A New Distributed Accountability and Auditability for Data Storage in Cloud Computing

[1]Madhuri . C, [2]A.KrishnaChaitanya
[1]M.Tech (C.S.E),VCE, Hyderabad
[2]Associate Professor, I.T Dept ,VCE , Hyderabad

Abstract: Cloud computing presents a new way of using and delivering model for IT over internet to users by providing dynamic scalability and virtualized resources where users are hidden from actual processing and hosting their services in cloud. In cloud often data is outsourced, leading to a number of issues related to accountability and authentication by the cloud service provider (CSP) and make data usage be transparent and tractable to the users. To provide users to widely adopt the cloud without the threat of losing their important data In this paper, we propose that along with accountability, ensure integrity as per new Cloud Information Accountability and Auditability (CIAA) framework of outsourced data in cloud, and more secure than to existing system.

Index Terms: -- cloud computing, accountability, cloud service provider , auditability , outsourced data storage.

1 INTRODUCTION

Cloud computing is a promising business model to add-on the current use and delivery of IT services model based on the Internet, by providing for dynamically scalable with virtualized resources as a service over the Internet. There is more number of distinct commercial and individual cloud computing services, such as Amazon, Google, Microsoft, Yahoo, and Sales force [1].Details of these services are abstracted from the users who no longer need to be experts of technology infrastructure. Moreover, users may not aware of which machines is actually processing and hosting their data. While accessing the benefits of this new technology, users also worry about losing control of their own data.

These lacks of control over the data leading to a number of issues related to accountability and auditability, including the handling of personally identifiable information and audit the data. By users' concerns, we provide an effective mechanism for users to monitor the usage data in the cloud. To ensure users data according to conventional environments access control approach, developed for closed domains such as the databases and operating systems, or by using a centralized server in distributed environments, are not applicable, due to the following features characterizing cloud environments. 1) Data handling can be outsourced by the cloud service provider (CSP) and to other entities in the cloud to allot the tasks to others, and so on. 2) The entities are allowed to go in and out of 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.

To overcome the above problems, recently Yang tang et al. [2] proposed a FADE, a secure overlay cloud storage system that achieves fine-grained, policy-based access control on the user's data and assured file deletion. It also relates outsourced data files with file access policies and assuredly deletes files to make them unrecoverable to anyone upon

revocations of file access policies. However, policy based access control is week in third cloud storage.

Further, they are not considering the problem of data integrity auditability, which is the one of important security requirement for cloud data storage.

To address Accountability and Auditability of outsourced data in cloud. we propose a new Cloud Information Accountability and Auditability (CIAA) framework based on CIA[3] in this paper . One of the main innovative features of the CIAA framework lies in its ability of maintaining lightweight and powerful Accountability and Auditability that combines aspects of access control, usage control, authentication and spot checking. By means of the CIAA, data owners can track not only whether or not the service-level agreements are being achieved but also implement access and usage control rules as required. Associated with the accountability feature, we also develop auditability mechanism to verify the integrity of data in cloud. In this audit process, the verifier periodically challenges the cloud server for integrity of data. The design of the CIAA framework presents significant challenges, including uniquely identifying CSPs, ensuring the reliability of the log, adaptability of a highly decentralized infrastructure, etc. Our basic approach toward addressing these issues is to control 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. The users will send their data along with any policies such as access control policies and logging policies that they want to implement with this JAR files, by 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 "tight 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 up with this issue, we provide the JARs with a central point of contact which forms a link between log files 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. Currently, we focus on image files since images represent a very common content type for end users and organizations (as is proven by the popularity of Flickr et al[4]) and are increasingly hosted in the cloud as part of the storage services offered by the utility computing paradigm featured by cloud computing. Further, images often reveal social and personal habits of users, or used for archiving important files from organizations. In addition, our approach can handle personal identifiable information provided they are stored as image files (they contain an image of any textual content, for example, the SSN stored as a .jpg file).

We tested our CIAA framework in a cloud test bed, the Emu lab test bed[4] with Eucalyptus as middleware[5] Our experiments demonstrate the effectiveness scalability and granularity of our approach. In addition, we also provide a detailed security analysis and discuss the reliability and potency of our architecture in the face of various nontrivial attacks, launched by malicious users or due to compromised Java Running Environment (JRE).

