STRENGTHENING USER'S CONTROL OF DATA IN THE CLOUD SERVICE

K. Selvakumar and V. Kavitha

***Abstract*— In this paper, to strengthen user's control of data in the cloud service, a novel approach, namely cloud information accountability (CIA) framework is proposed which secure data during transmission and storage. Experimental results are reported to demonstrate the efficiency and effectiveness of the proposed approach.**

***Index Terms*— accountability, cloud computing, data sharing**

I. Introduction

Cloud computing enables highly scalable service to be easily consumed over the internet on an as needed basis. A major feature of the cloud service is that the users data are usually processed secretly in unknown machines that users do not own or operate. On using cloud services users fears of loosing control of their data (in particular, financial and health data) and it become a significant barrier to wide adoption of cloud services[1-9]. To overcome the above problems, we propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability [9]. Unlike privacy protection technologies which are 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. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed. 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 where by 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, 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. 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. Currently, we focus on image files since images represent a very common content type for end users and organizations (as is proven by the popularity of Flickr [4]) and are increasingly hosted in the cloud as part of the storage services offered by the utility computing paradigm featured by cloud computing. Further, images often reveal social and personal habits of users, or are used for archiving important files from organizations. In addition, our approach can handle personal identifiable information provided they are stored as image files (they contain an image of any textual content, for example, the SSN stored as a .jpg file). We tested our CIA framework in a cloud test bed, the Emu lab test bed [3], with Eucalyptus as middle-ware. Our experiments demonstrate the efficiency, scalability and granularity of our approach. In addition, we also provide a detailed security analysis and discuss the reliability and strength of our architecture in the face of various nontrivial attacks, launched by malicious users or due to compromised Java Running Environment (JRE).

In [1], a fully functional identity based encryption service(IBE) Is proposed. The scheme has shown cipher text security in the random oracle model assuming and elliptic curve variant of the computational Diffe-Hellmen problem In[2], a game based Machine checked retention of the security of the Branch- Franklin (IBE) scheme to the Bilinear–Diffie-Hellman assumption is presented.

In [8], an attempt is made to strengthen users control of data in the cloud service using CIA. In this project, based on the works in [8], another attempt is made to strengthen users control of data in the cloud service using CIA. To strengthen user's control of data in the cloud service in this paper, a novel approach, namely cloud information accountability (CIA) framework is proposed. That is, to track the actual usage of data of a data owner, logging and auditing techniques are proposed. The proposed CIA framework provides end-to-end accountability in a highly distributed fashion with captcha, one time password during uploading

Manuscript received October 13, 2013.

K. Selvakumar , Department of Computer Science and Engineering, University College of Engineering Anna University (Nagercoil Campus), Nagercoil, Tamilnadu, India, 9566604812.

V. Kavitha, Department of Computer Science and Engineering, University College of Engineering Anna University (Nagercoil Campus), Nagercoil, Tamilnadu, India, 9487116703.

files, stegenagraphy to view user's name storing the log records in a secured java archive file (JAR) and delivering log records to the e-mail or g-mail of the data owner by implementing Java Running Environment (JRE).

In section II, our proposed CIA framework is presented. Auditing modes is given in section III. Experimental results are presented in section IV.

The objective of this paper is to provide a secure CIA framework for data sharing in cloud computing environment.

II. CLOUD INFORMATION ACCOUNTABILITY

Our proposed CIA framework is given in the following figure 1. This work is different from traditional logging methods which use encryption to protect log files. This CIA framework prevents various attacks such as detecting illegal copies of user's data. CIA framework given in Smitha et.al[8] is a particular case of our CIA framework.

CIA framework conducts computerized logging and auditing of relevant access performed by any entity carried at any point of time at any cloud service provider. It has two major components, namely, logger and log harmonizer.

