International Journal of Grid and Distributed Computing
Vol. 11, No. 10 (2018), pp.65-74
http://dx.doi.org/10.14257/ijgdc.2018.11.10.06

# Certificateless Dynamic Data Integrity Verification using Lattices in Cloud Storage

Sasikala Chinthakunta[1] and Shoba Bindu Chigarepalli[2]

[1]*Dept.of CSE, JNT University Anantapur-515002, Andhrapradesh, India*
[2]*Dept.of CSE, JNTUA College of Engineering, Anantapur-515002, Andhra Pradesh, India*
*sasikalareddy27@gmail.com, shobabindhu@gmail.com*

***Abstract***

*Because of the data outsourcing in cloud, user loss his control and physical possession of the data. So, data integrity and security is a major problem in Cloud Storage. To solve this problem several Remote Data Integrity Checking (RDIC) schemes have been proposed, but most of them do not support the integrity verification over dynamically updated outsourced data. This paper, presents a Certificateless dynamic data integrity checking scheme using Merkele Hash Tree (MHT) data structure and Lattice based homomorphic linear signatures. MHT is an authenticated binary search tree used for block authentication to perform data update operations such as Modification, Insertion and Deletion. And, we apply Random masking technique to preserve the privacy against TPA. The experimental results of the proposed scheme are compared with the state-of-the art; they show the efficiency of the method.*

***Keywords:*** *Cloud storage, Data dynamics, Integrity, Merkele Hash Tree, Third Party Auditor*

## 1. Introduction

One of the prominent services of the Cloud Computing is cloud storage. It permits the users to save their data on remote servers maintained by a third party. With this new model of computing, users enjoy on-demand high-quality services from a cloud, without any difficulty of local storage management and maintenance. Although cloud resources are far more powerful and reliable than user resources, many organizations are scared to shift to the cloud environment because of multi-tenancy, loss of control over the data and lack of trust on Cloud Service Provider (CSP). For monetary reasons, an untrusted CSP may delete the data that is not used or rarely accessed to save the storage space and even to maintain reputation he may protect data loss occurences. More ever power outages, Server failures, and security attacks of prominent cloud services occurred from time to time [2-3]. These issues make data integrity protection is a challenging task in cloud environment. Hence, constructing a new RDIC protocol in cloud environment is crucial to check the integrity of the data.

In the recent years, researchers introduced several RDIC protocols using Public Key Infrastructure (PKI). In those protocols, to initiate the integrity checking the Third Party Auditor (TPA) validate the certificate of the data user for authentication. In the cloud environment, public key certificate management (generation, storage, update, and revocation) is the responsibility of the user and it also brings a burden on TPA regarding computation and communication cost to verify the certificate. Usually, the major issue with PKI is certificate management. Thus, this type of RDIC protocols has become a serious

Received (April 15, 2018), Review Result (July 24, 2018), Accepted (September 14, 2018)

ISSN: 2005-4262 IJGDC

problem especially for source constrained users. In their further research work) to deflect certificate management issue in PKI based methods researchers proposed Identity based RDIC schemes using Identity-Based Signatures (IBS. Unfortunately, one of the major drawback of IBS is the key escrow problem because of its dependency on Private Key Generator (PKG) while generating the private keys. Therefore, by using Certificateless Signatures (CLS) instead of IBS, TPA can check the integrity of the outsourced data without suffering from certificate management problem in PKI. In CLS, private keys are created by combining the partial private key generated by PKG along with users secrete information, and the data owner's identity such as her email address or name is considered as the public key. Hence, in our method there is no need of certificates to validate the owner's public key, which is used to start the integrity verification task in the cloud.

Due to the dynamic nature of the data in the cloud, RDIC protocols should provide the support dynamic data operations such as insertion, deletion and modification. Therefore, a different type of data structures such as hash index tables, Double Linked Lists is used in the protocol design to achieve this goal. However, the data structures that are applied to RDIC protocols cannot effectively support dynamic data update operations, especially for periodic data update applications. So, this paper aims to propose a method to support dynamic data operations by using Merkle Hash Tree (MHT) that can minimize the computation cost over the user/TPA.

The rest of the paper is organized as follows: We describes the related works in Section 2, Section 3 defines the framework of the proposed method. In Section 4, we illustrate the implementation of dynamic data operations. In Section 5, we compare the experimental results of our protocol with the state- of- art. Finally, in Section 6 we discusses the conclusions of this work.

