ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

Dynamic And Public Auditing With Fair Attribution For Cloud Data

[1] K. Shanmugapriya, [2] Moothi Lakshmi Prasanna
[1] Assistant Professor, Deptt of CSE, Bharath University, Chennai, Tamil Nadu, India
[2] UG, Dept of CSE, Bharath University, Chennai, Tamil Nadu, India

Abstract: ***To protect outsourced data in cloud storage against corruptions, adding fault tolerance to cloud storage together with data integrity checking and failure reparation becomes critical. Recently,regenerating codes have gained popularity due to their lower repair bandwidth while providing fault tolerance. Existing remote checking methods for regenerating-coded data only provide private auditing, requiring data owners to always stay online and handle auditing, as well as repairing, which is sometimes impractical. In this paper, we propose a public auditing scheme for the regenerating-code-based cloud storage. To solve the regeneration problem of failed authenticators in the absence of data owners, we introduce a proxy, which is privileged to regenerate the authenticators, into the traditional public auditing system model. Moreover, we design a novel public verifiable authenticator, which is generated by a couple of keys and can be regenerated using partial keys. Thus, our scheme can completely release data owners from online burden. In addition, we randomize the encode coefficients with a pseudorandom function to preserve data privacy. Extensive security analysis shows that our scheme is provable secure under random oracle model and experimental evaluation indicates that our scheme is highly efficient and can be feasibly integrated into the regenerating-code-based cloud storage.***

Keywords: TPA, Regenerating code, Proxy, Authenticator, Public auditing, Key.

I. INTRODUCTION

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

LITERATURE SURVEY

A BERKELEY VIEW OF CLOUD COMPUTING

Provided certain obstacles are overcome, I believe Cloud Computing has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IThardware is designed and purchased. Developers with innovative ideas for new interactive Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about over-provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under- provisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companieswith largebatch-oriented tasks can get their results as quickly as their programs can scale, since using 1000 servers for one hour costs no more than using one server for 1000 hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT. The economies of scale of very large-scale datacenters combined with ``pay-as-you-go" resource usage has heralded the rise of Cloud Computing. It is now attractive to deploy an innovative new Internet service on a third party's Internet Datacenter rather than your own infrastructure, and to gracefully scale its resources as it grows or declines in popularity and

revenue. Expanding and shrinking daily in response to normal diurnal patterns could lower costs even further. Cloud Computing transfers the risks of over-provisioning or under-provisioning to the Cloud Computing provider, who mitigates that risk by statistical multiplexing over a much larger set of users and who offers relatively low prices due better utilization and from the economy of purchasing at a larger scale. We define terms, present an economic model that quantifies the key buy vs. pay-as-you-go decision, offer a spectrum to classify Cloud Computing providers, and give our view of the top 10 obstacles and opportunities to the growth of Cloud Computing.

Provable data possession at un-trusted stores

We introduce a model for provable data possession (PDP) that allows a client that has stored data at an un-trusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage system.

We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

Proofs of retrievability for large files

In this paper, We define and explore proofs of retrievability (PORs). A POR scheme enables an archive or back-up service (prover) to produce a concise proof that a user (verifier) can retrieve a target file F, that is, that the archive retains and reliably transmits file data sufficient for the user to recover F in its entirety.

A POR may be viewed as a kind of cryptographic proof of knowledge (POK), but one specially designed to handle a large file (or bitstring) F. We explore POR protocols here in which the communication costs, number of memory accesses for the prover, and storage requirements of the user (verifier) are small parameters essentially independent of the length of F. In addition to proposing new, practical POR constructions, we explore implementation considerations and optimizations that bear on previously explored, related schemes. In a POR, unlike a POK, neither the prover nor the verifier need actually have knowledge of F. PORs give rise to a new and unusual security definition whose formulation is another contribution of our work.

We view PORs as an important tool for semi-trusted online archives. Existing
cryptographic techniques help users ensure the privacy and integrity of files they retrieve. It is also natural, however, for users to want to verify that archives do not delete or modify files prior to retrieval. The goal of

a POR is to accomplish these checks without users having to download the files themselves. A POR can also provide quality-of-service guarantees, i.e., show that a file is retrievable within a certain time bound.

MR-PDP: Multiple-replica provable data possession

Many storage systems rely on replication to increase the availability and durability of data on untrusted storage systems. At present, such storage systems provide no strong evidence that multiple copies of the data are actually stored. Storage servers can collude to make it look like they are storing many copies of the data, whereas in reality they only store a single copy. We address this shortcoming through multiple-replica provable data possession (MR-PDP): A provably-secure scheme that allows a client that stores t replicas of a file in a storage system to verify through a challenge-response protocol that (1) each unique replica can be produced at the time of the challenge and that (2) the storage system uses t times the storage required to store a single replica. MR-PDP extends

previous work on data possession proofs for a single copy of a file in a client/server storage system (Ateniese et al., 2007). Using MR-PDP to store t replicas is computationally much more efficient than using a single-replica PDP scheme to store t separate, unrelated files (e.g., by encrypting each file separately prior to storing it). Another advantage of MR-PDP is that it can generate further replicas on demand, at little expense, when some of the existing replicas fail.

A high-availability and integrity layer for cloud storage We introduce HAIL (High-Availability and Integrity Layer), a distributed cryptographic system that allows a set of servers to prove to a client that a stored file is intact and retrievable. HAIL strengthens, formally unifies, and streamlines distinct approaches from the cryptographic and distributed-systems communities. Proofs in HAIL are efficiently computable by servers and highly compact-- -typically tens or hundreds of bytes, irrespective of file size. HAIL cryptographically verifies and reactively reallocates file shares. It is robust against an active, mobile adversary, i.e., one that may progressively corrupt the full set of servers.

We propose a strong, formal adversarial model for HAIL, and rigorous analysis and parameter choices. We show how HAIL improves on the security and efficiency of existing tools, like Proofs of Retrievability (PORs) deployed on individual servers. We also report on a prototype implementation.

II. SYSTEM ANALYSIS

EXISTING SYSTEM

- Many mechanisms dealing with the integrity of outsourced data without a local copy have been proposed under different system and security models up to now. The most significant work among these studies are the PDP (provable data possession) model and POR (proof of retrievability) model, which were originally proposed for the single-server scenario by Ateniese et al. and Juels and Kaliski, respectively.

- Considering that files are usually striped and redundantly stored across multi-serversormulti-clouds,

explore integrity verification schemes suitable for such multi-servers or multi-clouds setting with different redundancy schemes, such as replication, erasure codes, and, more recently, regenerating codes.

- Chen et al. and Chen and Lee

separately and independently extended the single-server CPOR scheme to the regeneratingcode-scenario; designed and implemented a data integrity protection (DIP) scheme for FMSR-based cloud storage and the scheme is adapted to the thin-cloud setting.

PROPOSED SYSTEM

- In this paper, we focus on the integrity verification problem in regenerating-code-basedcloud

storage, especially with the functional repair strategy. To fully ensure the data integrity and save the users' computation resources as well as online burden, we propose a public auditing scheme for the regenerating-code-based cloud storage, in which the integrity checking and regeneration (of failed data blocks and authenticators) are implemented by a third-party auditor and a semi-trusted proxy separately on behalf of the data owner.

- Instead of directly adapting the existing public auditing scheme to the multi-server setting, we design a novel authenticator, which is more appropriate for regenerating codes.

Besides, we "encrypt" the coefficients to protect data privacy against the auditor, which is more lightweight than applying the proof blind technique and data blind method.

- We design a novel homomorphism authenticator based on BLS signature, which can be generated by a couple of secret keys and verified publicly.

ADVANTAGES OF PROPOSED SYSTEM

- Utilizing the linear subspace of the

regenerating codes, the authenticators can be computed efficiently. Besides, it can be adapted for data owners equipped with low end computation devices (e.g. Tablet PC etc.) in which they only need to sign the native blocks.

- To the best of our knowledge, our scheme is the first to allow privacy-preserving public auditing for regenerating code- based cloud storage. The coefficients are masked by a PRF (Pseudorandom Function) during the Setup phase to avoid leakage of the original data. This method is lightweight and does not introduce any computational Our scheme completely releases data owners from online burden for the regeneration of blocks and authenticators at faulty servers and it provides the privilege to a proxy for the reparation.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

- Optimization measures are taken to improve the flexibility and efficiency of our auditing scheme; thus, the storage overhead of servers, the computational overhead of the data owner and communication overhead during the audit phase can be effectively reduced.

