ISSN: 2320-1363

An Efficient Accountability for Data Sharing In The Cloud Distribution System

Mohammed Younus Shariff*1, Akheel Mohammed*2, Dr. Vasumathi*3

Abstract

Cloud computing is the use of computing of sources that are delivered as a service over a network. It enables highly scalable services to be easily consumed over the internet on an as needed basis. A major characteristic of the cloud services is that user's data are usually processed remotely in unknown machines that users do not operate. While enjoying the convenience brought by this new emerging technology, user's fear of losing control of their own data can become a significant barrier to the wide adoption of cloud services. It can become a substantial roadblock to the wide adoption of cloud services. To adders this problem we propose a highly decentralization answerability frame work to keep track of the actual usage of the users data in the cloud. The cloud information accountability frame work 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. We propose an object centered approach that enables enclosing our logging mechanism together with user's data and policies. The proposed methodology will also take concern of the JAR file by converting the JAR into obfuscated code which will adds an additional layer of security to the infrastructure. Apart from that we are going to increase the security of users data by provable data possession for integrity verification.

Keywords: Cloud Computing, Data Sharing, Information accountability framework, Provable data possession.

1. Introduction

The cloud information accordingly framework proposed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, taken out at any point of time at any cloud service provider. It has two major components logger and log harmonizer. The JAR file includes a set of simple access control rules specifying whether and how the cloud services and possibly other data stakeholders are authorized to access the content itself. Apart from that we are going to check the integrity of the JRE on the system on the systems on which the logger components is initiated. While enjoying the convenience brought by this new technology users also start worrying about losing control of their own data. The data processed on cloud are often outsourced leading to a number of issues related to accountability, including the handling of personally identifiable information. Such fears are becoming a significant barrier to wide adoption of cloud services. This integrity checks are carried out by using oblivious hashing. The proposed methodology will also take concern of the JAR file by converting the JAR into obfuscated code which will adds an additional layer of security to the infrastructure. Apart from that we are going to enlarge the security of users' data by provable data possessions for integrity verification. Based on the configuration settings defined at the time of creation, the JAR will give usage control associated with logging, or will give only logging functionally. As for the logging, every time there is an access to the data, the JAR will automatically produce a log record. In existing system the cloud computing is the delivery of computing as a service rather than a product by which shared resources, software and information are given to computers and other devices as utility like the electricity grid over a network. In these days a single server deals with the multiple requests from the user. Here the server has to operate the both the request from the user simultaneously, so the processing time will be high. This may lead to deficit of data and pockets may be delayed and corrupted and also the data management and services are not trust worthy. While enjoying the convenience brought by this new technology, users also start bothering about losing control of their own data. The data operated on clouds are often outsourced, which lead to a number of issues related to accountability, including the management of personally identifiable information. To allay users concerns it is necessary to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example users required 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, or approaches with a centralized server in distributed environment are not suitable because of the following features characterizing cloud environment.

- Disadvantage: Although the cloud computing is vast developing technology, the database management system does not have trustworthiness.

We propose a novel automatic and enforceable logging mechanism in the cloud. To our knowledge, this is the first time a systematic approach to data accountability through the novel usage of JAR files is proposed.

This article is an extension of our previous conference paper. We have made the following new contributions. First we integrated integrity checks and obvious hashing technique to our system in order to strengthen the dependability of our system in case of compromised JRE. We are also updated the log records structure to provide additional guarantees of integrity and authenticity. We extend the security analysis to cover more possible attack scenarios. Then we report the results of new experiments and provide a thorough evaluation of the system performance.

ISSN: 2320-1363

2. Related Work

First review related for works addressing the privacy and security issues in the cloud. Then we briefly discuss works which adopt similar techniques as our approach but server for different purposes.

2.1 cloud privacy and security

Their basic idea is that the users private data are sent to the cloud in an encrypted from and 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. The authors focus is very different from ours in that they mainly leverage trust relationships for accountability along with authentication and relationships for accountability, along with authentication and anomaly detection. Researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of electronic commerce. Crispo and ruffo proposed an interesting approach related to accountability in case of delegation. Delegation is complementary to our work in that we do not aim at controlling the information workflow in the clouds. The best of our knowledge, the only work proposing a distributed approach to accountability is from lee and colleagues. The authors have proposed an agent based system specific to grid computing.

2.2 Related Techniques

Java based techniques for security of our methods are related to self defending objects. Self defending objects are an extension of the object oriented programming paradigm, where software objects that offer sensitive functions or hold sensitive data are responsible for protecting those functions or data. We provided a java based approach to prevent privacy leakage from indexing which could be integrated with the CIA framework proposed in this work since they build on related architecture. Appel and Felten proposed the proof carrying authentication framework. The PCA includes a high order logic language that allows quantification over predicates and focuses on access control for web services. Another work is by Mont et al. who access control using identifying based encryption. We also leverage IBE techniques but in a very different way. We do not rely on IBE to bind the content with the rules. We use it to provide strong guarantees for the encrypted content and the log files, such as protection against chosen plaintext and cipher text attacks. In this work we do not cover issues of data storage security which are a complementary aspect of the privacy issues.

3. Proposed system

To overcome the existing problems, we propose a novel method, namely Cloud Information Accountability framework, based on the notion of information accountability. Data owner can upload the data into the cloud server after encrypted the data. Users can subscribe into the cloud server with certain access polices such as read, write and copy of the original data. The Loggers and Log Harmonizer will have a track of the access logs and report to the data owner. 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.

- Advantage: we can share the data in a secured manner.

3.1 Data Flow

At the beginning each user creates a pair of public and private keys based on Identify Based Encryption. This IBE scheme is a Weil pairing based IBE scheme which protects us against one of the most prevalent attacks to our architecture as described. Using the generated key the user will create a logger component which is a JAR file to store its data items.

ISSN: 2320-1363

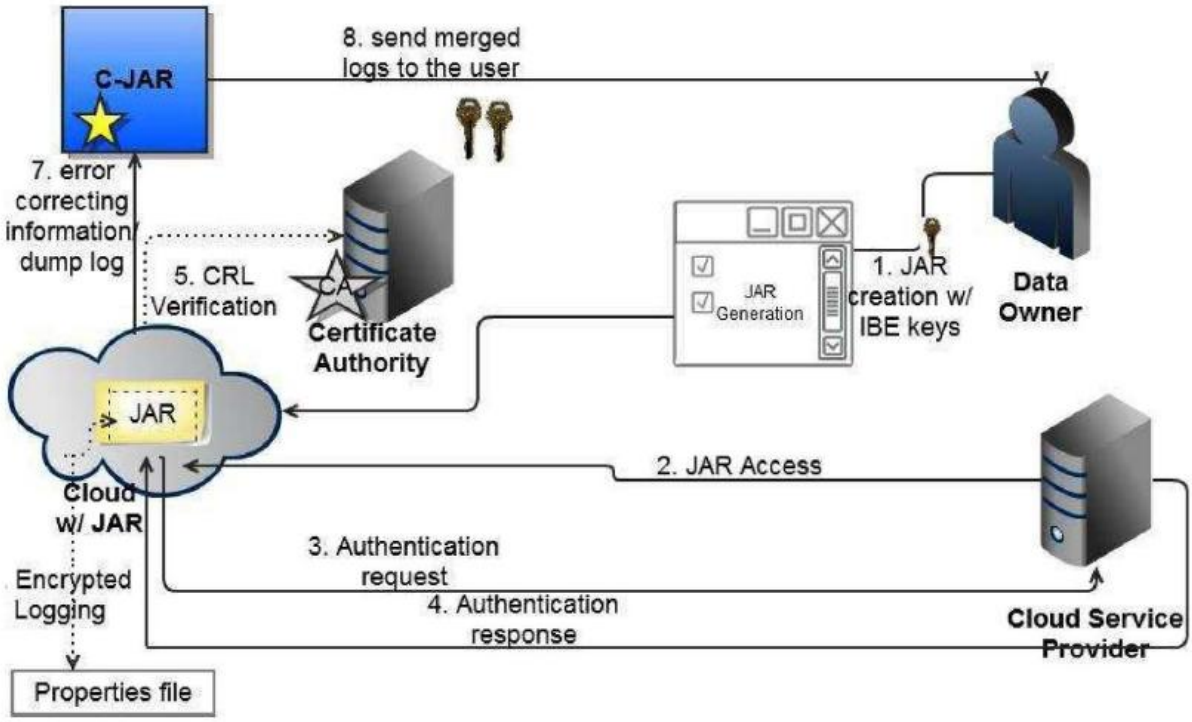

Fig 1. Overall Architecture.

Modification:

Automatic reporting of illegal action performance of any use to the data owner, along with data owner would generate the random numbers set for the every user. So if the user accessing the account the user has to give the random number set and that will be verified by the server. If the verification resultant is positive then only the user will be allotted to access their account.

4. Implementation

4.1 Evolution of Proposed Algorithm

The algorithm here used is Log Retrieval Algorithm for Push and Pull modes. The algorithm presents logging and synchronization steps with the harmonizer in case of pure Log. First the algorithm checks whether the size of the JAR has exceeded a stipulated size or the normal time between two consecutive dumps has elapsed. The size and time threshold for a dump are specified by the data owner at the time of development of the JAR. The algorithm also determines whether the data owner has requested a dump of the log files. If none of these events has happened it proceeds to encrypt the record and write the error correction information to the harmonizer. The interaction with the harmonizer begins with a simple handshake. If no response is received the log file records an error. The data owner is then altered via emails, if the JAR is configured to send error notifications. Once the handshake is done, the interation with the harmonizer proceeds using a TCP/IP protocol. If either of the aforementioned events has happened, JAR simply dumps the log files and resets all the variables, to make a space for a new record. In case of Access Log checks whether the CSP accessing the log satisfies all the conditions specified in the policies pertaining to it. If the conditions are fulfilled, access is granted otherwise, access is declined. Irrespective of the access control outcome, the attempted access to the data in JAR file will be logged. Our auditing mechanism has two fundamental advantages. First it guarantees a high level of availability of the logs. Second the usage of the harmonizer minimizes the amount of workload for human users in going through long log files sent by different copies of JAR files. In this system we are used different types of modules these are

4.1.1 User or Data Owners

The person or user is going to sea or download the data from the cloud server. Access the data from the cloud server the user have to be registered with the cloud server. So that the user have to register their details like username, password and a set of random numbers. This is the information that will store in the database for the future authentication. Data owner is the person who is going to upload the data in the Cloud Server. In order to upload the data into the Cloud server, the Data Owner have to be registered in the Cloud Server. Once the Data Owner registered in cloud server, the space will be assigned to the Data Owner.

4.1.2 Cloud Server

ISSN: 2320-1363

Cloud Server is the area where the user going to request the data and also the data owner will upload their data. When the users send the request regarding the data they want the request will need first send to the cloud server and cloud server and the cloud server will forward your request to the data owner. The cloud server will also manage the data owner and users information in their Database for future purpose.

4.1.3 Logger

The logger is maintained by the Cloud Server. Loggers have the details of the data owner and users who are accessing the cloud server. So the Logger will be more useful for many purposes. Like which data owner accessing the cloud server accessed at the particular time and the IP address from which the data is requested by user.

4.1.4 Certification Authority

The certificate authority is used to verify the Cloud server is recognized or not. The cloud server has to be recognized by the certificate authority. If not recognized the Cloud Server is a Fraudulent Server. The data owner can check the whether the recognized or not. Because the data owner is going to upload their data in the Cloud Server.

4.1.5 Access Privileges

Some Owners will provide read only, some of them will allow read and download. The Cloud Server will send the dynamic intimation when the user is accessing the data beyond their limits. This increases more security while sharing the data in the cloud.

4.1.6 Push and Pull Concept

The data owner may able to know who're all the accessing their data at the particular time period. During the registration phase the data owner will ask by the cloud server whether they're choosing the push or pull method. In pull method, the data owner has to send the request to the cloud server regarding the access details of their data up to the particular time.

4.1.7 Random Set Generation and Verification

When the user request the data to be downloaded from the Cloud Server, the user have to enter the random number set. If it is matched the user is allowed to download the data. Each and every time the random number set will vary. This ensures security while downloading the dat.

5. Experiment Results

We first bring out the settings of the test environment and then present the performance study of our system. Note that we do not consider the attack on the log harmonizer component, since it is saved separately in either a secure proxy or at the user end and the attacker typically cannot access it. As a result we consider that the attacker cannot extract the decryption keys from the log harmonizer.

In the experiments first we examine the time taken to create log file and then measure the overhead in the system. At the time of the authentication during encryption of a log record and at the time of the merging of the logs. Further JAR appears as a compressor of the files that it handles. In particular as proposed multiple files can be managed by the same logger component. To this extent we checked whether a single logger component used to managed more than one file result in storage overhead.

5.1 Log Creation Time

In the first round of experiments, we are concerned in finding put the time taken to create a log file when there are entities continuously accessing the data causing continuous logging. Results are shown in fig2. 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.

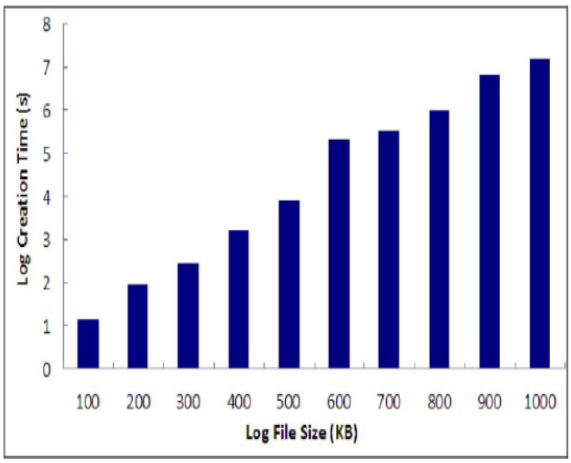

Fig .2. Time to merge log files.

5.2 Authentication Time

The performance can be further improved by caching the certificates. The time of authentication an end user is about the same when we consider only the actions required by the JAR, viz. acquiring a SAML certificate and then evaluating it. This is done to both the OpenSSL and the SAML certificates are handles in a similar fashions by the JAR. When we consider the users actions it averages at 1.2 minutes.

5.3 Time Taken to Perform Logging

This set of experiment studies the effect of log file size on the logging performance. We measured the average time taken to

ISSN: 2320-1363

allow an access plus the time to write the corresponding log record. 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 and is seemingly unrelated from the log size.

5.4 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 article 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. The results are shown in Fig.3. 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 70 1 MB files was 2.35 minutes.

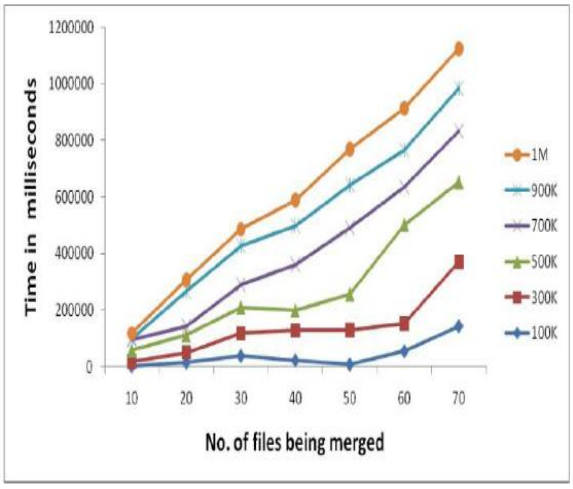

Fig.3. Time to create log files of different sizes.

The size of logger grows from 3500 to 4035 KB when the size of the 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. Notice that we priposely did not include large log files. Sa as to give attention on the overhead added by having multiple content files in a single JAR. Results are in Fig.4.

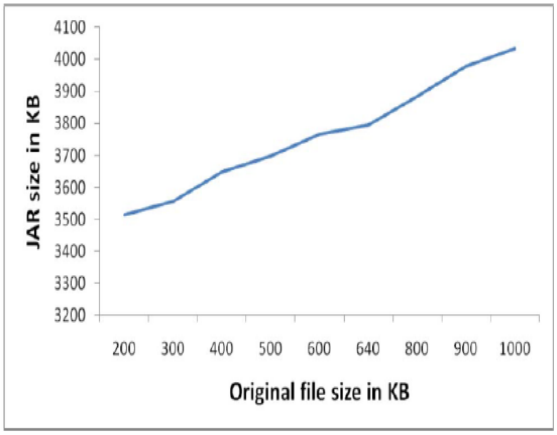

Fig.4. Size of the logger component

6. Conclusion

We introduced modern approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back end protection if needed. Apart from that we have enclosed PDP methodology to enhance the integrity of owners data. In future we plan to refine out approach to verify the integrity of JRE. For that we will look into whether it is possible to leverage the advantage of secure JVM being development By IBM and we would like to enhance our PDP architecture from user end which allows the users to check data remotely in an efficient manner in multi cloud environment.

References

1. Text Books

[1] Cloud Computing, Principles and Paradigms by John Wiley & Sons.

2. Conference Proceedings

[1] Ensuring Distributed Accountability for Data Sharing in the Cloud Author, Smitha Sundareswaran, Anna C.Squicciarini, Member, IEEE, and Dan Lin, IEEE Transactions on Dependable and Secure Computing ,VOL 9,NO,4 July/August 2012 .

[2] Hsio Ying Lin,Tzeng.W.G, **"**A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding ",IEEE transactions on parallel and distributed systems,2012.

[3] Yan Zhu, Hongxin Hu, Gail Joon Ahn, Mengyang Yu, "Coopera-tive Provable Data Possession for Integrity Verification in Multi-Cloud Storage" , IEEE transactions on parallel and distributed systems,2012.

3. Generic Website

ISSN: 2320-1363

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

[2] Emulab Network Emulation Testbed, www.emulab.net, 2012.