Dynamic List Based Data Integrity Verification in Cloud Environment

Akshay, KC and Balachandra Muniyal*

Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India
E-mail: akshay.kc@manipal.edu; bala.chandra@manipal.edu
**Corresponding Author*

Received 25 January 2022; Accepted 15 April 2022;
Publication 22 July 2022

Abstract

Cloud repository gives a proficient way to fathom issues of management and capacity, driven by high-speed information emergence. Consequently, a developing number of governing bodies and people lean towards storing their information within the cloud premises. In any case, due to the partition of information ownership and administration, it becomes exceptionally troublesome for the users or the owners to verify the integrity of data in a routine way. Hence, numerous analysts center on creating various protocols, that remotely check the astuteness of the information saved within the cloud. In this respect, a conceivable solution is proposed for dynamic reviewing by making use of a dynamic list-based index table to verify the integrity of the data which is more efficient than the state of the arts. Besides, with such a verification structure, it is proven that communication cost and storage cost at the client side is diminished effectively. The statistical analysis based on comprehensive tests illustrates that the proposed convention accomplishes the specified properties in comparison with the state of the arts.

Keywords: Cloud computing, cryptography, cloud security, cloud auditing, elliptic curve encryption, data integrity verification, information security.

Journal of Cyber Security and Mobility, Vol. 11_3, 433–460.
doi: 10.13052/jcsm2245-1439.1134

1 Introduction

In this digital era, it is exceptionally critical that, the information kept within the cloud is secure. No harm of any kind to the privacy or integrity of the data can be afforded. Especially after the pandemic COVID 19, there is a huge increase in digital data storage. It is not just sufficient if the data stored is secure, but it is required that the data stored is not modified in any way by a malicious user. Only the data owner has the right to modify the content, if necessary. That is why the confirmation for the integrity of the data stored is important. Data integrity check was performed by the data owners which resulted in the overhead for the owner. So, there is a need for a third-party auditor (TPA) who performed a data integrity check in lieu of the owner. But the risk of using a third-party auditor for a data integrity check is, he must be a trusted or certified, or verified party.

The essential and critical challenge within the cloud environment is keeping up the integrity of the information put away in it. To assert the wholeness of the data, auditing is performed. The challenge here is how efficiently the auditing can be performed. To broach this issue, Proof of Retrievability (PoR) and Provable Data Possession (PDP) accords were put forward by Ateniese et al. [1] and Juels et al. [2]. In an ordinary PDP process, the client, to begin with, produces metadata for a record, and afterward, the record in conjunction with the meta-information is passed into the cloud servers and expels from the local repository. PDP utilizes a challenge-reaction norm, i.e., the client challenges the cloud server and cloud server, in turn, gives verification for the verifier's challenge. PoR is the complementary method of PDP, and it is more suitable for dynamic auditing whereas PoR is suitable for static auditing.

The PDP process is of two types: public and private auditing. In private auditing, data is verified by the owner. The user or owner cannot give convincing data to the cloud server since there is no trust between them. Moreover, it is not recommended that the user does the integrity check frequently because there will not be any use of outsourcing data otherwise. To defeat this problem, public auditing came into existence as proposed by Anisetti et al. [3]. Here TPA verifies the integrity of the information stocked in the server in lieu of the user. Public auditing must achieve certain functional and safety measures such as preserving the secrecy of the data and dynamic scrutiny of the data. Even during the auditing process, the stored data must not be visible, even for the TPA. Moreover, since the cloud environment is used, there are many applications that manipulate the data frequently, there is a need for efficient dynamic auditing.

There were many methods proposed for the dynamic auditing by Erway et al. [4], Wang et al. [5], Zhu et al. [6], Tian et al. [7], Shen et al. [8]. These methods claimed to have reduced the computational cost, but the communication cost is still a problem in those methods.

The data owners delegate the work to TPA to reduce the workload on the owners' end. The data integrity check is performed whenever the user or owner requested the operation by passing the necessary details. There will be a communication process from the TPA to the data proprietor to confirm the validity of the requester. This is performed through various challenge-request processes between the TPA and the data owner. TPA challenges data owner to provide proof of ownership of the data with zero knowledge. Moreover, if requested by the owner, the cloud security provider must furnish evidence of possession of the information.

1.1 Paper Organization

The remaining part of the paper is organized as follows: Section 2 describes the related works relevant to the projected technique. Section 3 lists the research contributions in this paper. Section 4 provides the required background knowledge of the proposed work. Section 5 explains the methodology adopted for efficient data integrity check. Section 6 presents the experimentation performed using the proposed algorithm. Section 7 gives the comprehensive performance evaluations with some existent methods. Section 8 provides the statistical analysis of the proposed work. Finally, Section 9 provides the concluding remarks of this paper along with the future work.

2 Related Works

There are many researches which were emphasized on cloud data auditing. This section provides points of interest on various existing works.

2.1 Conventional Schemes

Atenises et al. [1], proposed an earliest related work, "Provable Data Possession (PDP)". It makes use of RSA – based homomorphic authenticator to verify the integrity of the data. Juels et al. [2], proposed an antonymous work, "Proof of retrievability (PoR)", which checks the integrity of the data, at the same time it ensures the retrievability of the data stored in the cloud using the error-correcting method.

Liu and Zic [26] have proposed a Proof of Retrievability (PoR) scheme that is constructed based on homomorphic encryption schemes. The claim here was that the PoR scheme could retrieve homomorphically encrypted data by generating probabilistic and homomorphic message authenticators. Moreover, the encrypted data can be handled by the cloud directly and the PoR scheme can verify the integrity of such outsourced computations over ciphertexts. The testing is done here based on the prototype which they developed. There was no comparison with any other existing work.

2.2 Schemes Based on Merkle hash Tree

Wang et al. [5] discussed a convention that concurs both public auditing and handling dynamic information. It utilizes Boneh – Lynn – Shacham (BLS) [27] signature in conjunction with Merkle Hash Tree (MHT). Even though integrity of the information is accomplished here, it comes up short to supply privacy to the information put away on the remote storage.

Du et al. [24] proposed a scheme that is a combination of proof of ownership and retrievability. The uploaded files are encoded using the erasure code. To keep the low-cost communication Merkle tree and homomorphic verifiable tags are generated and used. It is claimed that using the scheme reduces the computation cost especially for the larger files, but this method is increasing the storage cost.

2.3 Schemes with Third Party Auditor

Kwon et al. [9] presented a review procedure for dynamic distributed information in a cloud repository. They have a record table overseen by a third-party reviewer which may be a storage expense for TPA. In addition, the uploader should separate the information into chunks, produce labels, direct it to the clients who are sharing the information chunks. The uploader can establish the confirmation handle to TPA. The cost of communication is exceptionally high since there is a lot of message exchange between clients and data uploaders. Every time an information updation happens, the list table is updated within the third-party inspector.

