed dealers. These dealers sell the movies to customers tackle the problem of policy evolution in usage control when data is re-distributed.

## VIII.LINEAGE-RELATED STANDARDS

Federal metadata standards for geospatial data have included specifications for lineage as a component of data quality information since the Spatial Data Transfer Standard (SDTS) became a Federal Information Processing SDTS, developed for transferring geo referenced spatial data between dissimilar applications or computer systems, includes a brief text description of lineage as part of a data quality report.The SDTS influenced the Federal Geographic Data Committee (FGDC) Content Standard for Digital Geospatial Metadata (CSDGM) which defines explicit metadata elements corresponding to digital geospatial data sets, including lineage

**Localized location computation**

Some systems provide a location capability and insist that the object being located actually computes its own position. This model ensures privacy by mandating that no other entity may know where the located object is unless the object specifically takes action to publish that information.Placing the burden on the infrastructure decreases the

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

computational and power demands on the objects being located, which makes many more applications possible due to lower costs and smaller form factors. The policy for manipulating location data need not be dictated by where the computation is performed

### 8.1 SECURITY DISCUSSIONS

We now analyze possible attacks to our framework. Our analysis is based on a semi honest adversary model by assuming that a user does not release his master keys to unauthorized parties, while the attacker may try to learn extra information from the log files. We assume that attackers may have sufficient Java programming skills to disassemble a JAR file and prior knowledge of our CIA[2] architecture. We first assume that the JVM is not corrupted, followed by a discussion on how to ensure that this assumption holds true.

#### 8.8.1 Copying Attack

The most intuitive attack is that the attacker copies entire JAR files. The attacker may assume that doing so allows accessing the data in the JAR file without being noticed by the data owner. However, such attack will be detected by our auditing mechanism. 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 texts. log records are encrypted and breaking the encryption.

#### 8.8.2 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. Given the ease of disassembling JAR files, this attack poses one of the most serious threats to our architecture. Since we cannot prevent an attacker to gain possession of the JARs, we rely on the strength of the cryptographic schemes applied to preserve the integrity and confidentiality of the logs. Once the JAR files are disassembled. 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..

#### 8.8.3 Man-in-the-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.

#### 8.8.4 Compromised JVM Attack

An attacker may try to compromise the JVM. To quickly detect and correct these issues, we discussed in source code is not including the JAR files.This is, however, very hard to accomplish since, as mentioned earlier, log records are encrypted and breaking the encryption is computationally hard. Our solution relies on specifying rights and duties as intervals on events and parameters. To the best of our knowledge, we are the first to investigate the duality. which is not feasible when dealing with large amounts of data.

## IX. RELATED TECHNIQUES

Finally, in general outsourcing techniques have been investigated over the past few years. Although only is specific to the cloud, some of the outsourcing protocols may also be applied in this realm. In this work, we do not cover issues of data storage security which are a complementary aspect of the privacy issues. As you will learn below, many more tests are needed. Whenever possible, we should run customer acceptance test cases ourselves so that we can increase our confidence that they will work at the customer location.

## X.CONCLUSION

An object-centered approach that enables enclosing our logging mechanism together with users' data and policies, automatically logging any access to the data in the cloud together with an auditing mechanism and it allows the data owner to not only audit his content but also enforce strong back-end protection if needed. The data owner to audit even those copies of its data that were made without his knowledge and user's control also provide distributed auditing

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 2, Issue 7, July 2014**

mechanisms the efficiency and effectiveness of the proposed approaches. We can evaluate our performance by the following parameters they are Log Creation Time, Time Taken to Perform Logging, and Log Merging Time Size of the Data JAR Files.

## REFERENCES

[1] Ammann. P and Jajodia .S, "Distributed Timestamp Generation in Planar Lattice Networks," ACM Trans. Computer Systems, vol. 11, pp. 205-225, Aug. 1993.
[2] Ateniese. G, R., Burns.R, Curtmola ,.J,Herring..Z.Peterson.K and Song.D, "Provable Data Possession at Untrusted Stores," Proc. ACM Conf. Computer and Comm. Security, pp. 598- 609, 2007.
[3] Chun.B and Bavier.A.C, "Decentralized Trust Management and Accountability in Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS), 2004
[4] Corin.R, Etalle.S, Hartog.F, Lenzini.G, and Staicu.H, "A Logic for Auditing Accountability in Decentralized Systems," Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005
[5] Hightower.J and Borriello.G, "Location Systems for Ubiquitous Computing," Computer, vol. 34, no. 8, pp. 57-66, Aug. 2001
[6] Jaeger.P, Lin.T, and Grimes.J, "Cloud Computing and Information Policy: Computing in a Policy Cloud?," J. Information Technology and Politics, vol. 5, no. 3, pp. 269-283, 2009.
[7] Mont.M.C, Pearson.K, and Bramhall.O, "Towards Accountable Managementof the Identity and Privacy: Sticky Policies and Enforceable Tracing Services," Proc. Int'l Worksh
[8] Pretschner.A, Hilty.M, and Basin.D, "Distributed Usage Control," Comm. ACM, vol. 49, no. 9, pp. 39-44, Sept. 2006op Database and Expert Systems Applications (DEXA), pp. 377-382, 2003.
[9] Pretschner, F. Schuo¨ tz, C. Schaefer, and T. Walter, "Policy Evolution in Distributed Usage Control," Electronic Notes Theoretical Computer Science, vol. 244, pp. 109-123, 2009.
[10] Wang.Q, Wang.Q, Li.I, Ren.K, and Lou.K, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. European Conf. Research in Computer Security 9(ESORICS), pp. 355-370, 2009.