- Our scheme is provable secure under random oracle model against adversarie

SYSTEM ARCHITECTURE

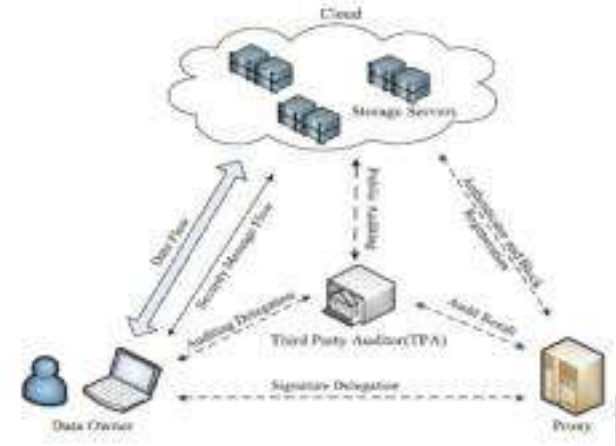


Figure 5.1

SYSTEM DESIGN

DATA FLOW DIAGRAM

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

- DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

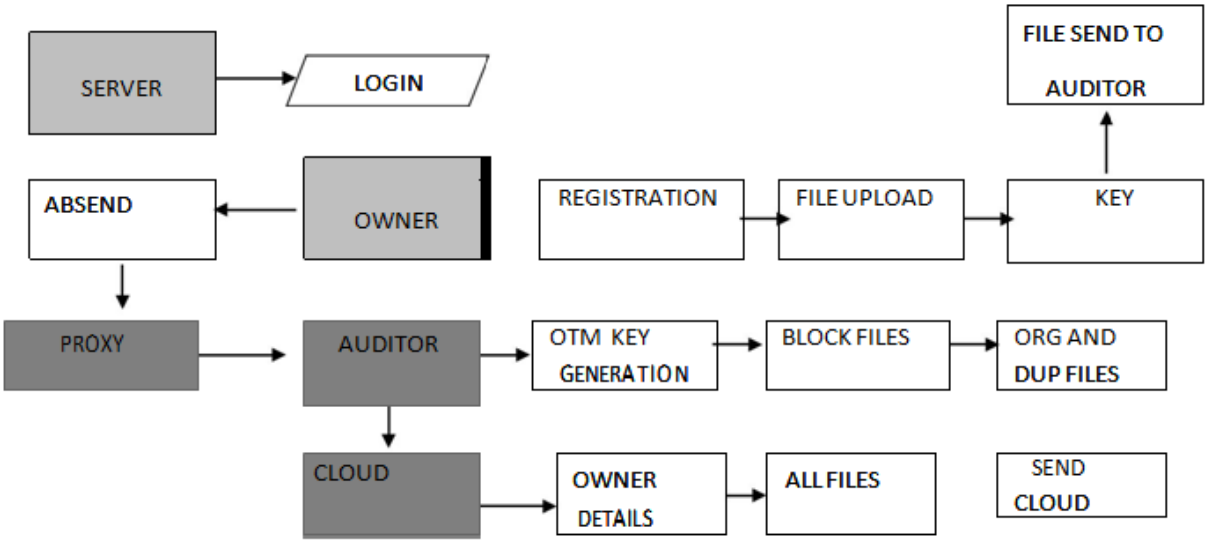


Figure 6.1

USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

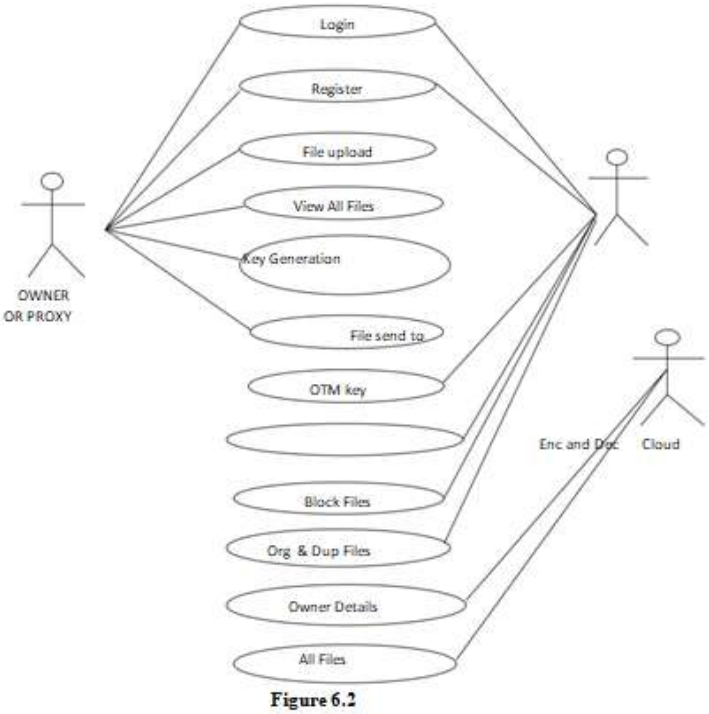


Figure 6.2

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

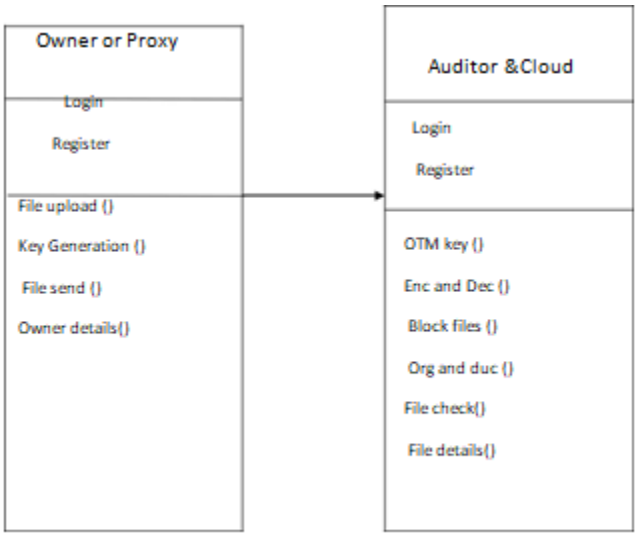


Figure 6.3

SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

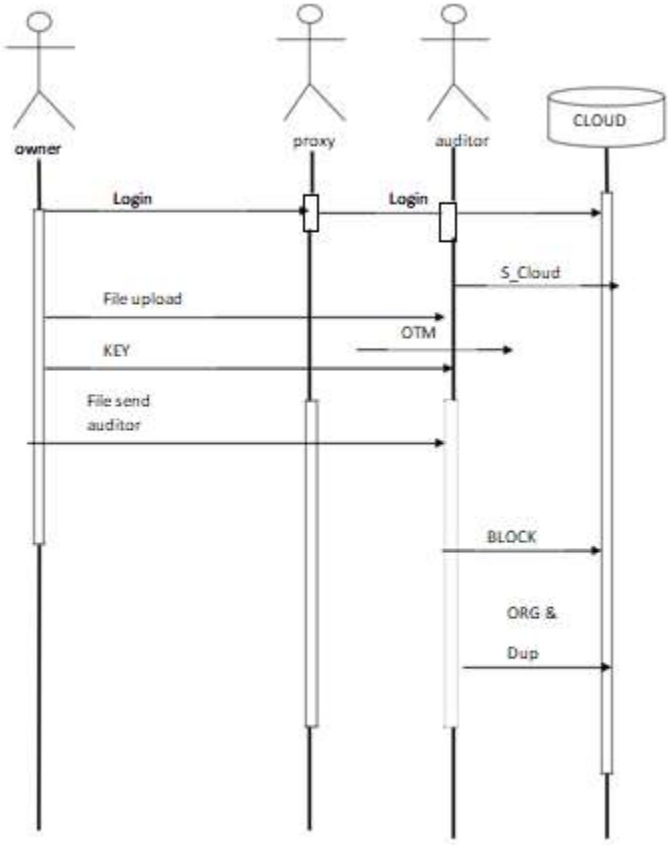


Figure 6.4 SEQUENCE DIAGRAM

ACTIVITY DIAGRAM

`Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

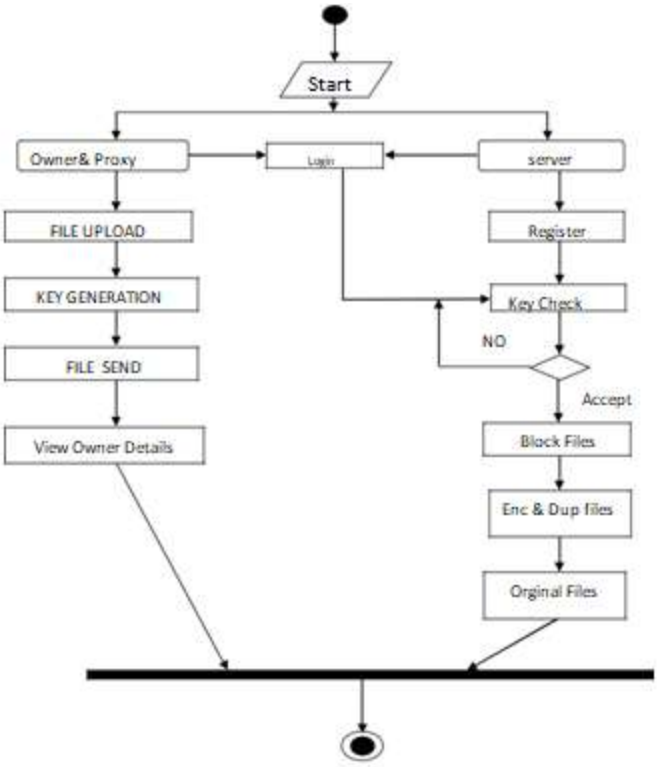


Figure 6.5

MODULES DESCRIPTION System Model

We consider the auditing system model for Regenerating-Code-based cloud storage, which involves four entities: the data owner, who owns large amounts of data files to be stored in the cloud; the cloud, which are managed by the cloud service provider, provide storage service and have significant computational resources; the third party auditor (TPA), who has expertise and capabilities to conduct public audits on the coded data in the cloud, the TPA is trusted and its audit result is unbiased for both data owners and cloud servers; and a proxy agent, who is semi-trusted and acts on behalf of the data