## 2. Related Work

Ateniese *et al.,* [4] first introduced Provable Data Possession scheme to check the integrity of the outsourced data using Homomorphic Verifiable Tags (HVTs). But, their scheme suffers from high communication and computation overhead and it does not support dynamic operations. Jules *et al.*, [5] addressed Proof of Retrievability (POR) method based on sentinels. But the main disadvantage of this method is, numbers of challenges are limited. In the further research, Ateniese *et al.,* [6] Proposed scalable PDP scheme using symmetric-key cryptography, but it supports partial dynamic data operations only. Bowers *et al.*, [7] proposed High Availability and Integrity Layer (HAIL), the main drawback of this approach is providing support for private verifiability only. Wang H. [8] Designed PDP model based on Proxy, here verification task dependent on proxy instead of user or TPA. In the recent years, many RDIC protocols have been focused to address the above issues [9-11].

Usually, the outsourced data in the cloud might be not only accessed but also updated frequently by users for various purposes. Hence, a RDIC protocol must provide the support for data dynamic operations such as modify, insert, delete without downloading the entire file. In [12], authors introduced a new RDIC protocol, which supports dynamic data update using Symmetric key operations. However, this method is recommended for modify, insert and delete operations, but it needs to re-compute all remaining blocks to update data blocks thus it imposes high computational cost on the user, therefore it is not suitable for large files. Erway *et al.*, [13], proposed a RDIC protocol to support fully dynamic operations by combining Skip list, dictionary and rank based information. However, this method failed to verify the integrity of the individual block. Wang *et al.*, [14] presented RDIC method to support dynamic operations using Merkle Hash Tree based on bilinear aggregate signatures. However, this method is vulnerable to leak data to the TPA and it has imposed a higher computation cost on the auditor due to large number of pairing operations. To solve the privacy issue in [14] Yang *et al.*, [15] proposed an efficient data integrity checking protocol

 

to support data dynamics using hash index table indices for each block of the outsourced data. However, to insert or delete a block, the user/TPA has to know the exact position of the block and then change the positions of the remaining blocks to the accurate positions this process incurred high computation cost on the user/TPA.

In this paper, we extend our work "Certificateless remote data integrity checking using lattice [1]" to support dynamic data operations using MHT structure for block authentication and while generating the proof to preserve the privacy against TPA we apply random masking technique. Our scheme involves lattice based signatures instead of pairing based signatures because they include matrix-matrix or matrix-vector multiplications or additions only. So, it significantly reduces the overall computation cost over the TPA and the CSP.

## 3. Frame Work of Certificateless Dynamic Remote Data Integrity Checking

The frame work of Certificateless dynamic data integrity checking protocol is illustrated as follows.

**Setup** (n): Given security parameter n, first the PKG runs TrapGen(n,m,q) algorithm to get a matrix A ∈ $Z_q^{n\times m}$ along with a short trapdoor basis $\mathbf{T}_A$ of the lattice $L_q^{\perp}$(A).

**Extract-Partial-Privatekey**(PParams,msk, ID): On input the public parameters PParams, the master secrete key and the user ID, PKG executes the Samplebasis (A, $\mathbf{T}_A$, s, $H_1$ (ID) ) algorithm to get a matrix $P_{ID} \in Z_q^{m\times k}$ and send it to the user and the user sets his partial private key as $p_{ID}$=$P_{ID}$.

**Set-Secrete-value** (PParams, ID): Given PParams and user ID, the user randomly chooses a matrix $S_{ID} \in Z_q^{m\times k}$ (which must satisfy $\| S_{ID} \| \leq$ b, where b is a positive integer) and set it as his secrete value $s_{ID}$ = $S_{ID}$.

**Set-Privatekey**(PParams,$p_{ID}$,$s_{ID}$): It takes the public parameters and users partial private key and secrete value as the input and the user computes his full private key sk=( $P_{ID}$,$S_{ID}$).

**Set-Public key(**PParams,$s_{ID}$): Given the public parameters and user secrete value as input the user computes his public key Pk=A$S_{ID}$.

**SignGen (**PParams,F,ID, sk**,** $\boldsymbol{f_i}$**)**: This algorithm is run by data owner, takes *PParams*, owner's identity *ID*, private key *sk*, meta information of file *F* and the data block $\boldsymbol{f_i}$ as input and outputs the signature of the data block $\sigma_i$.

