A New Ensuring Technique of Distributed Accountability for Data Sharing in the Cloud

Chaganty Mounika[#1], Sathuluri Sravya[#2]
[#1] B.Tech Scholar
Department of CSE ,
Gayatri Vidya Parishad, College of Engineering for Women
Kommadi,Visakhapatnam,AP,India.
[#2] *B.Tech Scholar*
Department of CSE ,
Gayatri Vidya Parishad, College of Engineering for Women
Kommadi,Visakhapatnam,AP,India.

Abstract

Cloud computing enables highly scalable services to be easily consumed over the Internet on an as-needed basis. A major feature of the cloud services is that users' data are usually processed remotely in unknown machines that users do not own or operate. While enjoying the convenience brought by this new emerging technology, users' fears of losing control of their own data (particularly, financial and health data) can become a significant barrier to the wide adoption of cloud services. In this paper , we propose an object-centered approach that enables enclosing our logging mechanism together with users' data and policies. We leverage the JAR programmable capabilities to both create a dynamic and traveling object, and to ensure that any access to users' data will trigger authentication and automated logging local to the JARs. To strengthen user's control, we also provide distributed auditing mechanisms. We provide extensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches.

Keywords: Cloud Computing, accountability, data sharing.

1. Introduction

Cloud computing enables highly scalable services to be easily consumed over the Internet on an as-needed basis. A major feature of the cloud services is that users' data are usually processed remotely in unknown machines that users do not own or operate. CLOUD computing presents a new way to supplement the current consumption and delivery model for IT services based on the Internet, by providing for dynamically scalable and often virtualized resources as a service over the Internet. To date, there are a number of notable commercial and individual cloud computing services, including Amazon, Google, Microsoft, Yahoo, and Sales force [1]. 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. Such fears are becoming a significant barrier to the wide adoption of cloud services [2].

To relieve 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. 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 theses entities can also delegate the tasks to others, and so on. 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.

To overcome the above problems, we propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability [3]. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, information account-ability focuses on keeping the data usage transparent and trackable. 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 account-ability 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 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, 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.

We tested our CIA framework in a cloud testbed, the Emulab testbed [4], with Eucalyptus as middleware [5]. 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 compro-mised Java Running Environment (JRE).

In summary, our main contributions are as follows:

- We propose a novel automatic and enforceable logging mechanism in the cloud. To our knowledge, this is the first time a systematic approach to data accountability through the novel usage of JAR files is proposed.
- Our proposed architecture is platform independent and highly decentralized, in that it does not require any dedicated

authentication or storage system in place.

- We go beyond traditional access control in that we provide a certain degree of usage control for the protected data after these are delivered to the receiver.
- We conduct experiments on a real cloud test bed. 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.

This paper is an extension of our previous conference paper [6]. We have made 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.

2. Background Theory

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

2.1 Cloud Privacy and Security

Cloud computing has raised a range of important privacy and security issues [1], [7], [2]. Such issues are due to the fact that, in the cloud, users' data and applications reside—at least for a certain amount of time—on the cloud cluster which is owned and maintained by a third party. Concerns arise since in the cloud it is not always clear to individuals why their personal information is requested or how it will be used or passed on to other parties. To date, little work has been done in this space, in particular with respect to accountability. Pearson et al. have proposed accountability mechanisms to address privacy concerns of end users [2] and then develop a privacy manager [8]. Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is deobfuscated by the privacy manager to reveal the correct result. However, the privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed. In [9], 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 account-ability, 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.

To the best of our knowledge, the only work proposing a distributed approach to accountability is from Lee and colleagues [10]. The authors have proposed an agent-based system specific to grid computing. Distributed jobs, along with the resource consumption at local machines are tracked by static software agents. The notion of account-ability policies in [10] is related to ours, but it is mainly focused on resource consumption and on tracking of sub jobs processed at multiple computing nodes, rather than access control.

