Distributed Accountability and Logging Mechanism for Data Sharing in the Cloud

K. Prabakaran, M. Viswanathan

***Abstract:** Cloud computing present an innovative technique to progress to their exploit and liberate replica for IT services base on the internet, by provided that for aggressively scalable and regularly virtualized resources because a service above the internet. It enables enormously scalable services toward be by no problem consumed above the internet on a desirable source. A most important characteristic of the cloud services to be facilitate user's data be usually process hazily in anonymous tackle that users do not hold or else control. Whereas enjoy the ability bring by this original maturing technology, we suggest an innovative truly decentralized within sequence dependability formation to maintain course of the actual observe of the user's data into the cloud. In fussy, we propose an object-centered shift to facilitate enable enclose our sorting mechanism mutually through user's data and policy. We influence the sorting mechanism toward together create a dynamic and nomadic object , also near make sure to several access to user's data determination to establish legalization along with mechanical sorting. To construct stronger user's control, we besides there spread audit mechanism. We offer broad audition study to facilitate to illustrate the superior association and triumph of the deliberate result.*

***Keywords-** Cloud computing, Object-centered, Sorting mechanism, Innovative technique.*

I. INTRODUCTION

Cloud computing promising computing model that enable well-situated along with on-demand accessing of network with the purpose of shared computing pool resources.

This allows data owners to move data to local computing also start choosing to host their data in the cloud. The cloud service providers offer SaS (Software-as-a-Service) to diminish the burden of huge local data storage to ease the maintenance cost by way of data storage outsourcing. CSP like Microsoft, Google, Amazon, and Yahoo are successfully dropped rates of available storage in an internet. To conquer this, we proposition a work of fiction approach, called Cloud Information Accountability (CIA). Unlike seclusion protection techniques that built on the hide-it-or-lose-it perception, in order accountability focus on maintenance the data usage apparent and visible. It provides end-to-end accountability in a extremely distributed craze and their pioneering features of CIA lies in its facility of maintain lightweight along with powerful accountability to combines aspect of usage control, authorization and authentication. Data owner that can be an organize generating sensitive data to stored in cloud. CSP manages data owners files to authorize and their clients who have right to access the data in remote. The Trusted-Third-party (TTP) authorized the users and data owners have communal mistrust relations with Cloud Service provider (CSP). The leakage towards on data must be secured the outsourced data private.

Manuscript received March 15, 2014.

K.Prabakaran, M.Tech-Scholar, Department of CSE, Vel Tech Dr.RR & Dr.SR Technical University, Tamil Nadu, India.

M.Viswanathan, PhD-Scholar, Department of CSE, Vel Tech Dr.RR & Dr.SR Technical University, Tamil Nadu, India.

To associates the accountability feature, we also develop two distinct modes for auditing: Push and Pull mode. The push mode refers to logs start on periodically sent to data owner or stakeholder even as pull mode refers to an approach whereby the user can recover the logs while needed. This issue leverage as well as extend the programmable potential of JAR files to automatically log the usage of user's data by some entity in cloud. Users send data with logging and control policies that they want to enforce, enclosed in JAR files, to CSP. The access of any data triggers an automated and authenticated logging mechanism. This type of enforcement refers "strong binding" since the policies and logging mechanism pass through the data. To copying this issue we providing the JARs with a central point of contact which forms a link between them and user, also it records the error correction information sent by the JARs that allows and monitor the loss of any logs from any of the JARs. Also JAR is not able to contact its central point the access that encloses data to deny. Our approach can handle personal identifiable information provided they are stored as image files that represent a common content end user and organization (The popularity of Flicker is proven).

Our main contributes are as follows:

- We intend a novel mechanical and enforceable logging mechanism in cloud.
- Decentralized server toward require any authentication or storage system.
- The architecture is a podium independent.
- The experiment conducts on real cloud test bed. It results efficiency, scalability and granularity of approach.

