PRIVACY MANAGER FOR DATA SHARING IN THE CLOUD

SHAHNA FATHIMA S , S M NANDHAGOPAL

Abstract-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. 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 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 whereby the user (or another authorized party) can retrieve the logs as needed.

Keywords -Accountability, Auditing

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.

a.Accountability

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. To address this problem, in this paper, we propose a novel highly decentralized information accountability framework to keep track of the actual usage of the users' data in the cloud.

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.

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 track able. Our proposed CIA framework provides end-to-end accountability in a highly distributed fashion.

2. RELATED WORK

a) Decentralized trust management

The most basic example of this is authorization to create a *slice*, a network of virtual machines, and to deploy a wide-area network service in it. Once users have been authorized to create slices and deploy services, we then need to monitor resource usage in each slice to ensure that resources are not being misused.

b)Auditing accountability

Our system consists of a group of communicating agents which create and share data and an authorization authority which may audit agents. The creation of data, as well as the communication between agents, is assumed to leave some evidence and hence is observable (from the perspective of the authorization authority). As we do not continuously monitor agents, the internal computations of agents are not considered to be observable. However, when auditing an agent, the data and policies currently stored by an agent become visible to the authorization authority. Thus, the model of an agent consists of storage, (unobservable) internal computation and (observable) actions such as communication.

c) Privacy protection

Privacy concerns arise whenever sensitive data is outsourced to the cloud. By using encryption, the cloud server (i.e. its administrator) is prevented from learning content in the outsourced databases. The new access control file can now be distributed over the Encryption Proxies. An Encryption Proxy accepts a new XACML file on the condition: the file is signed by two (or more) editors and all their signatures are good. If the file is not proper, or correctly signed, the system rejects the new file and falls back to the current rights (for availability reasons).

d) Theory of accountability and audit

We describe an operational model of accountability based systems. Honest and dishonest principals are described as agents in a distributed system where the communication model guarantees point-to-point integrity and authenticity. Auditors and other trusted agents (such as trusted third parties) are also modeled internally as agents. Behaviors of all agents are described as processes in process algebra with discrete time. Auditor implement ability is ensured by forcing auditor behavior to be completely determined by the messages that it receives.

3. EXISTING SYSTEM

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.

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.

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.

4.PROPOSED SYSTEM

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 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 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 testbed. 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.

5. System Model

The overall CIA framework, combining data, users, logger and harmonizer is sketched in. 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.

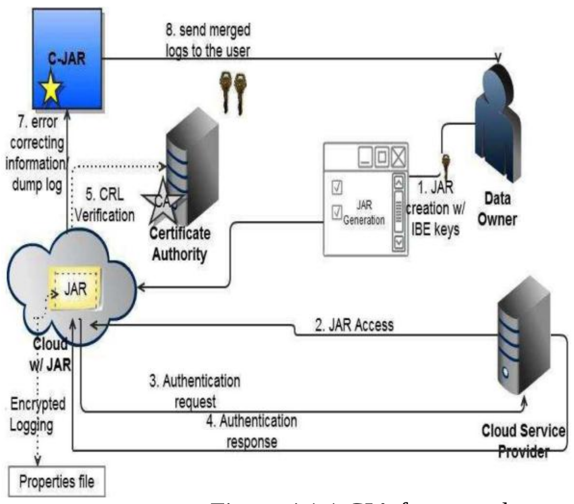

Figure 4.1.1 CIA framework

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 JARs or create different key pairs for separate JARs. Using separate keys can enhance the security (detailed discussion is in Section 7) 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.

5.1 Jar Usage

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. In our context, the data owners may not know the exact CSPs that are going to handle the data. Hence, authentication is specified according to the servers' functionality (which we assume to be known through a lookup service), rather than the server's URL or identity. For example, a policy may state that Server X is allowed to download the data if it is a storage server. As discussed below, the outer JAR may also have the access control functionality to enforce the data owner's requirements, specified as Java policies, on the usage of the data.

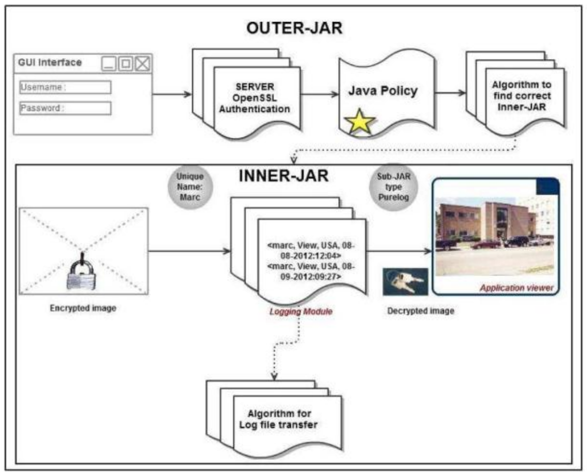
OUTER-JAR
GUI Interface
Username :
Password :
SERVER OpenSSL Authentication
Java Policy
Algorithm to find correct Inner-JAR
INNER-JAR
Unique Name: Marc
Sub-JAR type Purelog
<marc, View, USA, 08-08-2012:12:04>
<marc, View, USA, 08-09-2012:09:27>
Encrypted image
Logging Module
Decrypted image
Application viewer
Algorithm for Log file transfer

Figure 4.2.1 JAR usage

A Java policy specifies which permissions are available for a particular piece of code in a Java application environment. The permissions expressed in the Java policy are in terms of File System Permissions. However, the data owner can specify the permissions in user-centric terms as opposed to the usual code-centric security offered by Java, using Java Authentication and Authorization Services. Moreover, the outer JAR is also in charge of selecting the correct inner JAR according to the identity of the entity who requests the data.

5.1.1 Cloud storage

Cloud Storage is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. Hosting companies operate large data centers; and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs. The data center operators, in the background, virtualize the resources according to the requirements of the customer and expose them as virtual servers, which the customers can themselves manage. Physically, the resource may span across multiple servers.

5.1.2Cloud information accountability

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. 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.

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 sends 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.

5.1.3 Data flow

The overall CIA framework, combining data, users, logger and harmonizer is sketched in. 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.

5.1.4 Automated logging mechanism

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. In our context, the data owners may not know the exact CSPs that are going to handle the data. Hence, authentication is specified according to the servers' functionality (which we assume to be known through a lookup service), rather than the server's URL or identity.

6. CONCLUSION

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. To address this problem, in this paper, we propose a novel highly decentralized information accountability framework to keep track of the actual usage of the users' data in the cloud.

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 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. Chun and A.C. Bavier, (2004) "Decentralized Trust Management and Accountability in Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS).
2. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, (2005) "A Logic for Auditing Accountability in Decentralized Systems," Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201.
3. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, (2009) "Towards a Theory of Accountability and Audit," Proc. 14th European Conf. Research in Computer Security (ESORICS), pp. 152-167.
4. Pearson and A. Charlesworth, (2009) "Accountability as a Way Forward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing.