ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

A New Approach for Extremely Decentralized Data Accountability for Data Sharing in Cloud

[1] B.Lakshmi Tirapatamma , [2]Y.Sankararao

[1]Final M .Tech Student, [2]Asst Professor

[1,2] Dept of Computer Science and Engineering

[1,2] St.mary's group of institutions –Guntur

ABSTRACT: Cloud computing is a technology, which uses internet and remote servers to stored data and application. Cloud computing provides on demand services. Multiple users want to do business of their data using cloud but they get fear to losing their data. While data owner will store his/her data on cloud , he must get confirmation that his/her data is safe on cloud. To solve above problem in this paper we provide effective mechanism to track usage of data using accountability. Accountability is checking of authorization policies and it is important for transparent data access. We provide automatic logging mechanisms using JAR programming which improves security and privacy of data in cloud. Using this mechanism data owner may know his/her data is handled as per his requirement or service level agreement.

Keywords: Cloud Computing, SaaS, DaaS, PaaS, IaaS.

I.INTRODUCTION

Cloud Computing

It is the use of computing resources (hardware and software) which are available in a remote location and accessible over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

Cloud storage

An online network storage where data is stored and accessible to multiple clients. Cloud storage is generally deployed in the following configurations: public cloud, private cloud, community cloud, or some combination of the three also known as hybrid cloud.

Cloud Services

Software as a service (SaaS)

The software-as-a-service (SaaS) service-model involves the cloud provider installing and maintaining software in the cloud and users running the software from their cloud clients over the Internet (or Intranet). The users' client machines require no installation of any application-specific software - cloud applications run on the server (in the cloud). SaaS is scalable, and system administration may load the applications on several servers. In the past, each customer would purchase and load their own copy of the application to each of their own servers, but with SaaS the customer can access the application without installing the software locally. SaaS typically involves a monthly or annual fee.

Development as a service (DaaS)

Development as a service is web based, community shared development tools. This is the equivalent to locally installed development tools in the traditional (non-cloud computing) delivery of development tools.

Platform as a service (PaaS)

It is cloud computing service which provides the users with application platforms and databases as a service. This is equivalent to middleware in the traditional (non-cloud computing) delivery of application platforms and databases.

Infrastructure as a service (IaaS)

It is taking the physical hardware and going completely virtual (e.g. all servers, networks, storage, and system management all existing in the cloud). This is the equivalent to infrastructure and hardware in the traditional (non-cloud computing) method running in the

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

cloud. In other words, businesses pay a fee (monthly or annually) to run virtual servers, networks, storage from the cloud. This will mitigate the need for a data center, heating, cooling, and maintaining hardware at the local level.

Cloud Accounting

For many companies, clouds are becoming an interesting alternative to a dedicated IT infrastructure. However, cloud computing also carries certain risks for both the customer and the cloud provider. The customer places his computation and data on machines he cannot directly control; the provider agrees to run a service whose details he does not know. If something goes wrong - for example, data leaks to a competitor, or the computation returns incorrect results - it can be difficult for customer and provider to determine which of them has caused the problem, and, in the absence of solid evidence, it is nearly impossible for them to hold each other responsible for the problem if a dispute arises.

In this paper, we propose that the cloud should be made accountable to both the customer and the provider. Both parties should be able to check whether the cloud is running the service as agreed. If a problem appears, they should be able to determine which of them is responsible, and to prove the presence of the problem to a third party, such as an arbitrator or a judge. We outline the technical requirements for an accountable cloud, and we describe several challenges that are not yet met by current accountability techniques

II ACCOUNTABILITY

Accountability is a concept to make the system accountable and trustworthy by binding each activity to the identity of its actor [5]. Such binding should be achieved under the circumstance that all actors within the system are semi-trusted. That is, each identified actor may lie according to their own interest. Therefore the bindings must be supported by provable or non-disputable evidence.

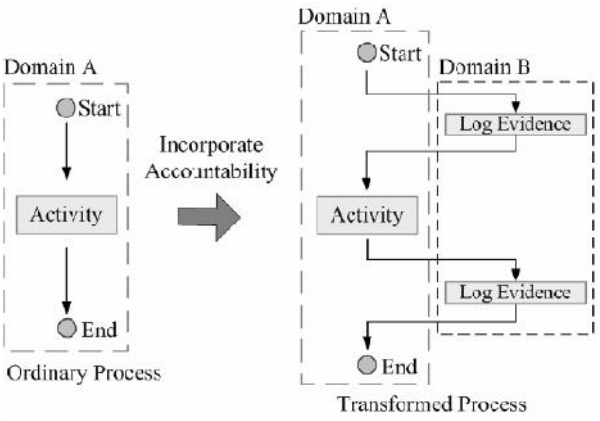

Fig 1: Example of incorporating accountability into process

In our approach, accountability can be incorporated into activity based process by requiring the actor (conductor) of the process to log non-disputable evidence about the activities in a separate domain from the domain of its own. Fig. 2 shows an example of such incorporation. In the example, domain A is required to perform logging operations before and after conducting the activity in its process. The evidence needs to be logged should contain enough information to describe the conducting activity. In the simple case in our example, intuitive enough, the evidence should include the states of the factors concerning the start of the activity (e.g. input variables) and the factors concerning its completion (e.g. output value).

As aforementioned, the logged evidence needs to be non-disputable so as to undeniably link the activity to its actor. That is, if by the logger or any other entity, the evidence can be proven to be associated with an activity conducted by a domain, no other entities can prove otherwise.

For Example, each of them has its own associated public-private key pair issued by certificate authorities. The logging procedure is illustrated in Fig. 3. First, the logger (domain A) signs the evidence (E) by its private key (K_A-) to create a digital signature of it (S_A). The evidence and its signature are then logged at a separate domain (domain B). When received, domain B creates a receipt by signing domain A's signature with domain B's private key (K_B-). At last, the receipt (S_B) is sent back to the logger (domain A) in the reply. Through this procedure, since the digital signature is un-forgeable, the signature of the evidence (S_A) and the receipt (S_B) enable both domains to prove the factor that domain A has logged this evidence at domain B. In another word, even though neither domain trusts the other, both of them can prove its own correctness with the signature it has kept.

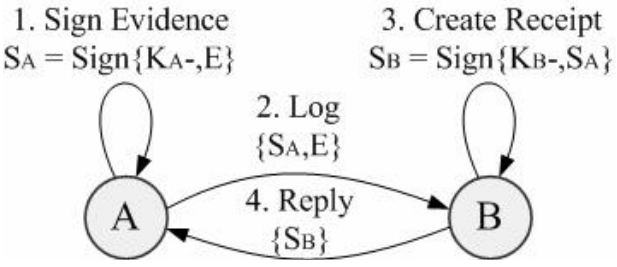

Figure 3. Evidence logging procedure

III. Phases of Cloud Accountability

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

We propose Cloud Accountability Life Cycle (CALC) as the following seven phases

Policy Planning

In the beginning, CSPs have to decide what information to log and which events to log on-the-fly. It is not the focus of this paper to claim or provide an exhaustive list of recommended data to be logged. However, in our observation, there are generally four important groups of data that must be logged: (1) Event data – a sequence of activities and relevant information, (2) Actor Data – the person or computer component (e.g. worm) which trigger the event, (3) Timestamp Data – the time and date the event took place, and (4) Location Data – both virtual and physical (network, memory, etc) server addresses at which the event took place.

Sense and Trace

The main aim of this phase is to act as a sensor and to trigger logging whenever an expected phenomenon occurs in the CSP's cloud (in real time). Accountability tools need to be able to track from the lowest-level system read/write calls all the way to the irregularities of high-level workflows hosted in virtual machines in disparate physical servers and locations. Also, there is a need to trace the routes of the network packets within the cloud.

Logging

File-centric perspective logging is performed on both virtual and physical layers in the cloud. Considerations include the lifespan of the logs within the cloud, the detail of data to be logged and the location of storage of the logs.

Safe-keeping of Logs

After logging is done, we need to protect the integrity of the logs prevent unauthorized access and ensure that they are tamper-free. Encryption may be applied to protect the logs. There should also be mechanisms to ensure proper backing up of logs and prevent loss or corruption of logs. Pseudonymisation of sensitive data within the logs may in some cases be appropriate.

Reporting and Replaying

Reporting tools generate from logs file-centric summaries and reports of the audit trails, access history of files and the life cycle of files in the cloud. Suspected irregularities are also flagged to the end-user. Reports cover a large scope: virtual and physical server histories within the cloud; from OS-level read/write operations of sensitive data to high-level workflow audit trails.

Auditing

Logs and reports are checked and potential fraud-causing loopholes highlighted. The checking can be performed by auditors or stakeholders. If automated, the process of auditing will become 'enforcement'. Automated enforcement is very feasible for the massive cloud environment, enabling cloud system administrators and end-users to detect irregularities more efficiently.

Optimizing and Rectifying

Problem areas and security loopholes in the cloud are removed or rectified and control and governance of the cloud processes are improved.

Cloud Accountability Abstraction Layers

We propose the following layers of accountability in a cloud:

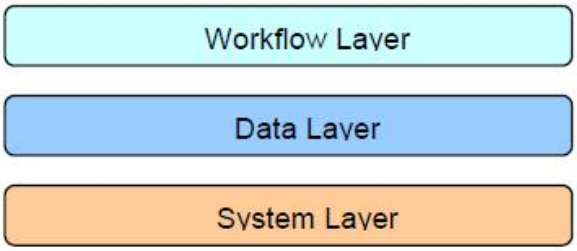

Fig 2 :Abstraction Layers of Accountability in Cloud Computing

System Layer

At the lowest level lie the system layer logs. The system layer consists of logging within the following components:

Operating System (OS)

OS system and event logs are the most common type of logs associated with cloud computing at the moment.

File System

Even though the file system is technically part of the OS, we explicitly include it as a major component in this system layer. This is because, in order to know, trace and record the exact file life cycle and history, we often have to track system read/write calls to the file system. From the system read/write calls, we can also extract the virtual and physical memory locations of the file, providing more information for further forensic investigations. The file-centric perspective [5] is also the area which is less emphasized by current tools. Cloud computing needs to have more emphasis on file-centric logging, and the tracing and logging of a file's life cycle (i.e. creation, modification, duplication, destruction).

Network Logs

As clouds are vast networks of physical and virtual servers over a large number of locations, we need to also monitor network logs within the cloud. Network logs are logs specific to data being sent and received over the network.

Data Layer

This layer contains the logging of data transactions and the life cycle of data. The difference with the system layer is that the system layer's file system logs track the life cycle of files, whereas the data layer actually tracks the life cycle of data and the contents of files. The same file can contain drastically different sets of data over time. Some examples of the data layer are: (1) data provenance, which records the so-called chains of custody [9] (e.g. the history of owners and authorized users) of the data found in the cloud and (2) database logs [2] (i.e. histories of updates and actions executed by a database management system to the database).

Workflow Layer

This layer primarily contains logs which reveal the robustness or weaknesses of the governance and controls of a workflow or business process [3] in an organization. It correlates with an organization's strategic and management levels [2]. It is the key layer audited by most IT auditors and internal audits. Examples include: (1) audit trails from transactions in business process and workflow management systems (2) audit trails from information systems for the customer organizations (e.g. ERP systems, Human Resource systems, etc) (3) continuous auditing and monitoring tools.

Automated Logging Mechanism

We control the programmable capability of JARs to conduct automated logging. A logger component is a nested Java JAR file which stores a user's data items and corresponding log files.

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.

Example 1. Suppose that Alice's photographs are classified into three categories according to the locations where the photos were taken. The three groups of photos are stored in three inner JAR J1, J2, and J3, respectively, associated with different access control policies. If some entities are allowed to access only one group of the photos, say J1, the outer JAR will just render the corresponding inner JAR to the entity based on the policy evaluation result.

Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item. We support two options:

- Pure Log. Its main task is to record every access to the data. The log files are used for pure auditing purpose.
- Access Log. It has two functions: logging actions and enforcing access control. In case an access request is denied, the JAR will record the time when the request is made. If the access request is granted, the JAR will additionally record the access information along with the duration for which the access is allowed.

Dependability of Logs

We aim to prevent the following two types of attacks. First, an attacker may try to evade the auditing mechanism by storing the JARs remotely, corrupting the JAR, or trying to prevent them from communicating with the user. Second, the attacker may try to compromise the JRE used to run the JAR files.

JARs Availability

To protect against attacks perpetrated on offline JARs, the CIA includes a log harmonizer which has two main responsibilities: to deal with copies of JARs and to recover corrupted logs.

Each log harmonizer is in charge of copies of logger components containing the same set of data items. The harmonizer is implemented as a JAR file. It does not contain the user's data items being audited, but consists of class files for both a server and a client processes to allow it to communicate with its logger components. The harmonizer stores error correction information sent from its logger components, as well as the user's IBE decryption key, to decrypt the log records and handle any duplicate records. Duplicate records result from copies of the user's data JARs. Since user's data are strongly coupled with the logger component in a data JAR file, the logger will be copied together with the user's data. Consequently, the new copy of the logger contains the old log records with respect to the usage of data in the original data JAR file. Such old log records are redundant and irrelevant to the new copy of the data. To present the data owner an integrated view, the harmonizer will merge log records from all copies of the data JARs by eliminating redundancy.

For recovering purposes, logger components are required to send error correction information to the harmonizer after writing each log record. Therefore, logger components always ping the harmonizer before they grant any access right. If the harmonizer is not reachable, the logger components will deny all access. In this way, the harmonizer helps prevent attacks which

attempt to keep the data JARs offline for unnoticed usage. If the attacker took the data JAR offline after the harmonizer was pinged, the harmonizer still has the error correction information about this access and will quickly notice the missing record.

In case of corruption of JAR files, the harmonizer will recover the logs with the aid of Reed-Solomon error correction code [4]. Specifically, each individual logging JAR, when created, contains a Reed-Solomon-based encoder.

Auditing Mechanism

To allow users to be timely and accurately informed about their data usage, our distributed logging mechanism is complemented by an innovative auditing mechanism. We support two complementary auditing modes: 1) push mode; 2) pull mode.

Push mode: In this mode, the logs are periodically pushed to the data owner (or auditor) by the harmonizer. The push action will be triggered by either type of the following two events: one is that the time elapses for a certain period according to the temporal timer inserted as part of the JAR file; the other is that the JAR file exceeds the size stipulated by the content owner at the time of creation. After the logs are sent to the data owner, the log files will be dumped, so as to free the space for future access logs. Along with the log files, the error correcting information for those logs is also dumped.

Pull mode: This mode allows auditors to retrieve the logs anytime when they want to check the recent access to their own data. The pull message consists simply of an FTP pull command, which can be issues from the command line. For naive users, a wizard comprising a batch file can be easily built. The request will be sent to the harmonizer, and the user will be informed of the data's locations and obtain an integrated copy of the authentic and sealed log file.

Algorithm for Push and pull Pure Log mode.

The pushing strategy is beneficial when there are a large number of accesses to the data within a short period of time. In this case, if the data are not pushed out frequently enough, the log file may become very large, which may increase cost of operations like copying data. The pushing mode may be preferred by data owners who are organizations and need to keep track of the data usage consistently over time. For such data owners, receiving the logs automatically can lighten the load of the data analyzers. The maximum size at which logs are pushed out is a parameter which can be easily configured while creating the logger component. The pull strategy is most needed when the data owner suspects some misuse of his data; the pull mode allows him to monitor the usage of his content immediately. A hybrid strategy can actually be implemented to benefit of the consistent information offered by pushing mode and the convenience of the pull mode.

- The algorithm presents logging and synchronization steps with the harmonizer in case of Pure Log.
- First, the algorithm checks whether the size of the JAR has exceeded a stipulated size or the normal time between two consecutive dumps has elapsed.
- The size and time threshold for a dump are specified by the data owner at the time of creation of the JAR.
- It checks whether the data owner has requested a dump of the log files. If none of these events has occurred, it proceeds to encrypt the record and write the error-correction information to the harmonizer.
- The communication with the harmonizer begins with a simple handshake. If no response is received, the log file records an error
- Once the handshake is completed, the communication with the harmonizer proceeds, using a TCP/IP protocol. If any of the aforementioned events (i.e., there is request of the log file, or the size or time exceeds the threshold) has occurred, the JAR simply dumps the log files and resets all the variables, to make space for new records.
- The Access Log checks whether the CSP accessing the log satisfies all the conditions specified in the policies pertaining to it. If the conditions are satisfied, access is granted; otherwise, access is denied. Irrespective of the access control outcome, the attempted access to the data in the JAR file will be logged.

For Example Ramu can specify that she wants to receive the log files once every week, as it will allow his to monitor the accesses to his files. Under this setting, once every week the JAR files will communicate with the harmonizer by pinging it. Once the ping is successful, the file transfer begins. On receiving the files, the harmonizer merges the logs and sends them to ramu. Besides receiving log information once every week, Ramu can also request the log file anytime when needed. In this case, he just need to send his pull request to the harmonizer which will then ping all the other JARs with the "pull" variable to 1. Once the message from the harmonizer is received, the JARs start transferring the log files back to the harmonizer.

Conclusion

In the cloud computing ensure that the trust of cloud from accountability. CIA framework accessible through the JAR files and data also encrypted format by using the Hierarchical identity based encryption. Similarly to the case of the design and realization of an IBE scheme, could not have a fully functional HIBE scheme. In addition the JAR file can authenticate. So, that it allows the powerful applications too many different mobile devices, information gathering capabilities very high and flexibility.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

REFERENCES

1. P. Ammann and S. Jajodia, "Distributed Timestamp Generation in Planar Lattice Networks," ACM Trans. Computer Systems, vol. 11,pp. 205-225, Aug. 1993.
2. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. ACM Conf. Computer and Comm. Security, pp. 598- 609, 2007.
3. E. Barka and A. Lakas, "Integrating Usage Control with SIP-Based Communications," J. Computer Systems, Networks, and Comm.,vol. 2008, pp. 1-8, 2008.
4. 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.
5. R. Bose and J. Frew, "Lineage Retrieval for Scientific Data Processing: A Survey," ACM Computing Surveys, vol. 37, pp. 1- 28, Mar. 2005.
6. 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.
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. OASIS Security Services Technical Committee, "Security Assertion Markup Language (saml) 2.0," http://www.oasis-open.org/committees/tc home.php?wg abbrev=security, 2012.
9. 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.
10. B. Crispo and G. Ruffo, "Reasoning about Accountability within Delegation," Proc. Third Int'l Conf. Information and Comm. Security (ICICS), pp. 251-260, 2001.

B.LAKSHMI TIRAPATAMMA was born in **guntur**, Andhra Pradesh, India. She received MCA from ANU Guntur,Andhra Pradesh,India. Presently, she is pursuing M.Tech in C.S.E from St.mary's group of institutions, Guntur,Andhra Pradesh, India. Her Research interest includes Cloud Computing,Computer Networks,and Data-warehousing.

Yenumula.Sankararao,
Assistant Professor in Computer Science And Engineering Department,
St.Marys' Group Of Institutions Guntur.