RESEARCH ARTICLE OPEN ACCESS

Data Sharing Using Cloud Information Accountability Framework

Chaitanya Chavali*, Lakshmi Prasad Koyi**, Dr.C. S Kumar***, Dr. N. Raghava Rao****

*Department of Computer Science and Engineering, QIS College of Engineering and Technology, Ongole-523272.

**Assistant Professor, Department of Computer Science and Engineering, QIS College of Engineering and Technology.

*** Professor, Department of Computer Science and Engineering, QIS College of Engineering and Technology.

****Professor, Department of Computer Science and Engineering, QIS College of Engineering and Technology.

Abstract
Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Unlike privacy protection technologies, information accountability focuses on keeping the data usage transparent and traceable. Our proposed CIA framework provides end-to end accountability in a highly distributed fashion. One of them an innovative feature of the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication, and security issues.

Index Terms: Cloud computing, Framework, Accountability, Data sharing, Cloud Service Provider.

I. INTRODUCTION

Cloud computing is the use of computing resources(H/w & S/w) that are delivered as a service over a network (typically the internet).Cloud computing refers to the delivery of computing and storage capacity as a services to a heterogeneous community of end-recipients the name comes from the use of clouds as an abstraction for the complex infrastructure. It provides remote services with a user's data, software and computation over a network. Cloud computing is the newest term for the long-dreamed vision of computing as a utility. Cloud computing is scalable services. Cloud computing is a computing platform that resides in a large data center and is able to dynamically provide servers the ability to address a wide range of needs, ranging from scientific research to e-commerce. Cloud computing is expanding rapidly as service used by a great many individuals and organizations internationally, policy issues related to cloud computing. Details of the services provided are abstracted from the users who no longer need to be experts of technology infrastructure. Moreover, users may not know the machines which actually process and host their data. While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data. The data processed on clouds are often outsourced, leading to a number of issues related to accountability, including the handling of personally identifiable information. It is necessary to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example, users need to be able to ensure that their data are handled according to the service level agreements made at the timeThey sign on for services. Conventional access control approaches developed for closed domains such as Databases and operating systems, or approaches using a centralized server in distributed environments, are not suitable, due to the following features characterizing cloud environments. First, data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the Cloud and these entities can also delegate the tasks to others, and so on. Outsourcing of data processing invariably raises governance and accountability questions. Second, entities are allowed to join and leave the cloud in a flexible manner. As a result, data handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments. Cloud computing is expanding rapidly as service used by a great many individuals and organizations internationally, policy issues related to cloud computing.

We propose a, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Privacy protection technologies built on the hide-it-or-lose-it perspective, information accountability focuses on keeping the data usage transparent and track able.

Our proposed CIA framework provides end-to end accountability in a highly distributed fashion. One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. Data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed. Associated with the accountability feature, we also develop two distinct modes for auditing: push mode and pull mode. The push mode refers to logs being periodically sent to the data owner or stakeholder while the pull mode refers to an alternative approach whereby the user (or another authorized party) can retrieve the logs as needed.

The design of the CIA framework presents substantial challenges, including uniquely identifying CSPs, ensuring the reliability of the log, security issues, accountability (The Obligation of an individual(or) Organization to account for its activities, accept responsibility for them.), adapting to a highly decentralized infrastructure, etc. Our basic approach toward addressing these issues is to leverage and extend the programmable capability of JAR (Java Archives) files to automatically log the usage of the users' data by any entity in the cloud. JAR file includes a set of simple access control rules specifying whether and how the cloud servers and possibly other data stakeholder's are authorized to access the content itself. JAR will provide usage control associated with logging (or) will provide only logging associated with logging functionality. Users will send their data along with any policies such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Any access to the data will trigger an automated and authenticated logging mechanism local to the JARs. We refer to this type of enforcement as "strong binding" since the policies and the logging mechanism travel with the data. This strong binding exists even when copies of the JARs are created; thus, the user will have control over his data at any location. Such decentralized logging mechanism meets the dynamic nature of the cloud but also imposes challenges on ensuring the integrity of the logging. To cope with this issue, we provide the JARs with a central point of contact which forms a link between them and the user. It records the error correction information sent by the JARs, which allows it to monitor the loss of any logs from any of the JARs. Moreover, if a JAR is not able to contact its central point, any access to its enclosed data will be denied.