owner to regenerate authenticators and data blocks on the failed servers during the repair procedure. Notice that the data owner is restricted in computational and storage resources compared to other entities and may becomes off-line even after the data upload procedure. The proxy, who would always be online, is supposed to be much more powerful than the data owner but less than the cloud servers in terms of computation and

memory capacity. To save resources as well as the online burden potentially brought by the periodic auditing and accidental repairing, the data owners resort to the TPA for integrity verification and delegate the reparation to the proxy.

Construction of Our Auditing Scheme

Our auditing scheme consists of three procedures: Setup, Audit and Repair. To correctly and efficiently verify the integrity of data and keep the stored file available for cloud storage, our proposed auditing scheme should achieve the following properties: Public Auditability: To allow TPA to verify the intactness of the data in the cloud on demand without introducing additional online burden to the data owner. Storage Soundness: To ensure that the cloud server can never pass the auditing procedure except when it indeed manage the owner's data intact. Privacy Preserving: To ensure that neither the auditor nor the proxy can derive users' data content from the auditing and reparation process. Authenticator Regeneration: The authenticator of the repaired blocks can be correctly

regenerated in the absence of the data owner. Error Location: To ensure that the wrong server can be quickly indicated when data corruption is detected.

Mitigating the Overhead of Data Owner

Despite that the data owner has been released from online burden for auditing and repairing, it still makes sense to reduce its computation overhead in the Setup phase because data owners usually maintain very limited computational and memory resources. As previously described, authenticators are generated in a new method which can reduce the computational complexity of the owner to some extent; however, there exists a much more efficient method to introduce further reduction. Considering that there are so many modular exponent arithmetic operations during the authenticator generation, the data owner can securely delegate part of its computing task to the proxy in the following way: The data owner first properly augments the m native blocks, signs for them, and thus obtains and, then it sends the augmented native blocks and to the proxy. After receiving from the data owner, the proxy implements the last two steps of SigAndBlockGen(•) and finally generates entire authenticators for each segment with secret value x. In this way, the data owner can migrate the expensive encoding and authenticator generation task to the proxy while itself maintaining only the first two lightweight steps; thus, the workload of data owner can be greatly mitigated

Enabling Privacy Preserving Auditable

The privacy protection of the owner's data can be easily achieved through integrating with the random proof blind technique or other technique. However, all

these privacy-preservation methods introduce additional computation overhead to the auditor, who usually needs to audit for many clouds and a large number of data owners; thus, this could possibly make it create a performance bottleneck. Therefore, we prefer to present a novel method, which is more light-weight, to mitigate private data leakage to the auditor. Notice that in regenerating-code-based cloud storage, data blocks stored at servers are coded as linear combinations of the original blocks with random coefficients. Supposing that the curious TPA has recovered m coded blocks by elaborately performing Challenge-Response procedures and solving systems of linear equations, the TPA still requies to solve another group of m linearly independent equations to derive the m native blocks. We can utilize a keyed pseudorandom function to mask the coding coefficients and thus prevent the TPA from correctly obtaining the original data. Specifically, the data owner maintains a secret key in the beginning of the Setup procedure and augments m original data blocks.

ALGORITHM SPECIFICATION

K-medoids

The k-medoids algorithm is a clustering algorithm related to the k- means algorithm and the medoid shift algorithm. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups). K-means attempts to minimize the total squared error, while k- medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses data points as centers (medoids or exemplars).

K-medoids is also a partitioning technique of clustering that clusters the data set of n objects into k clusters with k known a priori. A useful tool for determining k is the silhouette. It could be more robust to noise and outliers as compared to k-means because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distances. The possible choice of the dissimilarity function is very rich but in our applet we used the Euclidean distance.

A medoid of a finite dataset is a data point from this set, whose average dissimilarity to all the data points is minimal i.e. it is the most centrally located point in the set.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

The most common realisation of k-medoid clustering is the Partitioning Around Medoids (PAM) algorithm and is as follows:

1. Initialize: randomly select k of the n data points as the medoids
2. Assignment step: Associate each data point to the closest medoid.
3. Update step: For each medoid m and each data point o associated to m swap m and o and compute the total cost of the configuration (that is, the average dissimilarity of o to all the data points associated to m). Select the medoid o with the lowest cost of the configuration.

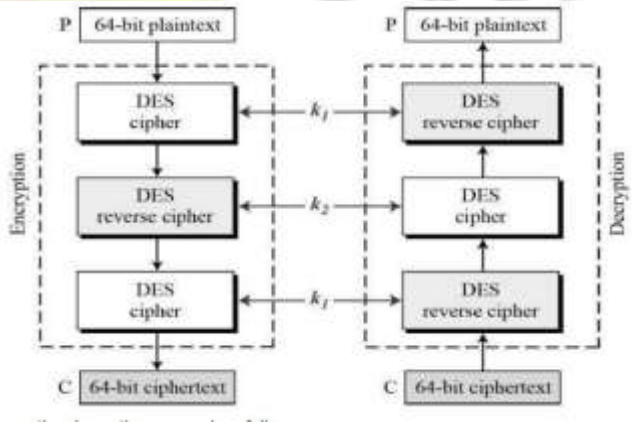
Repeat alternating steps 2 and 3 until there is no change in the assignments.

III. TRIPLE DES ALGORITHM

The speed of exhaustive key searches against DES after 1990 began to cause discomfort amongst users of DES. However, users did not want to replace DES as it takes an enormous amount of time and money to change encryption algorithms that are widely adopted and embedded in large security architectures. The pragmatic approach was not to abandon the DES completely, but to change the manner in which DES is used. This led to the modified

schemes of Triple DES sometimesknownas3DES. Incidentally, there are two variants of Triple DES known as 3-key Triple DES 3TDES and 2-key Triple DES 2TDES.

