PERIODIC LOGGING AND AUDITING MECHANISM FOR ENHANCE THE DATA SHARING IN CLOUD

Shani Sivadas*

R.LakshmiPriya*

Abstract—

Data storage into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage. At the same time such a service is also eliminating data owners' ultimate control over the fate of their data, which data owners with high service-level requirements have traditionally anticipated. That is data owners have fear of losing the control over their own data. To concentrate on this dilemma we develop a solution which is an accountability infrastructure for user's data in the cloud. The decentralized infrastructure tracks the usage of data in the cloud. To reinforce the user's control we use automated logging mechanism and distributed auditing mechanism together with data and policies. We clout the programmable capabilities of JAR files to assure that any usage of user's data will spark the authentication process and automated logging process which is local to any JAR files. To improve authentication in certificate cloud JAR apply RSA encryption with Push and Pull Algorithm. It will be improve secure data sharing in cloud.

Keywords— Accountability, Authentication, Cloud computing, Data sharing.

* PSN College of Engineering and Technology.

I. INTRODUCTION

Cloud computing is a recent advancement wherein IT infrastructure and applications are provided as "services" to end-users under a usage-based payment model. Cloud computing delivers infrastructure, platform, and software that are made available as subscription-based services. However, it is not so much that the term 'Cloud Computing' represents a host of new technologies, but rather that these technologies are combined and effectively upgraded so that they enable new IT services and new business models. The flexibility of cloud computing is a function of the allocation of resources on demand. This facilitates the use of the system's cumulative resources, negating the need to assign specific hardware to a task. Before cloud computing, websites and server-based applications were executed on a specific system. With the advent of cloud computing, resources are used as an aggregated virtual computer. This amalgamated configuration provides an environment where applications execute independently without regard for any particular configuration.

Cloud computing, in which services are carried out on behalf of customers on hardware that the customers do not own or manage, is an increasingly fashionable business model. The input data for cloud services is uploaded by the user to the cloud, which means that they typically result in users' data being present in unencrypted form on a machine that the user does not own or control. This poses some inherent privacy challenges[12]. Despite auditability being a crucial component of improving trust, current prominent providers (e.g. Amazon EC2/ S3 , Microsoft Azure) are still not providing full transparency and capabilities for the tracking and auditing of the file access history and data provenance [6]. That is data owners have fear of losing the control over their own data.

To diminish this problem it is very important to develop an effective mechanism for customers to track the usage of their own data in the cloud environment. In traditional approaches using a server which is centralized in distributed environments. But this is not suitable and develop some security issues. The characterizing features of the cloud computing such as outsourcing of private data to other entities by cloud service provider will arise security problems. And also data handling in the cloud goes through a complex and hierarchical service chain arises problems.

To alleviate these above problems we propose a accountability infrastructure called Decentralized Cloud Information Accountability (DCIA)framework which tracks the usage of user's data in the cloud. This proposed DCIA framework provides a facility to a track and check whether or not the service-level agreements are being honoured. It also enforce access and usage

control rules as needed. There are many challenges arises in the design of DCIA framework. The challenges are ensuring the reliability of the log, adapting to a highly decentralized infrastructure etc. The basic approach to DCIA is to extend the capability of JAR (Java Archives) files to automatically log the usage of user's data by any entity in the cloud. Data users will send their data along with any acess control policies and logging policies that enclosed in JAR files. JAR file is the compressed file format.

We can store many files in a JAR file. This file format is used to distribute a set of java classes. This file helps you to reduce the file size and collect many file in one by compressing files. Another feature of a Jar file is that any access to thedata will trigger automatically and It is logged in to the jar file itself. 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.

II.RELATED WORK

This section reviews the related works addressing the privacy and security issues in the cloud. Then, briefly discuss works which adopt similar techniques as in this approach but serve for different purposes.

1.SECURITY ISSUES RELATED TO CLOUD

There are numerous security issues for cloud computing as it encompasses many technologies including networks, databases, operating systems, virtualization, resource scheduling, transaction management, load balancing, concurrency control and memory management. Therefore, security issues for many of these systems and technologies are applicable to cloud computing[23]. Pearson et al. have proposed accountability mechanisms to address privacy concerns of end users [8] and then develop a privacy manager [12]. Their basic idea is that The privacy manager uses a feature called Obfuscation.The idea is that instead of being present unencrypted data in the cloud the user's private data is sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The result of the processing is de-obfuscated by the privacy manager to reveal the correct result. This solution is not suitable for all cloud applications. R. Corin, S. Etalle1, J proposed a language that allows agents to distribute data with usage policies in a decentralized architecture. The design involves a logic that allows audited agents to prove their

actions, and to prove their authorization to posses particular data. The paper present two notions of accountability ,agent accountability focuses on whether the actions of a given agent where authorized. data accountability expresses that a given piece of data was not misused. This framework can be used for distributing personal data as well as valuable digital assets[24].Bruno and Giancarlo Ruffo proposes a framework for the analysis of delegation protocols.The framework allows to analyses how accountability is transferred, and also having ability to trace how accountability is distributed among principals of a system. This approach starts with the notion of provability to formalize accountability. That is ability of consumers in a protocol to prove a statement to a third party[1]. Delegation is complementary to our work, in that we do not aim at controlling the information workflow in the clouds. In [7], the authors present a layered architecture for addressing the end-to-end trust management and accountability problem in federated systems. The authors' focus is very different from ours, in that they mainly leverage trust relationships for accountability, along with authentication and anomaly detection. Further, their solution requires third-party services to complete the monitoring and focuses on lower level monitoring of system resources.