Swapnali et al. [10] explained a strategy in which, encryption is carried out at the proprietor side, by breaking the information into pieces and after that scrambling them, producing the hash labels for each. Hash labels are then used and concatenated to generate the signature for a specific proprietor. Executing these steps, relatively might take more time. In addition, the proprietor side encompasses a lot of actions being carried out. The projected scheme additionally verifies information utilizing a third-party analyst who

has ought to re-create all the errands which are performed by the proprietor to cross confirm digital signature.

Shah et al. [17] projected a cloud storage system that supports the privacy-preserving public auditing and monitoring. They have claimed that their results empower the TPA to perform reviews for different clients at the same time, effectively and additionally review the integrity of information in the cloud. The investigation appears that plans are provably secure and profoundly well-organized. But the paper does not clearly state the computational cost required to implement the protocol.

2.4 Schemes without Third Party Auditor

Yuan et al. [11] extended a strategy to review the integrity for information sharing activities on the cloud defined by client disavowal, open examining, multiuser adjustment, elated error location likelihood, as well as viable computational or communication auditing execution. This strategy is said to resist client impersonation attacks. Here the technique permits multi clients to engage on data that might affect in a pantomime of client and the data can successfully be stolen or clashing information can be modified. But the paper does not clearly state the computational cost required to implement the protocol.

Snehal and group [12] presented a dynamic encryption strategy that does not have a neutral reviewer. The method they have put forward is, the judgment of the information is misplaced when the information is outsourced. At whatever point there is a threat, an encryption method is recognized and executed, depending on the sort and estimate of the information on the cloud. It employs an executive which could be a proxy server to handle the off-base addition or alteration. Moreover, a legitimate component to oversee the keys is not present.

Kaaniche [13] has developed an information integrity check strategy in her dissertation. She has proposed a design called cloudasec system that deals with information sharing within the public network. Confirmation of information ownership is given utilizing two approaches: set homomorphic verification and zero-knowledge proof. Her proposition further exhibits a purview on remote information checks in a cloud environment. But it does not give a purview on dynamic information corroborations.

Li and team [14] extended a lightweight safe information conspire for mobile computing. It employs ciphertext policy-attribute-based encryption (CP-ABE). A normal cloud environment uses the CP-ABE which is an access control technique. They utilized the ideological underpinning of the

access control tree and formed their transformation of the access control tree based on CP-ABE. The description fields have been exploited for the implementation of the lazy revocation method. They report that the overhead or processing time is diminished on mobile devices when information is distributed within the mobile cloud environment. But the paper does not clearly state the computational cost required to implement the protocol.

Anisetti and team [3] presented a scheme using a framework that is prompted by non-functional necessities, for persistent cloud benefit certification. These non-functional prerequisites are sketched out by a model of service beneath certification and a certification specialist. A consistency confirmation is carried out between prerequisites and models which gets to be the premise of a chain of trust upheld by the certification scheme. They have failed to uphold the communication and computational cost required in implementing their protocol.

Mrinal et al. [15] discussed a method to preserve the astuteness of information. They are covering up the information behind the picture while the information is being sent over to the collector. They are verifying only the integrity of the information. Confidentiality of the information is not legitimately dealt with.

Sun et al. [16] utilized a cryptographic primitive indistinguishability obfuscator to plan an open data integrity confirmation scheme; particularly, users' mystery data will be inserted into a jumbled program and hidden from the cloud server.

Akshay et al. [18], have proposed a method where the encryption of the data takes place during the integrity check and an analysis of the same has been performed. The analysis proved that the confidentiality of the data is properly handled and through the digital signature the authentication of the user who is requesting the data integrity check is performed automatically.

Wang and Di [19] proposed multi-agent-based storage in the cloud and a multi-copy data integrity check method. The bilinear mapping method is used to build or generate the keys. They have also proposed a multi-branch tree for authentication which makes use of multi-copy data signature to realize the signing, validation, and confirmation. A directed acyclic graph (DAG) is used which represents the task relationship in the task, allocation of resources, and workflow based on QoS. Based on the QoS preference settings, various jobs are scheduled.

Ganesh and Manikandan [20] designed a method for verifying the integrity and to authenticate over the remote information which is stored in the public clouds. But this scheme is implemented for mobile users.

They have claimed that the communication and computation overheads have been decreased by comparing with the previous works. The authentication process is implemented during the auditing process and during the block modification, deletion, and insertion.

Nithya et al. [21] presented a scheme for furnishing safe storage for the customer data in the cloud. They have used the RSA algorithm and digital fingerprint using the MD5 hash function. They have claimed that data encrypted using the RSA algorithm cannot be modified by a third party.

To summarize, many works or methods have been proposed for checking the integrity of the data stored in the cloud. But few of the works were on the static data stored on the cloud. Few other works did check on the data integrity, but they were implemented or proposed using a third-party auditor. The owner or the user requests for the data integrity check, only then the third-party auditor checks the integrity of the data. Moreover, there would be communication between the TPA and the owner to verify the ownership of the data and in the retrieval of data which is comparatively high. So, in this paper, a technique is proposed that is described and analyzed as a better algorithm. The communication cost and the computation cost of the data integrity check process are analyzed and proven to be better than the existing methods.

3 Research Contribution

The contributions in this paper are listed as below:

1. In the proposed research work, a data integrity verification system in the cloud environment is designed and developed.
2. The data verification process in the cloud auditor module by making use of a dynamic index table is explained.
3. Security analysis of the proposed method concerning forge attacks, replay attacks, and replacing attacks are discussed and validated.
4. A statistical analysis of the execution time required to carry out the verification process for the data of various sizes (KB) is performed and compared with the state of the art.

4 Background

4.1 Preliminaries

Users may have immense assets which are put away to store in the cloud. The cloud service provider has ample storage space and large-scale computing

mechanisms. The cloud auditor does the task of auditing on behalf of the users and provides fair and sincere results. The data owners outsource their data to CSP so that they can have the advantage of reliable remote storage of their data along with high-performance services provided by the CSP and to reduce their own storage/maintenance overhead at the users/owner's end. Since data is stored on the cloud rather than the local systems of the owner, the owner tends to determine the integrity and the properness of the data stored in the cloud.

Cloud auditor is said to be a trusted party and credible, but they are curious. Though the cloud auditor credibly fairly performs the audit, it might be curious about the private information of the user or the owner. It may even try to deduce the contents of the data stored. Moreover, CSP is not considered a trustworthy party. For various reasons such as keeping the reputation or to gain certain benefits, it might not reveal the fact of loss of data or manipulation of the data. There is a possibility that CSP may even try to launch the attacks on the cloud auditor such as:

- **Forge attack:** CSP might attempt to counterfeit the data and the respective tags to overcome the audit.
- **Replacing attack:** CSP may try to pass the audit by replacing the violated block of data along with its identifier tag with some other block of data.
- **Reply attack:** CSP might try to pass the audit by making use of the proof messages which were generated in the previous steps.

To accomplish efficient and secure auditing, the proposed method aims at meeting the following properties:

- Permit the cloud reviewer to look at the rightness and astuteness of the information put away within the CSP.
- If the CSP does not hold the data intact without any manipulation then, such CSP does not pass the auditing process.
- Allow the cloud auditor to perform dynamic auditing even when the users are dynamically manipulating the data.
- Make sure that the cloud auditor does not breach the privacy of the owners' data.
- Ensure that the cloud auditor has the capability of performing the auditing for multiple user integrity check requests.
- Ensure that the verification process happens with minimum communication and computation cost.

4.2 Secure Assumptions

The security of the proposed work is based on the following assumptions:

***Computational Diffie-Hellman (CDH) Assumption*:** Consider G as multiplicative cyclic groups of a large prime order p. Given g^x and g^y, where g is a generator of G and x, y $\in Z_p$, it is computationally intractable to compute g^{xy}. For any probabilistic polynomial-time adversary, ϕ, the probability of solving the CDH problem is negligible, i.e

$$P(\phi_{CDH}(g, g^x, g^y \in G) \Rightarrow g^{xy} \in G : \forall x, y \in Z_p) \leq \varepsilon \tag{1}$$

***Discrete Logarithm (DL) Assumption*:** Let G be the multiplicative cyclic groups of a large prime order p. Given k such that $k = g^x$, where g is the generator of G, and x $\in Z_p$, it is computationally intractable to compute x, the probability of solving the DL problem is negligible, i.e

$$P(\phi_{DL}(g, k \in G) \Rightarrow x \in Z, k = g^x) \leq \varepsilon \tag{2}$$

***BLS-Signature Scheme*:** This method is widely used used for public auditing protocols [4, 17, 19, 25, 28–30], which can enable a public verifier to verify the cloud data integrity without downloading the original data. This method is incontrovertibly secure based in the intractability of CDH problem. This method makes use of bilinear pairing and signature is an element of the elliptic curve group. The signatures generated using the BLS scheme are known as short signatures.

Consider a bilinear map function e: G x G $\rightarrow G_T$ where G and G_T are the groups of prime order, r. Assume that g is the generator of G. Presume an instance of CDH problem, g, g^p, g^q. The pairing function e does not help to generate or compute g^{pq} which is the solution for the CDH problem. Now because of this property it is derived that , it is intractable. But given g^z, it can be verified if $g^z = g^{pq}$ with zero knowledge about p, q, and z by checking if $e(g^p , g^q) = e(g, g^z)$ holds.

5 Methodology

In this section, the nucleus of the proposed list-based method using the user module described in Section 5.1, cloud auditor module detailed Section 5.2, and cloud service provider module in Section 5.3 is presented. Figure 1 depicts the architecture applied during the entire process.

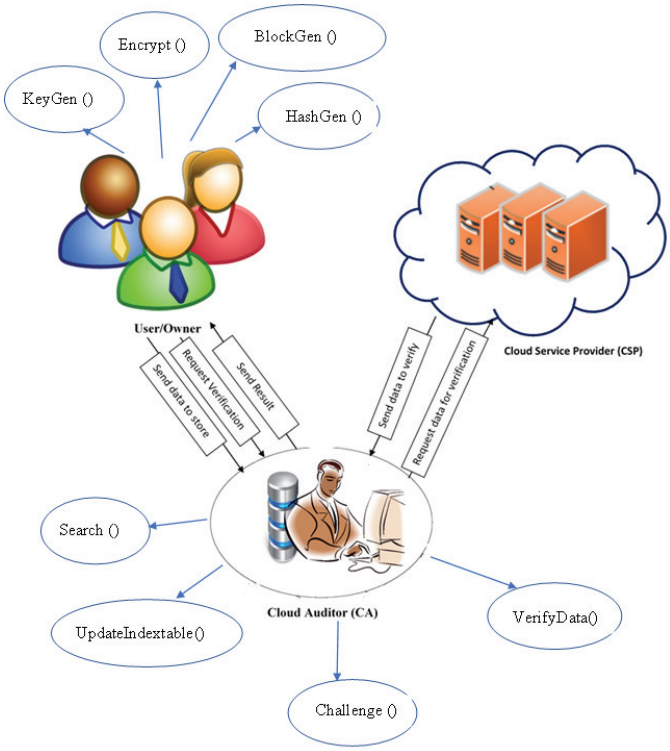

Figure 1 Architecture for the proposed method.

5.1 User Module

***Preambles*:**

1. ***KeyGen() (Generate keys)*:** User executes KeyGen to generate pair of keys – public and private keys (PK_u and PK_k) using Elliptic Curve Cryptography. Here PK_u and PK_k are generated using $\{E_p(a, b), G, n\}$ where $E_p(a, b)$ is the elliptic curve with a, b, points and a prime integer from 2^n, G is a point on elliptic curve whose order is a large value n. The order n of a point G on an elliptic curve is the smallest positive integer such that $n.G = 0$.
2. ***Encrypt():*** using the PK_k generated from the KeyGen, the user then encrypts the data to be stored to obtain $E_{PK_k}(m)$ where m is the plain text data to be stored in the CSP. Initially, the data m is encoded into a point P_m on an elliptic curve. The owner chooses a random positive integer k and produces the ciphertext, $E_{PK_k}(m)$, consisting of pair of

points.

$$E_{PK_k}(m) = \{k.G, P_m + k.P_m\} \tag{3}$$

3. ***BlockGen (Block Generation)*:** User generates a signature based on BLS (Boneh-Lynn-Shacham) Signature scheme. This scheme consists of three steps: key generation, signing, and verification.
   - ***Key Generation*:** The key generation algorithm chooses a random number p, between the range, [0, r-1]. The chosen integer p will be the private key and the respective public key is calculated as g^p.
   - ***Signing*:** Given the private key p, and some message m, the signature is computed by taking the hash of bitstring m, that is h = H(m) and then the signature would be:

$$\sigma = (u^{(H(vno_i \| ts_i))}.v^m)^x \tag{4}$$

     where vno_i is the version number, ts_i is the timestamp, and these two are concatenated and u, v, x are random numbers $\in$ Z. The version information and timestamp information will also be sent to the cloud auditor. In the proposed method, m_1 = (ID | E_{PK_k}(m) | fno | vno | TS). So, by taking the hash of the bitstring m_1, we get

$$h = H(ID|E_{PK_k}(m)|fno|vno|TS) \tag{5}$$

     where ID is the user ID, E_{PK_k}(m) is the encrypted message, fno is the file no, vno is the version number, TS is the timestamp.
     User generates a signature S_i using the following method:

$$S_i = h^p \tag{6}$$

   - ***Verification*:** Given a signature σ and a public key g^p, verify that e(σ, g) = e(H(m), g^p).
4. ***HashGen (Hash generation)*:** Apply the hash function, SHA – 512 on E_{PK_k}(m) and generate a message digest MD. User sends the E_{PK_k}(m), S_i and MD to the cloud auditor.