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 lays out our problem statement. Section 4 presents our proposed Cloud Information Accountability framework, and Sections 5 and 6 descriaa

provide the detailed algorithms for automated logging mechanism and auditing approaches, respectively. Section 7 presents a security analysis of our framework, followed by an experimental study in Section 8. Finally, Section 9 concludes the paper and outlines future research directions.

2. RELATED WORK

Here, we first review the related works addressing the privacy and security issues in the cloud. Then, we discuss about adopting the similar techniques as our approach but serve for different purposes.

The Cryptographic protection on outsourced storage, recent studies proposed to protect outsourced storage via cryptographic techniques. Plutus [7] is a cryptographic storage system, which allows secure file sharing over untrusted file servers. Ateniese et al. [8] and Wang et al. [10] proposed an auditing system that verifies the integrity of outsourced data. Wang et al. [12] proposed a secure outsourced data access mechanism that supports changes in user access rights and outsourced data. However, all the above systems require new protocol support on the cloud infrastructure, and such additional functionalities may make deployment more challenging.

Next, The Secure solutions that are compatible with existing public cloud storage services have been proposed. Yun et al. [13] proposed a cryptographic file system that achieves the privacy and integrity guarantees for outsourced data using a universal-hash-based MAC tree. This prototype is a system that can interact with an untrusted storage server via a modified file system. Jungle Disk et al. [14] protect the privacy of outsourced data, and their implementation use Amazon S3 [1] as the storage backend service. For these we have Cumulus, focuses on making effective use of storage space while providing essential encryption on outsourced data, but such systems do not consider file assured deletion and auditability in their designs.

Access control is an One approach to apply outsourced data is by attribute-based encryption, which associates fine-grained attributes with data. ABE is first introduced in [27], in which attributes are associated with encrypted data. Goyal et al. [15] extended the idea to key-policy ABE, in which attributes are associated with private keys, and encrypted data can be decrypted only when a threshold of attributes are satisfied. Pirretti et al. [26] implement ABE and conduct empirical studies, and also point out.

We ensure ABE and conduct empirical studies, and also specified this access control. Nair et al. [17] have also considered a similar opinion on ABE algorithm, and also he seek to implement a fine-grained access control of files based on identity-based public key cryptography. Policy-based deletion follows the similar notion of ABE, in which data can be accessed only if the subsequent attributes (i.e., atomic policies in our case) are satisfied. However, policy-based deletion focuses on how to delete data, while ABE focuses on how to access data based on attributes. A major feature of ABE is of users concern, decryption keys of the associated attributes so that they can access files that satisfy the attributes, and hence accessible studies of ABE seek to guarantee that no two users will collide if they are tied with different sets of attributes. But in this policy-based deletion, since each policy is possessed by multiple users, to revoke a policy the centralized administrator to manage the revocation is required. Boldyreva et al. [8] combine ABE with attribute revocation, and both of the studies need the use of some centralized key server to manage the attributes and the corresponding keys (i.e., policy-based control keys in our case). In FADE, each policy is associated with two keys. One is the access key, which is

issued to users, and another is the control key, which is maintained by the key server. Both Access and control keys are required to decrypt a file. Thus, the main focus of their work is to evaluate the feasibility of our system via practical implementation.

In Assured deletion, we discuss about time-based deletion, there are several related systems, Keypad [20] protects data in theft-prone devices (e.g., laptops, USB sticks) by encrypting such data and maintaining keys in independent, centralized key servers, similar to FADE [2].In assured deletion, it deletes all data of a protected device upon requests of deletion, and does not consider fine-grained deletion as in FADE. Nasuni supported assured deletion in backup snapshots in March 2011 [16]. However, there is no formal study about their implementation methodologies and performance evaluation.

To attain such security goals, FADE is built upon a set of cryptographic key operations that are self-maintained by a quorum of key managers that are regulated by third-party cloud provider and works effortlessly on today's cloud storage services.it doesn't focuses on integrity which is important for security in cloud.

3. PROBLEM STATEMENT

3.1 System Model

We represent network architecture for cloud storage service architecture is illustrated in Fig. 1. Three different network entities used in architecture are defined as follows

User: an entity, who has data to be stored in the cloud and relies on the cloud storage and computation, can be either enterprise or individual customers.

