Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 6, June 2013, pg.314 – 321

RESEARCH ARTICLE

Distributed Auditing for user data in the Cloud

Ms. Indushree T G[1], Mr. Anand R[2]
[1]IV SEM, M.Tech CSE, Department of CSE, CMRIT, Bangalore India
[2]Associate Prof, Department of ISE, CMRIT, Bangalore, India

[1] *indu.sai2316@gmail.com;* [2] *anandreddyr@gmail.com*

Abstract— Cloud computing is one of today's most exciting technology because of its cost-reducing, flexibility, and scalability. For infrastructure of cloud computing data security refers to policies, technologies, and controls deployed to protect data, applications. In this paper we discuss the evolvement of cloud computing paradigm and present a framework for secure cloud computing through accountability. Accountability is likely to become a core concept in the cloud. To address this problem, we propose a decentralized information accountability framework. We propose a logging mechanism along with user's data and policies. Whenever a user logs into the system, then a log record will be generated and those log records will be stored in a JAR (Java Archives) file. We propose a single JAR which has two functionalities like matching policies and providing/decrypting the file if that policy is matched. One more approach we are providing i.e. ABE technique which creates attributes and ensures security for user's related information. We provide extensive experimental studies that demonstrate the efficiency and effectiveness and the proposed approaches.

Key Terms: - Cloud computing; Accountability; ABE

I. INTRODUCTION

Cloud computing enables highly scalable and technology services that can be easily consumed by internet. A significant barrier to the adaptation of cloud services is thus the user fear of losing his own data and loss of privacy in the cloud. There are number of cloud computing services including Amazon, Google, Microsoft, Yahoo and Sales-force. The services which are abstracted by the user are no need to be the experts of technology infrastructure. This new technology is beneficial but users will start worrying about losing control of their own data. Governance and accountability questions are the main privacy issue of cloud computing.

Accountability plays a critical role in the development of trust during human interaction. Thus, accountability is viewed both as a tool to achieve practical security and as a first class design goal of services in federated distributed systems. There are two types of accountability Agent accountability and Data accountability. Agent accountability focuses on whether the actions of a given agent where authorized. For single data usage policy the data accountability describes the authorization requirements. In our paper we mainly concentrate on data accountability. Accountability will be achieved via a combination of private and public accountability. Public accountability is a active interaction between subjects of personal identifiable information, regulatory bodies, such as data controllers. Private accountability, in contrast is derived from the interaction between data controllers and data processors. For two reasons the approaches using a centralized server is not suitable. First, outsourcing of data by cloud service provider to storing data. Outsourcing of data into the cloud reduces the cost and complexity of long term large scale data storage, but it does not offer guarantee on data integrity and availability [2]. So that here proposed a trusted computing environment that provides secure cross platform. Authentication, encryption and decryption, compression are the security issues. Second, there will be a flexible manner of entities.

To overcome these problems, a new approach has been proposed namely Cloud Information Accountability (CIA). Here it uses Java Archives (JAR) files for automatically log the usage of user's data, it can provide distributed auditing mechanism and also can provide authentication of JAR for the strengthening of user's control, it develops powerful application even after they modify the code and the code of the copied code by the attacker. CIA framework combines the feature of access control, usage control and authentication. In CIA framework they have used a Java Archives (JAR) files which will automatically log t usage of the user data. JAR file includes user's data and their policies. By using this framework it helps to increase the trustiness of cloud. Cloud storage must be handling with full care of security. Cloud security can include access control, usage control and authentication. Considering the above related works cloud security can be provided by using many methods. Among that the most efficient mechanism is providing accountability for cloud data. By using CIA framework it trace the control of data by the data owner. Here the user, who subscribed to a certain cloud service, usually needs to send his data as well associated access control policies to the service provider after the data are received by the cloud service provider, the granted access is given to service provider, such as read, write and copy on the data. Logging and auditing technique is developed to track the data. There are two modes for auditing: push mode and pull mode in the accountability feature. The push mode refers to logs that are periodically sent to data owner. In contrast pull mode refers to retrieving the logs on as a needed basis. We provide the JARs with a central point of contact which forms a link between them and the user in a decentralized logging mechanism. This mechanism also records the error correction information which has been sent any of the JARs. Suppose, if any of the JAR is not able to contact its central point, then access to its enclosed data will be denied.

