iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** XI **Month of publication:** November 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Improving Security in Cloud Storage Using Eraser Code Techniques

Vidhya.K[1], Rethina sabapathi.[2],Thulasibai-u[3]
Sri Venkateswaracollege Of Engineering and Technology, India

Abstract: ***To achieve a secure and dependable cloud storage service, a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token and distributed erasure-coded data. This allows users to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving server. To support secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append. This approach ensures efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks. In this storage have fault tolerance along with data integrity checking and recovery procedures become critical.***
Keywords- Cloud storage, Eraser Code, Auditing, Data Security, Replica Server

I. INTRODUCTION

Several trends are opening up the era of cloud computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the Software as a Service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers.

In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. However, the fact that users no longer have physical possession of data in the cloud prohibits the direct adoption of traditional cryptographic primitives for the purpose of data integrity protection. Hence, the verification of cloud storage correctness must be conducted without explicit knowledge of the whole data files, Meanwhile, cloud storage is not just a third party data warehouse.

The data stored in the cloud may not only be accessed but also be frequently updated by the users, including insertion, deletion, modification, appending, etc. Thus, it is also imperative to support the integration of this dynamic feature into the cloud storage correctness assurance, which makes the system design even more challenging. Last but not the least, the deployment of cloud computing is powered by data centers running in a simultaneous, cooperated, and distributed manner. It is more advantages for individual users to store their data redundantly across multiple physical servers so as to reduce the data integrity and availability threats. Thus, distributed protocols for storage correctness assurance will be of most importance in achieving robust and secure cloud storage systems. However, such important area remains to be fully explored in the literature.

Cloud computing is the long dreamed vision of computing as a utility,where data owners can remotely store their data in the cloud to enjoy on-demand high-quality applications and services from a shared pool of configurable computing resources. While data outsourcing relieves the owners of the burden of local data storage and maintenance, it also eliminates their physical control of storage dependability and security, which traditionally has been expected by both enterprises and individuals with high service-level requirements.In order to facilitate rapid deployment of cloud data storage service and regain security assurances with outsourced data dependability, efficient methods that enable on-demand data correctness verification on behalf of cloud data owners have to be designed.

A. Our Contribution

CLOUD storage offers an on-demand data outsourcing service model, and is gaining popularity due to its elasticity and low maintenance cost. However, security concerns arise when data storage is outsourced to third party cloud storage providers. It is

desirable to enable cloud clients to verify the integrity of their outsourced data, in case their data have been accidentally corrupted or maliciously compromised by insider/outsider attacks. One major use of cloud storage is long-term archival, which represents a workload that is written once and rarely read. While the stored data are rarely read, it remains necessary to ensure its integrity for disaster recovery or compliance with legal requirements. Since it is typical to have a huge amount of archived data, whole-file checking becomes prohibitive. **Proof of retrievability** (POR) and **proof of data possession** (PDP) have thus been proposed to verify the integrity of a large file by spot checking only a fraction of the file via various cryptographic primitives.

Suppose that we outsource storage to a server, which could be a storage site or a cloud-storage provider. If we detect corruptions in our outsourced data (e.g., when a server crashes or is compromised), then we should repair the corrupted data and restore the original data. However, putting all data in a single server is susceptible to the single point- of-failure problem and vendor lock-ins. As suggested, a plausible solution is to stripe data across multiple servers. Thus, to repair a failed server, we can 1) read data from the other surviving servers, 2) reconstruct the corrupted data of the failed server, and 3) write the reconstructed data to a new server. POR and PDP are originally proposed for the single-server case. MR-PDP and HAIL extend integrity checks to a multi server setting using replication and erasure coding, respectively. In particular, erasure coding (e.g., Reed- Solomon codes) has a lower storage overhead than replication under the same fault.

II. RELATED WORK

The most recent and closely related work here, we consider the problem of checking the integrity of static data, which is typical in long-term archival storage systems. This problem is first considered under a single server scenario [3], giving rise to the similar notions POR and PDP, respectively. A major limitation of the above schemes Is that they are designed for a single-server setting. If the server is fully controlled by an adversary, then the above schemes can only provide detection of corrupted data, but
cannot recover the original data.

This leads to the design of efficient data checking schemes in a multiserver setting. By striping redundant data across multiple servers, the original files can still be recovered from a subset of servers even if some servers are down or compromised. Efficient data integrity checking has been proposed for different redundancy schemes, such as replication, erasure coding ,and regenerating coding .Specifically, although also consider regenerating-coded storage, there are key differences with our work.

First, their design extends the single-server compact POR scheme However, such direct adaptation inherits some shortcomings of the single-server scheme such as a large storageoverhead, as the amount of data stored increases with amore flexible checking granularity in Second, the storage scheme assumes that storage servers have encoding capabilities for generating encodeddata, while we consider a thin-cloud setting whereservers only need to support standard read/write functionalities for portability and simplicity.

The most closely related work to ours is HAIL which stores data via erasure coding. As stated in Section 1, HAIL operates on aper-file basis and it is nontrivial to directly apply HAIL to regenerating codes. In addition, our work focuses more on t he practical issues, such as how different parameters can be adjusted for the performance-security trade-off inpractical deployment.

III. ARCHITECTURE DESIGN

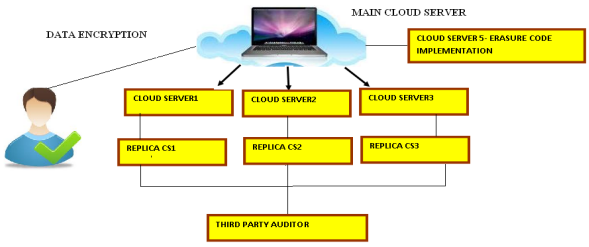


Figure 1 Architecture diagram of cloud Data storage

In the **EXISTING SYSTEM**, there is no big security provided in the Cloud server for data safety. If at all security exists, the third party auditor should be allowed to access the entire data packets for verification. There is no backup process.There exists drawbacks such as general encryption schemes protect data confidentiality, but also limit the functionalityof the storage system because few operations are supported over encrypted data and constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority and finally data robustness is a major drawback in the Existing

Cloud Storage Systems.Storing data in a third party's cloud system causes serious concern on data confidentiality.:

In the **PROPOSED SCHEME**, Cloud server will spilt the file into batches and allowed for encryption. The corresponding encrypted batches are kept in different Cloud servers and their keys are distributed in different key server. These encrypted batches are kept in replica servers as a backup. This encrypted data are converted into bytes and added parity bit process by the data owner in order to restrict TPA by accessing the original data. The Cloud server generates the token number from the parity added encrypted data and compared with the signature provided to the TPA to verify the Data Integrity. We also implement Erasure Code for the back-up of the data.In addition, we propose the encryption process of the data by the data owner before it reaches the Cloud server. This ensures proper double time security and it results in the following benefits.

A. Benefits

1) One way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. It is very robust because the message can be retrieved as long as one storage server survives.

2) A decentralized erasure code is suitable for use in a distributed storage system. After the message symbols are sent to storage servers, each storage server independently computes a codeword symbol for the received message symbols and stores it. This finishes the encoding and storing process. The recovery process is the same.

3) We propose a re-encryption scheme and integrate it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages. The tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding.

We consider a cloud data storage service involving three different entities: the cloud user, who has large amount of data files to be stored in the cloud; the cloud server, which is managed by the cloud service provider to provide data storage service and has significant storage space and computation resources; the third-party auditor, who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service reliability on behalf of the user upon request. Users rely on the CS for cloud data storage and maintenance.

They may also dynamically interact with the CS to access and update their stored data for various application purposes. As users no longer possess their data locally, it is of critical importance for users to ensure that their data are being correctly stored and maintained. To save the computation resource as well as the online burden potentially brought by the periodic storage correctness verification, cloud users may resort to TPA for ensuring the storage integrity of their outsourced data, while hoping to keep their data private from TPA. We assume the data integrity threats toward users' data can come from both internal and external attacks at CS. These may include: software bugs, hardware failures, bugs in the network path, economically motivated hackers, malicious or accidental management errors, etc. Besides, CS can be self-interested. For their own benefits, such as to maintain reputation, CS might even decide to hide these data corruption incidents to users. Using third-party auditing service provides a cost-effective method for users to gain trust in cloud. We assume the TPA, who is in the business of auditing, is reliable and independent. However, it may harm the user if the TPA could learn the outsourced data after the audit.

Note that in our model, beyond users' reluctance to leak data to TPA, we also assume that cloud servers have no incentives to reveal their hosted data to external parties. On the one hand, there are regulations, e.g., HIPAA, requesting CS to maintain users' data privacy. On the other hand, as users' data belong to their business asset [10], there also exist financial incentives for CS to protect it from any external parties. Therefore, we assume that neither CS nor TPA has motivations to collude with each other during the auditing process. In other words, neither entity will deviate from the prescribed protocol execution in the following presentation. To authorize the CS to respond to the audit delegated to TPA's, the user can issue a certificate on TPA's public key, and all audits from the TPA are authenticated against such a certificate.

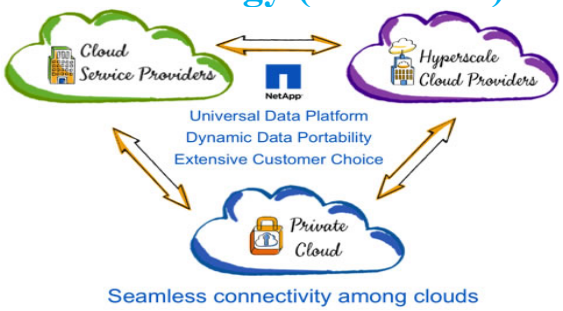


Figure2 cloud service provider

A cloud comprises processing, network, and storage elements, and cloud architecture consists of three abstract layers. Infrastructure is the lowest layer and is a means of delivering basic storage and compute capabilities as standardized services over the network. Servers, storage systems, switches, routers, and other systems handle specific types of workloads, from batch processing to server or storage augmentation during peak loads. The middle platform layer provides higher abstractions and services to develop, test, deploy, host, and maintain applications in the same integrated development environment. The application layer is the highest layer and features a complete application offered as a service.

In 1961, John McCarthy envisioned that "computation may someday be organized as a public utility." We can view the cloud computing paradigm as a big step toward this dream. To realize it fully, however, we must address several
significant problems and unexploited opportunities concerning the deployment, efficient operation, and use of cloud computing infrastructures.

The shift of computer processing, storage and software delivery away from desktop and local servers, across the
Internet and into next-generation data centers results in limitations new opportunities regarding data management. Data is replicated across large geographic distances, where its availability and durability are paramount for cloud service providers.

It's also stored at untrusted hosts, which creates enormous risks for data privacy. Computing power in clouds must be elastic to face changing conditions. For instance, providers can allocate additional computational resources.

IV. ALGORITHM DETAILS

The use of cryptographic hash functions like MD5 or SHA for message authentication has become a standard approach in many Internet applications and protocols. Though very easy to implement, these mechanisms are usually based on ad hoc techniques that lack a sound security analysis. We present new constructions of message authentication schemes based on a cryptographic hash function.

Our schemes, NMAC and HMAC, are proven to be secure as long as the underlying hash function has some reasonable cryptographic strengths. Moreover we show, in a quantitative way, that the schemes retain almost all the security of the underlying hash function. In addition our schemes are ancient and practical.

Their performance is essentially that of the underlying hash function. Moreover they use the hash function (or its compression function) as a black box, so that widely available library code or hardware can be used to implement them in a simple way, and replace ability of the underlying hash function is easily supported.

A. SHA(secured hash algorithm)

SHAis a set of cryptographic hash functions designed by the NSA (U.S. National Security Agency).[3] SHA stands for Secure Hash Algorithm. Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity. For example, computing the hash of a downloaded file and comparing the result to a previously published hash result can show whether the download has been modified or tampered with.A key aspect of cryptographic hash functions is their one-way nature: given a computed hash value, it is very difficult to derive the original data.