This paper is an extension of new contributions, sequentially to strengthen the dependability of our system in case of comprised JRE we combine obvious hashing and integrity check. We also reorganized the log records formation to provide additional guarantee of authenticity and integrity. After that we improve further potential attack scenario to widen the security analysis. Finally new experiments are reported and provided estimation through the system performance.

II. RELATED WORK

In this part, we initially assessment related works address the cloud privacy and security. Then
Discuss about Proof-Carrying Authentication (PCA) and Identity-Based Encryption (IBE).

2.1. Cloud Privacy and Security

The information housed on the cloud is often seen as valuable to individuals with malicious intent. There is a lot of personal information and potentially secure data and people store on their computers, and this information is now being transferred to the cloud. Pearson et al. have projected accountability mechanisms to address seclusion concern of end users and then develop a privacy manager. Hence the processing output is defocused by seclusion manager to retrieve the correct result. The author present a layered architecture for addressing the end-to-end trust management and accountability problem in federated systems, in that mainly leverage trust relationships for accountability, along with authentication and anomaly detection. Further, the monitoring and focuses on lower level monitoring of system resources. Crispo and Ruffo propose related to accountability in case of deletion. It is a balancing work does not control the information workflow in clouds. The researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of e-commerce. Lee and their colleagues proposing distributed approach to accountability. The notion of accountability policies is mainly focused on resource utilization and track of sub jobs proposed at aggregate computing nodes, quite than access control.

2.2. Proof-Carrying Authentication

In the interest of security, most computer systems restrict operations on resources like files and memory location to specific users, called access control. To the irrespective of exact setting, access control is usually engineered, that each operation on the controlled resource is intercepted by controlling program (called *reference monitor*) that allows the operation to succeed only if the calling program (called *requester*) has sufficient permissions to complete it. In order to take a decision, the reference monitor accomplishes two tasks: authentication and authentication. Accessing files and memory in local systems is relatively straightforward in authentication also a much harder in distributed scenarios. A proof is a textual and completely rigorous representation of logic reference and the requester provides the reference monitor a proof that is allowed access. The monitor verifies the proof is correct this does not require exploration and computation is straight forward. These approaches have been implemented in Grey systems at CMU and are called PIA. The PCA includes a higher order logic language that allows quantification over predicates, and focuses on access control for web services. The extent that helps maintaining safe, high concert, mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring. Mont et al. who proposed an approach for strongly coupling content with access control, using Identity-Based encryption (IBE) also leverage IBE techniques in different way. In additional work data provenance aim to guarantees data integrity through secure data provenance. Differently to propose data accountability, to monitor the data usage that ensures data access is tracked.

III. CLOUD INFORMATION ACCOUNTABILITY (CIA)

The CIA framework meets the design requirements is discussed in preceding section. The CIA framework proposed automated logging and distributed auditing of relevant access performed by an entity, carried out at any point of time at any CSP. It contains two major components: logger and harmonizer.

3.1. Major components:

The Components of CIA is logger and log harmonizer. The logger is strongly coupled with user data that downloaded when data are accessed and copied. The particular instance or copy of user data is responsible for logging access to that instance or copy. The log harmonizer forms central component that allows user to access log files. The logger requires only minimal support from server (e.g., JVM installed) in order to deployed. The highly distributed logging system meets first requirements in tight coupling between data and logger. The error correction information combine with encryption and authentication mechanism to provides a robust a reliable recovery mechanism. In this case, harmonizer sends the key to client in a secure exchange. It supports push and pull mode.