5.2 Cloud Auditor Module

The major task of a cloud auditor is to verify the integrity of the data. This feature of the cloud auditor has been extended, because of which the communication cost has been reduced between the CSP and the cloud auditor, User and CSP. This reduction in communication cost will largely reduce the

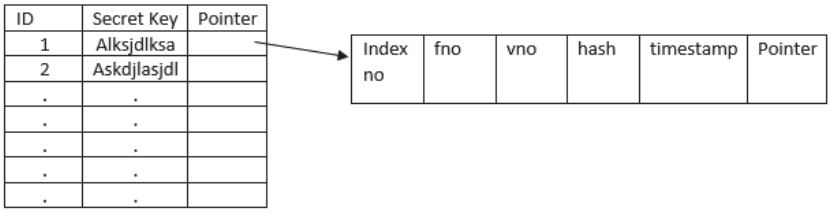

Figure 2 Index table in Cloud Auditor.

network overhead. Moreover, because of the tweaking of integrity check tasks of the cloud auditor, it is considered that the cloud auditor has a higher-end infrastructure to uphold all the activities performed by it.

- **Overview of Dynamic Index table**:
  The cloud auditor maintains an index table which is a list of lists data structure as shown in Figure 2. This index table is utilized by the cloud auditor for storing the latest versions of the data. The master list stores the Identification of the user/owner (ID), the secret key, SK, for each user, and then maintains the sub-list (pointer to the sub list). This secret key is generated per user as soon as the user registers into the system, using the AES encryption algorithm. Moreover, it is hidden even from the user.
  The sub-list in turn stores the index number (indexno), file number (fno), version no (vno), timestamp (TS), the hash of the data (MD), and the pointer to the next file of the user if any. If the user wants to store the new file, the cloud auditor first searches for the ID of the user and then traverses to the sub-list to insert the data at the beginning of the list. To delete a particular file or to update a file similar process is used, where the ID of the user is searched for first, and then traverse to the node which has the corresponding file number and then either delete the data or update the fields of the node, according to the requirements.
- **Data Integrity Check using Dynamic Index table**
  As soon as the cloud auditor receives the package S_i from the user, it checks for the ID of the user. It then searches for the ID in the index table and once it finds the ID, it stores the file details in the sub-list. After that, the E_{PK_k}(m) along with MD is encrypted using the SK of the respective user to get E_{sk}, i.e

$$E_{sk} = SK(E_{PK_k}(m) + MD) \tag{7}$$

Send the E_{sk} to CSP for storage. The task of CSP here is to store the data. There is no communication between the CSP and the user. This way it is possible to increase the confidentiality of the data or the data privacy.

Challenge: When the user requests for the file or the data or maybe for the data verification process, he/she requests it with the cloud auditor. The Cloud auditor first verifies the file number and ID of the user. If the verification is failed, the cloud auditor will not continue with the request for a data integrity check. Otherwise, it generates the following challenge: (chal = C = $\{$fno, $1 \leq$ c, c $\leq$ n$\}$, S = $\{s_i \; i \in C\}$, ξ) where C is the file numbers to be verified and S is randomly selected from C, ξ = g^{ρ} and $\rho \in Z$ is the random number. Cloud auditor calculates $\eta = y^{\rho}$ for verification. Cloud auditor sends the challenge to cloud service provider. CSP generates the proof and sends the result back to the cloud auditor for verification.

Verify: The cloud auditor computes the following equation to verify the proof sent from CSP.

$$TP = \lambda^{\rho}.e(u^{H(v_i \| t_i).s_i}, v^M, \rho) \tag{8}$$

If it holds, then the verification process is considered as succeeded otherwise, CSP has failed to pass the verification.

5.3 Cloud Service Provider

Upon receiving the challenge from the cloud auditor, CSP produces a response proof which consists of tag proof, file proof and an auxiliary auditing proof. The tag proof is given by

$$TP = \prod_{i \in C} e(\sigma_i, \rho)^{s_i} \tag{9}$$

and the file proof is given by

$$M = \sum_{i \in C} m.s_i + r \tag{10}$$

where r is the random padding/mask utilized to protect the privacy of the data. Finally the auxiliary proof is derived using the following equation:

$$\lambda = e(v, y)^{-r} \tag{11}$$

After computing these, the CSP sends $<$TP, M, $\lambda >$ to cloud auditor.

6 Experimentation

The proposed method is implemented using the cloudsim framework [22]. The primary step within the strategy is to initialize the cloudsim bundle. It is performed by setting the calendar, trace_flag, and the number of clients. Then invoke the init function by passing the initialized variables such as the number of clients, calendar, and trace_flag as parameters.

6.1 Algorithms

The next step is to define the data center and the broker as shown in Algorithm 1. To set up the data center, create the list to store the machines, defined by HostList. Each machine will have a set of PEs or CPUs or cores, which is defined by peList. Each CPU's unit of a cycle is defined in terms of MIPS. Create Host with its id, ram, bandwidth, storage space, and list of PEs and add them to the list of machines. addtoHostList() function creates the machine with the specifications. The datacenter broker is a proxy to the cloud customer. It acts on behalf of the cloud customer.

Algorithm 1 Data Center and Data Broker Creation

```
1:  function DATACENTER
2:      HostList ← create new arraylist(host)
3:      pelist ← create new arraylist(pe)
4:      int mips ← 1000
5:      addTo pelist (pe(0, create peProvisionersSimple(mips)))
6:      ram ← 2048; hostId ← 0; bw ← 10000; storage ← 1000000;
7:      addTohostList(Host( hostId, BwProvisionerSimple(bw), RamProvisionerSim-
    ple(ram),
                                    peList,
                                    VmSchedulerTimeShared(peList),
                                    storage ));
8:      Set Data Center characteristics.
9:  end function
10: function DATACENTERBROKER
11:     broker ← new DatacenterBroker("Broker");
12: end function
```

Once the data center and corresponding brokers are created, create the virtual machines, and assign them to each broker as shown in Algorithm 2. We need to explicitly mention the specifications of the virtual machine that is going to be created through, userid- the id of the virtual machine, i- the instance of the virtual machine, pesNumber- the CPU number, MIPS- raw

processing power of the CPU, bw- bandwidth, ram- the ram size identified for the virtual machine, vmm- the type of the virtual machine whether Xen or ESXi, size- image size of the virtual machine.

Algorithm 2 Creating Virtual Machines

```
1: LinkedList<Vm> list ← new LinkedList<Vm>()
2: Define parameters of virtual machine
3: for i ← 0 to vms in step of 1 do
4:     vm[i] = new Vm(userid, i, pesNumber, mips, bw, ram, vmm, size);
5:     list.add(vm[i]);
6: end for
```

Then, define the harddisk specifications. Assign the harddisk to a specific VM. To get the capacity and the available space of the hard disk created, inbuilt functions such as *getCapacity()* and *getAvailableSpace()* are used.