SHA-2 includes significant changes from its predecessor, SHA-1. The SHA-2 family consists of six hash functions with digests (hash

values) that are 224, 256, 384 or 512 bits: **SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256**.

SHA-256 and SHA-512 are novel hash functions computed with 32-bit and 64-bit words, respectively. They use different shift amounts and additive constants, but their structures are otherwise virtually identical, differing only in the number of rounds. SHA-224 and SHA-384 are simply truncated versions of the first two, computed with different initial values. SHA-512/224 and SHA-512/256 are also truncated versions of SHA-512, but the initial values are generated using the method described in FIPS PUB 180-4. SHA-2 was published in 2001 by the NIST as a U.S. federal standard . The SHA-2 family of algorithms are patented in US 6829355. The United States has released the patent under a royalty-free license.

In 2005, security flaws were identified in SHA-1, namely that a mathematical weakness might exist, indicating that a stronger hash function would be desirable. Although SHA-2 bears some similarity to the SHA-1 algorithm, these attacks have not been successfully extended to SHA-2.

Currently, the best public attacks break preimage resistance 52 rounds of SHA-256 or 57 rounds of SHA-512, and collision resistance for 46 rounds of SHA-256, as shown in the *Cryptanalysis and validation* section below.

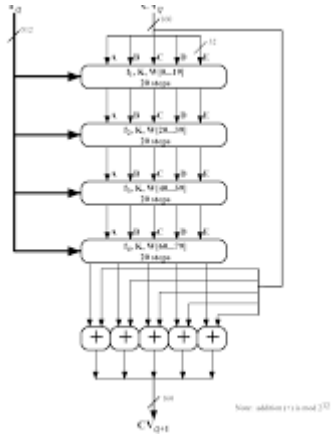


Figure3 secured hash algorithm

B. Message digest algorithm

MD5 which stands for **Message Digest** algorithm **5** is a widely used cryptographic hash function that was invented by Ronald Rivest in 1991. The idea behind this algorithm is to take up a random data (text or **binary**) as an input and generate a fixed size "hash value" as the output. The input data can be of any size or length, but the output "hash value" size is always fixed.

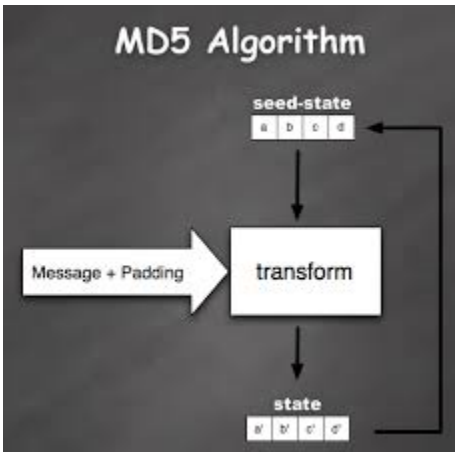


Figure4 working of MD5 algorithm

1) Applications of MD5 Hash: Before I tell you about how to use MD5, I would like to share one of my recent experience which made me start using MD5 algorithm. Recently I made some significant changes and updates to my website and as obvious, I generated a complete backup of the site on my server. I downloaded this backup onto my PC and deleted the original one on the server. But after a few days something went wrong and I wanted to restore the backup that I downloaded. When I tried to restore the backup, I was shocked! The backup file that I used to restore was corrupted.

That means, the backup file that I downloaded onto my PC wasn't exactly the one that was on my server. The reason is that there occurred some data loss during the download process. Yes, this data loss can happen often when a file is downloaded from the Internet. The file can be corrupted due to any of the following reasons:

- Data loss during the download process, due to instability in the Internet connection/server.
- The file can be tampered due to virus infections or
- Due to Hacker attacks.

So, whenever you download any valuable data from the Internet, it is completely necessary that you check the integrity of the downloaded file. That is, you need to ensure that the downloaded file is exactly the same as that of the original one. In this scenario, the MD5 hash can become handy. All you have to do is generate MD5 hash (or MD5 check-sum) for the intended file on your server.

After you download the file onto your PC, again generate MD5 hash for the downloaded file. Compare these two hashes and if they match, that means the file is downloaded perfectly without any data loss.

