International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 5, May - 2013

RDC Based Decentralized Data Storage In Cloud Computing

Jijin Soman[1], M. Premkumar[2], K. Ambika[3]
[1][2]Pervasive Computing Technology, [3]Asst. Prof. Computer Science and Engineering
BIT Campus, Anna University, Tiruchirappalli

Abstract

Cloud computing is an internet based computing where the resources are delivered as if it were an entity. Cloud service providers provide services for users to access their services from anywhere using internet. Users can thereby reduce the expenditure by converting capital expenditure into operational expenditure. But users' fear of losing control over their data leads to a prompt towards the transparent usage of their data in the cloud. Also checking the integrity of data stored remotely on un-trusted cloud servers has emerged as a critical issue. The accountability and auditing based framework in the cloud provides a user centric solution which monitor usage of user's data in the cloud. The resulting enhanced cloud information accountability framework is a highly distributed, powerful and lightweight framework featured with transparent auditing and remote integrity checking mechanism. The programmable capability of JAR file is leveraged to create a dynamic and travelling object for keeping track of usage of data. The push-pull auditing mechanism is provided to strengthen users control over their data in the cloud. A dynamic and remote integrity checking mechanism is also provided to periodically check integrity of outsourced data stored in cloud server.

1. Introduction

Cloud computing is a compilation of existing technologies and techniques, packaged within a new infrastructure archetype that offers improved scalability, business agility, elasticity, reduced management costs, faster startup time and just-in-time availability of resources. Cloud can be rapidly deployed with low startup costs or capital investments. It provides an on demand self-service where service costs are measured based on usage or subscription. The characteristics such as ubiquitous network access, location independent resource pooling, multi-tenant sharing of services or resources and rapid elasticity makes cloud computing more convenient for the users who access the cloud services.

Along with these conveniences users also facing some issues related cloud computing. Clients have no idea or control over what happens inside a cloud. They even don know in which machine their data are stored and which entity is handling their data. Even if the cloud provider is honest, it can have malicious system administrative, who can tamper with the VMs and violate confidentiality and integrity. Clouds are still subject to traditional data confidentiality, availability, privacy issues and integrity, plus some additional attacks.

Most security problems stem from Loss of control, Lack of trust and Multi-tenancy. These problems exist mainly in third party management models. Consumer's loss of control since his data, applications, resources are located with service provider. Also user identity management is handled by and security policies, user access control rules and enforcement are managed by the CSP. Consumer relies on the Cloud Service Provider to ensure data security and privacy, resource availability and monitoring and repairing of services or resources.

IJERT

Lack of trust arises since cloud relies on third party management schemes. Since tenants share a pool of resources and have opposing goals there may arise conflicts among them due to their difference in interests. The issues mostly need to solve are, can tenants get along together and 'play nicely'?, if they can't, can we isolate them?, and how to provide separation between tenants?

The solution to the aforementioned issues is a transparent framework which monitors the usage of user's data in the cloud. The closed environment approaches mainly developed for centralized system or data base won't adapt with cloud environment. This is because of two reasons such as data outsourcing and the nature of entities in cloud. The data is outsourced from cloud service providers to other entities. The entities can leave or join the cloud whenever in a flexible manner. So the traditional approaches can't handle the complex task delegation chain efficiently. CSP is beneficial and can concentrate on the core business due to data outsourcing. But outsourcing makes the sensitive data of user out of control of the cloud service provider. This causes a security risk in the integrity and confidentiality of users' sensitive data in the cloud.

Since data is outsourced into public cloud it should be protect from unauthorised access. Confidentiality can be maintained by using cryptographic algorithms. The access control and authorization policies can be included to avoid unauthorized access. A remote integrity checking mechanism and a push-pull auditing mechanism can strengthen user's control over their data in the cloud.

2. Related Works

Storing user's data in a third party's cloud server causes serious concern over data confidentiality. Data confidentiality can be protected by general encryption schemes. But only using conventional security measures, the issues in cloud environment cannot be handled. Accountability is an alternative to the traditional encryption schemes. P.T. Jaeger et al [5] explore nature of cloud computing, policy issues and problems of information policy. Authors in [5] examine the policy issues such as privacy, reliability, security, regulation, and access that arise in respond to rapid technological changes.

In the development of trust during human interaction, accountability plays an important role. Accountability acts as an alternative to traditional security algorithms. R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, in [6] gave an idea about how to develop foundations for distributed accountability systems.

Pearson and Charlesworth [13] propose a combined approach of procedural and technical solutions for demonstrating accountability to provide a solution to privacy and security issues within the cloud. Maintaining privacy while outsourcing data or using virtualization are seldom possible using conventional security measures in cloud computing. So Pearson and Charlesworth [13] propose an accountability concept that addresses privacy and security issues in the cloud. The idea is to ensuring protection of data by enforcing commitments to responsible data handling.

Data security is crucial when storing data in a third-party storage space. D. Boneh and M.K. Franklin [2] explained a pairing based cryptographic system in their paper. IBE is a public key cryptography where encryption is done using any string which is publicly known. IBE is followed in our paper for providing security to the data files. IBE is composed of four sub algorithms namely Setup, Extract, Encrypt and Decrypt [2]. A. Pretschner, et al in [12] explained about the requirements needed for usage control.

Q. Wang, et al [16] narrates about the need of a Third Party Auditor (TPA) to verify the integrity of user's data in the cloud. Remote data integrity checking is critical in cloud computing to verify integrity of users' data in the remote server. Public verifiability of dynamic data can be done using their approach. Ateniese, Giuseppe, et al [1] proposes a novel model for verifying integrity checking and robust auditing remotely using PDP and FEC techniques. This approach is adopting in

IJERT

our paper to strengthen the data integrity. Client can challenge upon their data in remote server to provide a proof of possession. Client pre-computes metadata for each block of data and store data in data centers along with these metadata.

3. Enhanced CIA Framework

Enhanced Cloud Information Accountability Framework is an extension to the novel Cloud Information Accountability Framework. CIA is a highly distributed lightweight framework that provides an end to end accountability [15]. This framework had developed for solving some issues like users fear of losing control over their data in the cloud. Users fear includes confidentiality, availability, privacy and security issues and integrity.

Confidentiality: Users' fear of losing control over their data in the cloud includes whether their sensitive data remain confidential or whether the cloud compromises the leakage of the client data. User cannot ensure whether the cloud provider itself is honest and won't peek into the data or not.

Integrity: User may fear whether the cloud provider is doing the computations correctly or not. Users may need to ensure that the cloud provider really stored their data without tampering with it.

Availability: When the provider is attacked in a Denial of Service attack whether the critical systems go down at the client or users data will be available without get tampered. Users also fear about whether their data will be safe when cloud provider goes out of business.

Auditability: Entity outside the organization now stores and computes data since data is outsourced to other entities for cloud owners beneficial. These entities can join or leave the cloud in a flexible manner. The data's can be either inside or outside the particular organization. It is difficult to audit data held outside organization in a cloud.

Security: Security is one of the most difficult tasks to implement in cloud computing. The attacks that can happen in the application side and in the hardware components make the security implementation more difficult. Entity outside the organization now stores and computes data, and so attackers can target the communication link between cloud provider and client.

This framework scenario is designed in such a way that it can overcome the issues that are commonly arising in distributed data sharing in the cloud. Access control, usage control and authentication policies are combined in this framework to provide a combined and efficient accountability in the cloud storage. To strengthen the control over users' data in the cloud an end to end auditing using push pull mechanism is provided. End to end accountability is ensured by implementing policies such as access control policy, authentication policy and automated logging. Distributed auditing mechanism fulfils the accountability by strengthen users control over their data in the cloud. To verify the integrity of their data in remote server, the owner can challenge for the proof of possession using a combined PDP and FEC mechanism. The techniques used in this framework are described below.

IJERT

3.1. Identity Based Encryption

IBE is a public-key encryption system in which an arbitrary string can be used as the public key. Any identity provided by the receiver is used as the public key to encrypt the data that is sent by the sender. Identity-Based Encryption (IBE) dramatically simplifies the process of securing sensitive communications. IBE is working as depicted in Figure 1.

Alice encrypts the email using Bob's e-mail address, "bob@b.com", as the public key. Upon receiving the message, Bob can contact the key server. The key server communicates with a directory or other external authentication source to authenticate Bob's identity and establish any other policy elements. After authenticating Bob, the key server then returns the private key, for decrypting the message. This private key can be used to decrypt all future messages received by Bob. Identity- based encryption scheme is

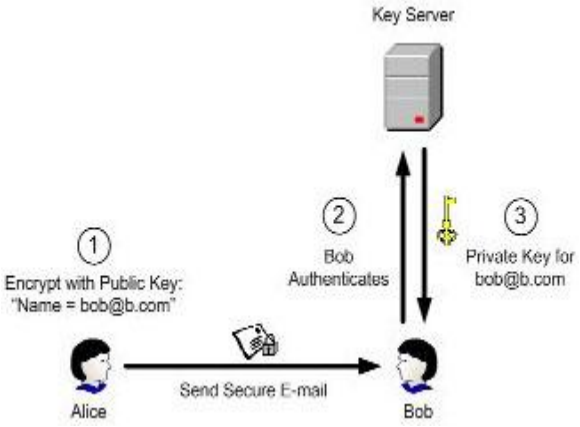

Figure 1. IBE Encryption Scheme.

specified by four randomized algorithms such as Setup, Extract, Encrypt and Decrypt.

Setup: Input a security parameter k and returns system parameters and master-key. Description of a finite message space M and a finite cipher text space C are the system parameters. System parameters are publicly known, while the master-key is generated and known to the Private Key Generator (PKG).

Extract: Input the master-key, a security parameter, and an arbitrary ID, and output a private key d. A private key from the given public key is extracted in Extract algorithm. The private decryption key d is derived from the arbitrary string ID that is used as a public key.

Encrypt: Takes security parameter, ID, and plain text and returns a cipher text.

Decrypt: Takes security parameter, cipher text and the private key d and return plain text. Decrypt a cipher text if have private key for identity. Public key are arbitrary string from system identities.

Chosen cipher text security: Chosen cipher text security is the standard approach of security for a public key encryption scheme. Hence, it is natural to require that an IBE scheme also satisfy this strong mode of security.

3.2. Automated Logging

A log record is automatically generated each time an access to the data is done by any of the entity. The generated log record is stored in a log file after encrypting it using the public key distributed by the data owner. The entity that accesses the data in the cloud sign the record for strengthen the security. Automated logging is done by extending the programmable capability of JAR file. The data is enclosed in a nested JAR file namely logger, along with authentication policy, access control policy and automated logging as in Figure 2.

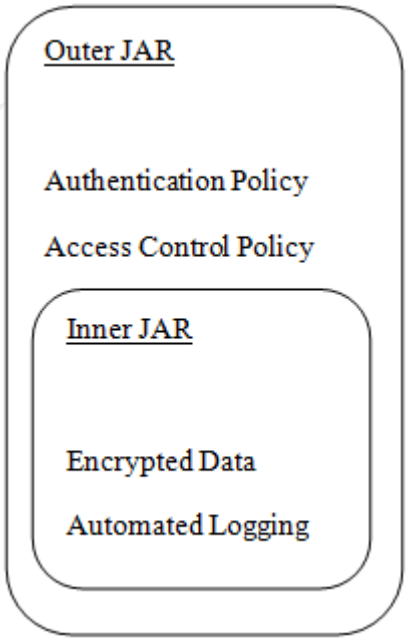

Figure 2. Logger.

Each log record r_i generated by logger components is individually uploaded into the log file after encryption. The generated log record has the form mentioned bellow.

r_i=<ID, Act, T, Loc, h((ID, Act, T, Loc)|r_i-1|…|r_1), sig>.

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. The component h((ID, Act, T, Loc)|r_i-1|…|r_1)

IJERT

corresponds to the checksum of the records preceding the newly inserted one. Act has one of the following four values: view, download, timed-access, and Location-based-access.

Auditing

Enhanced cloud information accountability provides a distributed end to end auditing algorithm for getting log files regarding the data owner's data from cloud server and end user. Push mode auditing and pull mode auditing are the two modes of auditing mechanisms.

The log files that are created when any of the entities accessing the data owners' data are periodically send to them in push mode auditing. The log files are pushed to the data owner in two situations. The first one is whenever the time lapse, which is set at the time of creation of logger by the data owner, exceeds and the second situation is whenever the log file is full.

The data owner can retrieve information about their data at any time on demand in the pull mode auditing. Whenever the user decides to check the recent and current access details the user can send a pull command.

3.3.1. Push-Pull Auditing Algorithm

A combined design of push mode and pull mode is push-pull auditing. Whenever the data owners want to audit they can pull the log file from both data user and cloud server. Whenever the time lapse that is set for periodic pushing exceeds or the log file is full, the log file that contain the records from the beginning to the current is pushed to the data owner and all information in the log file is erased. The log file information is send back to the data owner from both the data user and cloud server assuming that the cloud server cannot be trusted. The push-pull auditing algorithm is in Figure 3.

```
pull=0
rec=<ID, Act, T, Loc>
lsize=sizeof(log)
if((lsize<size)&&(pull==0)) then
    log=log+encry(rec)
    if(Act==Download|| TimedAccess||
    LocationBasedAccess) then
        if  LogHarmonizer is alive then
            push(encry(rec))
        else Exit(1)  end if
    end if
end if
if((lsize>=size)||(pull!=0)) then
    if LogHarmonizer is alive then
        push(log)
        (reset the parameters)
        log= NULL
        pull=0
    else Exit ()
    end if
end if
```

Figure 3. Push-Pull Auditing Algorithm.

3.4. Remote Data Checking

Remote Data Checking audits status of massive data and verifies the correctness of data object on un-trusted cloud storage. Verifying integrity or content for identifying damage is crucial to repair if any. Retrieving data to the owner side will cause I/O burden on server and wasting network traffic. So checking integrity remotely has more advantage. User couldn't trust the cloud service provider. Charging for terabytes and store

IJERT

gigabytes, discarding un-accessed data based on statistical reports, keeping fewer replicas than promised, hiding data loss for keeping institutional reputation and errors that are unnoticed by service provider were some of the reasons why user couldn't trust cloud service providers. RDC supports outsourced data storage. Provable Data Possession and other RDC mechanism can provide a secure and efficient auditing.

Data owner that has stored data at an un-trusted cloud server can verify that the server possesses his original data without retrieving it [1]. A probabilistic proof of possession is generated by sampling random sets of blocks from the data at the server, which reduces I/O costs in this model. A constant amount of metadata is maintained by the client to verify the proof. Since this protocol transmits a small and constant amount of data, it can minimize network communication. PDP is a lightweight model for RDC and supports large data sets. The data owner can challenge a cloud server to provide a proof of data possession for verifying that the cloud server possesses the original data stored by the data owner. RDC can be comprised diagrammatically as Figure 4.

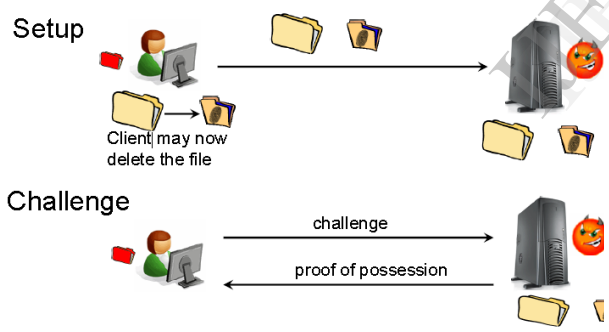

Figure 4. Remote Data Checking.

Spot checking is used for achieving this goal, in which the data owner randomly samples small blocks of the data and validate their integrity. Spot checking allows the client user can detect if small fraction of the data at the cloud server has been damaged. Provable data possession for remote data checking provides proof that a cloud server possesses a file. This is made possible using homomorphic verifiable tags [1]. The data owner pre-computes tags for each block of data and then stores the data and its tags in the storage server. Later data owner can validate that the cloud server possesses the data by generating a challenge against a randomly selected blocks of data. This model constitutes of two phases namely setup and challenge. The setup and challenge are diagrammatically described in Figure 5 and Figure 6.RDC using PDP can be enhanced by integrating forward error-correcting codes (FECs) with it.

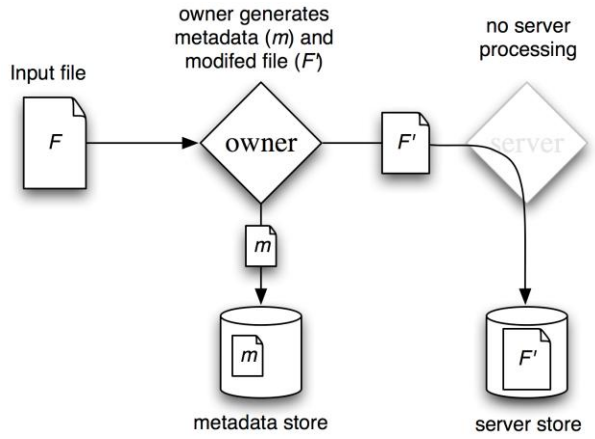

Figure 5. RDC Setup.

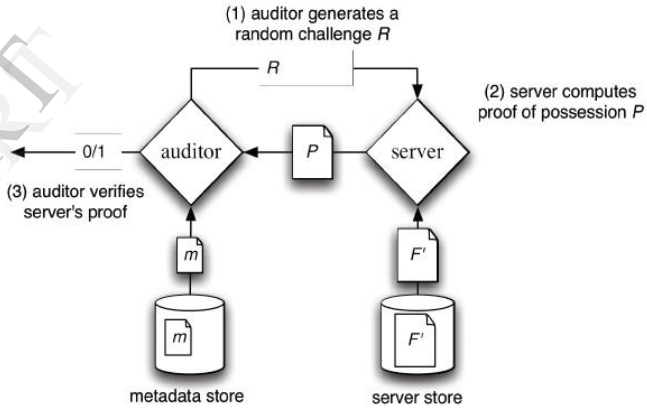

Figure 6. RDC Challenge.

The data owner stores his data on the cloud server is ordered collection of blocks F = (b_1, . . . , b_f). A homomorphic verifiable which is the building block of PDP is generated for each block of data and stored on the server together with the data file F. These tags are the verification metadata for the data file in the cloud server. Provable Data Possession Scheme is a collection of four algorithms, KeyGen, TagBlock, GenProof and CheckProof.

KeyGen: it is a probabilistic key generation algorithm that is run by the data owner side in the setup phase. Takes a security parameter *k* as input and returns a pair of public and secret keys.

TagBlock: Algorithm run by the data owner to generate the verification metadata. It takes a public key, a secret key, and a data block as input and returns the tag.

GenProof: Algorithm runs by the server for generating a proof of possession. It takes a public key, an ordered collection data blocks, a challenge, and an ordered collection of verification metadata corresponding to the data blocks as input and returns a proof of possession.

CheckProof: Algorithm run by the client in order to verify the proof of possession. It takes a public key, a secret key, a challenge, and a proof of possession as input and returns whether data is modified or not.

In order to strengthen the Proof of possession achieved by a RDC, forward error-correcting codes (FECs) is integrating with RDC [1].Integrating data checking with FEC improves possession guarantee. Spot checking in combined with systematic codes can help to achieve this. Reed-Solomon coding is a systematic encoding technique which keeps original file sequential. Encryption and permutation is the techniques used in RS codes for encoding of randomly selected input blocks. The encoding is represented in Figure 7.

4. System Architecture

Enhanced CIA framework composed of users, data, cloud server, Logger and Log Harmonizer. Logger is highly attached to the owner data. For implementing light weight accountability, the data is enclosed in a nested JAR file and send to the cloud service provider. The system architecture can be diagrammatically represented as in Figure 8.

At the beginning both the data user and the data owner has to register in cloud server. The cloud server distributes a pair of IBE keys to both of them. A master Key is also derived from the existing public key and parameters at the PKG. the data owner then upload his data in the cloud server after enclosing it in a JAR file along with authentication, access control and automated logging features.