Now, the virtual machines are created and assigned with brokers, it is required to create a network between them to provide a virtual scenario of the cloud environment. To create the network between the datacenters, a few APIs related to network topology are imported. Usually, there is a *brite* file which has the detailed matrix of the network which is read into the cloudsim. It builds the network according to the file. Otherwise, it is possible to explicitly create or build the network by using a function of network topology called addlink(). Addlink() reads four parameters: sourceId, destinationId, bandwidth and latency. SourceId is the source of the network link, destinationId is the destination of the network link, bandwidth is the number of bits per second that can be transferred in that link, and latency is the delay observed in milliseconds in the link.

In the owner or the user module, a key pair is generated using a KeyPair Generator function of ECC. To generate the key pair an elliptic curve over the finite field of 'mod' elements is to be constructed which is shown in Algorithm 3.

Utilizing the private key, the proprietor/user will scramble the message. Algorithm 4 renders the encryption method. The encryption technique used is the Elliptic Curve Cryptography algorithm which is asymmetric cryptography or public-key cryptography where based on the elliptic curve theory. It is used to create smaller, more efficient, and faster pairs of keys. It can provide the security of 1024bits of other algorithms in about 160bits.

Once, the encryption process is completed, apply the hash function, and generate the message digest, MD. The hash function used in the proposed method is SHA-1.

Algorithm 3 Generating elliptic curve

Input: a, b, p
Output: Elliptic curve

```
if !probalblePrime(p) then
    Throw Error
end if
if if mod(p) is true then
    compute order of group
    compute generator of group
end if
                                                    ▷ Generate curve
y_square ← (q.gety()).modPow(new BigInteger("2"), p);
x_cube ← (q.getx()).modPow(new BigInteger("3"), p);
dum ← ((x_cube.add(a.multiply(x))).add(b)).mod(p);
                                    ▷ Generate keys from the curve
sk ← new (ec.getp().bitLength(), Rand.om);
pk ← sk.getPublic();
```

Algorithm 4 Encrypting Message

Input: The message or file to be encrypted and the private key
Output: Cipher Text or Encrypted message

```
res ← new byte[ek.ec.getPCS()+numbytes];
ECPoint gamma ← ek.ec.getGenerator().multiply(rk);
ECPoint sec ← ek.beta.multiply(rk);
System.arraycopy(gamma.compress(),0,res,0, ek.mother.getPCS());
hash.update(sec.getx().toByteArray());
hash.update(sec.gety().toByteArray());
byte[] digest = hash.digest();
for j ← 0 to numbytes in step of 1 do
    res[j + ek.ec.getPCS()] ← (byte) (input[j]^digest[j]);
end for
```

The encrypted message appended with the ID of the user, fno, vno, and TS along with the MD is signed using the BLS signing technique. Then the signed package is sent to the cloud auditor.

The cloud auditor generates secret keys for every user who has registered in the system. AES encryption algorithm is used to generate the secret keys.

Using this secret key, the cloud auditor encrypts the ciphertext along with the corresponding MD and sends it to store in the CSP. When the owner requests for the integrity check, the cloud auditor will request the CSP for the data, and CSP, in turn, sends the data back to the cloud auditor.

Cloud auditor will decrypt the data received from the CSP using the same secret key of the user. After decrypting cloud auditor will get ciphertext and the message digest MD*. Now, compare the MD and MD* to verify. If they match then, the data is not tampered with, or else, the data has been manipulated. Finally, the cloud auditor sends the results back to the user.

7 Scheme Analysis

In this section, an analysis of the results obtained after implementing the algorithms described in Section 6 is discussed.

7.1 Security Analysis

The security of our proposed method is evaluated with the following:

***Unforgeability of BLS Method*:**

It is computationally impossible for any adversary to forge a valid BLS signature if the CDH assumption in the bilinear groups holds.

Proof: Boneh et al. [27], in their security analysis, have proven that BLS signature is unforgeable in bilinear groups when the CDH problem is hard. So, we will exclude the proof here.

Correctness

Correctness here means that the owner uploads the data into the cloud server by signing the data using the BLS signature scheme, which in turn verifies that, the owner is the legitimate cloud user.

If the CSPs follow the protocol honestly, then, CSPs will pass the auditing. Since the BLS signature is used for signing, it resists any adversary from trying to tamper with the name. We make use of the file number, name, version number, and timestamp of the corresponding file.

When the cloud auditor sends the data to CSP, it encrypts the data with the secret key SK to generate E_{psk}.

$$E_{psk}(E_{pk}, MD, fno, vno) \tag{12}$$

Suppose say, an adversary modifies the data. When the data is challenged by the cloud auditor, there will be a change in the version number and the message digest corresponding to the data, which results in

$$E^*_{psk}(E_{pk}, MD^*, fno^*, vno^*) \tag{13}$$

There is no way that, the same message digest is generated for the modified data set. So, when the cloud auditor receives the fake data from the CSP, and when the cloud auditor does the verification, it will fail in two cases. First, the message digest will not be a match. Second, the timestamp of the message will be different.

Further, based on the response sent by the CSP to cloud auditor as a reply to the challenge generated by the cloud auditor, the correctness can be derived as given below:

$$\begin{aligned}
&\lambda^{\rho}.e(u^{H(v_i\|t_i).s_i}.v^{M},\rho)\\
&\quad = e(v,y)^{-r}.e(u^{H(v_i\|t_i).s_i}.v^{\sum_{i\in C} m.s_i+r},\rho)\\
&\quad = e(u^{H(v_i\|t_i).s_i}.v^{\sum_{i\in C} m.s_i+r}.v^{-r},\rho)\\
&\quad = \prod_{i\in C} e((u^{H(v_i\|t_i)}.v^{m})^{x.s_i},\rho)\\
&\quad = \prod_{i\in C} e(\sigma_i,\rho)^{s_i} = TP
\end{aligned} \tag{14}$$

Resistance to forging attacks

The data is stored in CSP and when cloud auditor requests for the proof, CSP generates the proof and sends (TP, M, λ). If the auxiliary proof is not correct, the verification formula (8) will not hold. Moreover, BLS signatures are unforgeable, so tag proof TP cannot be forged. The block proof M is also unforgeable and it can be proven as follows: Consider that a fake proof is sent by the CSP to cloud auditor FP = (TP, M*, λ), where

$$M = \sum_{i\in C} m.s_i + r \neq M^{*} = \sum_{i\in C} m^{*}.s_i + r \tag{15}$$

Assume that the CSP passes the verification by cloud auditor, then

$$\begin{aligned}
&\lambda^{\rho}.e(u^{H(v_i\|t_i).s_i}.v^{M^{*}},\rho)\\
&\quad = e(v,y)^{-r}.e(u^{H(v_i\|t_i).s_i}.v^{\sum_{i\in C} m^{*}.s_i+r},\rho)
\end{aligned} \tag{16}$$

For the valid proofs we have

$$\begin{aligned}
&\lambda^{\rho}.e(u^{H(v_i\|t_i).s_i}.v^{M},\rho)\\
&\quad = e(v,y)^{-r}.e(u^{H(v_i\|t_i).s_i}.v^{\sum_{i\in C} m.s_i+r},\rho)
\end{aligned} \tag{17}$$

As per the property of bilinear maps, it can be derived that

$$\sum_{i \in C} m.s_i + r = \sum_{i \in C} m^*.s_i + r \tag{18}$$

Equation (18) contradicts the assumption made with the fake proof in equation (15). Hence it is proved that the forging attacks are neutralized. Similarly, it can be proven that the replay and replacing attacks can be efficiently stopped as explained by Shen et al. [31].