Images represent a very similar content type for end users and organizations for this reason forthwith they are focusing on image files. These images have been hosted in cloud as part of storage service. In augmentation, our approach also handles personal identifiable information and they have been stored as an image files (they contain an image of any textual content, for example, the SSN stored as a .jpg file)

In our paper, the contributions are as follows

- This is the first time we are proposing a usage of JAR files for the accountability and includes a logging mechanism
- In our architecture we are using decentralized server because it does not require any authentication or storage system.
- Our architecture is platform independent
- In addition to access control, we also provide usage control.

This paper is an extension of our previous conference paper. We have made the following new contributions. In order to strengthen the dependability of our system in case of comprised JRE we integrate oblivious hashing and integrity check. We also updated the log records structure to provide additional guarantees of integrity and authenticity. Second, to recover more possible attack scenario we extended the security analysis. Third, the results of new experiments are reported and provided a thorough evaluation of the system performance.

The rest of the paper is organized by sections: related work will be discussed in section 2. Proposed system is discussed in section 3.Our proposed Implementation is discussed in section 4, Experimental results in section 5, Section 6 describes conclusion and future work.

II. BACKGROUND AND RELATED WORK

Initially we will review related work addressing issues like data sharing in the cloud. User's data are usually processed in remote machines this is the main feature of cloud service. These unknown machines is not owned by the users and they cannot operate also the users will be afraid of losing control of their data in particular health and financial related data[1]. For the wide adoption of cloud services this can become a significant barrier. To overcome this problem they have proposed accountability framework that is going to keep track of data. They have also proposed JAR programmable framework that will trigger authentication whenever there is an access to the data.

There are two main problems that will arise in cloud environment. First is the data which has to be handled will be outsourced [4] and second is entities will enter and exit the cloud in a flexible manner. As a result handling data becomes complex. To address the above problems, they have proposed Cloud Information Accountability framework. This will keep usage of data transparent. There are two main modes for auditing that is push mode and pull mode. The push mode sends the log to the data owner. Whereas pull mode retrieves that log. The data which has to be send by user also contain policies.

They have focused on image files and for the storage purpose they have hosted in the cloud. Personal identifiable information is stored as image files and our approach is going to handle. This is the first time a systematic approach of JAR files has been proposed. The proposed architecture is platform independent. The cloud information accountability has two major components logger and log harmonizer. The logger is the

component and it is strongly coupled with user data. When the data is accessed it is downloaded and when data is copied it will be copied. The log harmonizer allows the user access to the log files and is responsible to form the central component. Proposed JAR consists of outer and inner JAR and outer JAR will handle authentication of entities which is willing to access the data stored and inner JAR Inner JAR contains the encrypted data.

Our proposed module contains many modules under that log record generation in which records will be generated with the help of logger. This record has four key terms view, download, timed access and location access. Second module is Dependability of JARs, it will prevent two types of attacks first is mechanism of storing the JARs remotely and corrupting it and second attack is running the JAR files by compromising with JRE. Third module is log correctness that is useful in verifying the integrity. The verification is carried out in two main methods. First is by repairing JRE and second is by inserting hash codes that is helpful in calculating hash value [6].

Our related work also concerned with privacy and security issues.

Pearson developed privacy manager [2] and he has also proposed the mechanisms for auditing. The layered architecture has been presented for end to end trust management addressing .In the electronic commerce they investigated accountability as a provable property by the researchers. Usage of policies is proposed by the authors .Accountability based distributed system is designed by the Jagadeesan .The only work proposing a distributed approach to accountability is from Lee and colleagues. Self-defending objects are related to our work. The java based approach has been proposed to prevent privacy leakage from indexing. Proposed an access control and for strongly coupling content using Identity Based Encryption (IBE). For secure data provenance our work looks similar but greatly differs in terms of goals, techniques and application domains. Investigation of usage control is carried out for the extension of current access control mechanisms. They mainly focused on conceptual analysis. Constraints at various level of granularity. These are the issues of privacy and security.

