International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 06, June-2019

Mathematical Implementation for Checking Data Correctness in Cloud Storage System

Sahil Dalwal
Department of CSE
Shoolini University

Associate Prof. Pankaj Vaidya
Department of CSE
Shoolini University

Abstract: **Cloud storage provides a convenient means of storing and retrieval of huge amount of data. With this facility, organizations and individuals can outsource their data to cloud. Though cloud storage gives significant benefits to users, there are many security concerns as well. Since the cloud users store their valuable business data in cloud, the security is an outmost concern. the cloud server has to be untrusted since it is accessed through a public network . In this paper we will show the mathematical implementation for checking the data integrity in cloud storage system. We propose the data integrity scheme in which a third party can audit the data stored in the cloud and assure the customer that the data is safe. This scheme ensures that the storage at the client side is minimal which will be beneficial for thin clients and it is also useful for peer- to - peer storage system, network file system, long term archives, web- service object stores, and database systems. Such verification systems prevent the cloud storage archives from misrepresenting or modifying the data stored by using frequent checks on the storage archives and The empirical results revealed that the implementation is capable of preventing loss of integrity of data in cloud. The further work will be done to improve the scheme for auditing multiple files from multiple clients simultaneously.**

Keywords: Cloud Computing, Cloud Storage, Data Integrity, Data Availability, Outsourcing.

1. INTRODUCTION

Cloud computing become a very emerged and integrated field in today's society as we all are using communication technology and networking in any way, whether it is our mobile phones, cctv camera or internet. Internet has widely changed the computing world. Starting from parallel computing to distributed computing and then to grid computing and finally to cloud computing. Cloud computing originated from Telecommunications Company which began to offer VPN (Virtual Private Network) using point to point data circuits. Cloud computing is available in three different offerings: cloud computing, cloud storage, and anything as a Service (XaaS). As we know that these days cloud has become a new model for delivering resources such as computing and storage to customers on demand. Rather than being a new technology in itself, the cloud is also a new business model with new technologies such as virtualization that take advantage of economies of scale and multi – tenancy to reduce the cost of using information technology resources. Just like cloud computing, cloud storage has also been rising in fame in recent times due to several of the same reasons as cloud computing. Cloud storage is a new business model for delivering virtualized storage to customers on demand. The formal term proposed by the Storage Networking Industry Association (SNIA) for cloud storage is Data Storage as a Service (DaaS) - as

" Delivery over a network of appropriately configured virtual storage and related data services, based on a request for a given service level."

A cloud storage service presents a container for data, and the user does not really care how the cloud provider implements, operates, or manages their resources within the cloud. A client, via the network, makes requests to the cloud storage to securely store and subsequently retrieve data at an agreed level of service. Although seemingly abstract and complex, cloud storage is actually simple. Cloud storage represents resources that are provided in potentially small increments with the appearance of unbound capacity. We have found many problems or risks that are occurred in cloud storage system. One of them is that users may not be able to maintain local copy of the data. When such data lost, they will not be able to take it back. There are some possibilities for fraudulent activities at CSP. CSP may hide the storage inconsistencies in cloud; he may even remove some of the information of users to rent that space for monetary gains.

In order to ensure that data stored in cloud storage system has integrity and availability, many methods came. Most of them depend upon on demand verification of data for its correctness. As cloud data is not historical in nature. It does mean that it is subjected to frequent modifications. Means data might be updated by data owners. As we can't trust fully on CSP, so they appoint a Third Party Auditor to check the availability of data and its correctness. In this, TPA verifies the data stored in the cloud and communicates with the clients.

In this paper we propose a mathematical implementation for checking data integrity and availability of data. Such implementation is useful for peer to peer storage system, network file systems, web service object stores, and data base systems. Such verification system prevents the cloud storage libraries from misrepresenting or modifying the data stored by using frequent checks on the storage archives.

2. RELATED WORK

Many security mechanisms came into existence as found in the literature. Proof of irretrievability [1] was proposed by Juels and Kaliski. It combines error correcting code and spot checking to ensure data integrity and irretrievability. On this

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 06, June-2019

model is built and homomorphic authenticator based on the random linear function is used for secure cloud services. This resulted in secure mechanisms with less cost. An improved framework is proposed by Bowers et al. Later they also extended this to apply in distributed environment [2]. The commonality of entire scheme is that they focused on static data. Their computational and communication cost is high. The reason for this is that every operation is done through error correcting code and random shuffling process. Of late Dodiset al. [3] studied the scheme to improve them. Provable Data Pocession (PDP) was developed by Ateniese et al. for untrusted storages. They used homomorphic tags which are based on public key cryptography for auditing the files. It proved to be computationally expensive as it takes more time to pre-compute tags. Later they improved the scheme for symmetric key-based cryptography [4]. This proved to have less overhead. However, the drawback of this scheme is that it worked only on single server. In case of server failure it could not provide guarantee for data. In two more recent studies and there was research on data dynamics. A skip-list based scheme was developed by Erway et al. [5] For provable data procession. However, this work comes under the traditional data integrity in which it is mandatory to maintain a local copy. This scheme covered multiple servers and ensured robustness of data. Another scheme known as P2P backup proposed by Lillibridge et al. Uses erasure codes for security. It also makes use of cryptographic keys. It can detect the data inconstancies in cloud storage. The drawback of their approaches that it needs exponentiation over the entire data file. This is not practical in the real world. In [6] and TPA (Third Party Auditing) was used to ensure data consistencies. Their scheme works for only encrypted files while auditors need to maintain the state for long time.

Recently Wang et al. proposed flexible techniques that work in distributed environment. This technique uses erasure correcting code for data integrity. It also sources data from Byzantine servers. This technique achieves security with less communication and computational overhead. Homomorphic tokens are also used. Error localization used in the mechanism. It enables the mechanism to identify misbehaving servers. Data Integrity Proofs in Cloud Storage by Sravan kumar and Asthosh Saxena provides a scheme for static storage of data. [7]. Juels et al. [8] described a formal “proof of retrievability” (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error-correcting code to ensure both possession and retrievability of files on archive service systems. Shacham et al. [9] built on this model and constructed a random linear function based on homomorphic authenticator which enables unlimited number of queries and requires less communication overhead. Bowers et al. [10] proposed an improved framework for POR protocols that generalizes both Juels and Shacham’s work. Later in their subsequent work, Bowers et al. [11] extended POR model to distributed systems. However, all these schemes are focusing on static data.

3. PROPOSED SCHEME

We propose a mathematical implementation for checking data correctness in the cloud. It involves the encryption of the few bits of data per data block therefore dipping the computational cost on the customers. This is based on the fact that high probability of security can be achieved by encrypting fewer bits instead of encrypting whole data. The customer storage operating cost is also minimizing as it does not accumulate any data through it and it reduces bandwidth requirements. Hence our scheme is useful for small memory devices and low power devices.

In our data integrity protocol the TPA wishes to accumulate just a particular key irrespective of the range of the information folder and two functions that produce an arbitrary series. The TPA does not store any data with it. What TPA does that before storing the file at the library TPA will preprocesses the file F and then he append some metadata to the file and stores that file to the library. After this, at the time of confirmation TPA will uses this metadata to confirm the reliability of user’s data. Now it is essential to note that our confirmation of data reliability performance will just checks the reliability of data. But the data can be stored, that is duplicated at redundant data centers to prevent the data loss from natural calamities. If the data has to be modified which involves updation, insertion and deletion of data at the client side, it requires an additional encryption of fewer data bits. So this scheme supports dynamic behaviour of data.

We have started my work by assuming the numerical values for the proposed scheme. Later on these numerical values are changed into bits.

4. CORRECTNESS PROOF BY SELECTING RANDOM BITS IN DATA BLOCKS

It involves the two phases:

1. **Setup phase:** Setup phase includes the generation of metadata and its encryption or appending of metadata.

I. Generation of Metadata: Let g be a function define as follow

$g(i,j) \longrightarrow \{1 \ldots .. M\}, \quad i \in \{1 \ldots n\}, \quad j \in \{1 \ldots k\}$

II. Encryption of metadata:

$H: i \longrightarrow \alpha, \alpha \in \{0 \ldots 2n\}$

To get a new k bit number M we will add α to get a new K bit number M

$M = k + \alpha$

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 06, June-2019

III. APPENDING OF METADATA:

After generating all the metadata bit blocks. We will concatenate the encrypted metadata. And these concatenated metadata then appended to the data file F before store the file F at the cloud storage server. Thus, when the file F appended with metadata then it becomes F' and it will be archived with the cloud.

Fig. 4.1 Encrypted file F' after Appending the Metadata to the File F.

2. **Verification phase:** Verification phase includes issuing a challenge to the cloud server and getting a response and checking its validity.

SETUP PHASE:
Let the verifier wishes to store the file F on the cloud server with the files. Let this file F consist of n number of data blocks. Then, first we initially preprocess the file and then we will make the metadata which will be append to the file F. Now, let's take that each of the n data blocks contains m number of bits in them.

The data file is named as F.

Number of blocks n in each file is 10.

Each block comprises of m bits that is 20 bits..
K number of bits out of m bits of 10 blocks is selected for the construction of Meta data.

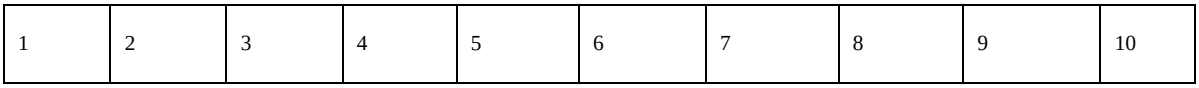
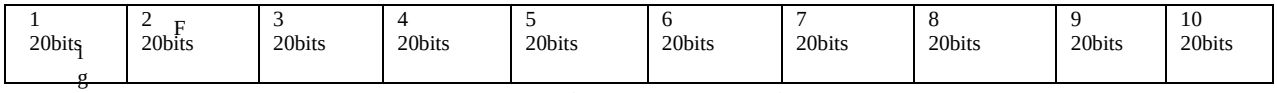


Fig. 4.2 File F contains 10 no. of blocks.



. 4.3 K bits of each block < m bits of each block

FOR DATA BLOCK 1st:
i = 1, m = 20 bits, K (metadata) = 12
m = 1 0 0 1 0 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0
K = 1 0 0 1 0 1 0 1 0 1 1 1
Now, for encrypting the metadata, h is function that generates a K bit numeral α. This function is a secret and known only to the TPA.
H: i ⟶ **α,** where α ϵ {0…. 2n}
Then**, α = 20, α = 1 0 1 0 0**
For the metadata m of **1st** block the number α is added to obtain a new K bit integer M.
M = K + α

```
 1 0 0 1 0 1 0 1 0 1 1
+            1 0 1 0 0
-----------------------
1 0 0 1 0 1 1 0 1 0 1 1
```

So, the new metadata is

M1 = 1 0 0 1 0 1 1 0 1 0 1 1

FOR DATA BLOCK 2nd:

m = 0 0 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 1 0 0 0

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 06, June-2019

K = 0 0 1 1 1 0 1 0 1 1 0 0

Now, for encrypting the metadata, we will follow the same procedure as mentioned above.

α = 25, α = 1 1 0 0 1

M = K + α

```
0 0 1 1 1 0 1 0 1 1 0 0
+             1 1 0 0 1
-------------------------
0 0 1 1 1 1 0 0 0 1 0 1
```

So, the new metadata is **M2 = 0 0 1 1 1 1 0 0 0 1 0 1**

The same procedure well be followed for data block 3 to data block 9. For every data block we just have to change the values of α.

FOR DATA BLOCK 10th:

m = 20, K = 12

m = 1 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0

K = 1 1 1 1 0 0 0 1 0 0 1 1

α = 40, α = 1 0 1 0 0 0

M = K + α

```
1 1 1 1 0 0 0 1 0 0 1 1
+         1 0 1 0 0 0
---------------------
1 1 1 1 0 0 1 1 1 0 1 1
```

So, the new encrypted metadata is M10 = 1 1 1 1 0 0 1 1 1 0 1 1

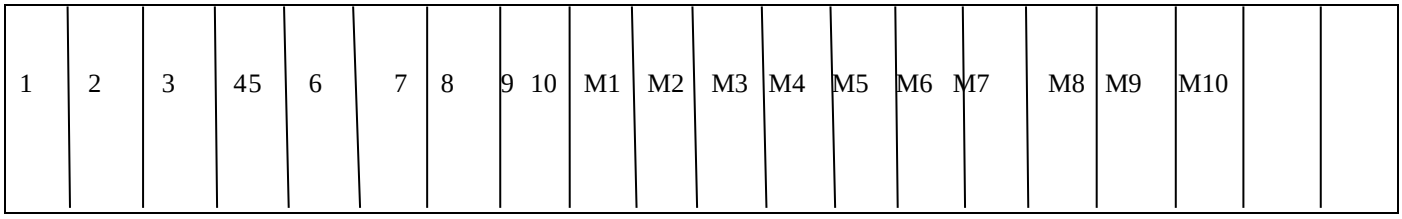


Fig. 4.4 the encrypted file F after appending the metadata to the data file F.

VERIFICATION PHASE:

Let the TPA wants to confirm the reliability of the file F. it throw a challenge to the library and asks it to reply. The challenge and the answer are comparing and if the result is TRUE the TPA accepts integrity proof. Else if the result of comparison is FALSE it rejects the integrity proof. This condition is applied when user wants to check the integrity of the whole file F.

My work is related to check the integrity of a specified data block of a data file F. in this condition suppose a user wishes to check the integrity of the 1st block. The TPA challenges the cloud storage server (CSP) by specifying the block number i.e 1st block and a bit number 12 created by via the function g which simply the TPA knows. After this, TPA also specifies the location at which the Meta data equivalent to the block 1st is append. Hence cloud storage server is required to send the bits for verification by the client. Then metadata send by the cloud is decrypted by via the numeral α. The corresponding bit in this decrypted metadata is compared with the TPA's encrypted metadata. Any difference between the two would mean a failure of the integrity of the client's data at the cloud storage.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 06, June-2019

VERIFICATION OF 1st DATA BLOCK:

First TPA challenges the css by specifying the block number 1st and a bit number 12 or also specifies the position of metadata corresponding to the block 1st. now, css has sent the required metadata. i.e .
M1 = 1 0 0 1 0 1 1 0 1 0 1 1
Now, this metadata will be decrypted by using the number α. So the value of α for the 1st data block is 20. i.e. α = 1 0 1 0 0
And, **K = M1 – α**

	1 0 0 1 0 1 1 0 1 0 1 1
-	1 0 1 0 0
	1 0 0 1 0 1 0 1 0 1 1 1

So, the decrypted metadata is Md = 1 0 0 1 0 1 0 1 0 1 1 1.

Now, we have to compare this css's decrypted metadata with the TPA's encrypted metadata.
I.e. **k = 1 0 0 1 0 1 0 1 0 1 1 1**
And **Md = 1 0 0 1 0 1 0 1 0 1 1 1**
Hence, the result of comparing the bits is TRUE. The TPA will accept the integrity proof. Means client's data is correct. There is no loss of integrity.

VERIFICATION OF 2nd DATA BLOCK:

M2 is the metadata that is sent by the cloud server. The term M2 is already defined above in the setup phase. M2 is encrypted metadata generated by the function h and this function is a secret and is well-known only to the TPA.
M2 = 0 0 1 1 1 1 0 0 0 1 0 1
Now, this metadata will be decrypted by using the number α. So the value of α for the 2nd data block is 25. i.e.,
α = 25, i.e. α = 1 1 0 0 1
K = M2 – α

	0 0 1 1 1 1 0 0 0 1 0 1
-	1 1 0 0 1
	0 0 1 1 1 0 1 0 1 1 0 0

So, the decrypted metadata is Md = 0 0 1 1 1 0 1 0 1 1 0 0

Now, we have to compare this css's decrypted metadata with the TPA's encrypted metadata.

K = 0 0 1 1 1 0 1 0 1 1 0 0
Md = 0 0 1 1 1 0 1 0 1 1 0 0
Hence, the result of comparing the bits is TRUE. The TPA will accept the integrity proof. Means client's data is correct. There is no loss of integrity.
The same procedure will be followed to check the data integrity of rest of the data blocks. i.e. from 3rd data block to 9th data block.

