An Overview of Data Integrity Approaches for Managing Outsourced Data on Insecure Cloud Platforms

Maral Vikas Balaso[1]*, Dr. Rakesh kumar Giri[2]

[1]*Research Scholar, Department of Computer Science & Engineering, Sunrise University, Alwar, Rajasthan
[2]Associate Professor, Department of Computer Science & Engineering, Sunrise University, Alwar, Rajasthan

***Corresponding Author:** Maral Vikas Balaso
*Research Scholar, Department of Computer Science & Engineering, Sunrise University, Alwar, Rajasthan

Abstract
Cloud computing has become an expansive and rapidly expanding domain that significantly influences the advancement of various emerging technologies and applications, including but not limited to the internet of things, sensors, artificial intelligence, social networks, and business applications. The exponential growth of technology and applications has led to a substantial increase in data production, which is dynamically updated. The aforementioned dynamic data is stored on third-party service provider-provided cloud storage. The reliability of third-party cloud storage is questionable, and the user lacks authority regarding the data's possession or integrity. The primary concern is the integrity of the data, which is not being purged, altered, or obliterated on purpose or by accident. The researchers have introduced a number of protocols, including Provable Data Possession (PDP) techniques, which offer a probabilistic approach to verifying the integrity of data at the block level. In conjunction with PDP, the researchers have implemented various data structures to accommodate the dynamic nature of the data. For metadata generation and node rebalancing of the data structures, integrity verification schemes impose substantial computational and communicational burdens due to the dynamic character of the immense amounts of data.

Keywords: Provable data possession, cloud computing, data dynamics.

Introduction

In the era of information technology in the twenty-first century, data is expanding exponentially daily. The phenomenon of data expanding at an exponential rate is referred to as "big data," which specifically denotes large, intricate, structured, and unstructured datasets generated in the course of routine business operations. This escalating volume of big data is an integral component of daily operations, generated by technologically advanced applications such as the internet, social networking sites, healthcare applications, and sensor networks, among others. Moreover, it is expanding at a rapid rate. In order to store and process this ever-evolving volume of big data, which requires immense processing and storage capacities, the nascent technology known as "Cloud Computing" is implemented. Cloud computing is an emergent technology that is having far-reaching implications in the fields of information technology, business, health care, software engineering, and data storage. The daily generation of vast quantities of data by enterprises, organizations, and individual users presents a formidable challenge for companies tasked with storing, processing, and ensuring the security of that data on local storage. Such endeavors necessitate substantial financial investments in the infrastructure required to store and process the data. Cloud computing primarily offers its customers three service delivery models: Infrastructure as a service (IaaS), Software as a Service (SaaS), and Platform as a service (PaaS). The National Institute of Standards and Technology (NIST) classifies four deployment models: private, public, hybrid, and community cloud. Cloud computing leverages the virtualization technique in order to furnish end users with resources in an efficient manner. Cloud computing is distinguished by its provision of on-demand services, resource aggregation, high scalability, flexibility, and cost-effective computational capabilities for storage, applications, and platforms. The primary functions of cloud services are data storage, data sharing, and application provisioning. The majority of enterprise organizations are cloud computing-driven, and as a result, they are migrating their application development and data storage (financial, personnel, healthcare) to the cloud.

Cloud Computing

The computational and networking industries have been significantly transformed by the advent of the World Wide Web (Internet), which enables users worldwide to share resources. The advent of cloud computing has significantly transformed the traditional computing paradigm by granting universal access to resources such as vast repositories of data storage, computational capabilities, applications, and services via the internet. Cloud computing can be defined as a vast repository of computational resources and storage that is accessible to the public via the internet; payment for these services is contingent upon their utilization. It provides unique advantages including enhanced network accessibility, scalability, adaptability, resource sharing, and efficiency in usage. As a consequence, there is a significant surge in the adoption of the cloud computing paradigm across diverse sectors, including scientific adoptions, health care

applications, and social networking.
The only requirement for a client to utilize cloud computing services is a computing device with a stable internet connection. Beyond its user-friendly interface, cloud computing offers numerous advantages to its clients, including worldwide accessibility, a standardized platform, substantial scalability, dynamic infrastructure, administrative capabilities, and most significantly, economical usage expenses. The cloud's seamless and practical connectivity has accelerated the expansion of numerous organizations. The progression of cloud computing services significantly influences the ability of organizations and individuals to accomplish their respective goals and objectives. The implementation of cloud computing has significantly propelled organizations forward in terms of revenue, cost, globalization, flexibility, and scalability.

Service Models of Cloud Computing
The functionality of the cloud computing infrastructure that is made available to the consumer on demand is the definition of a service model. Infrastructure as a service (IaaS), software as a service (SaaS), and platform as a service (PaaS) are the primary models of cloud services. The following are the specifics of the service models:
The core offering of cloud provider firms is IaaS. IaaS provides its customers with access to the hardware resources of the data center (network, storage, virtual server, processor, and memory). The cloud service provider (CSP) oversees the aforementioned resources, which the client can access via the internet. This is all accomplished through the use of virtualization, and clients only pay for the resources they utilize.
PaaS is a middleware paradigm that provides customers with services to execute applications, including frameworks, platforms, and virtual containers, so that they may construct their own applications. It reduces the expense of administering and marinating additional hardware and software necessary for application development by a significant margin. Platform as a Service (PaaS) offers clients pre-configured disk images and software stacks, which enable them to utilize the cloud's foundational resources including runtime components, libraries, and database engines. Google App Engine, AWS Elastic Beanstalk, and Adrenda are real-time examples of PaaS providers that offer software development kits (SDKs) for Python, Java, and.NET, respectively, as ready features. SaaS is the cloud computing application layer. SaaS satisfies the need of its clientele to access the hosted application via the internet and charges for utilization in accordance with resource consumption. Software as a Service (SaaS) is the most advantageous model for customers, as it enables them to achieve increased operational efficiency and decreased costs associated with self-managing the application [6]. As a result of decreased application costs and maintenance responsibilities, the SaaS layer is gaining significant traction among IT enterprises as a cloud business model.

Deployment Model of Cloud Computing
Deployment models represent the on-premises or off-premises physical presence of cloud service infrastructure. Community cloud, private, public, hybrid, and public clouds, as classified by the National Institute of Standards and Technology (NIST), comprise the majority of deployment models. The public cloud refers to cloud infrastructure that is accessible to organizations or consumers on a pay-per-use basis. The public cloud is capable of delivering any service, including PaaS, SaaS, IaaS, and more. OneDrive and Windows Azure HP Utilizable on demand, Hellion is a third-party cloud service provider on the market.
Private clouds are on-premises clouds that are utilized by the organization or enterprises and are managed and maintained by the organization. A team is devoted to the management and maintenance of their datacenters, which are utilized to supply cloud services to their enterprise applications. Prefersibly, they exhibit greater security with regard to the reliability of the secure service.
Hybrid cloud in which both cloud services are utilized an organization that operates both private and public databases is said to have a hybrid cloud structure. Thus, within such organizations, data is categorized according to various security measures. The most critical data that has the potential to cause significant harm to the organization is stored on-premises (private cloud), whereas less critical data is stored off-premises (public cloud), relieving the organization of the responsibility of managing less critical data.
The configuration of a community cloud is controlled and shared by multiple organizations that typically have a common objective or interest. The cloud may be deployed either on or off-site from the physical location of the organization. The management of the cloud is delegated to either the controlling organization or a third-party entity. It reduces the cost and security risk associated with private clouds and grants participating organizations unrestricted access to cloud-based data.

Data Integrity
Ensuring the confidentiality, integrity, and availability (CIA) of organizational information stored in cloud computing is a fundamental security feature. Data integrity is the principal concern of this research endeavor. Data integrity is the conviction that information remains consistent and accurate for the duration of its life cycle. Additional data integrity measures guarantee that the data remains unaltered, unintentionally or inadvertently destroyed, or lost. Decades of research have been devoted to examining the integrity of data, and integrity verification can be categorized into two primary approaches:
Deterministic approach in which the integrity of the entire file is examined. It ensures complete data ownership and is effective for verifying small-sized data.