**Challenge(id,F**): This algorithm is run by TPA, takes File *F* and File identity *id* as input and outputs the challenge message as chal={id, $f_i$} and uploads it to server.

**ProofGen(**PParams,F, $\boldsymbol{\phi}$,chal): This algorithm is run by cloud server, takes the *PParams*, meta data of file *F*, signature set $\phi$, and challenge as input and it outputs the proof of possession *P*.

**Proofcheck**(PParams, ID, F, Proof, Chal): This algorithm is run by TPA, takes the *PParams*, owner's identity *ID*, file F, chal, and the proof *P* as input, and verifies whether file is intact or not.

 

## 4. Certificateless Dynamic Data Integrity Checking

Cloud storage allows the users not only to access the outsourced data but also to update the data used by users for various application purposes. Therefore, RDIC that supports dynamic data update is also important. To achieve this, the scheme is constructed as follows: i) Generally, in the set phase itself data owner construct the MHT and further it is managed by TPA; ii) Whenever the dat owner wants to update the data, it sends an update message to the TPA for MHT updation; iii) Upon receiving the verification request from the data owner, TPA computes the challenge message and forward it to the CSP; iv) Using ProofGen algorithm, CSP computes the response including the updated block and its auxiliary authentication information and send it to the TPA; v) After getting the response from the CSP, first the TPA authenticate the data block using the MHT root and the auxiliary authentication information. If it fails then TPA output false, otherwise TPA performs integrity checking , if its false then quit otherwise TPA forwards the conformation message to the data owner. It indicates that, the data on the server and the abstract information on the TPA both are up-to-date.The Certificateless dynamic data integrity checking protocol involved four phases such as: i) Key generation phase; ii) Signature generation phase; iii) Dynamic data operations; iv) Verification phase. All the phses except Dynamic data operations are similar to the our work in [1]. So we focus on the implementation of dyamic data operations in detail in the following section.

### 4.1. Support for Dynamic Data Operations

Support for dynamic data operations is a one of the prominent feature of the RDIC protocols, it enables the users to update their outsourced data without having to download the entire file. In this section, we explain Merkle Hash Tree (MHT) data structure to perform dynamic update operations efficiently. The algorithm for dynamic data integrity is presented in Figure 1.

