ISSN:0975-9646

Ensuring Distributed Accountability for Data Sharing in Cloud Using AES and SHA

Vikas Vitthal Lonare[1], Prof.J.N.Nandimath[2],

[1] *PG Scholar, Computer Department, SKNCOE, Pune, Savitribai Phule Pune University,*
[2]*Professor, Computer Department, SKNCOE, Pune, India*

Abstract-Cloud computing is used to provide scalable services which are easily used over the internet as per the need basis. Cloud computing is a technology which uses internet and remote servers to store data and application. In cloud, there is no need to install particular hardware, software on user machine, so user can get the required infrastructure on his machine in cheap charges/rates. A major feature of the cloud services is that user's data are remotely processed in unknown machines that users are not operating. Cloud computing based solutions are becoming popular and adopted widely because of its low-maintenance and commercial characteristics. While using these services provided by cloud computing, users fear of losing their own data. The content of data can be financial, health, personal. To resolve this problem, we have proposed information accountability in decentralized format to keep track of the usage of the user's data over the cloud. It is object oriented approach that enables enclosing our logging mechanism together with user's data and apply access policies with respect to user's data. We use JAR programming which provides the dynamic and traveling object functionality and to track any access to user's data will call authentication and automated logging mechanism to the JAR files. Each access to user's data will be getting recorded in separate log file. To provide robust users control, distributed auditing functionality is also provided to track the usage of data. The proposed system also provides the authentication mechanism using external channels and also makes a log of user details from whom the cloud data is accessed. Proposed system uses the Advanced Encryption Standard AES Algorithm along with Secure Hash Algorithm to enhance and provide robust security for data stored in cloud. Data owner will provide the type of access and allowed location to view his data and based on that, authorized cloud users can access the data over the cloud environment. In order to avoid modification in original private key file, data user's can verify the integrity of received private key file from the data owner. Only data owner can retrieve the detail access log information and download log information of his data as per requirement at any time. This paper proposes a model to provide strong security mechanism for Authentication, Authorization and Accountability of data sharing in the cloud environment.

General Terms*: *Cloud Computing, Cloud Services, Data Sharing

Keywords*: *Distributed Accountability, Cloud, Encryption and Decryption, Key Generation, JAR Authentication, Secure Hash, Advanced Encryption Standard

1. INTRODUCTION

Cloud computing is the delivery of computing services over the Internet as per requirement. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage system, social networking sites, webmail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere and anytime. Cloud computing provides a shared pool of resources, including data storage memory space, networks, computer processing power, and specialized corporate and user applications. Cloud computing has been developed by the U.S. National Institute of Standards and Technology (NIST). Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources e.g., networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models. The characteristics of cloud computing include on demand self service, broad network access, resource pooling, rapid elasticity and measured service. The cloud computing service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Deployment models of cloud services are public cloud, private cloud, community cloud, hybrid cloud. Cloud services are popular because they can reduce the cost and complexity of owning and operating computers and networks. Some other benefits to users include scalability, reliability, and efficiency. From a technical perspective, cloud computing includes service oriented architecture (SOA) and virtual applications of both hardware and software. Within this environment, it provides a scalable services delivery platform. Cloud computing shares its resources among a cloud of service consumers, partners, and vendors. By sharing resources at various levels, this platform provides various services, such as an infrastructure cloud (hardware or IT infrastructure management), a software cloud (software, middleware, or traditional customer relationship management as a service), an application cloud (application, UML modeling tools, or social networks as a service), and a business cloud for instance, business processes as a service. While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data. Such fears are becoming a significant barrier to the wide adoption of cloud services. Our proposed Cloud Information Accountability CIA framework provides end-to end accountability in a highly distributed fashion. By using proposed model, we can provide strong security mechanism for Authentication, Authorization and Accountability of data sharing in the cloud environment. There are 7 phases of accountability as follows,

1. Policy setting with data
2. Use of data by users
3. Logging
4. Merge logs
5. Error correctness in log
6. Auditing
7. Rectify and Improvement.