2.2 Other Related Techniques

With respect to Java-based techniques for security, our methods are related to self-defending objects (SDO) [11]. Self-defending objects are an extension of the object-oriented programming paradigm, where software objects that offer sensitive functions or hold sensitive data are responsible for protecting those functions/data. Similarly, we also extend the concepts of object-oriented programming. The key difference in our implementations is that the authors still rely on a centralized database to maintain the access records, while the items being protected are held as separate files. In previous work, we provided a Java-based approach to prevent privacy leakage from indexing [12], which could be integrated with the CIA framework proposed in this work since they build on related architectures.

In terms of authentication techniques, Appel and Felten [13] proposed 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) [14]. 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.

In addition, our work may look similar to works on secure data provenance [15], [16], [17], but in fact greatly differs from them in terms of goals, techniques, and application domains. Works on data provenance aim to guarantee data integrity by securing the data provenance. They ensure that no one can add or remove entries in the middle of a provenance chain without detection, so that data are correctly delivered to the receiver. Differently, our work is to provide data accountability, to monitor the usage of the data and ensure that any access to the data is tracked. Since it is in a distributed environment, we also log where the data go. However, this is not for verifying data integrity, but rather for auditing whether data receivers use the data following specified policies.

3. Problem Statement

We begin this section by considering an illustrative example which serves as the basis of our problem statement and will be used throughout the paper to demonstrate the main features of our system.

Example 1. Alice, a professional photographer, plans to sell her photographs by using the SkyHigh Cloud Services. For her business in the cloud, she has the following requirements:

. Her photographs are downloaded only by users who have paid for her services.

. Potential buyers are allowed to view her pictures first before they make the payment to obtain the download right.

. Due to the nature of some of her works, only users from certain countries can view or download some sets of photographs.

. For some of her works, users are allowed to only view them for a limited time, so that the users cannot reproduce her work easily.

. In case any dispute arises with a client, she wants to have all the access information of that client.

. She wants to ensure that the cloud service providers of SkyHigh do not share her data with other service providers, so that the accountability provided for individual users can also be expected from the cloud service providers.

With the above scenario in mind, we identify the common requirements and develop several guidelines to achieve data accountability in the cloud. 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. Using conventional access control mechanisms, once the access rights are granted, the data will be fully available at the service provider. In order to track the actual usage of the data, we aim to develop novel logging and auditing techniques which satisfy the following requirements:

1. 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.

2. Every access to the user's data should be correctly and automatically logged. This requires integrated techniques to authenticate the entity who accesses the data, verify, and record the actual operations on the data as well as the time that the data have been accessed.

3. 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.

4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data. More importantly, log files should be retrievable anytime by their data owners when needed regardless the location where the files are stored.

5. 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.

4. Cloud Information Accountability (CIA)

The Cloud Information Accountability framework proposed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It has two major components: logger and log harmonizer.

4.1 Major Components

There are two major components of the CIA, 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. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X.

The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement. Furthermore, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, 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.

The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the 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.

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. These two modes allow us to satisfy the aforementioned fourth design requirement. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner.The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable JAR files. The JAR file implementation provides automatic logging functions, which meets the second design requirement.

4.2 Data Flow

The overall CIA framework, combining data, users, logger and harmonizer is sketched in Fig. 1. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption [4] (step 1 in Fig. 1). This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture as described in Section 7. 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 (steps 3-5 in Fig. 1), we use OpenSSLbased certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication [18], wherein a trusted identity provider issues certificates verifying the user's identity based on his username.

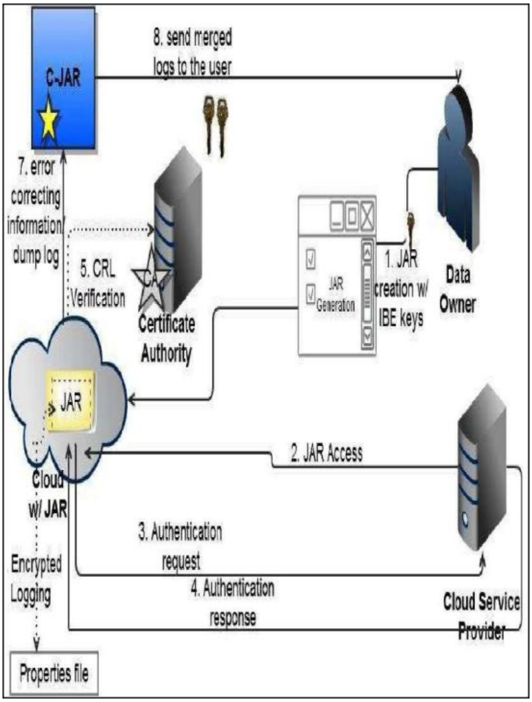

Fig 1. Overview of CIA Framework

5. Security Discussion

We now analyze possible attacks to our framework. Our analysis is based on a semi honest adversary model by assuming that a user does not release his master keys to unauthorized parties, while the attacker may try to learn extra information from the log files. We assume that attackers may have sufficient Java programming skills to disassemble a JAR file and prior knowledge of our CIA architecture. We first assume that the JVM is not corrupted, followed by a discussion on how to ensure that this assumption holds true.

6. Conclusion

Through this paper, we proposed innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

7. Future Enhancement

In the future, we plan to refine our approach to verify the integrity of the JRE and the authentication of JARs [19]. For example, we will investigate whether it is possible to leverage the notion of a secure JVM [20] being developed by IBM. This research is aimed at providing software tamper resistance to Java applications. In the long term, we plan to design a comprehensive and more generic object-oriented approach to facilitate autonomous protection of traveling content. We would like to support a variety of security policies, like indexing policies for text files, usage control for executables, and generic accountability and provenance controls.

8. References:

[1] P.T. Jaeger, J. Lin, and J.M. Grimes, “Cloud Computing and Information Policy: Computing in a Policy Cloud?,” J. Information Technology and Politics, vol. 5, no. 3, pp. 269-283, 2009.

[2] S. Pearson and A. Charlesworth, “Accountability as a Way Forward for Privacy Protection in the Cloud,” Proc. First Int’lConf. Cloud Computing, 2009.

[3] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G.J. Sussman, “Information Accountability,” Comm. ACM, vol. 51, no. 6, pp. 82-87, 2008.

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

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

[6] S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, “Promoting Distributed Accountability in the Cloud,” Proc. IEEE Int’l Conf. Cloud Computing, 2011.

[7] 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.

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

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

[10] W. Lee, A. Cinzia Squicciarini, and E. Bertino, “The Design and Evaluation of Accountable Grid Computing System,” Proc. 29th IEEE Int’l Conf. Distributed Computing Systems (ICDCS ’09), pp. 145-154, 2009.

[11] J.W. Holford, W.J. Caelli, and A.W. Rhodes, “Using Self- Defending Objects to Develop Security Aware Applications in Java,” Proc. 27th Australasian Conf. Computer Science, vol. 26, pp. 341-349, 2004.

[12] A. Squicciarini, S. Sundareswaran, and D. Lin, “Preventing Information Leakage from Indexing in the Cloud,” Proc. IEEE Int’l Conf. Cloud Computing, 2010.

[13] 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

[14] M.C. Mont, S. Pearson, and P. Bramhall, “Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services,” Proc. Int’l Workshop Database and Expert Systems Applications (DEXA), pp. 377-382, 2003.

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

[16] 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.

[17] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," Proc. Seventh Conf. File and Storage Technologies, pp. 1-14, 2009.

[18] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "Towards a Theory of Accountability and Audit," Proc. 14th European Conf. Research in Computer Security (ESORICS), pp. 152-167, 2009.

[19] J.H. Lin, R.L. Geiger, R.R. Smith, A.W. Chan, and S. Wanchoo, Method for Authenticating a Java Archive (jar) for Portable Devices, US Patent 6,766,353, July 2004.

[20] Trusted Java Virtual Machine IBM, http://www.almaden.ibm. com/cs/projects/jvm/, 2012.

9. About the Authors:

Chaganty Mounika is currently pursuing her B.tech in CSE at Gayatri Vidya Parishad, College of Engineering for Women, Visakhapatnam. Her area of interests includes Cloud Computing and Networks.

Sathuluri Sravya is currently pursuing her B.tech in CSE at Gayatri Vidya Parishad, College of Engineering for Women, Visakhapatnam. Her area of interests includes Cloud Computing and Security.