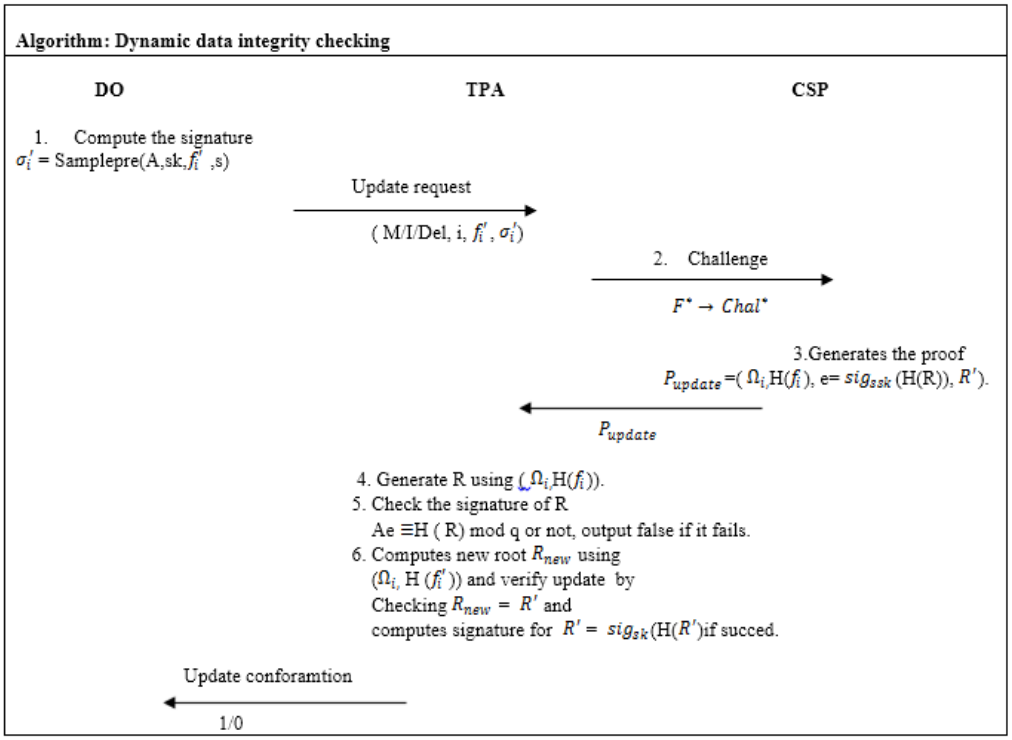


**Figure 1. Algorithm for Dynamic Data Integrity Checking**

#### 4.1.1. Data Modification (M)

It is the one of the frequently used operation in cloud storage. It referes to the replacement of specified blocks with new ones without having to download the all the blocks. Let the user/data owner wants to modify the block $f_i$ to $f_i'$ using the proposed method, then he executes the following steps.

**Step 1:**

i)First, the data owner/user computes the signature for the new block $f_i'$ as $\sigma_i'$ = Samplepre(A,sk,$h_i'$,s).

ii) Then construct the update message as (M, i,$f_i'$, $\sigma_i'$) and forwards it to the TPA.Where M represents the Modification operation.

**Step 2:**

Upon receiving the update message form the user, TPA executes the following steps to perform update operation:

i) The block $f_i$ is replaced with $f_i'$ and outputs $F'$, also $\sigma_i$ is replaced with $\sigma_i'$ and outputs $\emptyset'$.

ii) Then replace the H($f_i$) with H ($f_i'$) in the MHT and computes the new root $R'$ as shown in Figure 2.

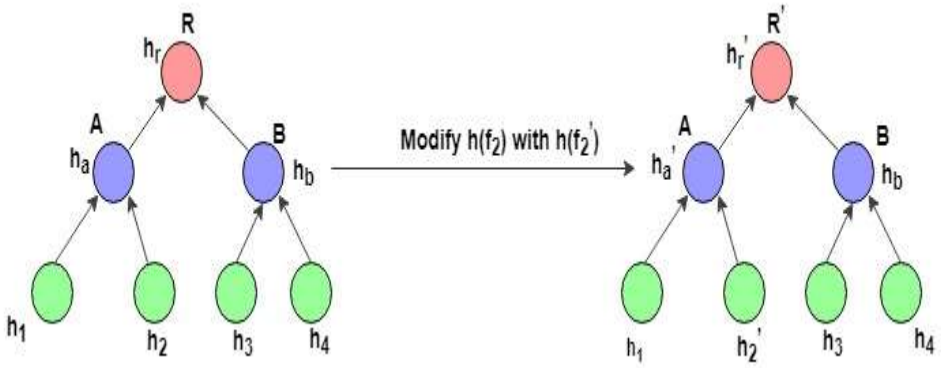


**Figure 2. Impact of Modify Operation**

iii) Upon receiving the verification request from the data owner, TPA send the challenge message to the CSP.

iv) As a response to the challenge server sends the Proof for the update operation as

$P_{update}$ = ( $\Omega_{i,}$H($f_i$), e=Sign(ssk,H(R), $R'$) to the TPA. Where $\Omega_i$ is the authentication information for the data block $f_i$ in the old MHT.

**Step 3:**

i) After getting $P_{update}$ message from the CSP, the TPA tries to construct root R using ($\Omega_{i,}$H($f_i$)) and verify the signature of R by checking Ae=H ( R) mod q.

ii) If the above equation fails, output false otherwise it can further computes the new root by using ($\Omega_{i,}$ H ($f_i'$)) and then compare the new root value with $R'$. If it is true, then TPA sends the update conformation to the data owner. Then the data owner computes the signature for new root as (sign(sk,H($R'$)) and forwards it to the CSP for update.

 

#### 4.1.2. Data Insertion

It allows a user to insert a new data block at a specified position in the file. Suppose the user wants to insert a new data block $f^*$ after the i [th] block $f_i$ of the file F. The procedure for inserting a new block after i [th] data block is as follows.

**Step 1:**

i) The data owner generates the signature for $f^*$ as

$$\sigma^* = \text{Samplepre}(A,sk,h_i^*,s).$$

ii) Then computes the update message as ( I, i, $f^*$ , $\sigma^*$) and forwards it to the TPA, where I represents the Insertion operation.

**Step2:**

Upon sreceiving the update message form the user, TPA executes the following tasks.

i)The TPA stores $f^*$ and in the MHT it adds a leaf node h( H($f^*$)) after the leaf node h(H($f_i$)) and outputs the new file $F'$ and new signature set $\emptyset'$ by adding $\sigma^*$ to it.

ii) Then computes the new root $R'$ based on MHT as shown in Figure 3.