VERIFICATION OF 10th DATA BLOCK:

M10 is the metadata that is sent by the cloud server. The term M10 is already defined above in the setup phase. M10 is encrypted metadata generated by the function h and this is a secret.
M10 = 1 1 1 1 0 0 1 1 1 0 1 1
α = 40, α = 1 0 1 0 0 0
K = M10 – α

	1 1 1 1 0 0 1 1 1 0 1 1
-	1 0 1 0 0 0
	1 1 1 1 0 0 0 1 0 0 1 1

So, the decrypted metadata is Md = 1 1 1 1 0 0 0 1 0 0 1 1.

Now, we have to compare this css's decrypted metadata with the TPA's encrypted metadata.

K = 1 1 1 1 0 0 0 1 0 0 1 1
Md = 1 1 1 1 0 0 0 1 0 0 1 1
Hence, the result of comparing the bits is TRUE. The TPA will accept the integrity proof. Means client's data is correct. There is no loss of integrity.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 06, June-2019

5. CONCLUSION AND FUTURE WORKS

The aim of this implementation is to identify cloud storage security and propose an efficient solution or scheme through which we can facilitate the clients regarding getting a proof of integrity of the user's data which user wants to store on the cloud storage servers with lesser costs and efforts. Through this scheme we have defined the work of a cloud. Means how a cloud works and how CSP and TPA linked to cloud. The motivation behind this research lies in the fact that for many organizations the final barrier to adopting cloud computing is whether it is sufficiently secure.

This paper examined the user's data protection requirements and TPA's work or CSP work, and security or privacy risks. An important conclusion of my work is that cloud is inherently neither secure nor insecure. The most important factor that a CSP can provide is the quality of management applied – just as this is the most important factor in any IT environment. After analyzing cloud storage security and privacy issues, user's data protection, and scheme for checking user's data correctness in cloud, we came up with our proposed mathematical implementation for checking the user's data correctness in the cloud on the parameters of integrity.

The following are the key aspects of my proposed implementation:

- Facilitate the client for getting a proof of integrity
- Computational and storage overhead of client as well as cloud is exceptionally reduced.
- Advantageous to thin clients like PDA's, Mobiles, and Laptops.
- Network bandwidth is minimized as the size of proof is comparatively less.
- Encryption of Meta data by using few no. of bits.
- Appending of Meta data
- Verification of bits
- Data Integrity check

FUTURE WORK

This proposed scheme reduces the computational and storage overhead at the client side as well as the computational transparency of the cloud storage server. This scheme also decreased the extent of the confirmation of data reliability which also helps to reduce the network bandwidth consumption. Means the storage at the client side is very much minimal. This feature proofs that are scheme is beneficial.

In this implementation we have worked only on two functions, one function is the bit generator function g, and the second function is h which will be used for encrypting the data. Many of the schemes which are proposed earlier have some lack of requirements. Like existed schemes require the archive to perform tasks that generally consumes a large computational power.

- Further research could be conducted for auditing multiple files from multiple clients simultaneously.
- This mathematical implementation can also be done through programming in any specific language like c, c++ etc.
- Another relevant research could be take place for dynamic data in cloud storage e.g. update, delete and append the data stored in the cloud.

This research is limited to data integrity risks in the cases of storing and sharing data between the cloud providers. Since there are always some compliance with the security policies and agreements, the thesis will concentrate on those requirements that are related to my work.

6. REFERENCES

[1] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.
[2] K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. ACM Conf. Computer and Comm. Security (CCS '09), pp. 187-198, 2009.
[3] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," Proc. Sixth Theory of Cryptography Conf. (TCC '09), Mar. 2009.
[4] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc.Fourth Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '08), pp. 1-10, 2008.
[5] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), pp. 213-222, 2009.
[6] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.
[7] Ashutosh Saxena, Sravan Kumar "Data Integrity Proofs in Cloud Storage" 978-1-4244-8953-4/11 ,2011 IEEE
[8] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. of CCS '07, pp. 584– 597,2007
[9] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. of Asia crypt '08, Dec. 2008.
[10] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Cryptology ePrint Archieve, Report/175, 2008, http://eprint.iacr.org/.
[11] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)