7.2 Performance Evaluation

This paper proposes a strategy where the communication between the owner and TPA is reduced. It proposes another algorithm where the storage overhead is reduced for the owner. The communication cost of the owner is O(1), the cloud auditor is O(n) and CSP is O(1). The owner encrypts the data using the Elliptic Curve Encryption method. Then finds the hash using SHA-512 and sends the ciphertext and the message digest to the Cloud Auditor. Cloud auditor gets the authentication of the owner through the session key using which the concatenation of Message Digest, (MD) and E_{psk} has been performed.

Using the shared session key, the cloud auditor decrypts the message. Cloud auditor maintains an index table that has the secret key for each user, which is known only to the cloud auditor. Cloud auditor then encrypts the data using the secret key of the corresponding data owner to obtain E_{psk}. After encrypting, E_{psk} is sent to CSP for storage. When any user requests for the data, the CSP sends E_{psk} to the cloud auditor. Cloud auditor in turn decrypts it to obtain Ep and MD. The Cloud auditor then verifies the MD with MD^* received from the owner. The outcome is then sent to the respective owner and the user.

In this segment, we are going to assess the performance of the proposed strategies and equate it with the state of the arts. Chen et al. [23], has compared their method with the other existing methods. Taking that into consideration, Table 1 summarizes the comparison of proposed strategy with the other existing methods.

***Communication Cost [Owner $\leftrightarrow$ Cloud auditor $\leftrightarrow$ CSP]*:**

The proposed protocol is analyzed for the communication process. The challenge-response messages between the cloud auditor and CSP are not present, since the CSP is used only to store the data. Unlike the regular methods where a challenge is sent to the CSP and CSP, in turn, generates

Table 1 Comparison of various state of arts with the proposed method

Techniques	Public Auditing	Data Privacy	Dynamic Auditing
PDP [4]	√	×	×
PoR [24]	×	NoD	×
IHT-PA [6]	√	√	√
DAP [25]	√	√	√
DPDP [4]	×	NoD	√
DHT-PA [7]	√	√	√
DLIT-PA [8]	√	×	√
AHT-PA [23]	√	√	√
Proposed	√	√	√

√: "Supports"; ×: "Doesnot support"; NoD: "Not Defined/No Demand".

evidence of verification and then sends it to the auditor who then verifies the proof sent by the CSP. This process has the communication cost of O(c) where c is the number of blocks or files which are challenged for verification. This challenge-response mechanism is not present in the proposed protocol. Here the cloud auditor requests for the data from the CSP and CSP transfers the data to the cloud auditor, so the cost of communication is O(c).

The second claim is that there is no communication between CSP and the user. If the user wants to store the data in CSP, he/she sends the data to a cloud auditor who then sends it to CSP after encrypting with a secret key. So, there is not a single communication happening between the user and CSP explicitly. So, the communication cost is zero.

Third, is the communication between cloud auditor and user. Here the communication cost is O(n) where n is the total number of files requested to store in CSP and the number of files requested from the CSP.

The communication costs for the verification phase of the methods are similar to one another including the proposed method. So, further analysis is performed on the computation of the methods and the comparison is shown in Figures 3 and 4.

***Computational Cost [Owner, Cloud auditor, CSP]*:**

The computation cost of the proposed method is comparatively high in the cloud auditor module because, the cloud auditor must maintain an index table, generate the secret key for each user, and evaluate the correctness and soundness of the user data.

The user module simply generates the keys and hash and signs the data before sending it to the cloud auditor. Apart from that, there is no computation as such.

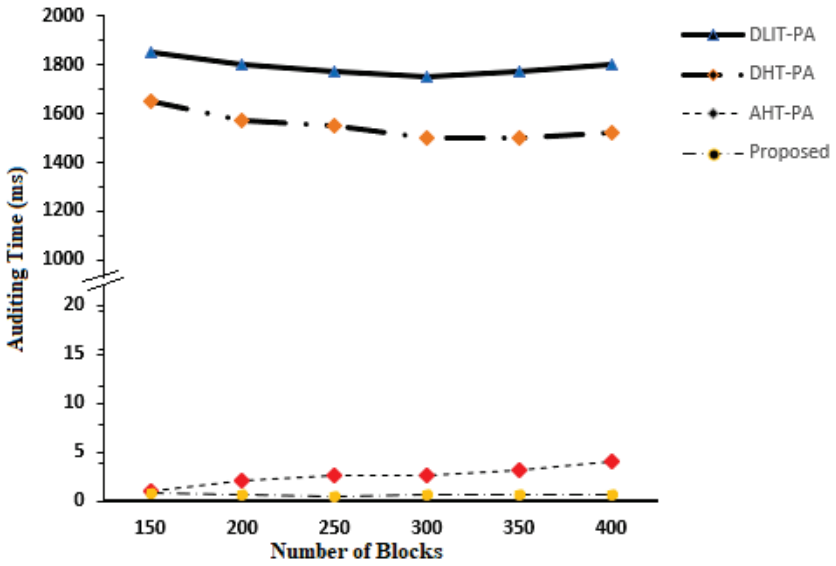

Figure 3 The average audit time per task for multiple auditing blocks.

In the CSP module, there is no computation since it only stores the data sent by the cloud auditor. Unlike the existing methods, where CSP must generate the proof for the data stored, the proposed technique, does not provide the proof, nor it will be challenged for proof by cloud auditor. In the CSP module, there is no computation at all. So, the computation cost of CSP is zero. But in the cloud auditor module, the computation cost is higher because of the verification process for all the files. So, the cost will be O(c) where c is the number of files requested by the user.

Figure 3 shows the comparison between different methods. As it can be seen, the proposed method and AHT-PA almost take the same time for auditing as per the given graph.

As it can be observed from the graph, the proposed method is almost similar to the AHT-PA, but it is still a better method because it takes lesser time compared to AHT-PA as depicted in Figure 4. The time required to perform block generation is also compared and shown in Figure 5.

8 Statistical Analysis

It can be proven that, based on the hypothesis testing, the proposed method takes less time. The data considered for the hypothesis testing is obtained as

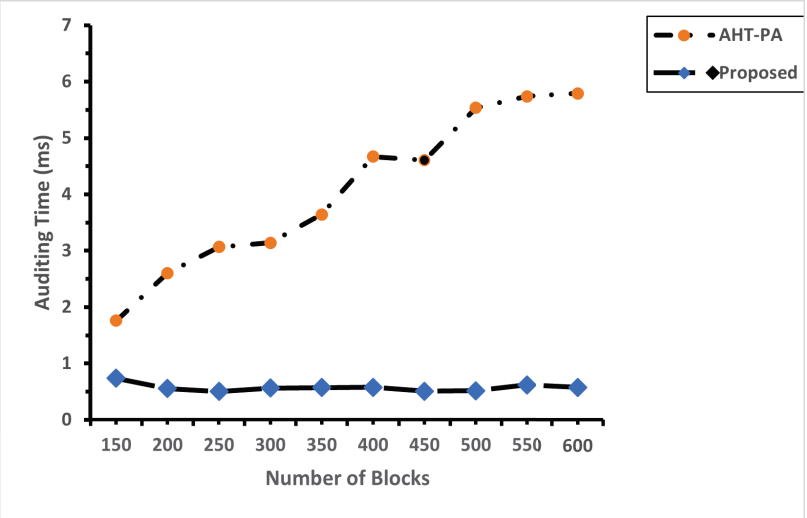

Figure 4 Average audit time per task for multiple auditing blocks comparison between AHT-PA and proposed method.

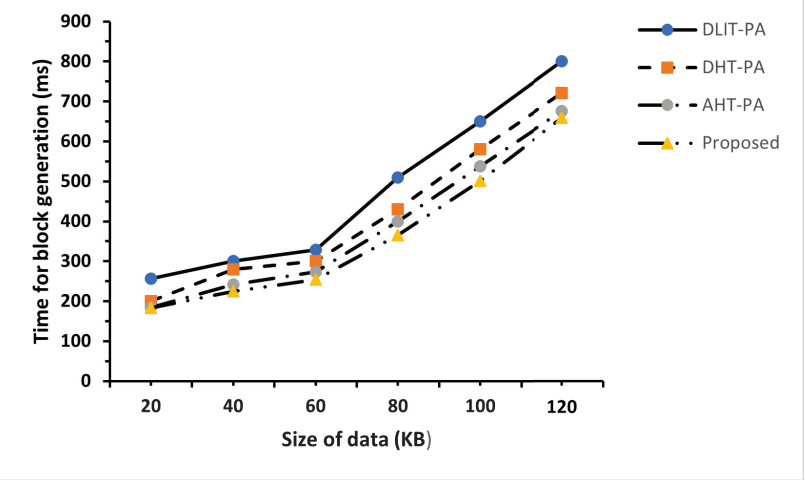

Figure 5 Block generation time for different file/block sizes.

per the graph depicted in Figure 4. The hypothesis defined for the validation is as follows:

H_0: There is no significant influence of the proposed method on the execution time of the data integrity check process.

H_1: There is a significant influence of the proposed method on the execution time of the data integrity check process. The procedure adopted is ANOVA. Table 2 provides the details about the various factors which are calculated

Table 3 details about the prerequisite value calculations for the F-calculation of ANOVA.

From the prerequisite values calculated in Table 3, the F-Ratio is given by

$$F\_Ratio = \frac{MSbetween}{MSwithin} = 225.8354773 \tag{19}$$

Therefore, $F_{calculated}$ is 225.8354773. This value is then verified with the t-table where

$$F(1, 18) = 4.4139. \tag{20}$$

Comparing the calculated and tabulated value of t-statistic:

$$F_{calculated} > F_{tabulated} \tag{21}$$

The values obtained in Equations (19) and (20) are substituted into Equation (21) and it is found that $F_{calculated}$ is higher than the $F_{tabulated}$, it is in the rejection region. Therefore, the null hypothesis is failed to be accepted. Hence, rejected which means that there is a significant influence of the proposed method on the execution time of the data integrity check process.

Table 2 Calculating the factors for ANOVA

Variables	Equation	Value Obtained
Total, T		65.736215892
Number of samples, n		20
Correction Factor, CF	$\frac{T^2}{n}$	216.062504
Total Sum of Square, SS_{Tot}	$\sum X_{ij}^2 - \frac{T^2}{n}$	155.409896
Sum of Squares Between, SS_{bet}	$\sum \frac{T_j^2}{n_j} - \frac{T^2}{n}$	143.937496
Sum of Squares Within, SS_{Within}	$\sum X_{ij}^2 - \sum \frac{T_j^2}{n_j}$	11.4724

Table 3 Prerequisite values to calculate $F_{calculated}$

Source of Variation	SS	d.f	MS
Between Sample	143.937496	(2–1) = 1	143.937496
Within Sample	11.4724	(20–2) = 18	0.637355555
Total	155.409896	(20–1) = 19	

9 Conclusion and Future Work

In this paper, an effective and secure data integrity verification process based on the lists has been discussed. Additional confidentiality is provided to the data while storing the data into the CSP. A theoretical analysis of the results obtained has been performed and the proposed approach is statistically evaluated. It is proven that there is a significant influence of the proposed method on the execution time during the data integrity verification. Hence, the proposed approach has comparatively small communication and computational complexity. Specifically, in the verification phase, the computation cost is less than the state-of-the-art. Since the proposed approach is list-based, the searching process can be improvised.

Compliance with Ethical Standards

- Disclosure of potential conflicts of interest: The authors do not have any conflicts of interest.
- Research involving Human Participants and/or Animals: There is no involment of human participants and/or animals in this research.
- The authors have no relevant financial or non-financial interests to disclose.
- All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

[1] Ateniese, G., Burns, R., Curtmola, R. et al., "Provable Data Possession at untrusted stores", in *Proceedings of the 14th ACM Conference on Computer and Communication Security (CCS '07)*, pp. 598–609, Virginia, Va, USA, November, 2007.

[2] Juels A., Kalski Jr., B. R., "PORs: Proofs of Retrievability for Large Files", in *Proceedings of 14th ACM Conference in Computer and Communications Security*, pp. 584–597, ACM, Alexandria, VA, USA, November, 2007.

[3] Anisetti, M., Ardagna, C., Damiani, E., and Gaudenzi, F., "A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification", in *IEEE Transactions on Services Computing,* 2017.

[4] Erway, C., Papamanthou, A. Küpçü C., and Tamassia R., "Dynamic provable data possession", in *Proceedings of the 16th ACM Conference on Computer and Comunications Security*, pp. 213–222, ACM, Chicago, Ill, USA, November, 2009.