III. PROPOSED SYSTEM

The main aim of the proposed system is to design an accountability framework for modeling a highly scalable cloud environment. This will enables policies in a JAR file. In this project work , the focus is on presenting the proposed technique , a scalable , reliable service model for policies in cloud that uses a logging mechanism and also providing security for the same. It has to show that the proposed system saves the time overhead, in which it won't lose the control of data in the cloud.

The aim to develop logging and auditing technique which satisfy the following requirements.

1. Implement CIA framework allowing user to generate keys.
2. Implement CIA framework allowing user to mention the access control rules.
3. Implement CIA framework to package the data, access rules and a key to a JAR component.
4. Every time user access data, log records must be generated, encrypted and stored in log area.
5. Implement mechanism to keep the log record safe, to avoid missing records and data corruption.
6. Implement a portal for the user to view the access statistics from the log files.

The method supposes the log record to avoid missing records, thereby it provides security for the successful usage of policy in the cloud.

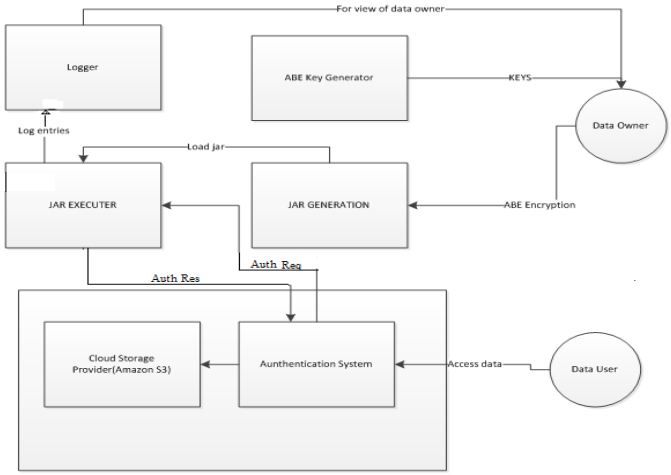

Figure 1: overview of accountability framework

The proposed system illustrates that a data owner who is the owner of all related data, accepts a key from a ABE key generator and a logger is responsible for the view of data owner. Logger contains the information of the person who has requested to access the data. The log entities should be initially sent to the logger component.

The key which is passed to the data owner will be encrypted using an ABE encryption technique. The encrypted key will be sent to the JAR which has been generated. The JAR file is loaded to the JAR executer.

The authentication system is provided in which JAR executer will enable authentication request and authentication response to that system. The user of the data will give access to the data if request is valid. The data which is sent to the authentication system will be finally sent to cloud storage provider (Azure).

IV. MODULES DESCRIPTION

The proposed system consists of following modules.

i) Log Record Generation

Log records are generated by the logger component. Logging occurs at any access to the data in the JAR, and new log entries are appended sequentially, in order of creation LR=<r_1… r_k>. Each record r_1 is encrypted individually and appended to the log file. In particular, a log record takes the following form:

r_1=<ID Act ,T ,Loc ,h((ID ,Act ,T ,Loc)|r_i-1|…|r_1), sig):

Here, r_1 indicates that an entity identified by ID has performed an action Act on the user's data at time T at location Loc. The component h((ID, Act ,T ,Loc{r_i-1|…|r_1) corresponds to the checksum of the records preceding the newly inserted one, concatenated with the main content of the record itself (we use I to denote concatenation). The checksum is computed using a collision-free hash function. The component sig denotes the signature of the record created by the server. If more than one file is handled by the same logger, an additional Obj ID field is added to each record .An example of log record for a single file is shown below.