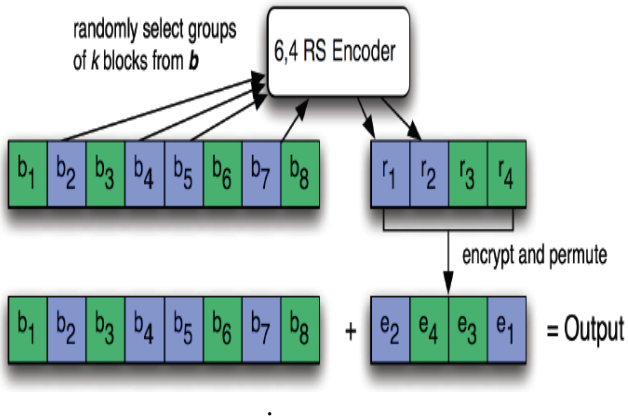

Figure 7. RS encoding.

The data in the JAR file is kin the encrypted form using Identity Based Encryption technique. Whenever any of the entities access the JAR file it is automatically logged. The generated log record is embedded in the log file after encrypted by the data user using the public key of the data owner. Before embedding encrypted record, it should be signed by the data user. The data is retrieved after decrypting it with the master key and sent to the data user. The data owner can audit the log file using push-pull auditing method. The content in the log file can be decrypted using master key before auditing. This provide an efficient accountability upon the data that data owner has sent to the CSP.

Since the data has been outsourced by the CSP it might be out of control of cloud service. For ensuring data integrity a remote data checking is implemented. The owner can claim for integrity of data by remotely selecting some blocks of data in the remote server. The data owner sends a challenge message after randomly selecting blocks from his data in the cloud. The cloud server should provide a proof of possession in reply with the challenge. The integrity of data is verified using spot checking mechanism. This process is done by pre-computing tags for each block of data in the cloud server for strengthen integrity checking, a forward error correction is integrated in the PDP.