Cloud Server (CS): an entity, which is managed by cloud service provider (CSP) to provide data storage service and has main storage space and computation resources

Third-Party Auditor: an optional TPA, who has knowledge and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

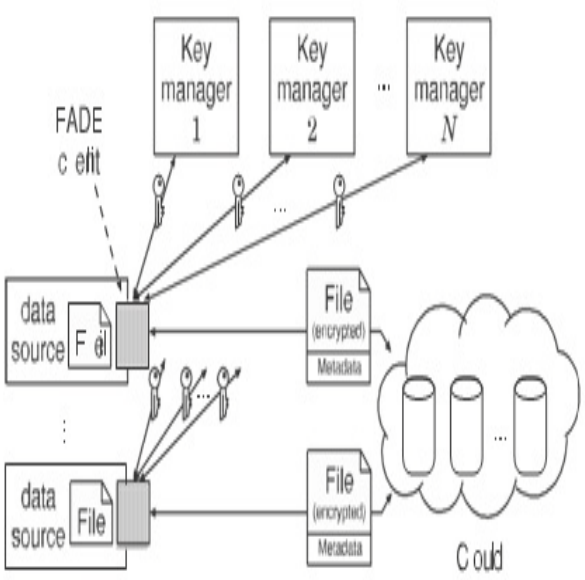


Fig.1. The FADE system architecture

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a Fig. 1. Cloud storage service architecture in which data stored in simultaneous, cooperated, and distributed manner. Data redundancy can be employed with a technique of erasure-correcting code to further tolerate errors or server crash as user's data grow in size and importance, later for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his desired data. In some cases, the user may need to perform block level operations on his data. The most common forms of these operations are that we are considering a block of data stored is to be update, delete, insert, and append. As users no longer possess their data locally, it is of critical point to ensure users

that their data are being accurately stored and maintained. That is, users should be equipped with security means so that they can make continuous accurate assurance (to enforce cloud storage service-level agreement) of their stored data even without the availability of local copies. In case if those users do not have time, feasibility, they can delegate the data auditing tasks to any optional trusted TPA of their respective choices. However, to securely establish such a TPA, any possible leakage of user's outsourced data toward TPA through the Auditing protocol should be prohibited.

In our model, we assume that the point-to-point communication channels between each cloud server and the user is authentic and reliable, which can be achieved actually with little overhead.

3.2 Adversary Model

From user's point of view, the adversary model has to capture all kinds of threats toward his cloud data integrity. Because in cloud data does not reside at local site but at CSP's address domain, these threats can arrive in two different sources: internal and external attacks. For internal attacks, a CSP can be self-interested, untrusted, and probably malicious. data that is desired cannot be moved or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors, Byzantine failures, and so on.

For external attacks, data integrity threats may come from outsiders who are beyond the control domain of CSP, for example, the economically motivated attackers. They may compromise a number of cloud data storage servers in different time intervals and subsequently be able to modify or delete users' data may be undetected by CSP. Therefore, its adversary is in our model having the capabilities, which captures both external and internal threats toward the cloud data integrity. Specifically the adversary is concerned in continuously corrupting the user's data files stored on individual servers. Once a server is comprised, an adversary can pollute the original data files.

By modifying or introducing its own fake data to prevent the original data from being retrieved by the user. This corresponds to the threats from external attacks. In the worst case scenario, the adversary can cooperate with all the storage servers so that he can intentionally modify the data files as long as they are internally consistent. In fact, this is equivalent to internal attack where all servers are assumed collision simultaneously from the early stages of application or service deployment to hide a data loss or corruption incident.

3.3 Design Goals

To ensure the Integrity and availability for cloud data storage under the aforementioned adversary model, we aim to design efficient mechanisms for dynamic data verification and operation and achieve the following goals:

1. Auditability: to ensure users that their data are indeed stored appropriately and kept integral all the time in the cloud.

2.Accountability: against Byzantine failures, malicious data modification and server colluding attacks, i.e., minimizing the effect brought by data errors or server failures.

3.4. Preliminaries and notations

- D - the data file to be stored in cloud, we assume that D can be denoted as matrix of 'm' equal sized data blocks, each consisting of '*l*' data blocks, these all data blocks belongs Galois Field GF (2^w) where w=8 or 16.

- $f_{key}(.)$- pseudo Random Function (PRF) indexed on some *key*, which is defined as
  $f : \{0,1\}^{*} \times key\text{-GF}(2^{w})$.
- π_{key} – pseudo Random Permutation (PRP) indexed under *key*, which is defined as
  $\pi : \{0,1\}^{\log 2(l)} \times key - \{0,1\}^{\log 2(l)}$.