Example: Suppose that a cloud service provider with ID John, located in Pune , read the image in a JAR file (but did not download it) at 4:52 pm on May 20, 2011. The corresponding log record is

<John, View, 2011-05-29 16:52:30, Pune, 45rftT024g, r94gm30130ff>

ii) JAR File Generation

The JAR file is generated in which both creates a dynamic and travelling object and to ensure that any access to the user data will trigger authentication and automated logging local to the JAR. These JARs will automatically log the usage of the user data by any entity in the cloud. The access policies will be sent by the users enclosed in JAR file to cloud service provider. The user will create a logger component which is a JAR file, to store its data items using a generated key.

The JAR file includes a set of simple access control rules specifying whether and how the cloud server and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, the JAR file will be sent to the cloud service provider that he subscribes to authenticate CSP to JAR, we use open SSL based certificate.

iii) Hash Function Creation

A hash function is any algorithm or subroutine that maps large data sets of variable length to smaller data sets of a fixed length. The values returned by a hash function are called hash values, hash codes, hash sums, checksums or simply hashes.

Hash value plays an important role in correcting the logs. Initially, the hash codes are inserted, which calculate the hash values of the program traces of the modules being executed by the logger component. This helps us detect modifications of the JRE, and are useful to verify if the original code. The integrity checks are carried out using oblivious hashing [6]. OH works by adding additional hash codes into the programs being executed. The hash function is initialized at the beginning of the program, the hash value of the result variable is cleared and the hash value is updated every time there is a variable assignment, branching, or looping.

The hash code captures the computation results of each instruction and computes the oblivious-hash value as the computation proceeds. These hash codes are added to the logger components when they are created. They are present in their JAR. The log harmonizer stores the values for the hash computations. The main functionality of log harmonizer is it forms the central component which allows the user access to log files. The values computed during execution are sent to it by the logger components. To verify the JRE has been tampered, the log harmonizer proceeds to match these values against each other. The execution values will not match, if the JRE is tampered. Additional layers of security are added when OH is added to the logger components. Any tampering of the logger components will also result in the OH values being corrupted.

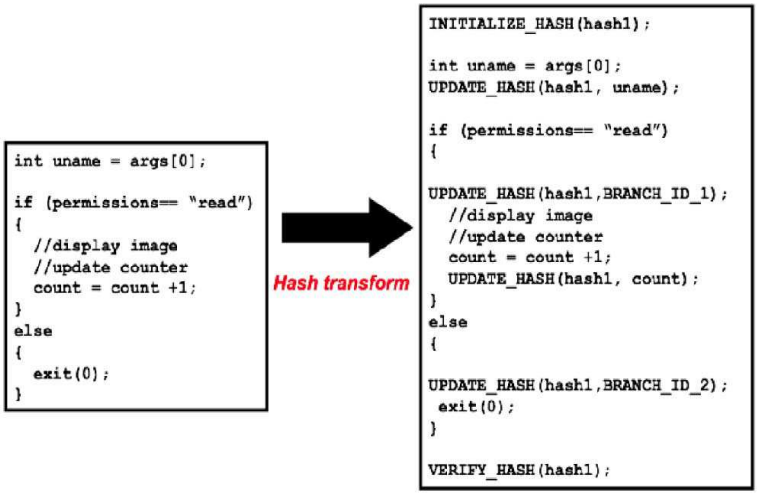

Figure 2: Oblivious Hash Algorithm

iv) ABE Algorithm

It is a type of public key Encryption in which the public key of a user and the cipher-text are dependent about attributes (e.g. the country he lives, the kind of subscription he has,). In a system the decryption of a cipher-text is possible only if the set of attributes of the user key matches the attributes of the cipher-text. Attribute based encryption can be used for log encryption. Instead of encrypting each part of a log with all the key of all the recipients, it is possible to encrypt the log only with attributes which match recipient's attributes. This primitive can also be used for broadcast encryption in order to decrease the number of key used. Each private key is associated with an access structure that specifies which type of cipher-texts the key can decrypt. We call such a scheme a Key-Policy Attribute-Based Encryption (KP-ABE).

An (Key-Policy) Attribute Based Encryption scheme consists of four algorithms.

Setup Attributes: This algorithm is used to set attributes for users. This is a randomized algorithm that takes no input other than the implicit security parameter. It defines a bilinear groupG1 of prime order p with a generator g, a bilinear map $e : G1 \times G1 \rightarrow G2$ which has the properties of bi-linearity, computability, and non-degeneracy. From these attributes public key and master key for each user can be determined. The attributes, public key and master key are denoted as

Attributes-U ={ 1,2. . . N}

Public key-PK = (Y, T_1, T_2, . . . , T_N)

Master key-MK = (y, t_1, t_2, . . . , t_N)

Where T I ∈ G_1 and ti∈Zp are for attribute i, $1 \leq i \leq N$, and Y ∈G_2 is another public key component. We have $T_{i=gti \text{ and}}$ Y =e(g, g)y, y∈Zp. While PK is publicly known to all the parties in the system, MK is kept as a secret by the authority party.

Encryption: This is a randomized algorithm that takes a message M , the public key PK , and a set of attributes I as input. It outputs the cipher text E with the following format:

E = (I, Ẽ, {E $_i$ ∈ I)

Where Ẽ=MY s,E $_i$=T *is.* And s is randomly chosen from Z.

Secret key generation: This is a randomized algorithm that takes as input an access tree T, the master key MK, and the public key K. It outputs a user secret key SK as follows. First, it defines a random polynomial $p_i(x)$ for each node I of T in the top-down manner starting from the root node r. For each non-root node j, $p_j(0)$ =p parent (j)(idx(j)) where parent (j) represents j's parent and idx (j) is j's unique index given by its parent. For the root node r ,pr (0) =y. Then it outputs SK as follows.

SK ={sk i }i ∈ L

Where L denotes the set of attributes attached to the leaf nodes of T and sk i=g $p_i(0)/t_i$.

Decryption: This algorithm takes as input the cipher text **E** encrypted under the attribute set **I**,the user's secret key **SK** for access tree **T** , and the public key **PK** . It first computes e(E i,sk i)=e(g, g)pi(0)s for leaf nodes. Then, it aggregates these pairing results in the bottom-up manner using the polynomial

interpolation technique. Finally, it may recover the blind factor Y s =e(g,g) ys and output the message **M** if and only if **I** satisfies **T**

Role of ABE in our Proposed System

In our proposed system, we are making use of Key policy based ABE. In earlier work, they have implemented IBE technique which was used to set the time. The main disadvantage of IBE in our system is the attributes has not been defined and the images has sent to all registered users. There was no particular group to set access policies. By making use of ABE technique we are going to define attribute and the users who belong to that particular attribute can access the images, others who are not under that particular category cannot access images. By using ABE technique in our system we are providing security and performance will get increased when compare to earlier work.

V. IMPLEMENTATIONS

The proposed system is implemented in windows XP with Pentium IV processor. The design environment is selected in cloud analyst. By making use of single JAR management will become easier and time consumption will become less drastically. So, in our proposed system instead of outer n inner JARs we are making use of single JAR. We are defining attributes in our proposed system. The data users will get registered to that particular attribute in which they belong to. Suppose a data owner wants to send an image. That image can be viewed by particular user who belongs to that designated group. Other group of people cannot access that image. Thereby we are achieving confidentiality, reliability, security. Moreover performance will get increased.

A download chart is designed in order to make out which users have accessed data owner files, how many times accessed, download count, date of downloading. All information will be in that chart. It keeps the record of data owner.

VI. EXPERIMENTAL RESULTS

In the experiments, the log file will be created and the time taken will be examined and also its overhead is measured in the system. The overhead will occur at three points with respect to the time First is during the authentication, second is during encryption of log file and finally during merging those log files. Our architecture is light weight with respect to the storage.

The experiment is conducted on cloud analyst tool. It mainly considers three timing characteristics namely response time, processing time and transmission time. First we need to define user bases, number of data centers and at last we need to access one file. Once the file is uploaded, it's going to create a JAR file. Our experimental results show the time taken to create a log file, time taken to create a JAR file and about encrypting that JAR file.

i) Response Time:

We are going to plot a graph. Along x axis we are considering hours and along y- axis we are considering mille seconds. It's going to take three parameters that is maximum response time in ms, minimum response time in ms and average response time in ms. In first step it's going to give total summary of response time. In second step it's going to give response time with respect to user base and region.

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	300.00	225.11	366.06
Data Center processing time:	0.34	0.02	0.66

Figure 3: Overall Response Time

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB1	300.45	225.11	361.63
UB2	298.94	231.11	364.68
UB3	299.32	232.62	366.06
UB4	301.19	234.12	364.63
UB5	300.01	225.12	363.13
UB6	300.09	232.61	363.11

Figure 4: Response Time by Region

ii) Processing and Transmission Time

According to individual user base it's going to calculate response time, processing time, transmission time and it also gives the cost of virtual machine, total data transmission of each data center and finally it will give the total cost of data centre.

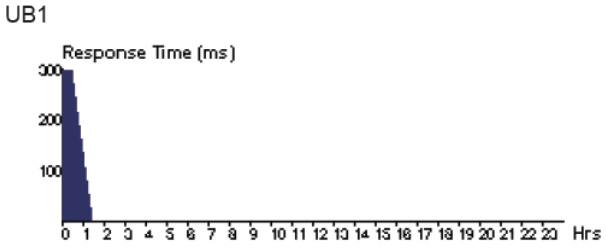

Figure 5: User Base Hourly Response Time

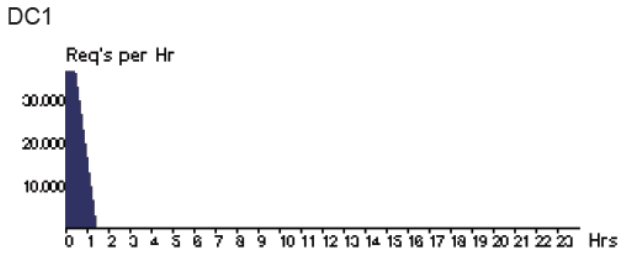

Figure 6: Data Center Hourly Processing Time

Data Center	VM Cost $	Data Transfer Cost $	Total $
DC1	0.50	0.38	0.89

Figure 7: Total Cost of Individual Data Center

VII. CONCLUSION AND FUTURE WORK

We have proposed an ABE technique and generation of single JAR. By proposing ABE technique the data is secured, cannot be outsourced. we have also achieved scalability, reliability and confidentiality to the users data. By the generation of single JAR time is consumed and management of cloud environment will be easier. Their won't be any leakage of data. By providing time and date to our file, we can control huge number of data being accessing. In future work we would like to add some security algorithms to check the correctness of the images.

As a future work we would like to account the user photographs in social networking websites (Face book, Linked-in). Our method log record generation can be used to see the users who have accessed our photographs which we have uploaded.

REFERENCES

[1] S. Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing, 2009.

[2] S. Pearson, Y. Shen, and M. Mowbray, "A Privacy Manager for Cloud Computing," Proc. Int'l Conf. Cloud Computing (CloudCom),pp. 90-106, 2009.

[3] X. Feng, Z. Ni, Z. Shao, and Y. Guo, "An Open Framework for Foundational Proof-Carrying Code," Proc. ACM SIGPLAN Int'l Workshop Types in Languages Design and Implementation, pp. 67-78, 2007.

[4] Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data Vipul Goyal¤ Omkant Pandeyy Amit Sahaiz Brent Waters

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

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

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

[8] S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang,"Promoting Distributed Accountability in the Cloud," Proc. IEEE Int'l Conf. Cloud Computing, 2011.
[9] Ensuring Distributed Accountability for Data Sharing in the Cloud Smitha Sundareswaran, Anna C. Squicciarini, Member, IEEE, and Dan Lin