IJERT

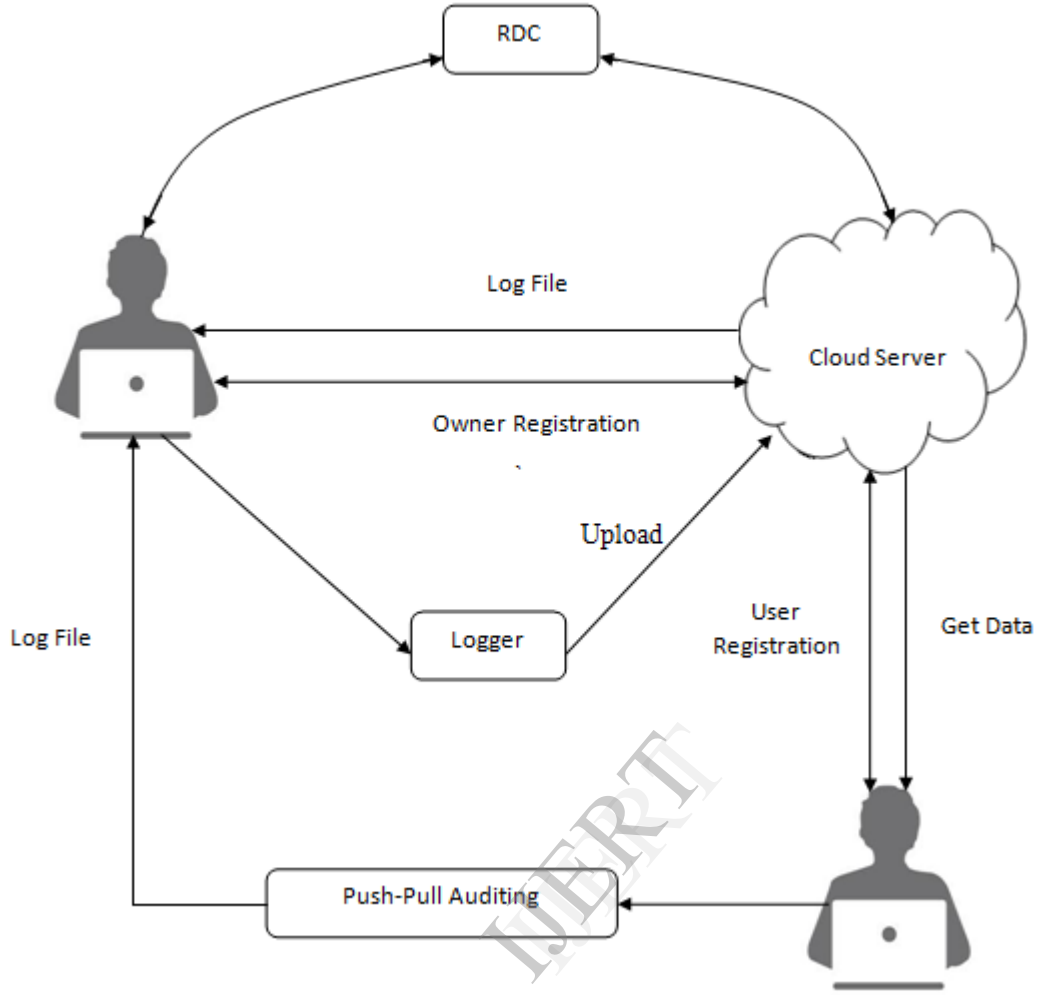

Figure 8. System Architecture.

5. Conclusion

Enhanced CIA framework provides a highly distributed data sharing approach that guarantees the integrity of the user's data in the cloud. Our light-weight framework allows the data owner to not only account and audit his data in remote server but also ensures the integrity of the same. Any access to the data in the cloud is automatically logged so that user can audit the logged information later. The integrity can be validated periodically by the data owner himself by generating challenge upon his data in the remote server.

We proposed approaches to address the small corruption problem for static data in our paper. If the data needs to be updated the proposed solution

References

[1] Ateniese, Giuseppe, et al. "Remote data checking using provable data possession." *AC Transactions on Information and System Security (TISSEC)*14.1 (2011): 12.

[2] Boneh and M.K. Franklin, "*Identity-Based Encryption from the Weil Pairing*," Proc. Int'l Cryptology Conf. Advances in Cryptology, pp. 213-229, 2001.

[3] R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, *"A Logic for Auditing Accountability in Decentralized Systems*," Proc. TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005.

[4] Hsiao-Ying Lin, et al. "*A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding*" Parallel and Distributed Systems, IEEE Transactions on , Volume 23 (6) Institute of Electrical and Electronics Engineers –Apr 25, 2012

[5] P.T. Jaeger, J. Lin, and J.M. Grimes, "*Cloud Computing and Information Policy: Computing in a Policy Cloud?,*" J. Information Technology and Politics, vol. 5, no. 3, pp. 269-283, 2009.

[6] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "*Towards a Theory of Accountability and Audit*," Proc. 14th European Conf. Research in Computer Security (ESORICS), pp. 152-167, 2009.

[7] Jijin Soman, "*Enhanced Data Sharing by Decentralized and Lightweight Framework on Accountability in Cloud Storage*" ERES international journal 2012

[8] W. Lee, A. Cinzia Squicciarini, and E. Bertino, "*The Design and Evaluation of Accountable Grid Computing System,*" Proc. 29th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '09), pp. 145-154, 2009.

[9] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance (Theory in Practice)*, first ed. O' Reilly, 2009.

[10] S. Pearson and A. Charlesworth, "*Accountability as a Way Forward for Privacy Protection in the Cloud*," Proc. First Int'l Conf. Cloud Computing, 2009.

[11] M. Premkumar, "*A Secure Cloud Storage System with increased Availability and Robustness*" ERES international journal 2012

[12] A. Pretschner, M. Hilty, and D. Basin, "*Distributed Usage Control*," Comm. ACM, vol. 49, no. 9, pp. 39-44, Sept. 2006.

[13] A. Squicciarini, S. Sundareswaran, and D. Lin, "*Preventing Information Leakage from Indexing in the Cloud*," Proc. IEEE Int'l Conf. Cloud Computing, 2010.

[14] S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, "*Promoting Distributed Accountability in the Cloud*," Proc. IEEE Int'l Conf. Cloud Computing, 2011.

[15] Sundareswaran, Smitha, Anna Squicciarini, and Dan Lin. "*Ensuring distributed accountability for data sharing in the cloud*." *Dependable and Secure Computing, IEEE Transactions on* 9.4 (2012): 556-568.

[16] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "*Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing*," Proc. European Conf. Research in Computer Security (ESORICS), pp. 355-370, 2009.

[17] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigen-baum, J.Hendler, and G.J. Sussman, "*Information Accountability*," Comm.ACM, vol. 51, no. 6, pp. 82-87, 2008.

IJERT