The results demonstrate the efficiency, scalability, and granularity of our approach. We also provide a detailed security analysis and discuss the reliability and strength of our architecture. The following new contributions. First, we integrated integrity checks and oblivious hashing (OH) technique to our system in order to strengthen the dependability of our system in case of compromised JRE. We also updated the log records structure to provide additional guarantees of integrity and authenticity. Second, we extended the security analysis to cover more possible attack scenarios. Third, we report the results of new experiments and provide a thorough evaluation of the system performance. Fourth, we have added a detailed discussion on related works to prepare readers with a better understanding of background knowledge. Finally, we have improved the presentation by adding more examples and illustration graphs.

II. ENHANCING THE ACCOUNTABILITY

Cloud computing may be a massive infrastructure which give several services to user while not installation of resources on their own machine. This is often the pay as you utilize model. Samples of the cloud services are Yahoo email, Google, Gmail and Hotmail. There are several users, businesses, government uses cloud, thus knowledge usage in cloud is massive. Thus knowledge maintenance in cloud is advanced. Several Artists desires to try to business of their art victimization cloud. As an example one amongst the creative person need to sell his painting victimization
Cloud then he need that his paintings should be safe on cloud nobody will misuse his paintings.

A. Cloud Ingredients

There is need to be compelled to offer technique that is ready to audit information in cloud. On the idea of accountability, we've an inclination to projected one mechanism that keeps use information clear suggests that data owner got to get information regarding use of his information. This process support accountability in distributed area, data owner should not problem regarding his information, he may acknowledge his information is handled per service level agreement and his information is riskless on cloud. Data owner will determine the authorization principles and policies and user will handle information victimization this rule and logs of each information access are created. Throughout this mechanism there are unit two main parts i.e. logger and log harmonizer. The feller is with the data owner's information, it provides work access to information and encrypts log record by pattern public key that's given by data owner and send it to log harmonizer. The log harmonizer is taking part in the

observance and rectifying, it generates the key it holds cryptography key decrypting the logs, and at the consumer side cryptography it sends key to shopper. Throughout this mechanism data owner will creates personal key and public key, pattern generated key owner will produce feller that will be a JAR file, it encloses his authorization principles and work policies with information send to cloud service provider.

Authentication of cloud service provider has been done exploitation open SSL based totally certificates once authentication of cloud service provider user are able to access information in JAR, log of each data usage has been generated and encrypted exploitation public key and it automatically send to log harmonizer for integrity log records are signed by entity that's exploitation the information and log records are decrypted and accessed by owner. In push state logs are automatically transferred to data owner and in pull state owner may claim logs, therefore he may observe information access at anytime, anywhere and he can do inspection of his information.

Overall Architecture

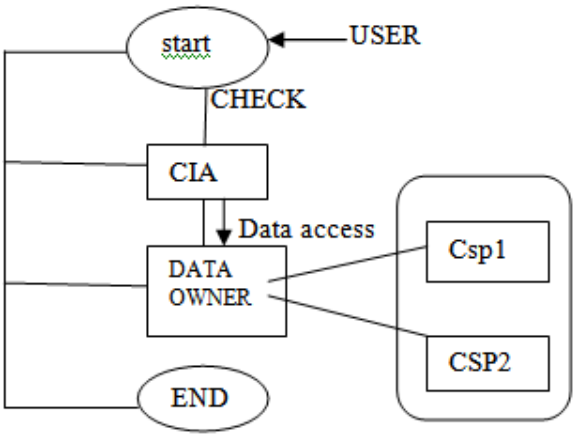

Fig.1

The main intention of architecture is, here data owner, cloud service provider, user. Every user can register first, i.e. data owner, normal user, csp. The overall CIA framework, combining information, users, logger and harmonizer is sketched in figure. At the start, every user creates a combine of public and personal keys supported Identity-Based encoding. This IBE scheme could be a Weil-pairing-based IBE scheme that protects us against one among the most current attacks to our design as described in exploitation the generated key, the user can produce a logger part that may be a JAR file, to store its data items. (FIG.1)