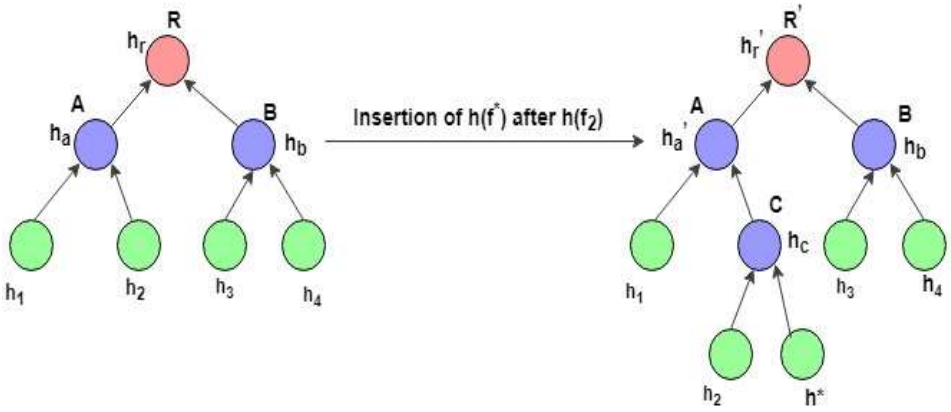


**Figure 3. Insertion of a New Block After i [th] Data Block**

iii) Upon receiving the verification request from the data owner, TPA send the challenge message to the CSP.

iv) As a response to the challenge, server sends the Proof for the update operationa as

$P_{update}$ = ( $\Omega_{i,}$H($f^*$), e=Sign(ssk,H(R), $R'$) to TPA.

**Step 3:**

i) After getting $P_{update}$ message from the CSP, the TPA tries to construct root R using ($\Omega_{i,}$H($f_i$)) and verify the signature of R by checking Ae=H ( R) mod q.

ii) If the above equation fails, output false otherwise the user check whether the server performed the insertion operation as required or not by computing the new root by using ($\Omega_{i,}$ H($f_i$),H ($f^*$)) and then compare the new root value with $R'$. If it is true, then TPA sends the update conformation to the data owner. Then the data owner computes the signature for new root as (sign(sk,H($R'$)) and forwards it to the CSP for update.

 

#### 4.1.3. Data Deletion

The delition operation is the reverse of insertion. It refers to deleting a specified block in the file F. Let the user wants to delete a block $f_i$ from the file, then he sends the update message to the TPA as (D, $f_i$). Then TPA removes $f_i$ from its storage space and also deletes the corresponding leaf node h(H($f_i$)) in the MHT and moves the latter blocks one block forward. After that he generates the new root $R'$ as shown in Figure 4.

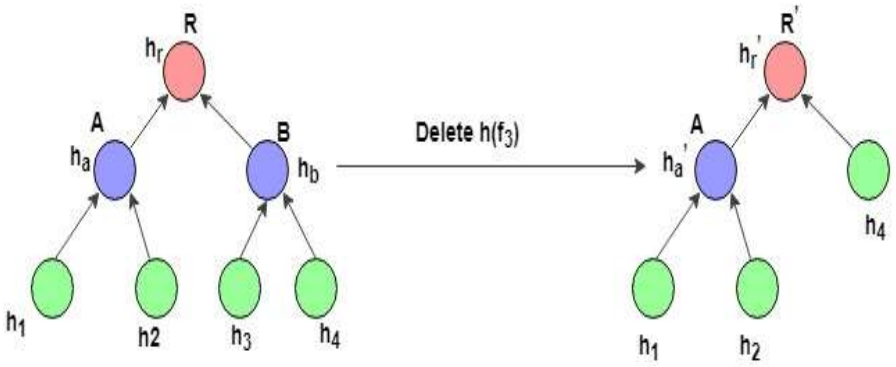


**Figure 4. Deletion of a Data Block From the File**

## 5. Performance Analysis

### 5.1 Evaluation

In this work, dynamic data integrity verification method allows the client to update the dynamic data by using modify ,insert and delete operations. To implement these operations, client must find the exact location of a requested block in the MHT, then computes the signature for that block and then re-balance the MHT depending on dynamic operation. So, the time required to execute dynamic operations is reffered as computational cost of dynamic data update. We now compare the proposed scheme with some existing methods in terms of computation cost, communication cost and computation cost of dynamic data update as shown in Table 1.