A. *Logger*

The logger is the component which is strongly coupled with the user's data , so that it is distributed when the data are accessed and is copied whenever the data are copied. It handles a particular instance or copy of user's data and is responsible for logging access to that instance or copy. Logger is responsible for generating error correction information for each log record and send to the log harmonizer.

B. *Log Harmonizer*

The log harmonizer forms the central component which allows the user's access to the log files. The log harmonizer is responsible for auditing. It generates master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternately, the decryption can be carried out on client end if the path between the log harmonizer and the client is not trusted. In this case the harmonizer sends the key to the client in a secure key exchange. The logger and the log harmonizer are both implemented as light weight portable JAR files. Automatic logging functions are provided by implementing JAR files.

III. AUDITING MODES

CIA supports two auditing modes, namely, push and pull modes.

A. *Log Push Mode*

The log file is pushed back to the data owner periodically in an customized fashion by the harmonizer. If there are multiple loggers for the same set of data items then the log harmonizer will merge the log records before sending back to the data owner. The push action will be triggered by time elapses by a certain period according to the temporal timer inserted as a part of the JAR file. The JAR file exceeds the size stipulated by the content, owner at the time of execution.

B. *Log Pull Mode*

This mode allows auditors to receive the logs any time when they want to check the recent access to their own data. The request will be send to the harmonizer and the user will be informed of the data's locations and obtain an integrated copy of the authentic and sealed file. The performance will be improved by inserting logging and auditing functions.

C. *Data Flow*

The data flow in the CIA frame work is sketched in figure 1 is as follows. Each user will create a logger component which is a JAR file to store its data items. The JAR file includes a set of simple access control rules to access the contents itself. Then he send the JAR to the CSP that he subscribes to .A trusted certificate authority certifies the CSP to authenticate the CSP to the JAR . Verifying the user's name the identity of the user, a trusted identity provider issues certificates. Once the authentication succeeds, the CSP will be allowed to access the data in the JAR files. In logging of CSP in the cloud each time their is an access to the data, the JAR will automatically generate a log record encrypt using the public key distributed by the data owner and it store it along with the date. Some error correction information will be send to the log harmonizer to handle possible log files corruption. The encrypted log files can be decrypted and they can be accessed by the data owner at any time with the help of harmonizer.

We use stegenography to view the name of the user who access the data of a data owner through the log records. The name of the data files(photographic images) will be hidden in the JAR file when the data owner sends his data to the cloud. This CIA frame work prevents various attacks such as, detecting illegal copies of user's data.

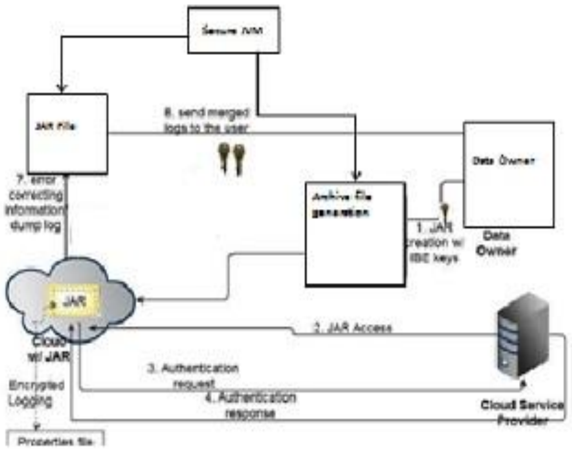
Secure JVM
JAR File
8. send merged logs to the user
Data Owner
7. error correcting information dump log
Archive file generation
1. JAR creation w/ IBE keys
Data Owner
JAR
Cloud w/ JAR
2. JAR Access
3. Authentication request
4. Authentication response
Encrypted Logging
Cloud Service Provider
Properties file

Fig 1: Data Flow(CIA Frame work)

IV. EXPERIMENTAL RESULTS

To demonstrate the efficiency and effectiveness of the proposed approaches, we examine the time taken to create a log file and measure the overhead in the system with respect to time and storage. The overhead with respect to time are during authentication, during encryption of a log record and during the merging of the log records. The overhead with respect to storage, only data to be stored are the actual file and the associated log files. In this paper, the size of the data files are of different sizes and four data owners are implemented for experimental study.

