Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 73 (2014) 26154-26157

Elixir ISSN: 2229-712X

Decentralized information accountability framework for information sharing in cloud environment

Manjunath Sholapur and B.G.Prasad
B.N.M.I.T, Bangalore.

ARTICLE INFO

Article history:
Received: 6 August 2013;
Received in revised form:
25 July 2014;
Accepted: 5 August 2014;

Keywords
Cloud computing,
Cloud services,
JAR programmable.

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 user's data are usually processed remotely in unknown machines that user's do not own or operate. While enjoying the convenience brought by this new emerging technology, user's 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. To address this problem, a highly decentralized information accountability framework is proposed to keep track of the actual usage of the user's data in the cloud. In particular, an object-centered approach that enables enclosing the logging mechanism together with user's data and policies. The JAR programmable capabilities are leveraged to both create a dynamic and traveling object, and to ensure that any access to user's data will trigger authentication and automated logging that is local to the JARs. To strengthen user's control, a distributed auditing mechanism is proposed to be implemented.

Introduction

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 Salesforce [1]. Details of the services provided are abstracted from the users who no longer need to be experts of technology infrastructure. Moreover, users may not know the machines which actually process and host their data. While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data. The data processed on clouds are often outsourced, leading to a number of issues related to accountability, including the handling of personally identifiable information. Such fears are becoming a significant barrier to the wide adoption of cloud services [2].

Objective

The main objective of this paper is create the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication.

Methodology

Feasibility Study

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing system and the proposed venture, the resources required to carry through, and ultimately the prospects for success. A detailed feasibility study was conducted to know the technical and financial feasibility of the project and it was found that the project is feasible to design, develop, use and maintain in all respects.

Requirement Analysis and Project Planning

The requirements of this project was analysed in detail which includes system requirements specification, software and hardware requirements. The project plan was developed with the help of requirements gathered in this phase.

Design

After successful analysis of the system requirement, design of the project started where various design constraints were analysed. The design phase consists of various modules to be developed for generation of data. The flow diagrams along with the activity diagrams are indicated to show the flow of control at various stages of the project.

Coding

The design of the system developed during the design phase is converted into code using the Java environment as and when required at different stages of the project. The coding is done according to the design strategy which aligns with the functional requirements that are categorized as in the previous step.

Testing

The program is tested by executing with the set of test cases in different set of setup environments and also stand alone systems. Then output of the program for the test cases is evaluated of determine if the program is performing as expected

Analysis

Analysis phase is a detailed study of various operations performed by a system and their relationships within and outside the system. One aspect of system analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. The emphasis in system analysis is on identifying what is needed from the system and not how the system will achieve its goal.

Existing system

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

Tele:
E-mail addresses: piyushpareek88@gmail.com

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. Self-defending objects (SDO) 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. 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. 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, and mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring content.

Disadvantages:

➢ Data handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments.

➢ 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.

➢ The end-to-end trust management and accountability problem in federated systems.

➢ The privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed.

➢ All these works stay at a theoretical level and do not include any algorithm for tasks like mandatory logging.

Proposed System

A Cloud Information Accountability (CIA) framework, based on the notion of information accountability. 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 tractable. 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. 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 user's data by any entity in the cloud.

Advantages:

➢ Data handling in the cloud is easy to access through an automatic and enforceable logging mechanism.

➢ Architecture is platform independent and highly decentralized, in that it does not require any dedicated authentication or storage system in place.

➢ The first time a systematic approach to data accountability through the novel usage of JAR files.

➢ The efficiency, scalability, and granularity of our approach.

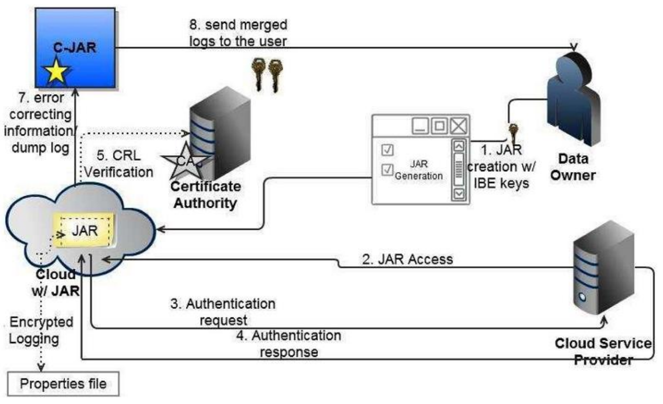


Figure 4.1- System Architecture

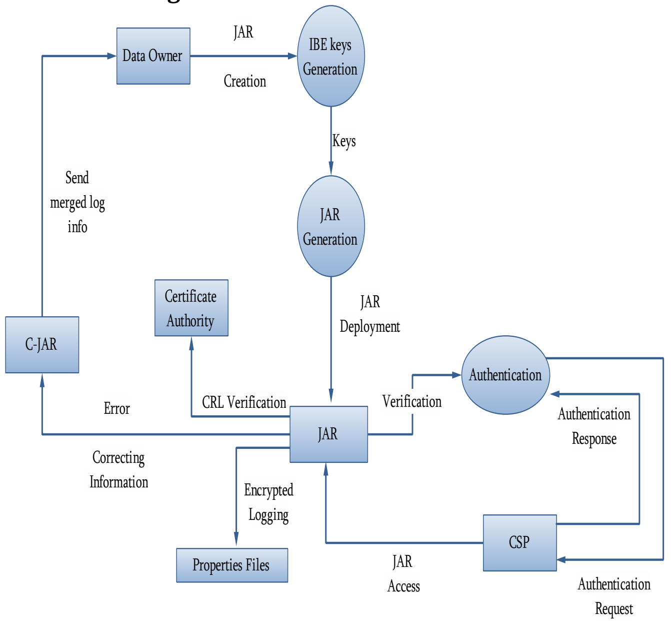
The overall CIA framework, combining data, users, logger and harmonizer is sketched. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption. This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks 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 Open SSL based 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, wherein a trusted identity provider issues certificates verifying the user's identity based on his username.

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. The data owner could opt to reuse the same key pair for all JAR's or create different key pairs for separate JAR's. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some 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.

Proposed framework prevents various attacks such as detecting illegal copies of user's data. Note that our work is different from traditional logging methods which use encryption to protect log files. 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.

The design process translates the requirements into a representation of the software that can be assessed for quality before coding begins. Once the requirements have been collected and analyzed, it is necessary to identify in detail how the system will be constructed to perform the necessary tasks.

Data Flow Diagram



FLOW DIAGRAM OF SYSTEM ARCHITECTURE

LEVEL 0

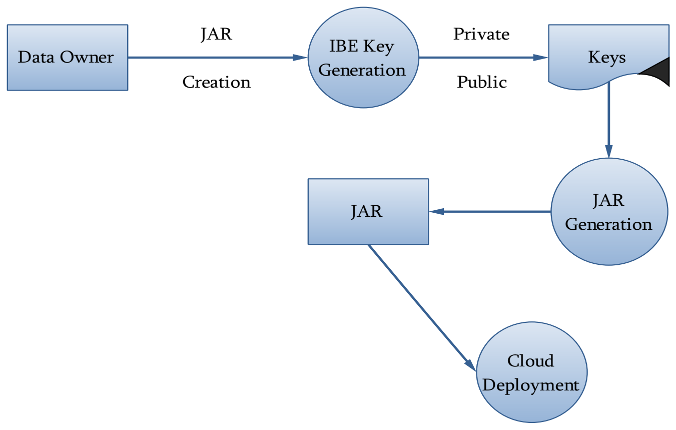
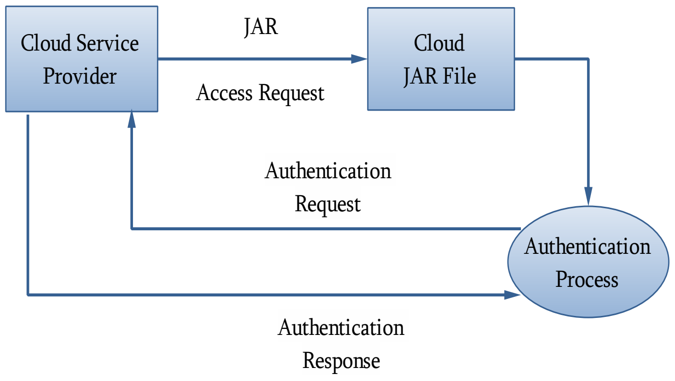


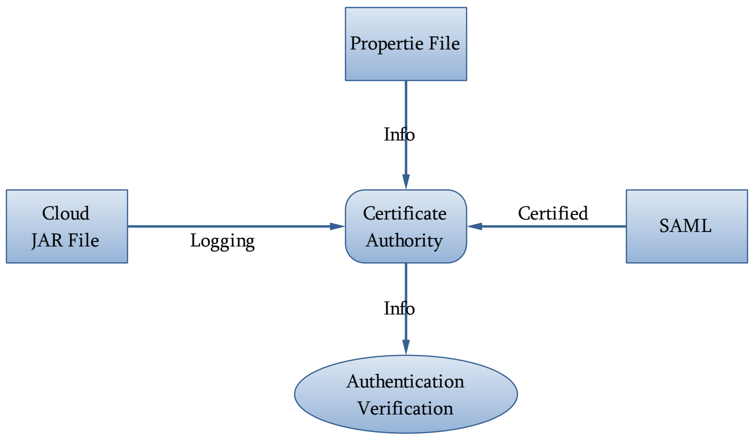
Figure 5.2- Flow Diagram Of Data Owner

Level 1



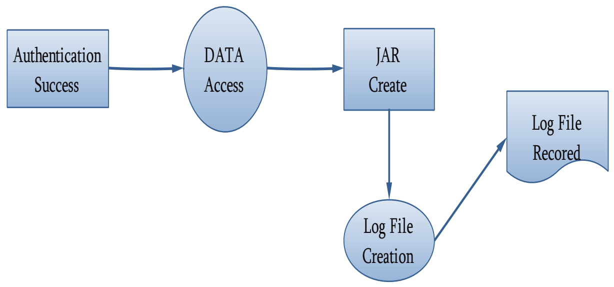
FLOW DIAGRAM OF CLOUD SERVICE PROVIDER

Level 2



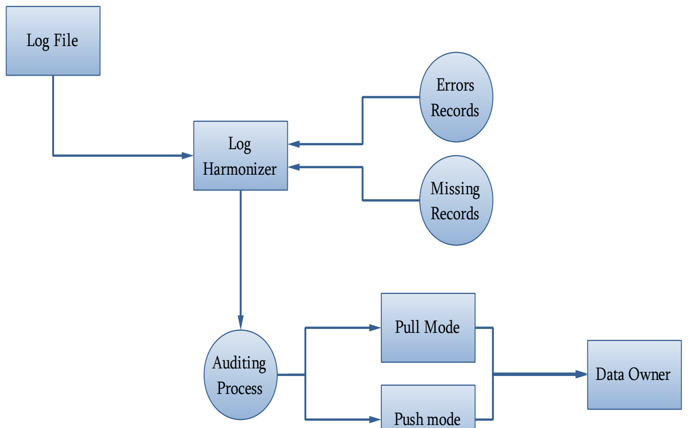
FLOW DIAGRAM OF CERTIFICATE AUTHORITY

Level 3



FLOW DIAGRAM OF LOG FILE RECORD

Level 4



FLOW DIAGRAM OF LOG HARMONIZER

The objective of implementation step is to create the code, test it for required output and debug the errors occurring during the execution of the program. System implementation involves testing the tool created on the setup and finding that the data is generated in the central manager database.

Integration

Integration is a process where in the code module for certain specification is tested separately and implemented into the existing big system to work well without any flaws introduced into the bigger system due to the new code integrated or flaws into the new module due to integration. Integration caters to the compatibility of the new to the existing code.

Testing

Testing is a process where in the code written is thoroughly checked for the various implementation errors and tested to find defects where the expected results do not occur for a certain module. Testing of the newly written code after integrating it with the existing code would yield few errors which need to be fixed and verified to ensure a fail-safe and correct working of the system as a whole.

Debugging

Debugging is a process which detects and corrects the syntactical and logical errors in a program. The syntax errors can be detected by the compiler. The diagnosis of logical errors is complicated by the delay which normally exists between the error insertion into the code and the appearance of the error as a defect while debugging. The most complex errors might get detected only in certain conditions that might be present in certain state of the system. Debugging needs a lot of good test cases to test and find embedded defects.

Conclusion And Future Enhancement

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. Future plan is to refine our approach to verify the integrity of the JRE and the authentication of JAR's. For example, investigate whether it is possible to leverage the notion of a secure JVM 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. To support a variety of security policies, like indexing policies for text files, usage control for executables, generic accountability and provenance controls.

References

1)G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Cloud Privacy and Security" Proc. ACM Conf. Computer and Comm. Security, pp. 598-609, 2007.

2)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.

3)S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, "Trust Cloud: A Framework for Accountability and Trust in Cloud Computing," Proc. IEEE Int'l Conf. Cloud Computing, 2011.

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