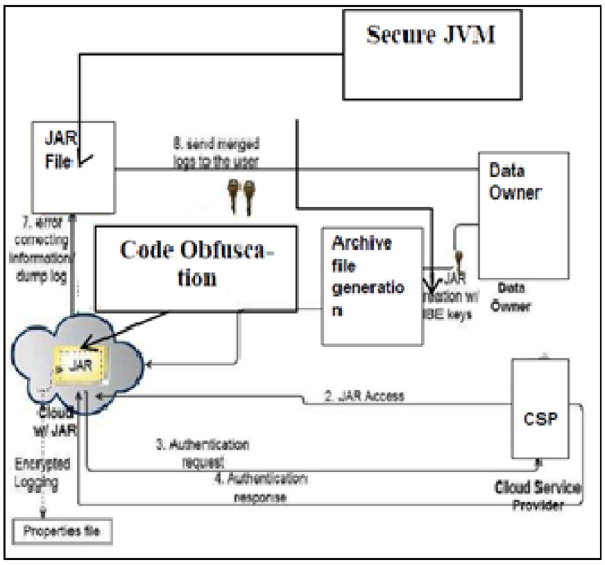


Fig.3.1.1: Overview of CIA framework.

The push mode log the file to pushed back to data owner periodically in automation. The pull mode is a demand approach where log file is obtained by data owner as often required. The multiple loggers for same data items will be merge by the log harmonizer before sending back to the data owner, it also responsible for log file corruption. To improve this performance logger and log harmonizer are both implemented as lightweight and portable JAR files, this implements automatic logging functions.

3.2. Data Flow:

The overall CIA framework, combining data, users, logger and harmonizer creates a pair of public and private keys based on IBE, which protect against most prevalent attacks. The JAR file includes set of access control specify how cloud servers, and possibly other stack holders are access the content. To authenticate the CSP to JAR using OpenSSL based certificates, wherein a trusted certificate authority certificates the CSP. Once the authentication succeeds, the CSP allow accessing the data enclosed in JAR. For each and every time accessing the logging, the JAR automatically generates log record, encrypt it using public key distributed by data owner and store it along with data. The unauthorized changes to encryption file prevents by attackers. The data owner use same key for all pair of JARs ores use different key pairs to divide a JARs. The encrypted log files can later be decrypted and the integrity is verified, this can be access by data owners or stake holders for audit with help of log harmonizer. This prevents various attacks such as detecting copies of user data with encryption mode; their logging mechanisms are neither automatic nor distributed. The federal system for logging require some data stayed on borders, that not suitable in cloud.

IV. DEPENDABILITY OF LOGS

In this part, we discuss to ensure the reliability of logs, in fussy we aim to prevent subsequent two types of attacks. Primary, attacker might try to dodge the auditing mechanism by store the JARs remotely, corrupting the JAR, or annoying to prevent them from communicating with the user. After that, the attacker may try to negotiation the JRE used to run the JAR files.

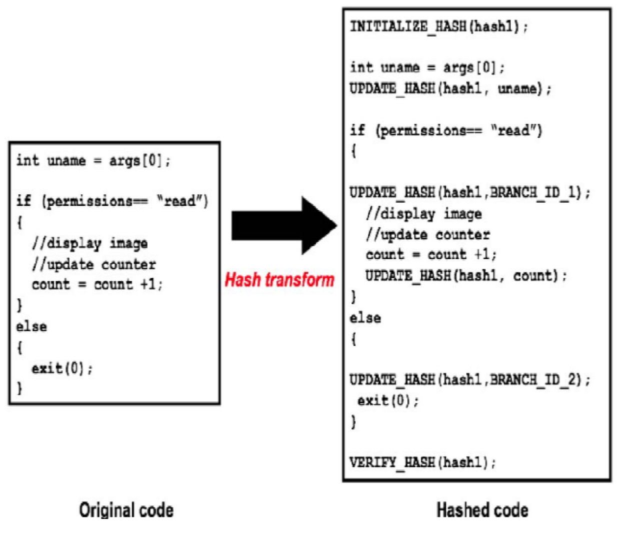
4.1: Log Correctness:

To verify the integrity of logger component consists two-step process:

- By launching the logger must repair along with every kind of access is given, to provide guarantee of integrity of the JRE.
- To calculate the hash values need to insert the hash codes, the program traces of the module being execute by logger module.

Once the logger component is launched it helps to detect modifications in JRE, and the useful to verify if the original code flow execution is altered.

The logger and log harmonizer work in tandem to carry out the integrity checks during runtime. The hash function is initialized at the beginning of program, this outcome variable is clear and there hash value is reorganized every time there is a variable assignment, branching, or looping.



```
int uname = args[0];

if (permissions== "read")
{
  //display image
  //update counter
  count = count +1;
}
else
{
  exit(0);
}
```

```
INITIALIZE_HASH(hash1);

int uname = args[0];
UPDATE_HASH(hash1, uname);

if (permissions== "read")
{

UPDATE_HASH(hash1,BRANCH_ID_1);
  //display image
  //update counter
  count = count +1;
  UPDATE_HASH(hash1, count);
}
else
{

UPDATE_HASH(hash1,BRANCH_ID_2);
 exit(0);
}

VERIFY_HASH(hash1);
```

Fig.4.1.1: Oblivious hashing applied to logger.

4.2: Log Retrieval for Push and Pull mode:

Pushing and pulling strategies have interesting tradeoffs. The pushing strategy is beneficial when there are a large number of accesses to data within a short period of time. The pull strategy is most needed when the data owner suspects some misuse of their data; the pull mode allows him to monitor the usage of their content immediately. Supporting both pushing and pulling modes helps protecting from some nontrivial attacks.

V. EXPERIMENTAL RESULTS

5.1: Log Creation Time:

The time to create a log file increases linearly with the size of log file. The time taken to create a log file where their entities continuously access the data, causing continuous logging

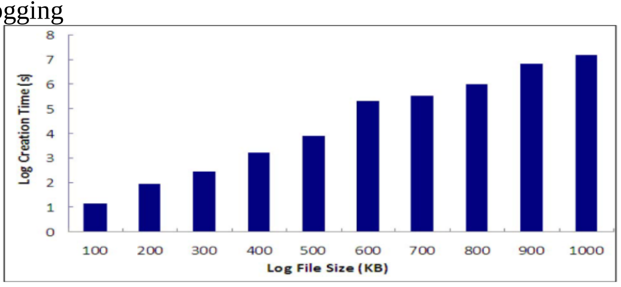


Fig.5.1.1: Time to create log files at unlike sizes.

Specifically, the time to create 100 KB file is about 114.5 ms time take to create 1 MB file averages at 731 ms. By the experiment amount of time is specified between dumps, keeping other variables like space constraints or network traffic in wits.

5.2: Authentication Time:

The time for authentication is too long, because accessing enclosed data at bottleneck. To assess this, the top node issue OpenSSl certificates for the computing nodes and we deliberate the total time for the OpenSSl authentication to be completed and the certificate revocation to be checked. For one access at a time, to find authentication time averages around 920 ms that proves not much overhead is added during the phase. Further the recital is enhanced by the caching certificates. When we consider the user actions obtaining SMALL certificates, it averages at 1.2 min's.

5.3: Log Merging Time:

To check the log harmonizer at bottleneck, we measure the amount of time to require merging log files. The exact number of records in common was random for repetition of the experimentation time was averaged over 10 repetitions.

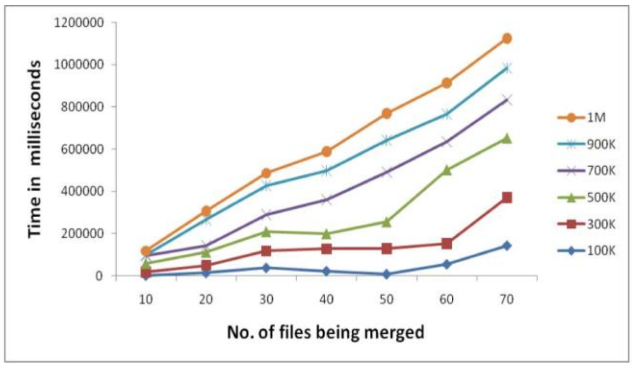


Fig.5.3.1: Time to merge log files.

