JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Reliable Data-Sharing Responsibility Mechanism in the Cloud computing.

Akheel Mohammed*and Ayesha

Abstract

The term "cloud computing" refers to the practice of utilizing remote data centers and other computing facilities to complete tasks. It paves the way for highly scalable services to be consumed on-demand over the internet. One defining feature of cloud services is that customers' data is typically processed on unrecognizable devices outside of their control. While the convenience of this new technology is undeniable, users' concerns about privacy may prevent them from fully embracing cloud services. It has the potential to become a major barrier to widespread use of cloud services. We propose a highly decentralized answerability framework to monitor the actual utilization of users' data in the cloud, which would be a significant step toward solving this issue. In this paper, we present a cloud information accountability framework that uses automated logging and distributed auditing to keep track of who accessed what, when, and where in the cloud. We propose an object-oriented method that allows us to wrap the user's data and policies in the same container as our logging technique. The proposed solution would also deal with the JAR file, turning it into obfuscated code that will further fortify the system's defenses. In addition, we will be implementing a system of proved data possession for integrity verification, which will greatly improve the safety of users' data.

Keywords: Cloud Computing, Data Sharing, Information accountability framework, Provable data possession.

1. Introduction

The system for cloud information auditing presented in this study uses automatic logging and distributed auditing to keep track of who accessed what, when, and where in the cloud. A logger and a log harmonizer are its two primary parts. To define if and how cloud services and maybe other data stakeholders can access the content, the JAR file provides a set of simple access control rules. Other than that, we'll make sure the JRE on the computers where the logger components are run is legitimate. While this new technology has many benefits, people are also starting to worry about giving up control of their personal data. Outsourcing the cloud's data processing raises responsibility concerns, especially when it comes to the protection of sensitive information like names and addresses. These concerns have emerged as a major roadblock to widespread use of cloud services. Oblivious hashing is used to perform these integrity checks. The proposed solution would also deal with the JAR file, turning it into obfuscated code that will further fortify the system's defenses. In addition, we will strengthen the safety of user information by using verifiable data holdings to ensure data integrity. The JAR will either provide use management in conjunction with logging, or it will provide solely logging functionality, depending on the configuration options defined at the time of creation. In terms of logging, the JAR will generate a log record whenever the data is accessed. Cloud computing, in the current system, is the distribution of computing as a service rather than a product, in which computers and other devices are provided with access to a common pool of hardware, software, and data via a network utility, analogous to how energy is distributed to homes and businesses. These days, a single server handles all of the user requests. Due to the server having to handle two requests at once, the total processing time is increased. As a result, data integrity issues, transaction delays, and compromised wallet data are all possible, and you can't put your trust in the underlying data infrastructure or service providers. While this new technology has many benefits, people are also starting to worry about giving up control of their personal data. Outsourcing the administration of sensitive data, such as that stored in the cloud, raises a number of accountability concerns. To put users' minds at ease, a reliable means for tracking how their data is being used in the cloud should be made available. For instance, due to the following characteristics of cloud environments, users need to be able to ensure that their data are handled according to the service level agreements made at the time they sign up for services, or approaches with a centralized server in distributed environment are not suitable.

Disadvantage: The database management system is not trustworthy in cloud computing, despite the fact that it is a rapidly expanding and improving technology.

A unique cloud-based automatic and legally binding logging technique is proposed. To the best of our knowledge, this is the first proposal of its kind, a systematic method for ensuring data accountability via the innovative use of JAR files.

This post is a follow-up to a conference paper we presented previously. The following are some of our recent improvements. To ensure the reliability of our system in the face of a corrupted JRE, we first incorporated integrity checks and an obvious hashing scheme. We have also improved the log records structure to give you more confidence in their honesty and legitimacy. We broaden the scope of the security analysis to include more types of attacks. We next detail the outcomes of further experiments and offer an in-depth analysis of the system's effectiveness.

2.Related Research

The first review of books and articles dealing with cloud computing privacy and security. We then briefly review other works that use methods analogous to our own, albeit for different ends.