[5] Wang, Q., Wang, C., Ren, C., Lou, W., and Li, J., "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", in *IEEE Transactions on Parallel and Distributed Systems*, 22(5), 847–859, (2011).

[6] Zhu, Y., Ahn, G. -J., Hu, H., Yau, S. S., An. H. G. and Hu. C. -J., "Dynamic Audit Services for Outsourced Storages in Clouds", *IEEE Transactions on Services Computing*, Vol. 6, No. 2, pp. 227–238, 2013.

[7] Tian H., Chen, Y., Chang, C. -C. et al, "Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage", in *IEEE Transactions on Services Computing*, Vol. 10, No. 5, pp. 701–714, 2017.

[8] Shen, J., Shen, J., Chen, X., Huang, X., and Susilo, W., "An Efficient Public Auditing Protocol with Novel Dynamic Structure for Cloud Data" in *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 10, pp. 2402–2415, 2017.

[9] Kwon et. al, "A Secure and Efficient Audit Mechanism for Dynamic Shared Data in Cloud Storage", in *The Scientific World*, May, 2014.

[10] More, S., Chaudhari, S., "Third Party Public Auditing Scheme for Cloud Storage", in *7th International Conference on Communication, Computing and Virtualization*, (2016).

[11] Yuan, J., and Yu, S., "Public Integrity Auditing for Dynamic Data Sharing With Multiuser Modification", in *IEEE Transactions on Information Forensics and Security,* 10 (8), (2015, August).

[12] Zargad, S. V., Tambile, A. V., Sankoli, S. S., and Bhongale, R. C., "Data Integrity Checking Protocol with Data Dynamics and Public Verifiability for Secure Cloud Computing", in *International Journal of Computer Science and Information Technologies (IJCSIT)*, 5 (3), 4062–4064, 2014.

[13] Kaaniche, N., "Cloud Data Storage Security Based on Cryptographic Mechanisms", Ph.D. dissertation, Informatique, Telecommunications et Electronique de Paris, 2014.

[14] Li, R., Shen, C., He, H., Gu, X., Xu, Z., and Xu, C. Z., "A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing", in *IEEE Transactions on Cloud Computing*, 6 (2), 344–357, April, 2018.

[15] Sarkar, M. K., and Chatterjee, T., "Enhancing data storage security in cloud computing through steganography", May, 2014.

[16] Sun, L., Xu, C., Zhang, Y. and Chen, K., "An efficient i$\mathcal{O}$-based data integrity verification scheme for cloud storage", in *Science China Information Science, 62*, doi: https://doi.org/10.1007/s11432-018-9500-0, 2019.

[17] Shah, H., Shah, J., and Desai, U., "Third party public auditing scheme for security in cloud storage", in *International Journal of Trend in Scientific Research and Development* (ijtsrd), 3, 179–184, April, 2019.

[18] Akshay, K., and Balachandra, M., "Analysis of Execution Time for Encryption During Data Integrity Check in Cloud Environment", in *Security in Computing and Communications*, Springer Singapore, pp. 617–627, isbn: 978-981-13-5826-5, 2019.

[19] Wang, C. and Di, X., "Research on Integrity Check Method of Cloud Storage Multi-Copy Data Based on Multi-Agent", in *IEEE Access*, Vol. 8, pp. 17170–17178, 2020.

[20] Ganesh, S. M., Manikandan, S. P., "An Efficient Integrity Verification and Authentication Scheme over the Remote Data in the Public Clouds for Mobile Users", in "Security in Communication Networks", Vol. 2020, doi: 10.1155/2020/9809874, 2020.

[21] Chidambaram, N., Raj, P., Themozhi, K., Anirtharajan, R., "Enhancing the Security of Customer Data in Cloud Environments Using a Novel Digital Fingerprinting Technique", in *International Journal of Digital Multimedia Broadcasting*, doi: http://dx.doi.org/10.1155/2016/8789397, 2016.

[22] "CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services", http://www.cloudbus.org/cloudsim/, Accessed: 2020.

[23] Chen, W., Tian, H., Chang, C., Nan, F., Lu, J., "Adjacency-Hash-Table Based Public Auditing for Data Integrity in Mobile Cloud Computing", in *Wireless Communications and Mobile Computing*, https://doi.org/10.1155/2018/3471312, 2018.

[24] Du, R., Deng, L., Chen, J., He, K., Zheng, M., "Proofs of Ownership and Retrievability in Cloud Storage", in *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, Ed. IEEE Computer Society, pp. 328–335, 2014.

[25] Yang, K. and Jia, X., "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing", in *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 9, pp.1717–1726, doi: 10.1109/TPDS.2012.278, 2013.

[26] Liu D., and Zic. J, "Proofs of Encrypted Data Retrievability with Probabilistic and Homomorphic Message Authenticators", in *IEEE Trustcom/BigDataSE/ISPA*, Ed. IEEE Computer Society, 2015.

[27] Boneh, Dan and Lynn, Ben and Shacham, Hovav, "Short signatures from the Weil pairing", in *International conference on the theory and application of cryptology and information security*, Pub. Sringer, pg. 514–532, 2001.

[28] Wang, C., Chow, S.S., Wang, Q., Ren, K. and Lou, W., "Privacy-preserving public auditing for secure cloud storage", in *IEEE transactions on computers*, 62(2), pp. 362–375, 2011.

[29] Wang, C., Wang, Q., Ren, K., Cao, N. and Lou, W., "Toward secure and dependable storage services in cloud computing", in *IEEE transactions on Services Computing*, 5(2), pp. 220–232, 2011.

[30] Yang, K. and Jia, X., "An efficient and secure dynamic auditing protocol for data storage in cloud computing", in *IEEE transactions on parallel and distributed systems*, 24(9), pp. 1717–1726, 2012.

[31] Shen, J., Shen, J., Chen, X., Huang, X. and Susilo, W., "An efficient public auditing protocol with novel dynamic structure for cloud data", in *IEEE Transactions on Information Forensics and Security*, 12(10), pp. 2402–2415. 2017.

Biographies

Akshay, KC received his B.E. degree in Information Science from Visvesvaraya Technological University, Belagavi. He has received his M.Tech. degree in Software Engieering from Manipal University. He is currently pursuing his Ph.D. degree in Cloud Security with the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, where he is currently an Assistant Professor (Senior). His research interests include information security, network security, algorithms, data structures, data base systems and software engineering.

Balachandra Muniyal received the B.E. degree in computer science and engineering from Mysore University and the M.Tech. and Ph.D. degrees in computer science and engineering from the Manipal Academy of Higher Education, Manipal, India. He carried out his M.Tech. project work in T-Systems Nova GmbH, Bremen, Germany. He was deputed to Manipal International University, Malaysia, in 2014. He is currently a Professor with the Department of Information & Communication Technology, Manipal Institute of Technology, Manipal. He has 25 years of teaching experience in various Institutes. He has more than 30 publications in national and international conferences/journals. His research interest includes network security.