3-KEY Triple DES Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K1, K2 and K3. This means that the actual 3TDES key has length 3×56 = 168 bits. The encryption scheme is illustrated as follows − The encryption-decryption process is as follows –



Encrypt the plaintext blocks using single DES with key K1. Now decrypt the output of step 1 using single DES with key K2. Finally, encrypt the output of step 2 using single DES with key K3. The output of step 3 is the cipher text. Decryption of a cipher text is a reverse process. User first decrypt using K3, then encrypt with K2, and finally decrypt with K1. Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES hardware implementation for single DES by setting K1, K2, and K3 to be the same value. This provides backwards compatibility with DES. Second variant of Triple DES 2TDES is identical to 3TDES except that K3is replaced by K1. In other words, user encrypt plaintext blocks with key K1, then decrypt with key K2, and finally encrypt with K1 again. Therefore, 2TDES has a key length of 112 bits. Triple DES systems are significantly more

secure than single DES, but these are clearly a much slower process than encryption using single DES.

APPENDIX

SAMPLE CODING

```
<%@page import="java.sql.Statement"%> <%@page import="java.sql.Connection"%> <%--
<%@page import="com.actions.mail" %> --%>
<%@page
import="dbpackeg.databasecon"%> <%
   String g = request.getQueryString(); String userid =
session.getAttribute("UID").toString(); System.out.println("g " + g);
   System.out.println("userid " + userid);

   Connection con =databasecon.getconnection();
   Statement st = con.createStatement(); int insert = st.executeUpdate("insert into
req values('" + request.getQueryString() + "','" + session.getAttribute("UID") + "')");
   if (insert != 0) {
                //      new mail().mailsend(g, "cloudcomputing96@gmail.com");

response.sendRedirect("Files.jsp?msg=mail sent To TPA");
   } else {
      out.println("Error");
   }
%>
```

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

SCREENSHOTS

HOME

OWNER REG

Owner login

Owner page

File upload

Uploaded files

File send to TPA

Sending Proxy key

TPA login

TPA verification

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

TPA home

TPA files

Decrypt to view upload file

Checking the file

Blocks

Original/duplicate file

Cloud file

Proxy login

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

Proxy key from owner

Proxy home

Proxy blocks

Proxy file view

IV. CONCLUSION

In this paper, we propose a public auditing scheme for the regenerating-code-based cloud storage system, where the data owners are privileged to delegate TPA for their data validity checking. To protect the original data privacy against the TPA, we randomize the coefficients in the beginning rather than applying the blind technique during the auditing process. Considering that the data owner cannot always stay online in practise, in order to keep the storage available and verifiable after a malicious corruption, we introduce a semi-trusted proxy into the system model and provide a privilege for the proxy to handle the reparation of the coded blocks and authenticators. To better appropriate for the regenerating-code-scenario, we design our authenticator based on the BLS signature. This

authenticator can be efficiently generated by the data owner simultaneously with the encoding procedure. Extensive analysis shows that our scheme is provable secure, and the performance evaluation shows that our scheme is highly efficient and can be feasibly integrated into a regenerating-code-based cloud storage system.

REFERENCES

[1] M. Armbrust et al., "Above the clouds: A Berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, Tech. Rep.UCB/EECS-2009-28, 2009.

[2] G. Ateniese et al., "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), New York, NY, USA, 2007, pp. 598–609.

[3] A. Juels and B. S. Kaliski, Jr., "PORs:Proofs of retrievability for large files," in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 584–597.

[4] R. Curtmola, O. Khan, R. Burns, and G.Ateniese, "MR-PDP: Multiple-replica provable data possession," in Proc. 28th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jun. 2008, pp. 411–420.

[5] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in Proc. 16th ACM Conf. Comput. Commun. Secur., 2009, pp. 187–198.

[6] J. He, Y. Zhang, G. Huang, Y. Shi, and J. Cao, "Distributed data possession checking for securing multiple replicas in geographically dispersed clouds," J. Comput. Syst. Sci., vol. 78, no. 5, pp. 1345– 1358, 2012.

[7] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote data checking for network coding-based distributed storage systems," in Proc. ACM Workshop Cloud Comput. Secur. Workshop, 2010, pp. 31–42.

[8] H. C. H. Chen and P. P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 407–416, Feb. 2014.

[9] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 9, pp.1717–1726, Sep. 2013.

[10] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," IEEE Trans. Parallel Distrib. Syst., vol. 23,no. 12, pp. 2231–2244, Dec.