2.1 Privacy and safety in the cloud

The idea behind them is that sensitive user information can be securely transmitted to the cloud and then processed there. The correct processing result is revealed after being deobfuscated by the privacy manager. However, the privacy manager has limited functionality in that it cannot ensure data security after it has been published. The authors' main focus is on leveraging trust connections for accountability in addition to authentication and anomaly detection, which is considerably different from our own. Most academic work on accountability has focused on how to make it a cryptographically proven quality, notably in the setting of online transactions. Crispo and ruffo put forth a novel theory on responsibility in delegation. Since we are not attempting to manage the cloud-based flow of data, delegation is a useful tool that complements our efforts. To the best of our knowledge, only the work by lee and colleagues proposes a decentralized method of establishing responsibility. The authors suggest a grid-centric agent-based system.

2.2 Related Methods

Our approaches' security is based on Java-based mechanisms for protecting self-defending objects. It is an extension of the object-oriented programming paradigm that software objects that provide sensitive functions or store sensitive data are tasked with defending themselves. Since the CIA system given in this study is based on similar architecture, we gave a java-based solution to the problem of privacy leaking during indexing. The proof carrying authentication framework was proposed by Appel and Felten. The PCA is concerned with web service access management and features a high order logic language that permits quantification over predicates. In addition, Mont et al.'s approach provides an identifying-based encryption method for access control. We use IBE methods as well, though in a very unique fashion. We do not count on IBE to enforce compliance between content and regulations. We utilize it to safeguard the encrypted data and the logs from threats like tampered plaintext and ciphertext attacks. While concerns about the safety of data storage are related to privacy, they are outside the scope of this work.

3. The Proposed system

To address these issues, we suggest a unique approach built on the concept of information accountability: the Cloud Information Accountability framework. After data encryption, the data owner can then upload the data to the cloud server. Users can subscribe to the cloud server and be granted varying degrees of access to the original data (read, write, copy). The Loggers and Log Harmonizer will monitor and report on the data's access logs. In this paper, we offer a framework for ensuring the accountability of data stored in the cloud by automatically recording and auditing any relevant access made by any organization at any time. Both the Logger and the Log harmonizer are essential parts of it.

Benefit: we can exchange information in a safe and reliable environment.

3.1 Flow of Data

Each user starts by generating their own unique public and private Identifier-Based Encryption (IBE) keys. One of the most common attacks against our design is mitigated by this IBE scheme, which is a Weil pairing based IBE scheme. A logger component, in the form of a JAR file containing the necessary items, will be created by the user using the generated key.

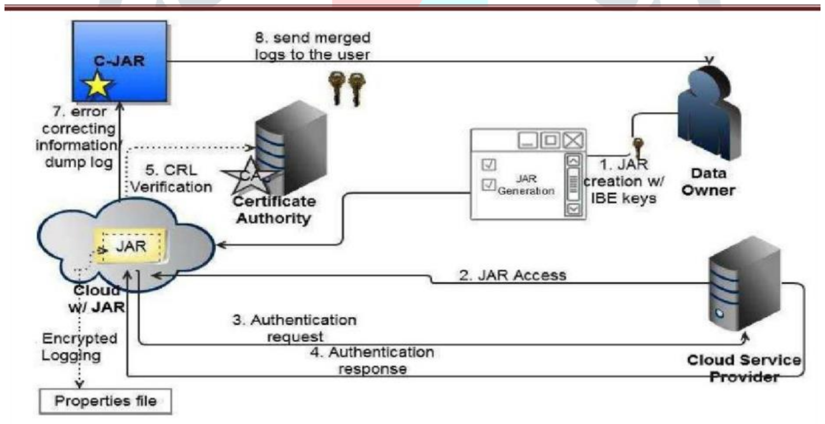

Figure 1. Overall Architecture.

Modification:

Each user's random numbers would be generated by the data owner, and any illegal actions would be reported automatically. When logging in, the user must enter the random number they generated, which will be checked by the server. If the verification is successful, then the user's account will be accessible only to them.