B. Flow of Data

The JAR file includes a collection of easy access management rules specifying whether and the way the cloud servers, and probably different information stakeholders (users, companies) are licensed to access the content itself. At the same time, he transfers the JAR file to the cloud service provider that he subscribes to. To certify the CSP to the JAR (FIG.1.1) we have a tendency to use open SSL-primarily based certificates, whereby a trustworthy certificate authority certifies the CSP. Within the event that the access is requested by a user, we have a tendency to use SAML-based authentication, whereby a reliability identity provider problems certificates confirmative the user's identity supported his username.

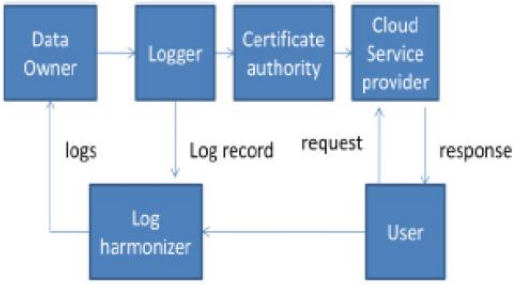

FIG.1.1 **Accountability Mechanism in cloud**

Once the authentication succeeds, the service providers (or the user) are going to be allowed to access the information enveloped within the JAR. Depending on the configuration settings outlined at the time of creation, the JAR can give usage management related to logging, or can give solely work practicality. As for the work, when there's associate access to the information, the JAR can mechanically generate a log record, encipher it victimization the general public key distributed by the data owner, and store it alongside the information. The encoding of the log file prevents unauthorized changes to the file by attackers.

The data owner could opt to reuse the same key pair for all JARs or create different key pairs for different JARs. Using separate keys are able to improve the authorization without introducing any overhead except in the starting phase. In inclusion, some error correction data will be sent to the log harmonizer to handle possible log file corruption. To ensure reliability of the logs, each record is signed by the entity accessing the content.

In earlier, own records are hashed together to create a chain formation, can easily identify possible errors or losts files. The encrypted log records may be decrypted afterward and their integrity checked. They will be accessed by the data owner and other authorized stakeholders at any time

for auditing purposes with the aid of the log harmonizer. Researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particular in the context of electronic commerce. The authors propose the usage of policies attached to the data and present logic for accountability data in distributed settings. Logic for accountability data in distributed settings, similarly. This IBE scheme could be a Weil-pairing-based IBE scheme that protects us against one among the most current attacks to our design as described in exploitation the generated key, the user can produce a logger part that may be a JAR file, to store its data items.

Our proposed framework prevents various attacks such as detecting illegal copies of users' information. Hence our work is distinct from normal logging methods which use encryption to secure log records. Their logging techniques are neither automatic nor shared. They request the information to lie within the boundaries of the centralized system for the logging to be able, which is not appropriate in the cloud State transition diagram is machine that shows no of states, machine take input from outside world and every input will turn out machine to travel next step. Following transition diagram shows the various states of Accountability mechanism in cloud i.e. however it changes from one state to next state.

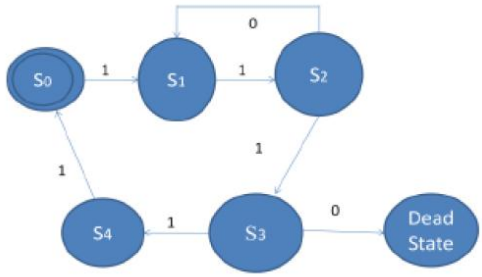

Fig.2. State Transition Diagram

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 principles.
S2: Authentication of CSP to JAR file.
S3: Authentication of user. S4: owner can see merge log

C. Fog computing methodology

In this paper which proposes a different approaches for securing data in the cloud using offensive decoy method. We supervise information access in the cloud and detect abnormal data access patterns. When unsecured access is identified and after checked by raising queries, we utilize a mislead attack by forwarding large amounts of decoy information to the attacker. This prevents the utilization of the user's own information. Hypothesis supervises in a local file context provides evidence that this approach may provide unprecedented levels of user data security in a Cloud environment. We use this technology to launch disinformation attacks against harmful groups, protecting from noticeably the real sensitive customer data from fake worthless data. In this paper, which propose two ways of using Fog computing to prevent attacks such as the Twitter attack, by retrieving decoy data inside the Cloud by the Cloud service customer and within personal online social networking profiles by individual user? The basic idea is that we can limit the damage of stolen data if we decrease the value of that stolen information to the attacker. We can achieve this through a 'preventive' mislead attack. We assume that protected Cloud events may be implemented by given two additional security features:

1) User Behavior Profiling:

It is expected that access to a user's information in the Cloud will exhibit a normal access. User profiling is a popular method that can be applied here to design how, when, and how much a client utilizes their data in the Cloud. Such 'normal user' behavior can be continuously checked to determine whether abnormal access to a user's information is happening or not. This procedure of behavior-basis protection is commonly used in fraud detection applications. Such prominence usually consists of metered information, how many documents are commonly read. These user- distinguish features may serve to detect abnormal Cloud access based partially upon the scale and scope of data transferred.

2) Decoys:

Decoy information, such as decoy documents, honey files, honey pots, and various other bogus information can be generated on demand and serve as a means of detecting unauthorized access to information and to 'poison' the thief's ex-filtrated information. Serving decoys will confound and confuse an attacker into believing they have ex-filtrated useful information or not. This method may be integrated with user behavior profiling technology to secure a user's information in the Cloud. Whenever exceptional access to a cloud events are observed, decoy data can be get back by the Cloud and delivered in such a way as to appear completely lawful. The real user, who is the owner of the data, would readily identify when decoy information is being returned by the Cloud, and might alter the

Cloud's responses through a variety of means, such as challenge questions, to inform the Cloud security system that it has inaccurately detected an unsecured access. In the situation where the access is accurately identified as an unauthorized access, the Cloud security system would deliver unbounded amounts of bogus information to the attacker, thus protecting the user's real information from unsecured disclosure.

The decoys contribute two features: (1) validating whether data access is authorized when abnormal information access is detected, and (2) confusing the attacker with bogus information. These posit that the combination of these two security features will provide unprecedented levels of security for the Cloud. No current Cloud security mechanism is available that provides this level of security.

III. MODULES

The major buildings modules of proposed systems are five they are.

3.1. Data Owner Module
3.2. Jar Creation Module
3.3. Cloud Service Provider Module
3.4. Disassembling Attack
3.5. Man-in-the-Middle Attack

3.1. Data Owner Module:

In this module, the data owner uploads their data in the cloud server. The new users can register with the service provider and create a new account and so they can securely upload the files and store it. For the security purpose the data owner encrypts the data file and then store in the cloud. The Data owner can have capable of manipulating the encrypted data file. And the data owner can set the access privilege to the encrypted data file. To allay users' concerns, it is essential to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example, users need to be able to ensure that their data are handled according to the service level agreements made at the time they sign on for services in the cloud.

3.2. Jar Creation Module

In this module we create the jar file for every file upload. The user should have the same jar file to download the file. This way the data is going to be secured. 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. 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. 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. 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.

3.3. Cloud Service Provider Module

The cloud service provider manages a cloud to provide data storage service. Data owners encrypt their data files and store them in the cloud with the jar file created for each file for sharing with data consumers. To access the shared data files, data consumers download encrypted data files of their interest from the cloud and then decrypt them.

3.4. Disassembling Attack

In this module we show how our system is secured by evaluating to possible attacks 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, the attacker is in possession of the public IBE key used for encrypting 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. However, the adoption of the Weil Pairing algorithm ensures that the CIA framework has both chosen cipher text security and chosen plaintext security in the random oracle model.

Therefore, the attacker will not be able to decrypt any data or log files in the disassembled JAR file. Even if the attacker is an authorized user, he can only access the actual content file but he is not able to decrypt any other data including the log files which are viewable only to the data owner.1 From the disassembled JAR files, the attackers are not able to directly view the access control policies either, since the original source code is not included in the JAR files. 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. Would the attacker erase or tamper a record, the integrity checks added to each record of the log will not match at the time of verification, revealing the error. Similarly, attackers will not be able to write fake records to log files without going undetected, since they will need to sign with a valid key and the chain of hashes will not match.

3.5. Man-in-the-Middle Attack.

In this module, 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 which will become obsolete at the time point of reuse. The second type of attack will also fail since renegotiation is banned in the latest version of open SSL and cryptographic checks have been added.

IV. PERFORMANCE STUDY

In this section, we first introduce the settings of the test environment and then present the performance study of our system.

4.1 Experimental Settings

We tested our CIA framework by setting up a small cloud, using the Emulab test bed. In particular, the test environment consists of several open SSL-enabled servers. One head node which is the certificate authority and several computing nodes. Each of the servers is installed with Eucalyptus. Eucalyptus is an open source cloud implementation for Linux-based systems. It is loosely based on Amazon EC2, therefore bringing the powerful functionalities of Amazon EC2 into the open source domain. We used Linux-based servers running Fedora 10 OS. Each server has a 64-bit Intel Quad Core Xeon E5530 processor, 4 GB RAM, and a 500 GB Hard Drive. Each of the servers is equipped to run the open JDK runtime environment with IcedTea6 1.8.2.

4.2.2 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, 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.

Considering one access at the time, we find that the authentication time averages around 920 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 performance can be further improved by caching the certificates.

V. CONCLUSION

We proposed to improve different types of approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. We would like to support a variety of security mechanism, indexing policies for text files, usage control for executables, time to time logging and closing of section.

REFERENCES

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

[2] P. Ammann and S. Jajodia, "Distributed Timestamp Generation in Planar Lattice Networks," ACM Trans. Computer Systems, vol. 11, pp. 205-225, Aug. 1993.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. ACM Conf. Computer and Comm. Security, pp. 598-609, 2007.

[4] E. Barka and A. Lakas, "Integrating Usage Control with SIP-Based Communications," J. Computer Systems, Networks, and Comm., vol. 2008, pp. 1-8, 2008.

[5] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Int'l Cryptology Conf. Advances in Cryptology, pp. 213-229, 2001.

[6] R. Bose and J. Frew, "Lineage Retrieval for Scientific Data Processing: A Survey," ACM Computing Surveys, vol. 37, pp. 1-28, Mar. 2005.

[7] P. Buneman, A. Chapman, and J. Cheney, "Provenance Management in Curated Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), pp. 539-550, 2006.

[8] B. Chun and A.C. Bavier, "Decentralized Trust Management and Accountability in http://www.oasis-open.org/ committees/tc home.php? Wg abbrev=security, 2012.

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

Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS), 2004.

[9] OASIS Security Services Technical Committee, "Security Assertion Markup Language (small) 2.0,"

[11] B. Crispo and G. Ruffo, "Reasoning about Accountability within Delegation," Proc. Third Int'l Conf. Information and Comm. Security (ICICS), pp. 251-260, 2001.

[12] Y. Chen et al., "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive," Proc. Int'l Workshop Information Hiding, F. Petit colas, ed., pp. 400-414, 2003.