We tested the time to merge up to 70 log files of 100 KB, 300 KB, 500 KB, 700 KB, 900 KB and 1024 KB each. The time increases almost linearly to the number of files and size of files, with the least time being taken for merging two 100 KB log files at 59 ms, while the time to merge 70 log files of 1 MB was 2.35 min's.

5.4: Size of the Data JAR files:

Finally, we probe whether a single logger, used to handle more than one file, results in storage overhead. We measure and the size of loggers by changeable the number and size of data items held by them. We tested the increase in size of the logger containing 10 content files of the same size as the file size increases. The size of logger grows from 3,500 to 4,035 KB when the size of content items changes from 200 KB to 1MB. Overall, due to the compression provided by JAR files, the size of the logger is dictated by the size of the largest files it contains. Observe that we knowingly did not include large log files, so as to focus on the overhead added by having multiple content files in a single JAR.

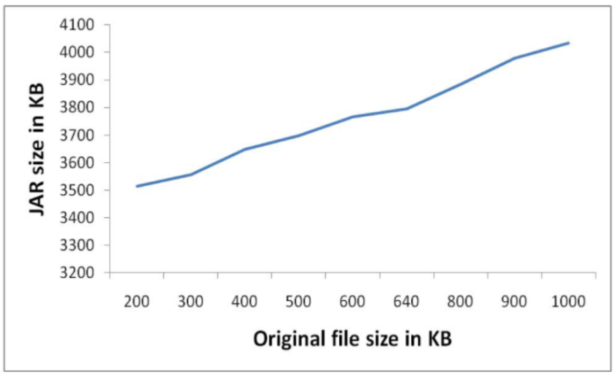


Fig.5.4.1: Size of the logger component.

VI. CONCLUSION

We introducing effective mechanism that automatically log any access to the data in the cloud together with an auditing mechanism. Data owner can audit his content on cloud, and make sure that content is safe. Apart from that we have enclosed PDP methodology to enhance the integrity of owner's data using this mechanism usage data is transparent. To developing a cloud in future store the data in enormous security mode to reduce log record generation by installing the JRE and JVM, to authenticate the JAR.

REFERENCES

1. Andrew W.Appel and Edward W.Felten, "Proof-Carrying Authentication. In G.Tsudik, editor, Proceedings of the 6th Conference on Computing and Communications Security, pages 52-62, Singapore, Nov 1999. ACM Press.
2. D.Boneh and M.K.Franklin, "Identity-Based Encryption from the Weil Pairing," Proc .Int'l Cryptography Conf. Advances in Cryptology, pp.213-229, 2001.
3. Hsio Ting Lin, Tzeng.W.G, "A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding," IEEE transactions on Parallel and Distributed systems, 2012.
4. J.H.Lin, R.L.Geiger, R.R.Smith, A.W.Chan and S.Wanchoo, "Method for Authentication a Java Archive (JAR) for Portable devices," US Patent 6, 766, 353, July 2004.
5. S.Pearson and A.Charlesworth, "Accountability as a Way Forward for Privacy Protection in Cloud," proc. First Int'l Conf. Cloud Computing, 2009.
6. S.Sundareswaran, A.Squicciarini and D.Lin, "Preventing Information Leakage from Indexing in the Cloud," Proc. IEEE Int'l Conf. Cloud Computing, 2010.
7. S.Sundareswaran, A.Squicciarini, D.Lin and S.Huang, "Promoting Distributed Accountability in the Cloud," Proc, IEEE Int'l Conf. Cloud Computing, 2011.
8. SmithaSundareswaran, Anna C.Squicciarini, Member, IEEE and Dan Lin, "Ensuring Distributed Accountability for Data Sharing in the Cloud," IEEE transactions on Dependable and Secure Computing, Vol9, No.4 Jul/Aug 2012.

AUTHORS POFILE

K. Prabakaran M.Tech-Scholar, Department of CSE, Vel Tech Dr.RR & Dr.SR Technical University, Tamil Nadu, India.

M.Viswanathan PhD-Scholar, Department of CSE, Vel Tech Dr.RR & Dr.SR Technical University, Tamil Nadu, India.