4.Implementation

4.1 The Development of the Suggested Algorithm

In this case, we employ a Log Retrieval Algorithm that operates in both Push and Pull modes. In the case of pure Log, the method shows logging and synchronization steps with the harmonizer. The algorithm initially determines if the JAR has grown too large or if the period between dumps has been longer than usual. At the time of JAR creation, the data owner defines the minimum and maximum allowable dump sizes and durations. The program can also tell if the data owner has asked for a dump of the logs. In the absence of any of these conditions, it will continue with the encryption of the record and the transmission of the mistake correction data to the harmonizer. A handshake is the first step in interacting with the harmonizer. When no reply is received, an error is recorded in the log file. When an issue occurs, the data owner is notified by email if the JAR is set up to do so. After the initial handshake is complete, a TCP/IP protocol is used to carry on the conversation with the harmonizer. If either of those things has happened, JAR simply deletes the log files and clears out all the variables so that a new record can be made. Verify that the CSP accessing the Access Log meets the requirements outlined in the relevant policies. If the requirements are met, permission is granted; otherwise, it is denied. The attempted access to the JAR file data will be noted regardless of the outcome of the access control. There are two primary benefits to our auditing system. To begin, it ensures a very high degree of log availability. Second, the harmonizer reduces the time it takes humans to examine massive amounts of data contained inside the log files that are transmitted by many versions of JAR files. Many kinds of modules, including but not limited to

4.1.1Owners/Users of the Data

The sailor/user is departing for sea/downloading data from the cloud server. A user must sign up with the cloud server in order to access the data stored there. Users will be required to sign up with information such as a username, password, and random numbers. For future authentication purposes, this data will be stored in a database. The owner of the data is the person responsible for storing it in the cloud. The Data Owner must sign up for a Cloud Server account before any data may be sent there. The space will be allocated to the Data Owner once they have enrolled on the cloud server.

4.1.2Hosted Web Server

The cloud server is where both the user and the data owner will store their information. To get the information you need, users must first submit a request to a cloud server, which will then relay the message to the data's rightful owner. The cloud server will also save user and data owner details in a database for later use.

4.1.3Logger

The Cloud Server is responsible for maintaining the log. The cloud server's loggers record information about the data's owner and any users who use the server. So, the Logger will serve many more functions. Including the identity of the data's rightful owner, the time and date of the request, and the IP address of the requesting user.

4.1.4Certifying Body

To ensure the Cloud server is legitimate, the certificate authority is consulted. The certificate authority must validate the cloud server. A fraudulent Cloud server is one that goes undetected by the user. The data owner can verify if the claims were true. Since the data owner will be storing their information in the cloud.

4.1.5 Authorized Access

Depending on the Owner, you may merely be able to read the content or even download it. Once the Cloud Server detects that a user has exceeded their permitted data access, they will send a dynamic warning. As a result, cloud data exchange is made much safer.

4.1.6 Concept of Push and Pull

The owner of the data may be able to see a list of everyone who accessed the file at a given moment. The cloud server will inquire during the signup process as to whether the data owner prefers the pull or push approach. With the pull technique, the data owner initiates contact with the cloud service provider to inquire about the current state of their data's accessibility.

4.1.7Generating Random Sets and Checking Them

The user must enter the random number set while making the data download request from the Cloud Server. If it's a good match, the user will be able to get their hands on the files. The pool of random numbers is changed each time. This safeguards the download of the data.

5.The End of the Experiment

Our system's performance is analyzed once we reveal the test environment conditions. The log harmonizer is stored in a separate location, either in a secure proxy or on the user end, making it inaccessible to the attacker and hence not considered in this attack. Therefore, we believe the log harmonizer is secure and the attacker cannot access the decryption keys.

We begin our studies by measuring the overhead in the system and then analyzing the time it takes to generate a log file. Both when each individual log record is being encrypted and when all of the logs are being combined. In addition, JAR appears to be a file compressor for the files it manages. In particular, the suggested solution allows a single logger component to handle data for several files. We investigated whether using a single logger component to handle many files increased the size of the system.

5.1Time of Log Generation

In the first set of tests, we are interested in measuring how long it takes for a log file to be created when there are entities that are constantly accessing the data, which results in continuous logging. The outcomes are depicted in fig2. To be more precise, it takes on average 114.5 ms to produce a 100 Kb file, and 731 ms to develop a 1 MB file. Using the results of this experiment as a starting point, one can calculate the minimum interval between dumps while also taking into account factors like available storage space and network congestion.

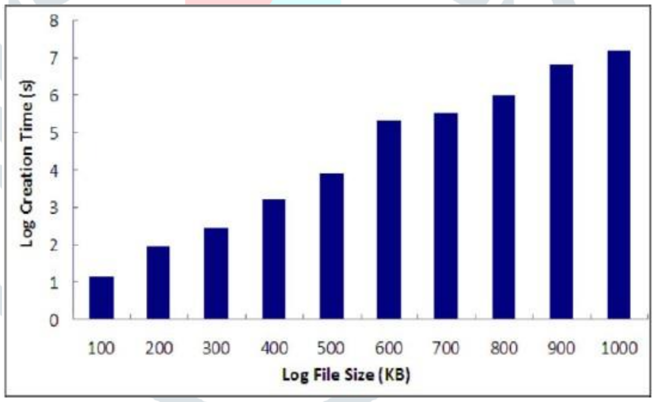

Fig .2. Time to merge log files.