C. HMAC

In cryptography, a **keyed-hash message authentication code** (**HMAC**) is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key. As with any MAC, it may be used to simultaneously verify both the *data integrity* and the *authentication* of a message. Any cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA1 accordingly. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, the size of its hash output, and on the size and quality of the key.

An iterative hash function breaks up a message into blocks of a fixed size and iterates over them with a compression function. For example, MD5 and SHA-1 operate on 512-bit blocks. The size of the output of HMAC is the same as that of the underlying hash function (128 or 160 bits in the case of MD5 or SHA-1, respectively), although it can be truncated if desired.

Definition

$$HMAC(K, m) = H\left((K \oplus opad) | H((K \oplus ipad) | m)\right)$$

where

H is a cryptographic hash function,
K is a secret key to the right with extra zeros to the input block size of the hash function, or the hash of the original key if it's longer than that block size,
m is the message to be authenticated,
| denotes concatenates
⊕ denotes (XOR)

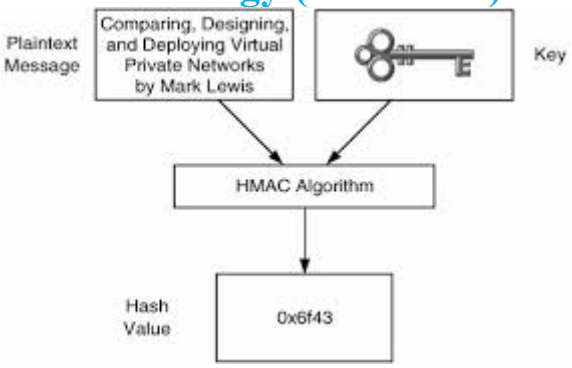


Figure 5 working of HMAC algorithm

V. CONCLUSION

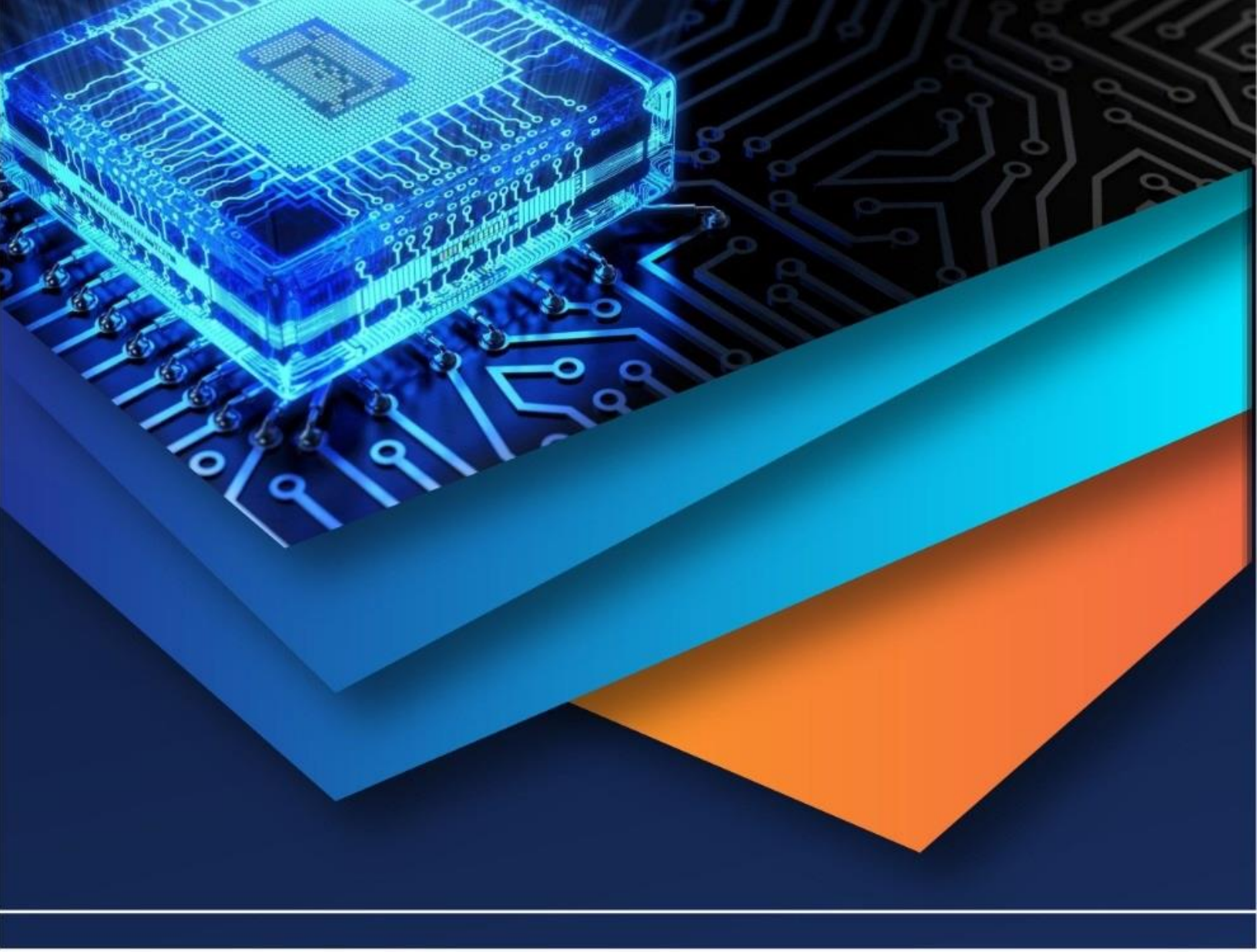
In this paper, we investigate the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasurecoded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s).

VI. ACKNOWLEDGEMENT

I feel very pleasure to thank MR.RETHINA SABAPATHY to this project successful.

REFERRENCES

[1] P. Mell and T. Grance, “Draft NIST Working Definition of Cloud Computing,” 2009; http://csrc.nist.gov/groups/SNS/cloud-computing/index.html
[2] M. Armbrust et al., “Above the Clouds: A Berkeley View of Cloud Computing,” Univ. California, Berkeley, Tech. Rep. UCBEECS-2009-28, Feb. 2009.
[3] Amazon.com, “Amazon s3 Availability Event: July 20, 2008,” July 2008; http://status.aws.amazon.com/s3-20080720.html
[4] M. Arrington, “Gmail Disaster: Reports of Mass Email Deletions,” Dec. 2006; http://www.techcrunch.com/2006/12/28/gmail-disaster-reports-of-massemail-deletions
[5] M. Krigsman, “Apple’s MobileMe Experiences Post-Launch Pain,” July 2008; http://blogs.zdnet.com/projectfailures/?p=908
[6] A. Juels, J. Burton, and S. Kaliski, “PORs: Proofs of Retrievability for Large Files,” Proc. ACM CCS ‘07, Oct. 2007, pp. 584–97.
[7] G.Ateniese et al., “Provable Data Possession at Untrusted Stores,” Proc. ACM CCS ‘07, Oct. 2007, pp. 598–609.
[8] M. A. Shah et al., “Auditing to keep Online Storage Services Honest,” Proc.USENIX HotOS ‘07, May 2007.
[9] G. Ateniese et al., “Scalable and Efficient Provable Data Possession,” Proc. SecureComm ‘08, Sept. 2008.
[10] H. Shacham and B. Waters, “Compact Proofs of Retrievability,” Proc. Asia- Crypt ‘08, LNCS, vol. 5350, Dec. 2008, pp. 90–107.
[11] K. D. Bowers, A. Juels, and A. Oprea, “Hail: A High-Availability and Integrity Layer for Cloud Storage,” Proc. ACM CCS ‘09, Nov. 2009, pp. 187–98.
[12]C.Wang et al.,”Ensuring Data Storage Security in Cloud Computing,” Proc. IWQoS ‘09, July 2009, pp. 1–9.
[13] Q. Wang et al., “Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing,” Proc. ESORICS ‘09, Sept. 2009, pp. 355–70.
[14] C. Erway et al., “Dynamic Provable Data Possession,” Proc. ACM CCS ‘09, Nov. 2009, pp. 213–22.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY