International Journal of Electronics Communication and Computer Engineering
Volume 4, Issue (6) NCRTCST-2013, ISSN 2249–071X

National Conference on Recent Trends in Computer Science and Technology (NCRTCST)-2013

Integrity Verification & Distributed Accountability in High Performance Distributed Clouds

B. Rajani, K. Nagasindhu, K. Saikrishna
Department of Computer Science, CMRCET, Hyderabad

***Abstract* – Cloud computing enables highly scalable services to be easily consumed over the Internet on an as-needed basis. A major feature of the cloud services is that users' data are usually processed remotely in unknown machines that users do not own or operate. While enjoying the convenience brought by this new emerging technology, users' fears of losing control of their own data (particularly, financial and health data) can become a significant barrier to the wide adoption of cloud services. To address this problem, in this paper, we propose a novel highly decentralized information accountability framework to keep track of the actual usage of the users' data in the cloud and for the data sharing in which procedural and technical solutions are co-designed to demonstrate accountability by the various researchers to resolving privacy and security risks within the cloud. In particular, we propose an object-centered approach that enables enclosing our logging mechanism together with users' data and policies. This paper presents a review on new way to supplement the current consumption and delivery model for IT services based on the Internet, by providing for dynamically scalable framework and often virtualized resources as a service over the Internet. By the time there are a number of notable commercial and individual cloud computing services, including Google, Microsoft and Yahoo. We leverage the JAR programmable capabilities to both create a dynamic and traveling object, and to ensure that any access to users' data will trigger authentication and automated logging local to the JARs. In Addition we propose a dynamic audit service for verifying the integrity of un trusted and outsourced storage. Our audit service, constructed based on the techniques, fragment structure, random sampling and index-hash table, can support provable updates to outsourced data, and timely abnormal detection. We also propose an efficient approach based on probabilistic query and periodic verification for improving the performance of audit services.**

***Keywords* – Cloud Computing, Accountability, Data Sharing, Integrity Verification.**

I. INTRODUCTION

In recent years, the emerging cloud-computing paradigm is rapidly gaining momentum as an alternative to traditional information technology. Cloud computing provides a scalability environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. One fundamental aspect of this paradigm shifting is that data are being centralized and outsourced into clouds. This kind of outsourced storage services in clouds have become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data. The cloud storage service (CSS) relieves the burden of storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to users since their data or archives are stored into an uncertain storage pool outside the enterprises. These security risks come from the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices.

However, they are still susceptible to security threats both from outside and inside the cloud for the benefits of their possession, there exist various motivations for cloud ser- vice providers (CSP) to behave unfaithfully toward the cloud users furthermore, the dispute occasionally suffers from the lack of trust on CSP. Consequently, their behaviors may not be known by the cloud users, even if this dispute may result from the users' own improper operations . Therefore, it is necessary for cloud service providers to offer an efficient audit ser- vice to check the integrity and availability of the stored data. Traditional cryptographic technologies for data integrity and availability, based on hash functions and signature schemes, cannot work on the outsourced data without a local copy of data. In addition, it is not a practical solution for data validation by downloading them due to the expensive transaction, especially for large-size files. Moreover, the solutions to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users. Therefore, it is crucial to realize public auditability for CSS, so that data owners may resort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and data assurance in clouds.

In this paper, we introduce a dynamic audit service for integrity verification of un trusted and outsourced storages. Our audit system, based on a novel audit system architecture, can support dynamic data operations and timely abnormal detection with the help of several effective techniques, such as fragment structure, random sampling, and index-hash table. Furthermore, we propose an efficient approach based on probabilistic query and periodic verification for improving the performance of audit services. A proof of concept prototype is also implemented to evaluate the feasibility and viability of our proposed approaches. Our experimental results not only

All copyrights Reserved by NCRTCST-2013, Departments of CSE
CMR College of Engineering and Technology, Kandlakoya(V), Medchal Road, Hyderabad, India.

validate the effectiveness of our approaches, but also show our system has a lower computation cost, as well as a shorter extra storage for integrity verification.

The traditional cryptographic technologies for data integrity and availability, based on Hash functions and signature schemes [4, 11, 13], cannot work on the outsourced data without a local copy of data. In addition, it is not a practical solution for data validation by downloading them due to the expensive communications, especially for large size files. Moreover, the ability to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users. Therefore, it is crucial to realize public auditability for CSS, so that data owners may resort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and credibility in clouds.

To implement public auditability, the notions of proof of retrievability (POR) and provable data possession (PDP) have been proposed by some researchers. Their approach was based on a probabilistic proof technique for a storage provider to prove that clients' data remain intact. For ease of use, some POR/PDP schemes work on a publicly verifiable way, so that anyone can use the verification protocol to prove the availability of the stored data. Hence, this provides us an effective approach to accommodate the requirements from public auditability. POR/PDP schemes evolved around an untrusted storage offer a publicly accessible remote interface to check the tremendous amount of data. There exist some solutions for audit services on outsourced data. For example the proposed efficient method on content comparability for outsourced database, but it wasn't suited for irregular data. Wang et al. [8] also provided a similar architecture for public audit services. To support their architecture, a public audit scheme was proposed with privacy-preserving property. However, lack of rigorous performance analysis for constructed audit system greatly affects the practical application of this scheme. For instance, in this scheme an outsourced file is directly split into n blocks, and then each block generates a verification tag. In order to maintain security, the length of block must be equal to the size of cryptosystem, that is, 160-bit=20- Bytes. This means that 1M-Bytes file is split into 50,000 blocks and generates 50,000 tags [7], and the storage of tags is at least M-Bytes. It is clearly inefficient to build an audit system based on this scheme. To address such a problem, a fragment technique is introduced in this paper to improve performance and reduce extra storage. Another major concern is the security issue of dynamic data operations for public audit services. In clouds, one of the core design principles is to provide dynamic scalability for various applications. This means that remotely stored data might be not only accessed but also dynamically updated by the clients, for instance, through block operations such as modification, deletion and insertion. However, these operations may raise security issues in most of existing schemes, e.g., the forgery of the verification metadata (called as tags) generated by data owners and the leakage of the user's secret key. Hence, it is crucial to develop a more efficient and secure mechanism for dynamic audit services, in which possible adversary's advantage through dynamic data operations should be prohibited. Note that this paper only addresses the problems of integrity checking and auditing. Other security services, such as user authentication and data encryption, are orthogonal to and compatible with audit services.

II. Architecture and Techniques

We introduce an audit system architecture for outsourced data in clouds as shown in Figure 1. In this architecture, we consider a data storage service involving four entities: data owner (DO), who has a large amount of data to be stored in the cloud; cloud service provider (CSP), who provides data storage service and has enough storage space and computation resources; third party auditor (TPA), who has capabilities to manage or monitor the outsourced data under the delegation of data owner; and authorized applications (AA), who have the right to access and manipulate stored data. Finally, application users can enjoy various cloud application services via these authorized applications. We assume the TPA is reliable and independent through the following audit functions: TPA should be able to make regular checks on the integrity and availability of the delegated data at appropriate intervals;

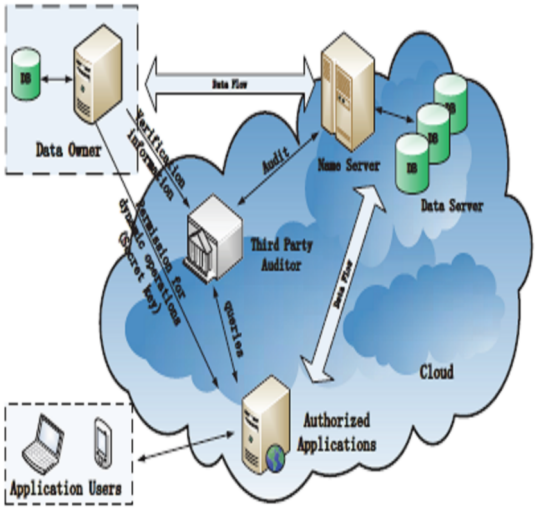

Fig.1. The Audit System Architecture

TPA should be able to organize, manage, and maintain the outsourced data instead of data owners, and support the dynamic data operations for authorized applications; and TPA should be able to take the evidences for disputes about the inconsistency of data in terms of authentic

All copyrights Reserved by NCRTCST-2013, Departments of CSE

records for all data operations. To realize these functions, our audit service is comprised of three processes:

Tag Generation: the client (data owner) uses the secret key sk to pre-process a file, which consists of a collection of n blocks, generates a set of public verification parameters (PVP) and index-hash table (IHT) that are stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy.

Periodic Sampling Audit: by using an interactive proof protocol of retrievability, TPA (or other applications) issues a "Random Sampling" challenge to audit the integrity and availability of outsourced data in terms of the verification information (involves PVP and IHT) stored in TPA.

Audit for Dynamic Operations: An authorized applications, who hold data owner's secret key sk, can manipulate the outsourced data and update the associated indexhash table (IHT) stored in TPA.

The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the authorized applications and forge the valid audit records. In general, the authorized applications should be cloud application services inside clouds for various application purposes, but they must be specifically authorized by data owners for manipulating the outsourced data. Since the acceptable operations require that the authorized applications must present authentication information for TPA, any unauthorized modifications for data will be detected in audit processes or verification processes. Based on this kind of strong authorization-verification mechanism, we neither assume that CSP is trust to guarantee the security

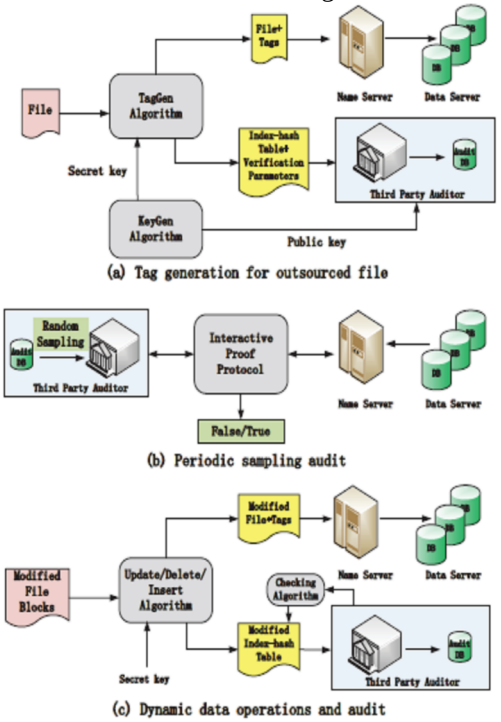

Fig.2. The Process of Audit System

of stored data, nor assume that a date owner has the capability to collect the evidence of CSP's faults after errors have been found.

The ultimate goal of this audit infrastructure is to enhance the credibility of cloud storage services, but not to increase data owner's burden and overheads. For this purpose, TPA should be constructed in clouds and maintained by a cloud storage provider (CSP). In order to ensure the trust and security, TPA must be secure enough to resist malicious attacks, and it also should be strictly controlled to prevent unauthorized access even for internal members in clouds. A more practical way is that TPA in clouds should be mandated by a trusted third party (TTP). This mechanism not only improves the performance of audit services, but also provides the data owner with a maximum access transparency. This means that data owners are entitled to utilize the audit service without further costs besides storing a secret-key and some secret information.

The above processes involve some procedures: KeyGen, TagGen, Update, Delete, Insert algorithms, as well as an interactive proof protocol of retrievability . In order to improve security and performance, we make use of following techniques to construct corresponding algorithms and protocols.

3.1 Fragment Structure and Secure Tags

To maximize the storage efficiency and audit performance, a general fragment structure is introduced into our audit system for outsourced storage. An instance for this framework which is used in this scheme is showed in Figure 3: an outsourced file F is split into n blocks {m1,m2, ….,mn}, and each block mi is split into s sectors {mi,1,mi,2, ….,mi,s}. The fragment framework consists of n block-tag pair (mi, i), where i is a signature tag of block mi generated by some secretsT = (T1,T2,….Ts). Finally, these block-tag pairs are stored in CSP and the encryption of the secrets _ (called as PVP) are in TTP. Although this fragment structure is simple and straightforward, but the file is split into n×s sectors and each block (s sectors) corresponds to a tag, so that the storage of signature tags can be reduced with increase of s. Hence, this structure can reduce extra storage for tags and improve the audit performance. There exist some schemes to convergence s blocks to generate a secure signature tag, e.g., MAC-based, ECC or RSA schemes [1, 7]. These schemes, built from collision-resistance signatures (see Appendix A) and the random oracle model, have higher scalability, performance and security.

3.2 Periodic Sampling Audit

In contract with "whole" checking, random "sampling" checking greatly reduces the workload of audit services, while still achieve an effective detection of misbehavior. Thus, the probabilistic audit on sampling checking is preferable to realize the abnormality detection in a timely manner, as well as rationally allocate resources. The fragment structure shown in Figure 3 can provide the

All copyrights Reserved by NCRTCST-2013, Departments of CSE

support of probabilistic audit as well: given a random chosen challenge (or query) Q = {(i, vi)}i I , where I is a subset of the block indices and vi is a random coefficient, an efficient algorithm is used to produce a constant-size response (μ1, μ2, …, μs, ′), where μi comes from all {mk,i, vk}k I and _′ is from all {k, vk}k I . Generally, this algorithm relies on homomorphic properties to aggregate data and tags into a constant size response, which minimizes network communication. Since the single sampling checking may overlook a very small number of data abnormality, we propose a periodic sampling approach to audit outsourcing data, which is called as Periodic Sampling Audit. In this way, the audit activities are efficiently scheduled in an audit period, and a TPA needs merely access small portions of file to perform audit in each activity. Therefore, this method can detect the exceptions in time, and reduce the sampling numbers in each audit.

3.3 Index-Hash Table

In order to support dynamic data operations, we introduce a simple index-hash table (IHT) to record the changes of file blocks, as well as generate the hash value of block in the verification process. The structure of our index-hash table is similar to that of file block allocation table in file systems. Generally, the index-hash table consists of serial number, block number, version number, random integer, and so on . Different from the common index table, we must assure that all records in this kind of table differ from one another to prevent the forgery of data blocks and tags. In addition to record data changes, each record in table is used to generate a unique Hash value, which in turn is used for the construction of signature tag I by the secret key sk. This kind of relationship between Xi and i must be cryptographic secure, and we can make use of it to design our verification protocol and algorithms are depicted in figures.

Although the index-hash table may increases the complexity of an audit system, it provides the strongest assurance to monitor the behavior of untrusted CSP, as well as efficient evidence of computer forensics, due to the reason that anyone cannot forge the valid Xi (in TPA) and (in CSP) without the secret key sk. In practical applications, the designer should consider that the index-hash table is kept into the virtualization infrastructure of cloud-based storage services.

IV. DISTRIBUTED ACCOUNTABILITY

It is essential to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example, users need to be able to ensure that their data are handled according to the service- level agreements made at the time they sign on for services in the cloud. Conventional access control approaches developed for closed domains such as databases and operating systems, or approaches using a centralized server in distributed environments, are not suitable, due to the following features characterizing cloud environments. First, data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and theses entities can also delegate the tasks to others, and so on. Second, entities are allowed to join and leave the cloud in a flexible manner. As a result, data handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments.

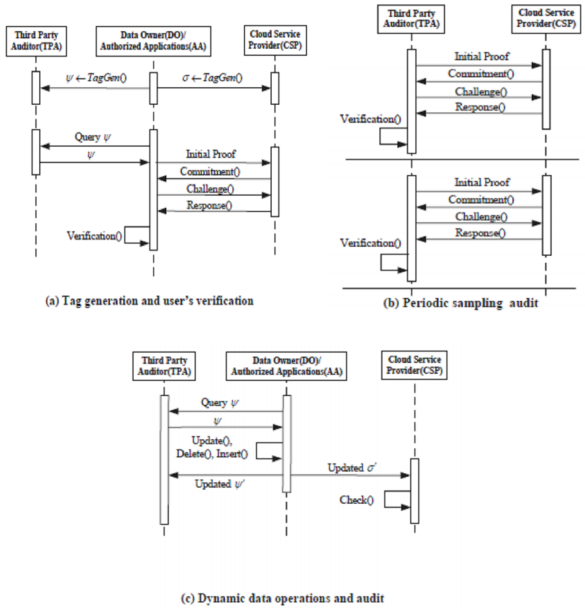

Fig.3. The Workflow of Audit System

To overcome the above problems, we propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, information account- ability focuses on keeping the data usage transparent and trackable. Our proposed CIA framework provides end-to- end accountability in a highly distributed fashion. One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful account- ability that combines aspects of access control, usage control and authentication. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed. Associated with the accountability feature, we also develop two distinct modes for auditing: push mode and pull mode. The push mode refers to logs being periodically sent to the data owner or stakeholder while the pull mode refers to an alternative approach whereby the user (or another authorized party) can retrieve the logs as needed.

The design of the CIA framework presents substantial challenges, including uniquely identifying CSPs, ensuring the reliability of the log, adapting to a highly decentralized

All copyrights Reserved by NCRTCST-2013, Departments of CSE
CMR College of Engineering and Technology, Kandlakoya(V), Medchal Road, Hyderabad, India.

infrastructure, etc. Our basic approach toward addressing these issues is to leverage and extend the programmable capability of JAR (Java ARchives) files to automatically log the usage of the users' data by any entity in the cloud. Users will send their data along with any policies such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Any access to the data will trigger an automated and authenticated logging mechanism local to the JARs. We refer to this type of enforcement as "strong binding" since the policies and the logging mechanism travel with the data. This strong binding exists even when copies of the JARs are created; thus, the user will have control over his data at any location. Such decentralized logging mechanism meets the dynamic nature of the cloud but also imposes challenges on ensuring the integrity of the logging. To cope with this issue, we provide the JARs with a central point of contact which forms a link between them and the user. It records the error correction information sent by the JARs, which allows it to monitor the loss of any logs from any of the JARs. Moreover, if a JAR is not able to contact its central point, any access to its enclosed data will be denied. 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 [14]) 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 are 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).

In addition, we also provide a detailed security analysis and discuss the reliability and strength of our architecture in the face of various nontrivial attacks, launched by malicious users or due to compromised Java Running Environment (JRE).

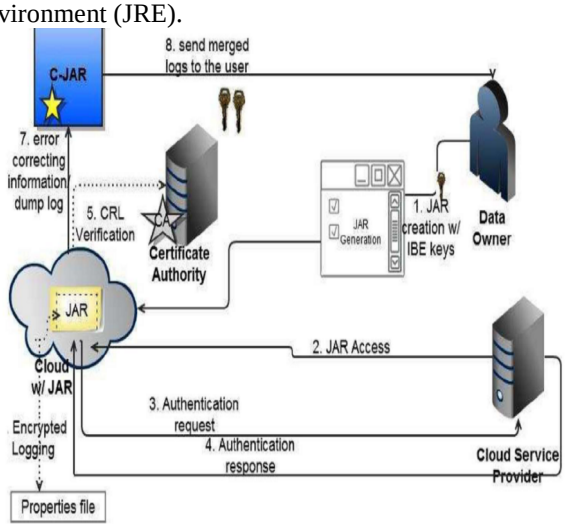

Fig.4. Framework of Cloud Information Accountability

In summary, our main contributions are as follows: We propose a novel automatic and enforceable logging mechanism in the cloud. To our knowledge, this is the first time a systematic approach to data accountability through the novel usage of JAR files is proposed. Our proposed architecture is platform independent and highly decentralized, in that it does not require any dedicated authentication or storage system in place. We go beyond traditional access control in that we provide a certain degree of usage control for the protected data after these are delivered to the receiver.

The results demonstrate the efficiency, scalability, and granularity of our approach. We also provide a detailed security analysis and discuss the reliability and strength of our architecture. This paper is an extension of our previous conference paper. We have made the following new contributions. First, we integrated integrity checks and oblivious hashing (OH) technique to our system in order to strengthen the dependability of our system in case of compromised JRE. We also updated the log records structure to provide additional guarantees of integrity and authenticity. Second, we extended the security analysis to cover more possible attack scenarios. Third, we report the results of new experiments and provide a thorough evaluation of the system performance. Fourth, we have added a detailed discussion on related works to prepare readers with a better understanding of background knowledge. Finally, we have improved the presentation by adding more examples and illustration graphs.

4.1 Cloud Privacy and Security

Cloud computing has raised a range of important privacy and security issues. Such issues are due to the fact that, in the cloud, users' data and applications reside-at least for a certain amount of time-on the cloud cluster which is owned and maintained by a third party. Concerns arise since in the cloud it is not always clear to individuals why their personal information is requested or how it will be used or passed on to other parties. To date, little work has been done in this space, in particular with respect to accountability. Pearson et al. have proposed accountability mechanisms to address privacy concerns of end users and then develop a privacy manager. Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is deobfuscated by the privacy manager to reveal the correct result. However, the privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed. In [7], the authors present a layered architecture for addressing the end-to-end trust management and accountability problem in federated systems. The authors' focus is very different from ours, in that they mainly leverage trust relationships for account- ability, along with authentication and anomaly detection. Further, their solution requires third-party services to complete the

All copyrights Reserved by NCRTCST-2013, Departments of CSE
CMR College of Engineering and Technology, Kandlakoya(V), Medchal Road, Hyderabad, India.

monitoring and focuses on lower level monitoring of system resources.

Researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of electronic commerce [10], A representative work in this area is given by [9]. The authors propose the usage of policies attached to the data and present a logic for accountability data in distributed settings. Similarly, Jagadeesan et al. recently proposed a logic for designing accountability-based distributed systems. In [10], Crispo and Ruffo proposed an interesting approach related to accountability in case of delegation. Delegation is complementary to our work, in that we do not aim at controlling the information workflow in the clouds. In a summary, all these works stay at a theoretical level and do not include any algorithm for tasks like mandatory logging.

To the best of our knowledge, the only work proposing a distributed approach to accountability is from Lee and colleagues. The authors have proposed an agent-based system specific to grid computing. Distributed jobs, along with the resource consumption at local machines are tracked by static software agents. The notion of account-ability policies in is related to ours, but it is mainly focused on resource consumption and on tracking of subjobs processed at multiple computing nodes, rather than access control. In order to track the actual usage of the data, we aim to develop novel logging and auditing techniques which satisfy the following requirements:

1. The logging should be decentralized in order to adapt to the dynamic nature of the cloud. More specifically, log files should be tightly bounded with the corresponding data being controlled, and require minimal infrastructural support from any server.
2. Every access to the user's data should be correctly and automatically logged. This requires integrated techniques to authenticate the entity who accesses the data, verify, and record the actual operations on the data as well as the time that the data have been accessed.
3. Log files should be reliable and tamper proof to avoid illegal insertion, deletion, and modification by malicious parties. Recovery mechanisms are also desirable to restore damaged log files caused by technical problems.
4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data. More importantly, log files should be retrievable anytime by their data owners when needed regardless the location where the files are stored.
5. The proposed technique should not intrusively monitor data recipients' systems, nor it should introduce heavy communication and computation overhead, which otherwise will hinder its feasibility and adoption in practice.

4.2 Cloud Information Accountability

In this section, we present an overview of the Cloud Information Accountability framework and discuss how the CIA framework meets the design requirements discussed in the previous section. The Cloud Information Accountability framework pro- posed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It has two major components: logger and log harmonizer. There are two major components of the CIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the user access to the log files.

The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is down- loaded by user X. The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement. Furthermore, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, the logger is also responsible for generating the error correction information for each log record and send the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement. The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. In this case, the harmonizer sends the key to the client in a secure key exchange.

It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is

All copyrights Reserved by NCRTCST-2013, Departments of CSE
CMR College of Engineering and Technology, Kandlakoya(V), Medchal Road, Hyderabad, India.

obtained by the data owner as often as requested. These two modes allow us to satisfy the aforementioned fourth design requirement. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner. The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable JAR files. The JAR file implementation provides automatic logging functions, which meets the second design requirement.

The overall CIA framework, combining data, users, logger and harmonizer is sketched in Fig. 1. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption [4] (step 1 in Fig. 1). This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture as described in Section 7. Using the generated key, the user will create a logger component which is a JAR file, to store its data items.

The JAR file includes a set of simple access control rules specifying whether and how the cloud servers, and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. To authenticate the CSP to the JAR (steps 3-5 in Fig. 1), we use OpenSSL- based certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication [8], wherein a trusted identity provider issues certificates verifying the user's identity based on his username.

Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data. The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption. To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity verified. They can be accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer.

As discussed our proposed framework prevents various attacks such as detecting illegal copies of users' data. Note that our work is different from traditional logging methods which use encryption to protect log files. With only encryption, their logging mechanisms are neither automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud.

V. Conclusion and Future Research

We proposed Integrity Verification and innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

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.

[11] Y. Chen et al., "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive," Proc. Int'l Workshop Information Hiding, F. Petitcolas, ed., pp. 400-414, 2003.

All copyrights Reserved by NCRTCST-2013, Departments of CSE
CMR College of Engineering and Technology, Kandlakoya(V), Medchal Road, Hyderabad, India.

[12] S. Etalle and W.H. Winsborough, "A Posteriori Compliance Control," SACMAT '07: Proc. 12th ACM Symp. Access Control Models and Technologies, pp. 11-20, 2007.

[13] X. Feng, Z. Ni, Z. Shao, and Y. Guo, "An Open Framework for Foundational Proof-Carrying Code," Proc. ACM SIGPLAN Int'l Workshop Types in Languages Design and Implementation, pp. 67-78, 2007.

[14] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," Proc. Seventh Conf. File and Storage Technologies, pp. 1-14, 2009.

All copyrights Reserved by NCRTCST-2013, Departments of CSE
CMR College of Engineering and Technology, Kandlakoya(V), Medchal Road, Hyderabad, India.