**Table 1.Comparision of Proposed Scheme with Previous Schemes**

| Metric | | Scheme | | |
|---|---|---|---|---|
| | | **Yang et al[ 14]** | **Tian et al[15]** | **Our scheme** |
| **Communication cost** | | O( c ) | O ( c) | O ( c) |
| **Computation cost for verification** | CSP | O(cn) | O ( cn ) | cO(log n) |
| | TPA | O (c ) | O (c) | cO(log n) |
| **Computation cost on user** | Modify | O (c ) | O (w) | cO(log n) |
| | Insert | O ( m) | w O (log n) | wO(logn) |
| | Delete | O(m) | O (c) | cO(log n) |

where m is the total number of blocks in a file, n is the number of sectors ,c is the number of challenge blocks,w is the number of update blocks.

The Yang method [14] incurs heavy computation cost for dynamic data update, because they applied Index table data structure, to insert or delete a block i, the user have to shift

 

$(m - i)$ indices in the table. Even though, Tian *et al.*, [15] method is efficient, to reduce the burden on the user while performing insert and delete operations they used dynamic hash table structure. Further to improve the performance we designed certificateless dynamic data integrity verification method based on Merkle Hash Tree structure. As discussed in section 4, to insert or delete a block in our method the user incurs computation cost of O(log n).

### 5.2. Experimental Results

To evaluate the performance of the proposed method, we have used the .NET programming language & NTRU-CRYPTO library by using the system 2.33 GHz Intel Core 2 Duo processor with 3 GB RAM and Ubuntu 14.04 OS. We assume the size of the file as 16 MB is segregated into $l$ = 800 blocks of size20 KB, | n|=20 bits and |q| = 60 bits. Here we consider a different number of updated blocks (c) is a parameter that affects the efficiency of the protocol.

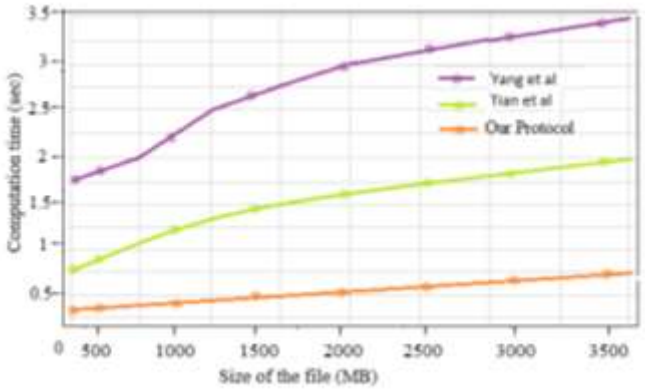


**Figure 5. Comparison of Computation Time in Terms of Frequent Update**

Figure 5, demonstrates the efficiency of the proposed scheme regarding the data update operation. For the insetion or deletion of a block, in the Yang scheme, TPA first find the accurate position of the block then shift the remaining block positions to execute either insertion or deletion operations. If the update operation is performed repeatedly it will induce huge computation cost over the TPA when compared to our scheme,where we need to re calucalate the hash values from leaf to root while performing update operation.

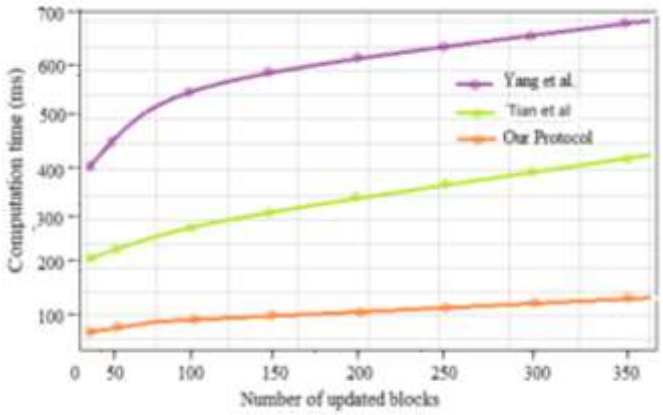


**Figure 6. The Impact of File Size on Dynamic Data Update**

Figure 6, Illustrate the effect of file size ranging from 500MB to 4000MB on computation cost of dynamic data update when 100 blocks of the data are randomly updated in the file. In the Yang method to insert or delete a data block TPA has to shift the more number of blocks that induces huge computation cost over the TPA than the proposed scheme, where we are using MHT structure and Homomorphic signatures to perform insertion or deletion operation that incurs minimum computation cost over the TPA.