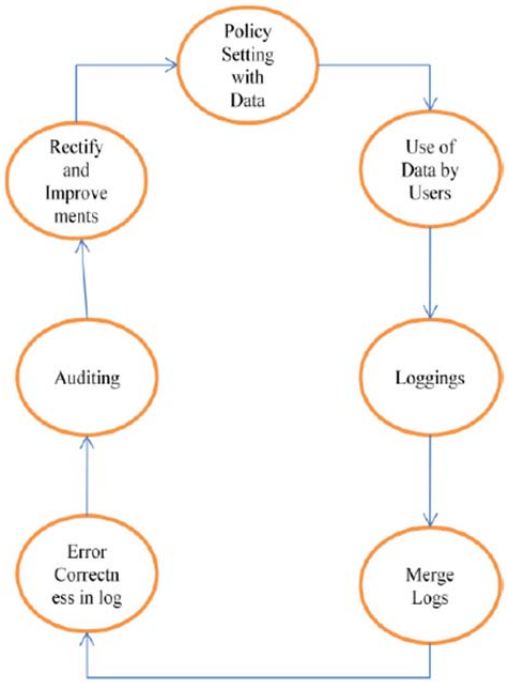

Fig1: Phases of Accountability

In the Fig1 Steps of accountability is given. These are 7 steps each step is important to perform next step. Accountability is nothing but validation of user actions, means user having rights for accessing this data or not. Suppose user will do misuse of data or resources then network or data owner will take action on it so, businesses and government should not bother about their data on cloud.

2. Related Work

Smitha S, Dan Lin proposes a novel highly decentralized information accountability framework [1] to keep track of the actual usage of data in cloud using JAR files. They have used oblivious hashing and SAMPL authentication. They have implemented the Push-Pull log mode.

Qian Wang, Cong Wang present third party auditor scheme in cloud computing using RSA and Bilinear Diffie Hellman techniques [2].

Parikshit Prasad, R Lal present a system in which data classification done by Owner before storing data [3]. Data categorized on the basis of Confidentiality, Integrity and Availability based on the Classification Algorithm.

P L Rini, Anand N Provides innovative approach for automatically logging and auditing mechanism using BASE64 encoding algorithm [6] to protect the data from attackers.

A. Squicciarini , S. Sundareswaran and D. Lin[12], authors gives a three layer architecture which protect information leakage from cloud, it provides three layer to protect data, in first layer the service provider should not view confidential data in second layer service provider should not do the indexing of data, in third layer user specify use of his data and indexing in policies, so policies always travel with data.

B. Chun and A. C. Bavier[13], authors present accountability in federated system to achieve trust management. The trust towards use of resources is accomplished through accountability so to resolve problem for trust management in federated system they have given three layers architecture, in first layer is authentication and authorization in this authentication does using public key cryptography. Second layer is accountability which perform monitoring and logging. The third layer is anomaly detection which detects misuse of resources. This mechanism requires third party services to observe network resources.

3. Proposed System

Cloud computing is a large infrastructure which provide many services to user without installation of resources on their own machine. This is the pay as you use basis model. Examples of the cloud services are Yahoo email, Google, Gmail and Hotmail. There are many users, businesses, government uses cloud, so data usage in cloud is large. So data maintenance in cloud is complex. Many Artists wants to do business of their art using cloud. For example, one of the artist want to sell his painting using cloud then he want that his paintings must be safe on cloud and no one can misuse his paintings. Here our proposed model can be used to achieve this security issue. The proposed model is suitable for the image files shared over the cloud environment. A user, who subscribed to a cloud service, usually needs to send his/her data as well as associated access control policies to the service provider. These access control policies should be define by the data owner. After the data received by the cloud service provider, the service provider will have granted access rights, such as read, write, and copy, on the user's data. In order to track the actual usage of the data; we proposed model which satisfy the following requirements:

1. The logging should be decentralized in order to adapt to the dynamic nature of the cloud. More specifically, log files should be tightly bounded with the corresponding data being controlled, and require minimal infrastructural support from any server.
2. Every access to the user's data should be correctly and automatically logged. This requires integrated techniques to authenticate the entity that accesses the data, verify, and record the actual operations on the data as well as the time that the data have been accessed.
3. Log files should be reliable and tamper proof to avoid illegal insertion, deletion, and modification by malicious parties. Recovery mechanisms are also desirable to restore damaged log files caused by technical problems.
4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data. More importantly, log files should be retrievable at anytime by their data owners when needed regardless the location where the files are stored.
5. The proposed technique should not intrusively monitor data recipients' systems, nor it should introduce heavy communication and computation overhead, which otherwise will hinder its feasibility and adoption in practice.
6. The proposed system should provide strong data security mechanism to protect data from network intruders. This includes strong encryption and decryption algorithm as well as private key verification to avoid unauthorized access to key and data over the cloud.
7. Only authorized users can access the data based upon the access privileged and location for which they are authorized to access the data.

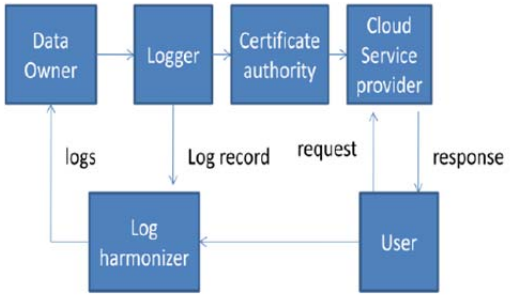

Fig2: Accountability in cloud

In Fig 2 working of accountability mechanism in cloud is shown. When user will access data then log of each access is created by logger and periodically sent to log harmonizer, log harmonizer send these logs to data owner and data owner can see logs and take appropriate action if he wants. State transition diagram is machine which shows no. of states, machine take input from outside world and each input can produce machine to go next step. Following transition diagram shows the different states of accountability mechanism in cloud i.e. how it changes from one state to next state.

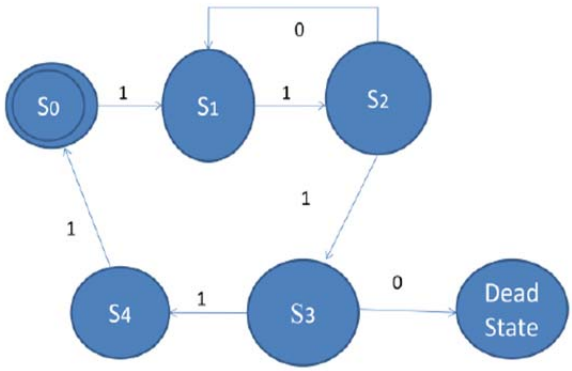

Fig3: State Transition Diagram

Where,
0: Unsuccessful
1: Successful
Transitions are:
S0: Data Owner will send data to logger.
S1: Data Owner will create logger which is a jar.
File to store data and policies.
S2: Authentication of CSP to JAR file.
S3: Authentication of user.
S4: owner can see merge log.
Input: = {0, 1}
Representation of
A= *(*{S0, S1, S2, S3, S4,} {0, 1}, δ, S0, S4)
Input given 11011011
Expected output
δ(S0,1) = S1
δ(S1,1) = S2
δ(S2,1) = S3
δ(S3,1) = S4
δ(S4,1) = S0
In accountability mechanisms the log records are generated as access of data in jar happened then it create log record log rec (Lr).
Lr = r1, r2, r3, r4... rk.
Parameters uses for log record are:
rk = (id, action, T, loc, h((id, action, T, loc)ri-1…r1), sig)
Where,
rk = log record
id = user identification
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
Checksum of each record is calculated and it is stored with data.
Checksum is computed using hash function
H[i] = f(H[i − 1] ,m[i]),

To achieve authentication and non-repudiation purpose within cloud computing environment, digital signature has assumed great significance. There are various digital signature algorithms which involves the generation of message digest (hash). MD5 and SHA-1 are well known digital signature generation algorithms and comparative study of these are described with the help of table:

Characteristics	MD5(Message Digest 5)	SHA-512
Message Digest Length	128	512
Attack (For Original message from message digest)	2^{128}	2^{512}
Attack (Find two message for same message digest)	2^{64}	2^{256}
Successful Attack	Some attempt reported	No such claim
Speed	Faster	Slow
Software Implementation	Very easy	Easy

Table1: Comparison of MD5 and SHA

The study shows that MD5 is much faster than SHA-512 digital signature algorithm, but with respect to security concerns SHA-512 is more secure than MD5 and no claim of successful attacks with optimal time complexity on SHA-512 has been done so far. The study of various cryptography (Symmetric/Asymmetric) encryption and digital signature algorithms helps to choose the best one from each category to be used in proposed cryptographic module. The symmetric and asymmetric encryption algorithms to be used are AES and ECC respectively. The SHA-512 digital signature generation algorithm is used in combination with ECC asymmetric key encryption algorithm. These algorithms are described as follows:

AES (Advanced Encryption Standard): The basic steps in algorithm are stated as:
a) Key Expansion - round keys are derived from the cipher key using Rijndael's key schedule
b) Initial Round AddRoundKey - each byte of the state is combined with the roundkey using bitwise xor
c) Rounds-
1. SubBytes - a non-linear substitution step where each byte is replaced with another according to a lookup table.
2. ShiftRows - a transposition step where each row of the state is shifted cyclicallya certain number of steps.
3. MixColumns - a mixing operation which operates on the columns of the state, combining the four bytes in each column.
4. AddRoundKey
d) Final Round (no MixColumns) - 1. SubBytes 2. ShiftRows 3. AddRoundKey
e) Key generation- This module handles key generation by the cryptographic module at client side. The server generates unique keys for users once they authenticate themselves with the server. The key is generated using instances of AES key generator class. This key is then transferred to the cloud client via the mail-server through a mail which receives and stores a copy for it for decrypting purpose.

3.1 Proposed Architecture

The proposed architecture consists of Cloud Service Provider CSP, Glassfish Server, MySQL Database and related peripherals. We have separate modules as Data Security Module and Key Verification module to enhance the security of data over cloud environment.

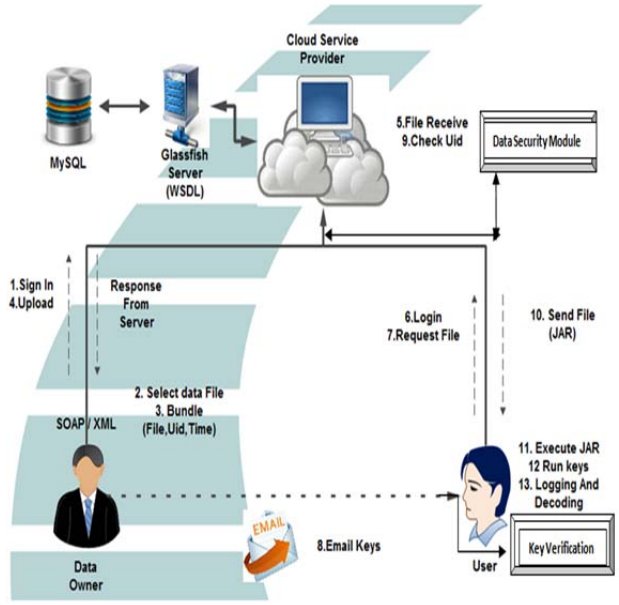

Fig4: Architecture of proposed system

3.2 Working

Data owner will sign in and select the data file which he wants to upload on cloud. This data file will be combining with some details like File name, user ID, Time. Data owner will receive the unique private key file for his file. He will upload the file and assign the access related permissions with respect to file. Data file will be converted into JAR (Java Archive) format. Then, this file will be get processed by the data security module. Here, strong encryption and decryption process is used to secure the file. Processed file will be getting stored in database by the CSP. Once any data user wants to access file then he has to login with valid credentials and request for the file. Only data owner can send the private key file to the data user via email. User details will be verified by the CSP and then JAR file will be sent to the user upon successful verification. We have added Key Verification module to check whether the received key has been modified in between transition or not. Received key should match with the original key. Data user will validate the key and provide the valid key details in order to access the JAR file. Data user can access the data as per the access rights he has. No other user can access the JAR file because; all this access will be defined by data owner only. All the activities done by the data user will be get logged and that log file will be retrieve by data owner as per the need. So data owner can view two different logs like Access Log and Download Log as per requirement.

Access Log Details:

{AccessID, UserID, File Name, AccessType, Status, Date-Time, Location, Hash Verification}

Download Log Details:

{DownloadID, UserID, File Name, Date-Time, Location, Hash Verification}

3.2.1 Advanced Encryption Standard Algorithm: Advanced Encryption Standard (AES) is the current standard for secret key encryption. AES was created by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, replacing the old Data Encryption Standard (DES). The Federal Information Processing Standard 197 used a standardized version of the algorithm called Rijndael for the Advanced Encryption Standard. The algorithm uses a combination of Exclusive-OR operations (XOR), octet substitution with an S-box, row and column rotations, and a MixColumn. It was successful because it was easy to implement and could run in a reasonable amount of time. Some of the applications of AES are still inflexible to various type of cracking techniques, which makes it a better choice even for top secret information. AES data encryption is more scientifically capable and graceful cryptographic algorithm, but its main force rests in the key length. The time necessary to break an encryption algorithm is straightly related to the length of the key used to secure the communication. AES allows you to choose a various type of bits like 128-bit, 192-bit or 256-bit key, making it exponentially stronger than the 56-bit key of DES.

AES Encryption Algorithm:

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
Begin
byte state[4,Nb]
state = in
AddRoundKey(state, w[0, Nb-1])
for round = 1 step 1 to Nr–1
SubBytes(state)
ShiftRows(state)
MixColumns(state)
AddRoundKey(state,w[round*Nb,(round+1)*Nb-1])
end for
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
out = state
end
```

AES Decryption Algorithm:

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
byte state[4,Nb]
state = in
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
for round = Nr-1 step -1 downto 1
InvShiftRows(state)
InvSubBytes(state)
AddRoundKey(state,w[round*Nb,(round+1)*Nb-1])
InvMixColumns(state)
end for
InvShiftRows(state)
InvSubBytes(state)
AddRoundKey(state, w[0, Nb-1])
out = state
end
```

The below table shows the comparison of DES and AES algorithms based upon the various parameters.

	DES	AES
Date	1976	1999
Block size	64	128
Key length	56	128, 192, 256
Number of rounds	16	9,11,13
Encryption primitives	Substitution, permutation	Substitution, shift, bit mixing
Cryptographic primitives	Confusion, diffusion	Confusion, diffusion
Design	Open	Open
Design rationale	Closed	Open
Selection process	Secret	Secret, but accept open public comment
Source	IBM, enhanced by NSA	Independent cryptographers

Table2: Comparison of AES and DES

3.2.2 Secure Hash Algorithm: A hashing algorithm is a cryptographic algorithm that can be used to provide data integrity and authentication. SHA typically used in password based systems to avoid the need to store plaintext passwords. A hashing algorithm is a deterministic function that takes in an arbitrary length block of data, and returns a fixed-size string, which is called the hash value. A message is transmitted with its hash, allowing the recipient to hash the message and compare outputs. By signing the hash before sending, the sender can prove that the message has not been tampered. The sender can hash a file before sending recipient. The recipient will then hash the file received and check the hashes match. This can also be used for the storage of files, to ensure files have not been corrupted or modified. A secure hashing algorithm has three main properties:

1. Preimage resistance: Given a hash value, it should be difficult to find any message that hashes to that value. 2. Second-preimage resistance: Given an input, it should be difficult to find another input, which is different to the first, where they both hash to the same value. 3. Collision resistance: It should be difficult to find two different inputs such that have the same hash values.

Elliptic Curve Cryptography (ECC) with SHA-512: An elliptic curve is given by an equation in the form of

$$y^2=x^2+ax+b \quad \text{where } 4a^3+27b^2\neq 0$$

The finite fields those are commonly used over primes (FP) and binary field (F2n). The security of ECC is based on the elliptic curve discrete logarithm problem (ECDCP).

4. Performance Study

We first bring out the settings of the test environment and then present the performance study of our system. We have tested our application in small private cloud network. In the experiments, we first examine the time taken to create log file and then measure the overhead in the system. With respect to time, the overhead can occur at three points: at the time of the authentication, during encryption of JAR file and at the time of 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 provided by the actual files and the associated logs. Further, JAR appears as a compressor of the files that it handles. As proposed, multiple files can be managed by the same logger component. To this extent, we checked whether a single logger component, used to manage more than one file, results in storage overhead.

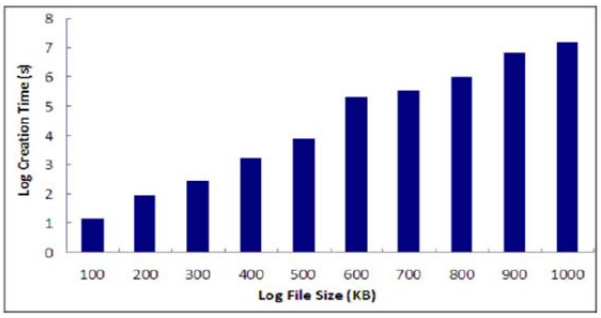

Fig5: Log Creation Time

4.1 Log Creation Time

In the first round of experiments, we are concerned in finding out the time taken to create a log file when there are entities continuously accessing the data, causing continuous logging. Resultants are shown in Fig. 5. It is not surprising to identify that the time to create a log file increases linearly with the size of the log file. Specifically, the time to develop 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 figure out the amount of time to be specified between dumps, keeping other variables like space constraints or network traffic in mind.

4.2 Authentication Time

The next point that the overhead can occur is during the authentication of a CSP. If the time taken for this authentication is too long; it may become a bottleneck for accessing the enclosed data. To evaluate this, considering one access at the time, we got that the authentication time averages around 520 ms which proves that not too much overhead is added during this phase. As of present, the authentication takes place each time the CSP needs to access the data. The time for authenticating an end user is about the same when we consider only the actions required by the JAR. When we consider the user actions (i.e., submitting his username to the JAR), it averages at 500ms.

4.3 Time Taken to Perform Logging

This set of experiments studies the effect of log file size on the logging performance. We measured the average time taken to allow an access plus the time to write the corresponding log record. The time for allowing any access to the data items in a JAR file includes the time to evaluate and enforce the applicable policies and to locate the requested data items. In the experiment, we let multiple servers continuously access the same data JAR file for a minute and recorded the number of log records generated. Every access is just a view request and hence the time for executing the action is negligible. As a resultant, the average time to log an action is about 10 seconds, which involves the time taken by a user to double click the JAR or by a server to run the script to open the JAR. We also took the log encryption time which is about 300 ms (per record) and is seemingly unrelated from the log size.

4.4 Log Merging Time

To check if the log harmonizer can be a bottleneck, we have taken the amount of time required to merge log files. In this experiment, we confirmed that each of the log files had 10 to 25 percent of the records in common with one other. The exact number of records in common was random for each repetition of the experiment. The time was averaged over 10 repetitions. We tested the time to merge up to 70 log files of 100 KB, 300 KB, 500 KB, 700 KB, 900 KB, and 1 MB each. We can see that the time increases almost linearly to the number of files and size of files, with the least time being acquired for merging two 100 KB log files at 59 ms, while the time to merge 1 MB files was 2.00 minutes.

4.5 Size of the Data JAR Files

Finally, we investigate whether a single logger, account to handle more than one file, results in storage overhead. We have figured out the size of the loggers (JARs) by varying the number and size of data items held by them. We tested the accession in size of the logger containing 10 content files (i.e., images) of the same size as the file size expands. Intuitively, in case of large size of data items held by logger, the overall logger also expand 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, because of the compression provided by JAR files, the size of the logger is commanded by the size of the largest files it holds. Results are in Fig.6.

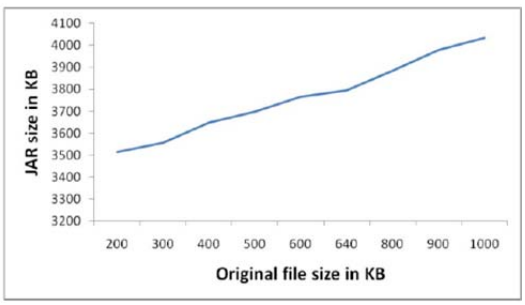

Fig6: JAR File Size

4.6 Overhead Added by JVM Integrity Checking

We investigate the overhead added by both the reinstallation/repair process, and by the time taken for computation of hash codes. The time taken for JRE installation/repair averages around 6,500 ms. This time was figured by taking the system time stamp at the beginning and end of the installation/repair. To calculate the time overhead added by the hash codes, we simply measure the time acquired for each hash calculation. This time is taken to average around 9 ms. The number of hash commands varies on the basis of size of the code. If the code does not change with the content, the number of hash commands remains constant.