4. Proposed Protocol

To ensure accountability and auditability of data storage in cloud computing, we propose a new CIAA framework adopted from CIA [3]. It is designed based on access control, usage control, authentication and spot checking.

Our scheme consists of two phases:

1) Accountability 2) Auditability

4.1 Accountability

In this section, we explain automated logging mechanism [3] and then present techniques. It contains two methods: a) Logger Structure b) Log record Generation

a) The Logger Structure

A logger section is a nested Java JAR file, which stores a user's data and corresponding log files. As shown in Fig. 2, our proposed JAR file consists of one outer JAR nested with one or more inner JARs. The main task of the outer JAR file is to handle validation of entities which want to access the data stored in the JAR file. In our framework, the user may not aware of exact CSPs that handle the data. Hence, authentication is specified according to the server's functionality rather than the server's URL or identity. For example, a policy may state that Server X is allowed to download the data if data is in storage server. As the outer JAR file may also have the access control functionality to implement the users' requirements, specified as Java policies, on data usage. A Java policy specifies permissions that are available for a specified piece of code in a Java application environment. The permissions are expressed in the Java policies are of File System Permissions. However, the user can specify the permissions in user-centric terms as opposed to the usual code-centric security accessible by Java, using Java Authentication and Authorization methods. Further, the outer JAR is also in charge of selecting the right inner JAR according to the users' identity who requested the data.

Example1: Suppose that users' photographs are classified into three categories according to the locations where actually the photos were taken. The three groups of photos are stored in three inner JAR J1, J2, and J3, correspondingly, related with different access control policies. If some entities are allowed to access only one group of the photos, say J1, the outer JAR will choose their respective inner JAR to the entity based on the policy evaluation result. Each inner JAR contains the encrypted data, class files to support log files retrieved and display enclosed data in an appropriate format, and a log file for every data encrypted item.

We support two options[3]:

Pure Log: Its task is to record each access to the data. The log files are used for pure auditing cause.

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 of request when it is made. If the access request is granted, the JAR will also record the access information along the period of time for which the access is allocated.

The two kinds of logging modules allow the user to implement definite access conditions either proactively (in case of Access Logs) or reactively (in case of Pure Logs). For example, services like billing

may need to use Pure Logs. The Access Logs will be needed for services which require enforcing service-level agreements such as limited visibility to some sensitive content at certain location.

To hold these functions, the inner JAR contains a class file for writing the log records, and another class file which corresponds with the log harmonizer, the encrypted data, the third class file for displaying or downloading the data (based on whether it's a Pure Log, or an Access Log), and the public key of the IBE key pair that is required for encrypting the log records. There are no secret keys that are ever stored in the system. The outer JAR may control one or more inner JARs, in addition to a class file for authenticate the servers or the users, another class file finding the right inner JAR, a third class file checks the JVM's validity using oblivious hashing. Further, a class file is used for running the GUI for user authentication and the Java Policy.

b) Log Record Generation

Log records are generated by the logger component. Logging occurs only when any access to the data in the JAR, and new log entries are appended in sequence order creation LR = $[r_1,r_n]$ each record r_i is encrypted individually and appended to the log file. Specifically, a log record takes the following forms: Here, r_i indicates that an entity identified by ID has performed an action Act on the user's data at time T at location Loc correspond to the checksum of the records preceding the recently inserted one, concatenated with the main content of the record itself. The checksum is computed using a collision-free hash function [10]. The component symbol denotes the signature of the record created by the server. If more than one file is handled by the same logger, an extra Obj ID field is added to each record. An example of log record for a single file is shown below. Suppose if a cloud service provider with ID Kronos, located in USA, read the image in a JAR file (but did not download it) at 4:52 pm on May 20, 2011. The corresponding log record is Kronos, View, 2011-05-29 16:52:30, USA, 45rftT024g, r94gm30130ffi.The location is converted from the IP address for improved readability.