5.2 Time of Authentication

Certificates can be cached to further boost performance. When only the JAR-required steps, such as obtaining a SAML certificate and then analyzing it, are taken into account, the time it takes to authenticate an end user is almost the same. This is done in a similar method by the JAR for both OpenSSL and SAML certificates. The average time for user activity is 1.2 minutes.

5.3 Effort Required for Logging Activities

The impact of log file size on logging performance is investigated in this set of experiments. We timed how long it took to allow an access and how long it took to write the accompanying log entry. Since every interaction is merely a view request, the time required to carry it out is quite small. In this way, the time it takes to record an event—whether it be the time it takes a user to double-click the JAR or the time it takes a server to run the script to open the JAR—is typically around 10 seconds. We also measured the time it took to encrypt the log, which was around 300 ms despite having no apparent correlation to the size of the file.

5.4 It Takes Logs to Merge

To determine if the log harmonizer is a bottleneck, we measured how long it took to combine log files. Here, we show that 10–25% of the records in the different log files are identical. Each each run of the experiment yielded a different average number of shared records. Ten repeats were used to calculate the average time. We timed how long it took to combine 70 log files ranging in size from 100 KB to 1 MB. Figure 3 displays the obtained outcomes. The time required to merge two 100 KB log files takes the least amount of time (59 ms) and increases practically linearly with the number of files and file size. When 70 files of 1 MB each were combined, the process took 2.35 minutes.

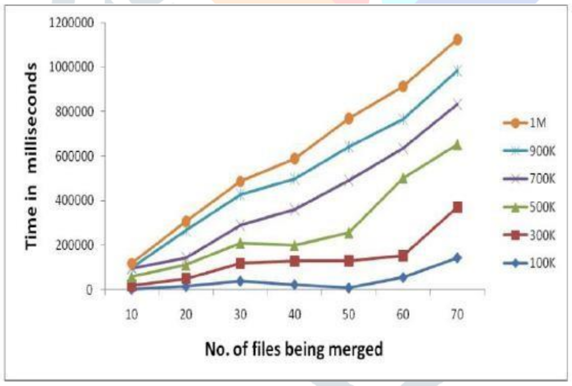

Fig.3. Time to create log files of different sizes.

When moving from 200 KB to 1 MB for content items, the logger size increases from 3525 to 4035 KB. Because of the compression offered by JAR files, the logger's overall size is determined by the largest files it stores. Please take note that we avoided including massive log files on purpose. In order to focus on the additional load that having numerous content files in a single JAR creates. The results are shown in Fig.4.

6 Conclusion

We implemented state-of-the-art methods, including an auditing tool, for automatically logging all cloud data access. With our method, the data owner can perform thorough content audits and, if necessary, implement robust back end protection. In addition, we have included PDP approach to improve the authenticity of owners' data. We hope to improve our method for ensuring JRE's authenticity in the future. To that end, we shall investigate whether or not the benefits of secure JVM being created can be utilized. By IBM, and we'd like to improve our PDP architecture from the end user's perspective so that data can be checked remotely and efficiently across several clouds.

References

1. **Text Books**

[1] Cloud Computing, Principles and Paradigms by John Wiley & Sons.

2. **Conference Proceedings**

    1. Ensuring Distributed Accountability for Data Sharing in the Cloud Author, Smitha Sundareswaran, Anna C.Squicciarini, Member, IEEE, and Dan Lin, IEEE Transactions on Dependable and Secure Computing ,VOL 9,NO,4 July/August 2012 .
    2. Hsio Ying Lin,Tzeng.W.G, **"**A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding ",IEEE transactions on parallel and distributed systems,2012.
    3. Yan Zhu, Hongxin Hu, Gail Joon Ahn, Mengyang Yu, "Coopera-tive Provable Data Possession for Integrity Verification in Multi-Cloud Storage" , IEEE transactions on parallel and distributed systems,2012.

3. **Generic Website**