5. CONCLUSION

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. One of the main features of system is, it enables the data owner to audit even those copies of his data that were made without his knowledge. Robust and Secure data share in cloud using secure external channel authentication. Encryption algorithms play an important role in data security on cloud and by comparison of different parameters used in algorithms, it has been found that AES algorithm uses least time to execute and it is the robust algorithm to secure the data over cloud. This algorithm has speedy key setup time and good key agility. It requires less memory for implementation, making it suitable for restricted-space environments. There are no serious weak keys in AES. Statistical analysis of the cipher text has not been possible even after using huge number of test cases. No differential and linear cryptanalysis attacks have been yet proved on AES. AES provides security to cloud users as encrypted data in the cloud is safe from many security attacks. SHA is more secure than MD5 and no claim of successful attacks with optimal time complexity on SHA has been done so far. By using the proposed system, only authorized data users can access the data as per the access rules defined by the data owner. Hence proposed system provides robust and secure data share for image files on cloud environment using AES and SHA Algorithm.

6. FUTURE ENHANCEMENT

-Support a variety of security policies, like indexing policies for text less, usage control for executables and generic accountability and provenance controls.

-Refine approach to verify Integrity of JRE and Authentication of JAR files in order to support multiple file formats.

-Reduce the size of JAR files to avoid communication delay and increase the speed of JAR file uploading and downloading from cloud environment.

REFERENCES

[1] Smitha Sundareswaran, Anna C. Squicciarini, Member, IEEE, and Dan Lin "Ensuring Distributed Accountability for Data Sharing in the Cloud" IEEE Trans actions On Dependable And Secure Computing, Vol. 9, No. 4, July/August 2012

[2] Qian Wang, Student Member, IEEE, Cong Wang, Student Member, IEEE, Kui Ren, Member, IEEE, Wenjing Lou, Senior Member, IEEE, and Jin Li "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing" IEEE Transactions On Parallel And Distributed Systems, Vol. 22, No. 5, May 2011

[3] Parikshit Prasad,R Lal "3 Dimensional Security in Cloud Computing" IEEE 2011

[4] Xiao Zhang, Hong-tao Du ,Jian-quan Chen, Yi Lin, Lei-jie Zeng Ensure Data Security in Cloud Storage IEEE 2011

[5] Yubo Tan , Xinlei Wang Research of Cloud Computing Data Security Technology IEEE 2012

[6] P L Rini, Anand N " Encoding Personal Information On Data Sharing In Cloud Using BASE64 Algorithm" GRET 2013

[7] MdMasoom Rabbani, Ilango Paramasivam "Enhancing Accountability for Distributed Data Sharing in the Cloud" IJET Jun-Jul 2013

[8] P Sobha Rani, P. Suresh Babu "Achieving Information Accountability in Cloud Computing Environment" IJCER April 2013

[9] Drishya S G, Kavitha Murugeshan "Towards Achieving Secured and Decentralized Accountability in Cloud Computing" IJCTT May 2013

[10] Prema Mani, Janahanlal P Stephan "Enhanced Accountability Framework for Data Sharing in the Cloud" ICCSE April 2013

[11] Drishya S G, Kavitha Murugeshan "Towards Achieving Secured and Decentralized Accountability in Cloud Computing" IJCTT May 2013

[12] A. Squicciarini , S. Sundareswaran and D. Lin, " Preventing Information Leakage from Indexing in the Cloud," *Proc. IEEE Int'l* Conf. Cloud Computing, 2010

[13] B. Chun and A. C. Bavier ,"Decentralized Trust Management and Accountability in Federated System," *Proc. Ann. Hawaii Int'l Conf. System Science (HICSS), 2004*

[14] Drishya S G, Kavitha Murugeshan "Towards Achieving Secured and Decentrali zed Accountability in Cloud Computing" IJCTT May 2013