## 6. Conclusions

In this paper, we proposed a Certificateless dynamic data integrity checking method to support dynamic data operations. We utilize Merkle Hash Tree(MHT) data structure for block authentication to perform modify, insert and deletle operations.Our protocol utilizes random masking technique to preserve the privacy against the TPA and Homomorphic Linear Authenticators for the integrity verification. And it is implented using lattices, they imposes less computation cost over the TPA and CSP. Our experimental results shows that the proposed method is secure and efficient.

## References

[1] C. Sasikala and C. Shoba Bindu, "Certificateless Remote Data Integrity Checking Using Lattices in Cloud Storage", Journal of Neural Computing and Applications **(2018)**, https://doi.org/10.1007/s00521-018-3546-6,pp.1-7.
[2] A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing", Communications of the ACM, vol. 53, no. 4, **(2010)**, pp. 50-58.
[3] B. Krebs, "Payment Processor Breach May Be Largest Ever", Online at http://voices.washingtonpost.com/ securityfix/ 2009/01/payment processor breach mayb.html, **(2009)**.
[4] G. Ateniese, R. Burns, Curtmola, J. Herring, L. Kissner, Z. Peterson and Dawn Song, "Provable Data Possession at Untrusted Stores", In the Proceedings of ACM CCS, **(2007)**, pp. 598-610.
[5] A. Juels and B.S. Kaliski,"Proofs of Retrievability for Large Files", Proceedings of 14th ACMConf. Computer andComm.Security (CCS'07 ) **2007**, pp. 584-597.
[6] G. Ateniese, R. D. Pietro, L. V. Mancini and G. Sudik, "Scalable and efficient provable data possession", Proceedings of the 4th international conference on security and privacy in communication networks, **(2008)**, pp. 1-10.
[7] K. D. Bowers, A. Juels and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage", Proceedings of the 16th ACM conference on computer and communications security, Chicago, **(2009)**, pp. 187-98.
[8] H. Wang, "Proxy provable data possession in public clouds", IEEE transaction on Service Computing, **(2012)**, pp. 1-8.
[9] C. Wang, Q. Wang, K. Ren and W. Lou, "Ensuring data storage security in cloud computing", Proceedings of IWQoS, **(2009)**, pp. 1-9.
[10] Y. Zhu, H. Hu, G. J. Ahn and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 12, **2012**, pp. 1-14.
[11] S. G. Worku and C. Xu, "Secure and Efficient Privacy-Preserving Public Auditing Scheme for Cloud storage", Journal of Computer and Electrical Engineering [online] **(2014)**, http://dx.doi.org/10.1016/j.compeleceng.2013.10.004.
[12] C. Erway, A. Kupcu, C. Papamanthou and R. Tamassia, "Dynamic provable data possession", Cryptology ePrint Archive, Report 2008/432, **(2008)**.
[13] Q. A. Wang, C. Wang, K. Ren, J. Lou and J. Li, "Enabling public auditability and data dynamics for security in cloud storage", IEEE transaction on Parallel and Distributed systems, vol. 22, no. 5, **(2011)**, pp. 847-859.
[14] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing", IEEE Trans. Parallel Distributed systems, vol. 24, no. 9, **(2013)**, pp. 1717-1726.
[15] H. Tian, H. Jiang and Y. Chen, "Dynamic- Hash-Table based public auditing for secure cloud storage", IEEE Transaction on Services Computing, vol. 10, no. 5, **(2017)**.

 

## Authors

**C. Sasikala**, she is currently a Research scholar of JNT University Anatapur. Her research interests are in the fields of Cloud Computing and Network Security.

**C. Shoba Bindu.** She is presently working as Professor in the Department of CSE, JNTU, Anantapur. She received her B.Tech Degree in Electronics & Communication. Engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in 1997. M.Tech in Computer Science & Engineering from Jawaharlal Nehru Technological University, Anantapur, India, in 2002. She was awarded Doctorate in the year 2010, from JNTU, Anantapur. Her research interests are in the fields of cloud computing, Network Security, Da Analytics and Wireless Communication Systems.