To guarantee the integrity of the log records, we verify the access time, locations and actions. In particular, the time of access is determined using the Network Time Protocol (NTP) [22] to avoid restraint the correct time by a malicious entity. The location of the CSP can be determined by using IP address. The JAR can perform an IP lookup and use the series of the IP address to find the most possible location of the CSP. More advanced techniques for determining location can also are used [16] also, if a trusted time stamp management infrastructure can be set up or controlled, it can be used to record the time stamp in the accountability log [23]. The most critical part is to record the actions on the users' data. In the current system, we carry four types of actions, i.e., view, download, timed_access, and Location-based access. For each action, we present a specific method to correct the record or to implement it depending on the type of the logging mechanism, which are elaborated as follows:

View: The entity (e.g., the cloud service provider) is read only data but is not allowed to save a raw copy of logs anywhere permanently. For this type of action, the Pure Log will simply write a log record about the accessing of data, while the Access Logs will enforce the action through the enclosed access control module .It is to remind that the data are encrypted and stored in the inner JAR. When there is a view-only access request, the inner JAR will decrypt the data on the fly and create a temporary decrypted file. The decrypted file will then be displayed to the entity using the Java application viewer in case if file is displayed to a individual users.

Further, to prevent from the use of some screen capture software, the data will be hidden when the application viewer screen go out of focus. The content is displayed using the headless mode in Java on the command line when it is accessible to a CSP.

Download: The entity is allowed to save a raw copy of the data and the entity does not have control over that copy, neither to log records about accessing of that copy. If Pure Log is adopted, the user's data will be directly downloadable in a pure form using a link. When a user clicks this download link, the JAR file associated with the data will decrypt that data and provide that to the user in raw form. In case of Access Logs, the entire JAR file is provided to the use.

Timed_access: This action is combined with view-only and accessibility and it indicate that data is available only for a certain period of time. The Pure log will just record the access starting time and its duration, while the Access Log will enforce that the access is allowed only for certain period of time. The duration of the access is allowed to calculate by with the use of the NTP. To implement the limit on the duration, the Access Log records the starting time using the NTP, and uses a timer to stop the access.

Location -based_access. In this case, the Pure Log will record the locations of the users. The Access Log will verify the location for each of user's access. The access is granted and the data are made available only to user sited locations specified by the data owner.

4.2Auditability

Once authentication is verified, now, we have to audit the integrity of data stored in cloud with help of third party auditor (TPA).The third party auditor (TPA), who has knowledge and capabilities that cloud users may not possess and trusted to assess the cloud storage security service on behalf of the user upon request. The CSP providing the cloud data storage based services, for their own benefits the CS might neglect to keep or intentionally delete rarely accessed data files which belong to ordinary cloud users. Besides, the CS may also decide to hide the data corruptions caused by server hacks or Byzantine failures to maintain reputation.

We assume the TPA, who is in the business of auditing, is reliable and independent, and thus has no incentive to collude with either the CS or the users during the auditing process. The TPA should be able to efficiently audit the cloud data storage without local copy of data and without bringing in additional on-line burden to cloud users [10].

The verification starts from the TPA by sending a authenticate challenge to the cloud storage service provider (CSP), here it computes the proof of verification and sends it back to the TPA. After verifying the proof, the TPA sends the result to the client. The detailed descriptions this verification process is given in protocol [10]

5. SECURITY ANALYSIS

We now study the possible attacks to our CIAA framework[3]. Our analysis is based on a semi trusted adversary model by assuming that a server does not release user master keys to untrusted parties, while the attacker may try to learn extra information from the log files and detect the data modifications on user files. We suppose that hackers may have sufficient Java programming skills to disassemble a JAR file and prior information about our CIAA architecture , assume that the JVM is not corrupted, and how to ensure that our assumption hold correct.

5.1 Copying Attack

The most instinctive attack is that the attacker copies entire JAR files. The attacker may assume that doing so allows accessing the data in the JAR file without being noticed by the data owner. However,

such attack will be detected by our auditing mechanism. Recall that every JAR file is required to send log records to the harmonizer will send the logs to data owners periodically. If attackers move copies of JARs to places where the harmonizer cannot connect, copy files will immediately go inaccessible. Thus, the logger component provides more transparency than Conventional log files encryption; it make data owner to detect if attacker has tried to create the copies of a JAR, and harmonizer make them offline or inaccessible.

5.2 Disassembling Attack

Another possible attack is to disassemble the JAR file of the logs and attempt to remove useful information out of it or destroy the log records and it's the most serious attack to our architecture. The cryptographic schemes are useful to preserve the integrity and confidentiality of the logs. If these JAR files are disassembled, then attacker is in controls the public IBE key used for encrypting the log files, and *.class files so, the attacker has to depend on learning the private key or subverting the encryption to read the log records.

To compromise the confidentiality of the log files, the attacker may try to identify which encrypted log records correspond to his actions by mounting a chosen plaintext security attacker to gain some pairs of encrypted log records and plain texts. However, the adoption of the Weil Pairing algorithm ensures that the CIAA framework has both chosen cipher text security and chosen plaintext security in the random oracle model [10]. From the disassembled JAR files, the attackers are not able to directly view the access control policies either, since the original source code is not included in the JAR files. If the attacker wants to infer access control policies, the only possible way is through analyzing the log file and it is very hard to accomplish. Attackers will not be able to write fake records to log files without going undetected, since they will need to sign with a valid key and the chain of hashes will not match.

The Reed-Solomon encoding[10] used to create the redundancy for the log files, the log harmonizer can easily detect a corrupted record or log file. Finally, the attacker ma y tries to modify the Java Class loader in the JARs is in order to subvert the class files when they are being loaded. This attack is prevented by the sealing techniques offered by Java. Sealing ensures that all packages within the JAR file come from the same source code [22].

Even if an attacker can read from it by disassembling it—he cannot "reassemble" it with modified packages. In case the attacker guesses or learns the data owner's key from somewhere, all the JAR files using the same key will be compromised. Thus, using different IBE key pairs for different JAR files will be more secure and prevent such attack.

5.3 Man-in-the-Middle At tack

An attacker may interrupt messages during the certification of a service provider with the certificate authority, and reply it in order to masquerade as a legal service provider. There are two steps where the attacker can reply the messages. 1) the actual service provider has totally disconnected and at the end of certificate authority session. But attacks this will not succeed since the certificate usually has a time stamp which will become obsolete at the time of reuse and 2) the actual service provider is disconnected but the session is not finished, so the attacker may try to renegotiate the connection and this attack will also fail since renegotiation is banned in the latest version of Open SSL and cryptographic checks are added.

5.4 Compromised JVM Attack

An attacker may try to compromise the JVM, so has to quickly detect and correct these issues, we discussed in Section 4.2 how to integrate oblivious hashing to guarantee the correctness of the JRE [24] OH adds hash code to capture t he computation on results of each instruction and computes the oblivious hash value as the computation proceeds.

These two techniques allow for a quick detection of errors due to malicious JVM, therefore mitigating the risk of running subverted JARs to further strengthen the solution, one can extend OH usage to guarantee the correctness of the class files loaded by the JVM.

5.5. Security Strength against Data Corruptions

In our framework, servers are required to operate only on specified blocks in each challenge-response protocol execution. We will show that this "sampling" strategy on selected blocks instead of all can greatly reduce the computational overhead on the CSP, while maintaining high detection probability for data corruption.

Suppose servers are misbehaving due to the possible compromise or Byzantine failure. Then we assume the adversary modifies the data blocks in z blocks out of the *l* rows in the data file. Let c be the number of different blocks for which the user asks for checking in a challenge. Let X is a discrete random variable that is defined to be the number of blocks chosen by the user that are matches the blocks modified by the adversary. So, the detection probability that at least one of the blocks picked by the user matches one of the blocks modified by the adversary is: $P_{X=}1-(l-z/l)^{c}$

6. PERFORMANCE ANALYSIS

In this section, we discuss about settings of the test environment and performance of our system.

We tested our CIAA framework by setting up a small cloud, using the Emu lab test bed [5]. In particular, the test environment consists of several Open SSL-enabled servers: one head node which is the certificate authority, and several computing nodes. Each of the servers is installed with Eucalyptus [6]. Eucalyptus is an open source cloud execution for Linux-based systems. It is loosely based on Amazon EC2[26], therefore bringing the powerful functionalities of Amazon EC2 into the open source domain. We used Linux-based servers running Fedora 10 OS. Each server has a 64-bit Intel Quad Core Xeon E5530 processor, 4 GB RAM, and a 500 GB Hard Drive. Each of the servers is equipped to run the Open JDK runtime environment with IcedTea6 1.8.2.

6.1. Experimental Results

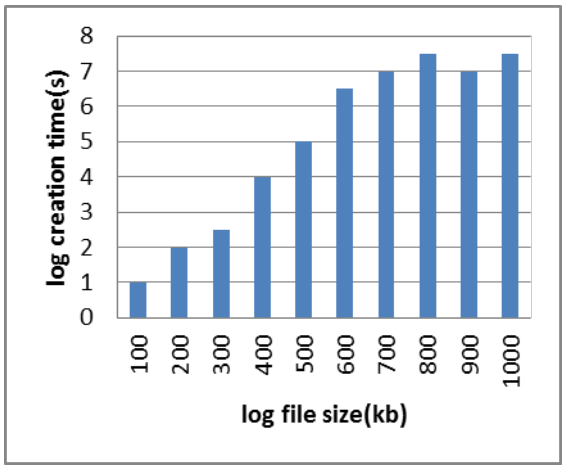
In the experiments, we first create a log file and time to create it and then measure the overhead in the system. With respect to time, the overhead can occur at three points: 1.authentication, 2.encryption of a log, and 3.merging of the logs. And also with storage overhead, we observe that our architecture is very lightweight, the data to be stored are given by the actual files and the associated logs then JAR act as a compressor of the files that it handles. In particular, as introduced in Section 3, multiple files can be handled by the same logger component. To this extent, we investigate whether a single logger component, used to handle more than one file, results in storage overhead.

a) Log Creation Time

Here we are concerned with finding out the time taken to create a log file when there are entities continuously accessing the data, causing continuous logging. Results are shown in Fig. 5.

It is unexpected to see that the time to create a log file increases linearly with the size of the log file. Distinctively, the time to create a 100 Kb file is about 114.5 ms while the time to create a 1 MB file averages at 731ms,as the baseline, one can

decide the amount of time to be specified between dumps, making other variables like space constraints or network traffic while experimenting.



b) Authentication Time

The other overhead can occur is during the authentic of a CSP .if authentication takes long time, it may become a bottleneck for accessing the enclosed data and to evaluate this, the head node issued Open SSL[25] certificates for the computing nodes .

We evaluate the total time for the Open SSL authentication to be completed and checking the certification by considering one access at the time, we find that the authentication time averages around 920ms which proves that not much overhead is added during this phase. It checks for each access thereby performance can be further improved by caching the certificates.

The time for authenticating an end user is about the same as we consider only the actions by the JAR viz., and obtain a SAML certificate to evaluate it because both the Open SSL and the SAML certificates are handled in a same fashion by the JAR. Then we consider the user actions (i.e., submitting his username to the JAR), it averages at 1.2 minutes.

c) Time Taken to Perform Logging

In this experiment measure the average time taken to grant an access plus the time to write the corresponding log record and performance of logging mechanism and it's time for granting any access to the data items in a JAR file includes the time to evaluate and implement the appropriate policies and to locate the requested data .

In the experiment, we let multiple servers continuously access the same data JAR file for a minute and recorded the number of log records generated. Each access is just a read only request and hence the time for executing the action is negligible and result, the average time to log an action is about 10 seconds, which includes the time taken by a user to double click the JAR or by a server to run the script to open the JAR and also measured the log encryption time which is about 300 ms (per record) and is apparently unrelated from the log size.

d) Size of the Data JAR Files

Finally, we examine whether each logger, used to handle more than one file, results in storage overhead.

To measure the size of the loggers (JARs) by varying the number and size of data items held is tested the increase in size of the logger containing 10 content files (i.e., images) which increases 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 1 MB and the size of the logger is dictated by the size of the largest files it contains as provided by JAR files

e) Overhead of Integrity Checking

To calculate the time overhead added by the hash codes, we only measure the time taken for each hash function and if time is found to average around 7ms then number of hash commands varies based on the size of the code, and it does not change with the content but with the number of hash commands stay constant.

7. CONCLUSION

We propose new Cloud Information Accountability and Auditability (CIAA) framework for ensuring auditing, accountability and integrity to data stored in cloud. It will be more transparent and secure than to existing system. Moreover, one of the main features of our work is that it enables the data owner to auditability data that were made without his knowledge.

REFERENCES

[1]. H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A Case for Cloud Storage Diversity," Proc. ACM First ACM Symp.Cloud Computing (SoCC), 2010.

[2]. Y. Tang, P.P.C. Lee, J.C.S. Lui, and R. Perlman, "FADE: Secure Overlay Cloud Storage with File Assured Deletion", IEEE Trans. On DEPEN DABLE AND SECURE COMPUTING, VOL. 9, NO. 6, NOVEMBER /DECEMB ER 2012.

[3]. Smitha S, Anna C. S. "Ensuring Distributed Accountability for Data Sharing in t he Cloud" , IEEE TRANSA CTIONS ON DEPEN DABLE AND SECURE COMPUTING, VOL. 9, NO. 4, JU LY/AUGUST 2012

[4]. Flickr, http://www.flickr.com/, 2012.

[5]. Emulab Network Emulation Testbed, www.emulab.net, 2012

[6]. Eucalyptus Systems, http://www.eucalyptus.com/, 2012.

[7]. M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc.Second USENIX Conf. File and Storage Technologies, 2003.

[8]. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M.Zaharia, "A View of Cloud Computing." Comm. ACM, vol. 53,no. 4, pp. 50-58, Apr. 2010.

[9]. G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. (SecureComm), 2008.

[10]. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc.IEEE INFOCOM, Mar. 2010.

[11]. J . Bethencourt , A . Sah a i, a nd B. Waters, " Ciphertext- Policy At tribute-Based Encryption," Proc. I EEE Sy mp. Security and Privacy, May 2006.

[12]. W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and Efficient Access to Outsourced Data," Proc. ACM Workshop Cloud Comput-ing Security (CCSW), Nov. 2009

[13]. A. Yun, C. Shi, and Y. Kim , "On Protecting Integrity and Confidentiality of Cryptographic File System for Outsourced Storage," Proc. ACM Workshop Cloud Computing Security. (ASIACCS), Apr. 2010.

[14]. P. Gutmann, "Secure Deletion of Data from Magnetic and Solid-State Memory," Proc. Sixth USENIX Security Symp. Focusing on Applications of Cryptography, 1996.

[15]. JungleDisk, http://www.jungledisk.com/, 2010.

[16]. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,"Proc. 13th ACM Conf. Computer and Comm. Security (CCS), 2006.

[17]. M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure Attribute-Based Systems," Proc. 13th ACM Conf. Computer and Comm. Security (CCS), 2006.

[18]. S. Nair, M.T. Dashti, B. Crispo, and A.S. Tanenbaum, "A Hybrid PKI-IBC Based Ephemerizer System," Int'l Federation for Informa-tion Processing, vol. 232, pp. 241-252, 2007.

[19]. A. Boldyreva, V. Goyal, and V. Kumar, "Identity-Based Encryp-tion with Efficient Revocation," Proc. 15th ACM Conf. Computer and Comm. Security (CCS), 2008.

[20]. R. Geambasu, J.P. John, S.D. Gribble, T. Kohno, and H.M. Levy,
"Keypad: Auditing File System for Mobile Devices," Proc. Sixth
Conf. Computer Systems (EuroSys), Apr. 2011.

[21]. Nasuni, "Nasuni Announces New Snapshot Retention Function-ality in Nasuni Filer; Enables Fail-Safe File Deletion in the Cl oud ," htt p:/ /ww w. nas u ni. c om /n ew s/ p res s- rel ea se s/ na su ni -an no u nc es - n ew - s na p sh ot - r et e nt i on - f u nc t i on al it y - i n- na su n i- f il e r -enables-fail-safe-file-deletion-in-the-clo ud/, Mar. 2011.

[22]. NTP: The Network Time Protocol, http://www.ntp.org/, 2012.

[23]. S. Pearso n and A. Charle sworth, "Account abil ity as a Way Forward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing, 2009.
[24]. Trusted Java Virtual Machine IBM, http://www.almaden.ibm.com/cs/projects/jvm/, 2012.
[25]. OpenSSL, http://www.openssl.org/, 2010.
[26]. Amazon S3, http://aws.amazon.com/s3, 2010.
[27]. S. Kamara and K. Lauter, "Cryptographic Cloud Storage," Proc.14th Int'l Conf. Financial Cryptography and Data Security, 2010.
[28]. LibAWS++, http://aws.28msec.com/, 2010.
[29]. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography. CRC Press, Oct. 1996.
[30].
[31]. B. Schneier, "File Deletion," http://www.schneier.com/blog/archives/2009/09/file_deletion.html, Sept. 2009.
[32]. A. Shamir, "How to Share a Secret," Comm. ACM, vol. 22, no. 11,pp. 612-613, Nov. 1979.
[33]. W. Stallings, Cryptography and Network Security. Prentice Hall,2006.