A. *Log Creation Time*

We find out the time taken to create a log file when there are entities continuously accessing the data causing continuous logging. The results are given in the figure 2.

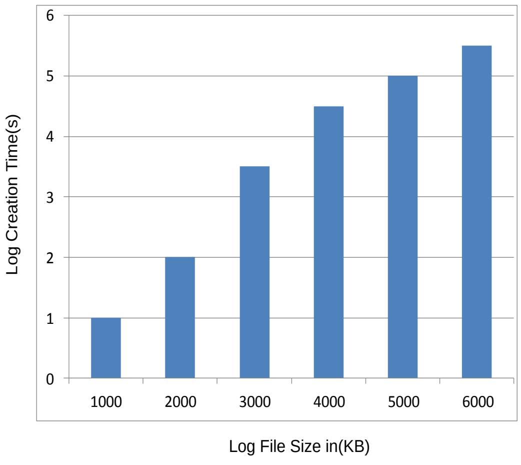

Fig 2: Time to create log files of different sizes

It is observed that, from figure 2, the time to create log files increases linearly with the size of the log files. In addition, it is observed that, the time taken to create log files decreases linearly with respect to the size of the newly uploaded files into the data of the four data owners after finding the log creation time for the initially stored data in the four data owners. In particular, the log time get increased when four owners are introduced, it is shown in the Table 1.

Table 1: Log creation time

Work	Data Owners	File Size	Log creation time
Smitha et.al[8]	**1**	**100KB**	**114.Ms**
Present work	**4**	**140KB**	**169.Ms**

B. *Authentication Time*

We consider both the authentication time of the CSP and an end user. During the authentication of a CSP the overhead can occur. It is given in the Table 2. It is observed that, the authentication time in the present work get reduced than in the works of Smitha et. al[8], on using four data owner.

Table 2: Authentication time

Work	Data Owners	Authentication Time CSP	USER
Smitha et.al[8]	**1**	**920ms**	**1.2seconds**
Present work	**4**	**920ms**	**1.0seconds**

C. *Time taken to perform logging*

In this experiment, we let multiple servers continuously access the same data JAR file for a minute and recorded the number of log records generated. The following Table 3, shows the time taken to perform logging of the present work is better than the works of Smitha et. al.[8].

Table 3: Time taken to perform logging

Work	Data Owners	Average time log an action	Log encryption time per second
Smitha et.al[8]	**1**	**10seconds**	**300MS**
Present work	**4**	**8 seconds**	**278MS**

D. *Log Merging Time*

In this experiment we measure the amount of time required to merge two log files. The two files will have common records with each other. We tested time to merge up to 100 log files of size each 10KB, 500KB, 800KB and 1MB.The results are shown in Figure 3.

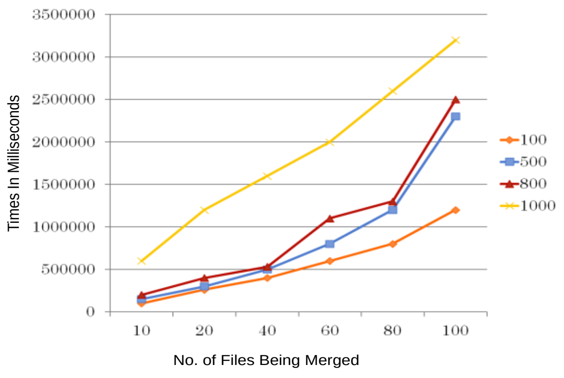

Fig 3: Time to merge log files

We observe that the log merging time increases linearly as the number of files increases with respect to the increase in size of files. Table 4 shows time get increased due to four data owners.

Table 4: Log merging time

Work	Data Owners	File Size	Log merging time
Smitha et.al[8]	**1**	**100KB**	**5.3MS**
Present work	**4**	**140KB**	**1.87MS**

E. *Size of the Data JAR Files*

We measure the overhead results in storage, when a single logger handles more than one file We measure the size of the logger(JAR) by varying the number of JARs and the size of the data inside the JAR. Figure 4, shows that the size of the logger(JAR) grows as the size of the original file increases in KB.

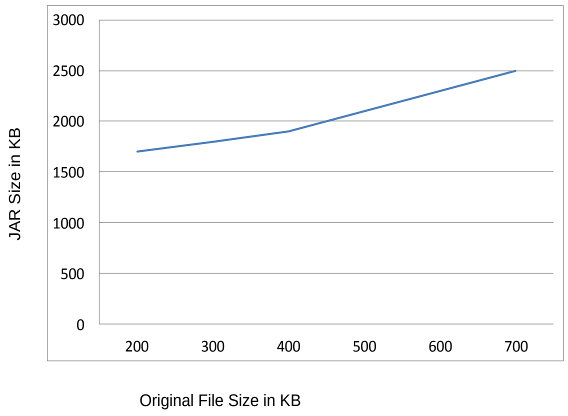

Fig 4: Size of the logger component

F. Over all time taken to complete the work

The total time taken to complete \the entire work is lesser than that of work of Smitha et.al[8], and it is shown in the Table 5.

Table 5: Time to complete the work

Work	**Data Owners**	**Overall time taken to complete the work**
Smitha et.al[8]	**1**	**6.5 minutes**
Present work	**4**	**6.0 minutes**

We use four data owners it is quite natural to take more time and. more storage space. In general, on comparing all the experimental results, our approach is more efficient and effective.

V. Conclusion

With an auditing mechanism, we proposed innovative approaches for logging any access to the data in the cloud. In our approach, the data owner can audit his contents and enforce back-end protection whenever needed. The main features of our approach is that it enables the data owner to audit even those copies of his data that were made without his knowledge. Moreover, we provided more security to the system which strengthen user's control of data in the cloud.

In future, we plan to refine our approach to verify the integrity of his JRE and authentication of JARs. We also plan to investigate practical encryption schemes that will allow encryption of log records in such a way that the logging cloud can execute necessary quires on the encrypted logs which is sufficient for privacy or confident ability.

Acknowledgment

I wish to acknowledge the financial support from Anna University, Chennai-600 025, India project for young faculty members under research support scheme.

References

[1] Boneh D and Franklin M.K, 'Identity-based encryption from the well pairing', Proc. Int'l Cryptography Conf. Advances in Cryptography, pp. 213-229, 2012
[2] Buneman P, Chapman A and Cheney J, 'Provenance management in crated databases., Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'06), PP. 539-550, 2006
[3] Emulab Netwok Emulation Testbed, www.emulabnet, 2012
[4] Fliker, Http//www.fliker.com/., 2012
[5] Jaeger P.T, Lin J and Grimes J.M, 'Cloud computing and information policy: computing in a policy cloud?'J. Information Technology and Policies, Vol. 5, No. 3, pp. 269-283, 2009
[6] Pearson S and Charleswurth A,(2009), 'Accountability as a way forward for privacy protection in the cloud', Proic. First Int'l Conf. Cloud Computing., 2009
[7] Sang Y, Cunming R and Gansen Z, 'Strengthen cloud computing security with management using hierarchical security-based cryptography', CloudCom, LNCS, 5931, PP.167-177, 2009
[8] Smitha S, Anna C.S and Dan L,'Ensuring distributed accountability for data in the cloud', IEEE Transactions on Dependable and Secure Computing, Vo. 9, No. 4, pp. 556-568, 2012
[9] Weitzncr D.J, Abelson H, Berners-Lee T, Feigen-baum J, Hemdeler J and Sussman G.J, 'Information Accuntability', Comm. ACM, Vol. 51,No. 6, pp.82-87, 2008