2.TECHNIQUES RELATED

Appel and Felten proposed [2] the Proof-Carrying authentication (PCA) framework. The PCA includes a high order logic language that allows quantification over predicates, and focuses on access control for web services. While related to ours to the extent that it helps maintaining safe, high-performance, mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring content. Another work is by Mont et al. who proposed an approach for strongly coupling content with access control, using Identity-Based Encryption (IBE) [5]. We also leverage IBE techniques, but in a very different way. We do not rely on IBE to bind the content with the rules. Instead, we use it to provide strong guarantees for the encrypted content and the log files, such as protection against chosen plaintext and ciphertext attacks.

III DECENTRALIZED CLOUD INFORMATION ACCOUTABILITY FRAMEWORK

Framework 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. There are two major components of the DCIA, the first being the logger, and the second being the log harmonizer. There are two major components of the DCIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the user access to the log files.

There are two major components of the DCIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the user access to the log files. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption [5] . This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture. Using the generated key, the 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 specifying whether and how the cloud servers, and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. To authenticate the CSP to the JAR .we use OpenSSLbased certificates, wherein a trusted certificate authoritycertifies the CSP. Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data The encryption of the log file prevents unauthorized changes to the file by attackers.

CLR VERIFICATION (HASH FUNCTION):

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 CLR uses the public key to decrypt the digital signature and get the hash value of the assembly that was present during compilation. 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.

DATA OWNER :

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. It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is obtained by the data owner as often as requested. As for the logging, each time there is an access to the data; the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs.

USERS

A user, who subscribed to a certain cloud service, usually needs to send his/her data as well as associated access control policies (if any) to the service provider. After the data are received by the cloud service provider, the service provider will have granted access rights, such as read, write, and copy, on the data. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the user access to the log files.

SECURITY DISCUSSION

The most intuitive attack is that the attacker copies entire JAR files. The attacker may assume that doing so allows accessing the data in the JAR file without being noticed by the data owner.

However, such attack will be detected by our RC4 Encrypt Decrypt Algorithm. Recall that every JAR file is required to send log records to the harmonizer. In particular, with the push mode, the harmonizer will send the logs to data owners periodically. That is, even if the data owner is not aware of the existence of the additional copies of its JAR files, he will still be able to receive log files from all existing copies. If attackers move copies of JARs to places where the harmonizer cannot connect, the copies of JARs will soon become inaccessible. This is because each JAR is required to write redundancy information to the harmonizer periodically. If the JAR cannot contact the harmonizer, the access to the content in the JAR will be disabled. Thus, the logger component provides more transparency than conventional log files encryption; it allows the data owner to detect when an attacker has created copies of a JAR, and it makes offline files unaccessible.

The logger is also responsible for generating the error correction information for each log record and send the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement. Our proposed framework prevents various attacks such as detecting illegal copies of users' data. Note that our work is different from traditional logging methods which use encryption to protect log files. Now we use RC4 Encryption algorithm which is responsible for decrypting the logs and to authenticate the CSP to the JAR where in a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, wherein a trusted identity provider issues certificates verifying the user's identity based on his username. This algorithm is used for both encryption and decryption as the data stream is simply XORed with the generated key sequence. The key stream is completely independent of the plaintext used. It uses a variable length key from 1 to 256 bit to initialize a 256-bit state table. The state table is used for subsequent generation of pseudo-random bits and then to generate a pseudo-random stream which is XORed with the plaintext to give the cipher text. It protects us against one of the most prevalent attacks to our architecture. Using the generated key, the 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 specifying whether and how the cloud servers and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. With only encryption, their logging mechanisms are neither

automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud.

RC4 Encryption algorithm provides the security against the encryption 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 separate JARs. In addition, Reed Solomon Error Correcting Code is give to provide error correction information will be sent to the log harmonizer to handle possible log file corruption .To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity verified. They can be accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer.

CONCLUSION

The proposed system including automated logging mechanism and distributed auditing mechanism improves the security protection in the cloud.And also it will give the strong back end protection. one of the main features of this work is that it enables the data owner to audit even those copies of its data that were made without his knowledge. The most intuitive attack such as copying Jar files JAR files will be detected by our RC4 Encrypt Decrypt Algorithm.

REFERENCES

[1]]. Mowbray, M., Pearson, S.: A client-based privacy manager for cloud computing. In: COMSWARE ’09. ACM(2009)

[2] 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.

[3] T. Mather, S. Kumaraswamy, and S. Latif, Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance (Theory in Practice), first ed. O’ Reilly, 2009.

[4] NTP: The Network Time Protocol, http://www.ntp.org/, 2012.

[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] P. Buneman, S. Khanna and W. Tan, "Data provenance: Some basic issues," FST TCS 2000: Foundations of Software Technology andTheoretical Computer Science, 2000, pp. 87-93..

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

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

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

[10] S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, "Promoting Distributed Accountability in the Cloud," Proc. IEEE Int'l Conf. Cloud Computing, 2011.

[11] A. Pretschner, F. Schuo¨ tz, C. Schaefer, and T. Walter, "Policy Evolution in Distributed Usage Control," Electronic Notes Theoretical Computer Science, vol. 244, pp. 109-123, 2009.

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

[13] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigen-baum, J. Hendler, and G.J. Sussman, "Information Accountability," Comm. ACM, vol. 51, no. 6, pp. 82-87, 2008.