A probabilistic approach is utilized to verify the integrity of the file, with only a limited number of essential blocks being examined. While it cannot provide absolute integrity assurance, it is effective for data files.
The straightforward approach to ensure the integrity of a file is to compute its message authentication code (MAC), which is the first traditional method. Prior to delegating the file to a remote cloud storage provider, the data proprietor performs a MAC computation on the entire file. The MAC of the file is retained in the local storage of the data proprietor when the file is deleted. Verification of data integrity necessitates the verifier to initiate a retrieval request from cloud storage for the file in question. The verifier then recalculates the outsourced file's MAC. Verifying the integrity of data, it compares the locally stored MAC to the recalculated MAC that is outsourced.
The data proprietor divides the file into n blocks before calculating the MAC of each block using a secret key; this is the second straightforward method. The proprietor transfers both the file and the MAC to the cloud server before erasing the MAC and the file from local storage. The proprietor of the data stores only the secret key. The verifier requests the file block and its corresponding MAC from the remote server in order to conduct verification.
Using the secret key, the verifier computes the MAC and compares it to the corresponding MAC received from the server. The aforementioned conventional methods may function admirably when the data is small in size and does not undergo frequent modifications after being stored on the server. But with regard to the data, both methods are impractical and riddled with severe defects.
The initial method incurs significant communication expenses; for instance, if a 10GB or 100GB file is outsourced, the data proprietor would be required to obtain the file from the cloud storage each time the integrity is verified. This is impracticable due to limitations in bandwidth and data consumption. The second approach is limited in its ability to account for the transient nature of data updates. In summary, the aforementioned approaches have significant drawbacks stemming from data, including substantial communication and computational expenses, as well as an inability to accommodate the dynamic nature of data.

Data Integrity Schemes

Research on data integrity verification schemes commenced with static data and subsequently expanded to include dynamic data, which encompasses operations such as creation, update, and deletion. Additional support was provided for public and private verifiability, with or without the inference of third-party auditing. Due to the fact that the introduction of a TPA introduced privacy vulnerability, protocols for maintaining data integrity and privacy were developed.
In data integrity verification, scholars have employed various techniques to generate metadata, including homomorphic tags, bilinear pairing algebraic signatures foundation codes, erasure codes RS codes based on cauchy metrics, and others. The researchers have implemented the ITable, skip list divide and conquer table Merkle hash tree in order to facilitate dynamic data revisions. Typically, integrity schemes consist of the subsequent stages:

Preprocessing Phase

To generate metadata, the original data is preprocessed using a predefined algorithm. The original file and metadata (for verification purposes) are both uploaded to the cloud service provider.

Verification Phase

The challenge request is transmitted by the auditor (TPA or proprietor) to the CSP, which generates the proof of possession utilizing the original data and metadata. The auditor is presented with the challenge proof in order to verify the integrity of the data stored in the cloud.

Provable Data Possession Characteristics

The functionality that a data integrity scheme offers to authenticate the ownership of data can be used to classify its characteristics. Security services, features, performance metrics, data verification coverage, and the state of verifiability are additional functional categories. It is responsible for ensuring data integrity and may also address availability and confidentiality.
The attributes that ought to be incorporated are robust integrity and soundness, which ensure that the information provided is accurate and verify data without the need to obtain the actual files. In relation to the computational cost of metadata data and data verification, communication expenses of data exchange, storage expenses, and detection probability, the performance of the protocol must be optimal. Coverage for data verification includes both static and dynamic data verification. As illustrated in Figure 1, the state of verifiability is contingent upon whether it offers public or private verifiability.

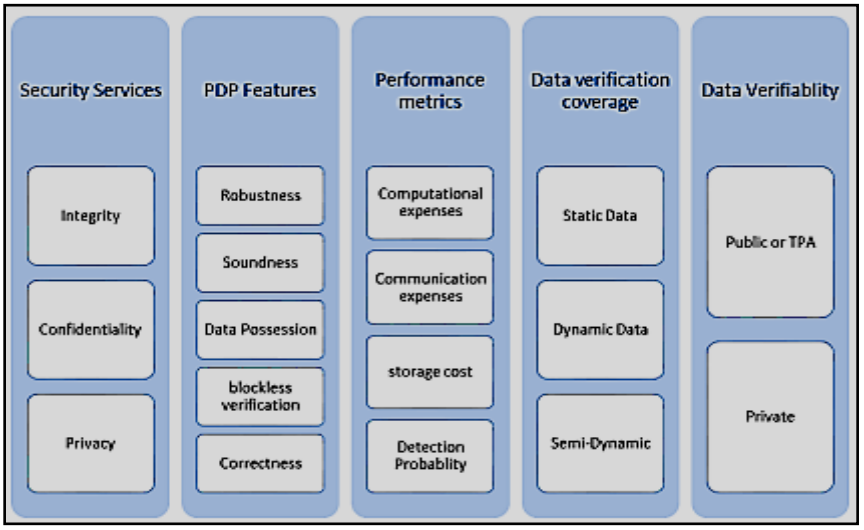


Figure 1: Provable Data Possession Characteristics (Sookhak et al.,2014)

Provable Data Possession Techniques

Pledgeable data possession (PDP) refers to the methods utilized to guarantee the ownership of data stored in the cloud. This segment provides an overview of the most recent PDP techniques developed by various researchers to validate the integrity of outsourced data without requiring the retrieval of the original data from cloud storage. This section will additionally examine how data integrity verification protocols account for the dynamic character of data through the utilization of various data structures.

Homomorphic verifiable tag based PDP

The initial model of PDP was introduced, which conducts data verification without requiring the download of the source data from an untrusted cloud server. PDP was introduced by the author as a solution to integrity checking issues caused by other protocols' deterministic approach and the server's costly computation of the entire file. Owner incurs local storage overhead and costly communication complexity by retaining metadata for a subsequent auditing task.

The author classified PDP into the following four polynomial time algorithms KeyGen, TagBlock, GenProof, and Check Proof according to its definition:

The algorithm KeyGen (1k) → (pk, sk) is executed by the client in order to initialize the scheme. The objective of this algorithm is to produce the pair of public and private keys (pk, sk) that will be utilized by the scheme to generate and validate proofs, generate metadata, and generate metadata.

TagBlock (pk, sk, f) → Tm: Tagblock is a client-side function that generates metadata tags in accordance with the input file and a pair of public and private keys.

GenProof(pk, F, cal, o) →V: Following the challenge message, GenProof is executed on the cloud server in order to calculate the proof of the provided challenge. It accepts the public key, f blocks, a challenge, and the metadata elements associated with the f blocks that are being challenged.

CheckProof (pk, sk, chal,V) → "success", "failure": executes once the client has obtained the proof of challenge. It accepts as input a pair of public and private keys, the challenge, and evidence of possession transmitted from the cloud storage.

Dynamic provable data possession (DPDP), an extended variant of the protocol introduced by [14] to accommodate the constraints of data dynamics, was proposed by [12]. In addition to its foundation on homomorphic verifiable tags, the protocol also introduces a secondary construction that utilizes an authenticated dictionary and an RSA tree [15]. The protocol operates on an n-block file F. It permits the deletion of any block in the file, modification of an existing block, and insertion of a new block at any ith position.

Additionally, a secure and efficient method for possessing provable data was suggested, which relies on homomorphic verifiable identifiers. The protocol enables public verifiability for data dynamic operations (insert, delete, update, insert), meaning that an authorized third party can determine whether or not the data is intact. Privacy is of the utmost importance when the TPA is involved, given that it can retrieve data from data proof. Additionally, the author assures confidentiality in the protocol's design.

Identity based PDP

The scheme was proposed with the data owner's identity as its foundation. It is suggested that identity-based PDP be implemented in order to simplify certificate management. The protocol's certificate administration contributes to its inefficiency. The system under consideration operates under the random Oracle paradigm with large public exponents and RSA assumptions. It also provides support for variable-sized data blocks and public auditing.

The protocol comprises the following principal entities: (i) the user, (ii) the cloud server, (iii) the third party auditor (TPA), and (iv) the private key generator (PKG). The identity-based integrity verification algorithm is implemented

procedurally as follows:
Configure it to generate a master public key and master secret key. Extract generates the secret key in accordance with the user's identity. TagGen generates the metadata identifiers for individual file blocks from the provided input of the file and identity ID.
It is the duty of Challenge to generate a challenge in accordance with the user ID. ProofGen generates the challenge's proof.
ProofCheck is responsible for validating the server-generated proof [17]. PDP based on Symmetric Key Cryptography proposed an additional A scalable and efficient PDP that operates exclusively on symmetric key cryptography and cryptographic hash functions; does not necessitate mass encryption; and supports dynamic data operations such as block appending, deletion, and modification. In the past, protocols employed asymmetric key cryptography, which required substantial computation power for large files and failed to account for dynamic data updates. Data, data owner, server, hash function (e.g., SHA-1, SHA-2), authenticated encryption/decryption scheme that provides both privacy and authenticity, pseudo-random function that efficiently computes random values, and pseudo-random permutation indexed under key are the fundamentals outlined in [19] for scalable and efficient PDP. AES, which generates a random sequence from a specified range, is regarded as an effective PRP.

BLS Signature
PDP was suggested by [7], which permits public auditing and data dynamics. The primary participants in the protocol are the file-storing client, the data-storing Cloud Storage server (CSS), and the third-party auditor (TPA), which can verify the integrity of the data by challenging the CSS. The rationale behind utilizing a third party is that the client may lack the necessary time, resources, or feasibility to monitor their data in the cloud. Therefore, any trusted TPA that possesses the client's public key can serve as a verifier.
To ensure that the data could be verified by the public, the author implemented a homomorphic authenticator based on BLS signatures and public key encryption. To process dynamic data, the author implemented a Merkle hash tree (MHT) [20]. The primary objective of MHT is to guarantee the integrity and security of the collection of elements by preventing any alterations or damage. The element hashes are stored in the leaves of MHT, which is designed as a binary tree. As the base node is composed of child nodes, any modification made to a child node will be promptly observed at the root node.
The primary operation of the protocol is to partition a file F into n sections. Generate the public and private key pair using the KeyGen() function. SigGen() accepts the private key and file blocks as input and returns a block-specific signature. The client then generates the origin of the MHT, which is a hash of the offspring nodes that correspond. The client signs the root with its private key before transmitting to the cloud storage file blocks, signature, and MHT. The data blocks are challenged by the verifier to the server, which then generates the proof using the data blocks, signature, and MHT.

Algebraic Signature based PDP
A dynamic PDP based on an algebraic signature was proposed in [3]. An algebraic signature is essentially an algebraic-property-containing hash function. The primary characteristic utilized in the development of data verification schemes is that calculating the sum of the signatures of a given number of random blocks yields an equivalent result to calculating the sum of the signatures of the corresponding block [21].
Additionally, [22] put forth a five-phase algorithmic signature scheme known as Setup, TagBlock, Challenge, ProofGen, and Proof Verify. While this approach effectively verifies static data, it fails to accommodate the dynamic characteristics inherent in big data.

Data Dynamics
To address the issue of dynamic data, numerous researchers have implemented various data structures. An explanation of the state-of-the-art solution is provided below.

Merkle Hash Tree
The process of modifying data in the cloud is as follows: the client initiates a request to the server to modify a specific portion of file. The initial client computes the modified block's signature before transmitting the updated data block accompanied by the computed signature. The server modifies the updated data block with the previous one, updates the corresponding signature and hash, and generates a new root upon request (see Figure 2).

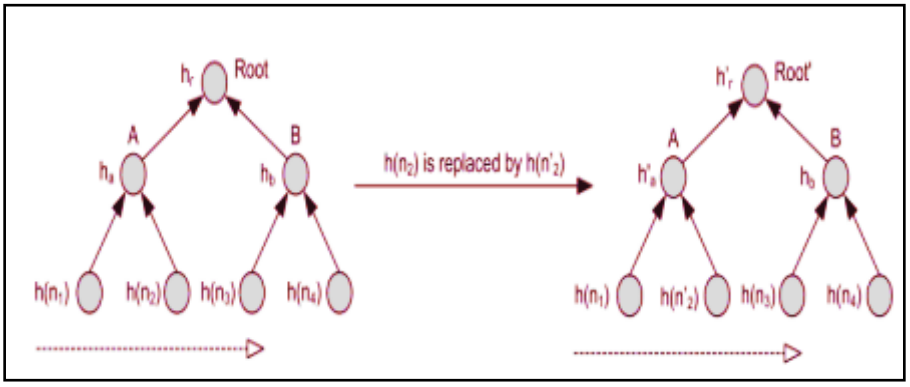


Figure 2: MHT Modification Operation

Data insertion and data modification are analogous in that the client generates the signature and transmits the corresponding data block. As illustrated in Figure 3, the insertion of the data block and signature occurs at the nth position.

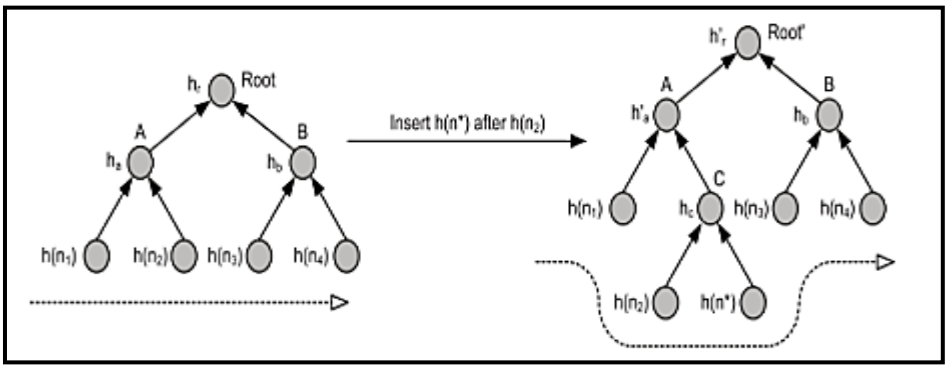


Figure 3: MHT Block Insertion Operation (Wang et al., 2012)

Any data element that the proprietor requests to be deleted is actually removed. The server deletes the data block and its corresponding signature following the deletion request. In addition, the server rebalances the tree and removes the corresponding block's hash in order to preserve the binary tree's property (Fig. 4).

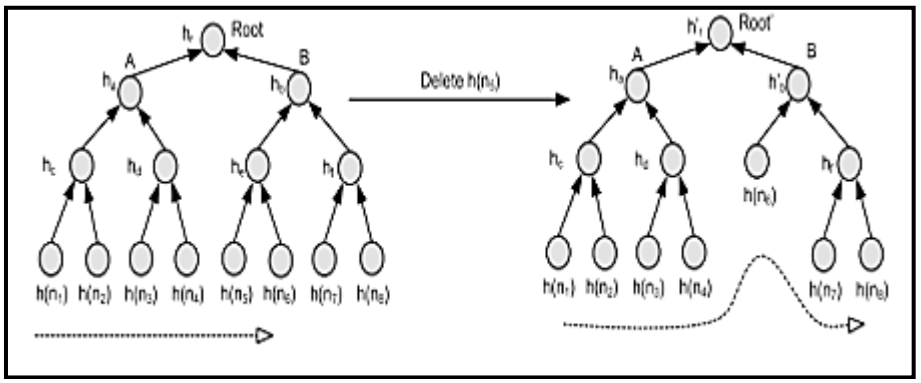
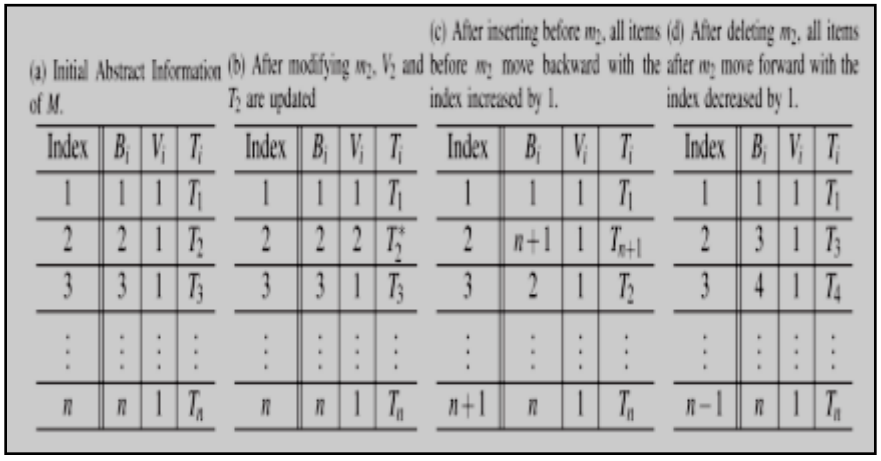


Figure 4: MHT Block Deletion Operation (Wang et al., 2012)

ITable

By utilizing ITable, the dynamic data modifications in [16] are rendered possible. ITable comprises entries including the version number of the data block, the original number of file block Bi, and the time stamp of the block's insertion or update. Fig. 5 illustrates the table updates in ITable.



(a) Initial Abstract Information of M.

Index	B_i	V_i	T_i
1	1	1	T_1
2	2	1	T_2
3	3	1	T_3
⋮	⋮	⋮	⋮
n	n	1	T_n

(b) After modifying m_2, V_2 and T_2 are updated

Index	B_i	V_i	T_i
1	1	1	T_1
2	2	2	T_2^*
3	3	1	T_3
⋮	⋮	⋮	⋮
n	n	1	T_n

(c) After inserting before m_2, all items before m_2 move backward with the index increased by 1.

Index	B_i	V_i	T_i
1	1	1	T_1
2	$n+1$	1	T_{n+1}
3	2	1	T_2
⋮	⋮	⋮	⋮
$n+1$	n	1	T_n

(d) After deleting m_2, all items after m_2 move forward with the index decreased by 1.

Index	B_i	V_i	T_i
1	1	1	T_1
2	3	1	T_3
3	4	1	T_4
⋮	⋮	⋮	⋮
$n-1$	n	1	T_n

Figure 5: ITable Data Dynamic Operations

Divide and Conquer Table

Insertion of a new data block occurs subsequent to the identification of the ith block, followed by an update to the file's metadata in DCT.

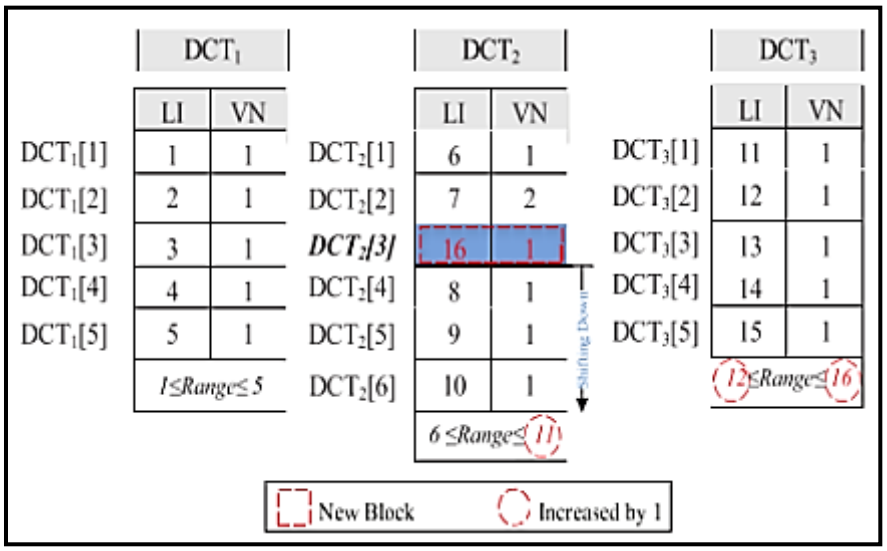


Figure 6: Block Insertion in DCT

The data is appended to the conclusion of the file via the data append operation. As shown in Fig. 7, at DCT, the entries in the final DCT table are also modified with the index of the appended block and the version number of the new block. When the DCT is at capacity, a new DCT table is generated.

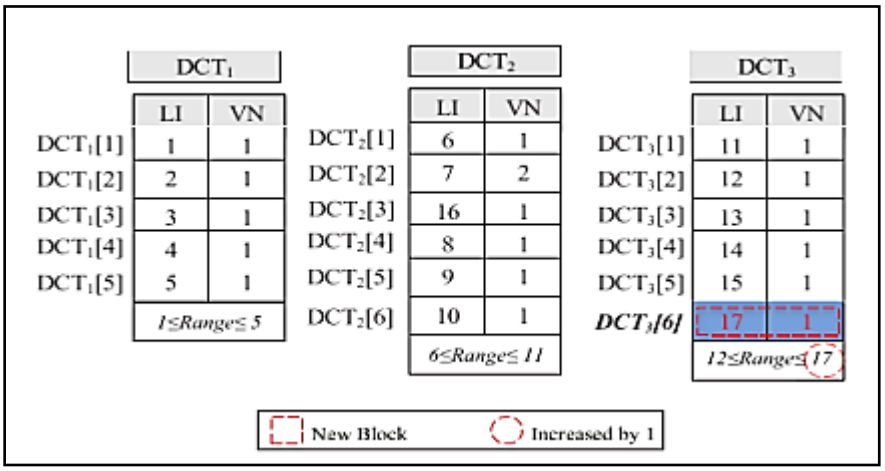


Figure 7: Block Append in DCT

By executing the data delete operation, a specified portion is removed from the original file. The proprietor requests

deletion of the data block index. Before the data fragment is purged from cloud storage, its corresponding signature is also removed. The entry from DCT is subsequently removed through a search. As illustrated in Figure 8, the remaining blocks are rebalanced by relocating upward.

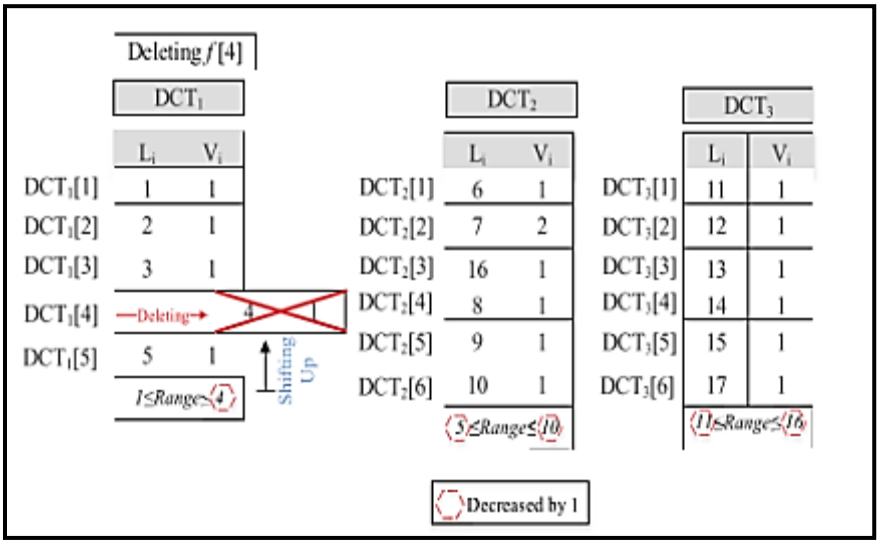


Figure 8: Block Deletion in DCT

Comparison

A comparison of various extant works on cloud data integrity verification is presented in Table I. Additionally, the table details the constraints of each individual task.

Table I. Comparison of Existing Work on Data Integrity

Presented	**Approach**	**Dynamic Updates approach**	**Weakness**
Ateniese *et al*.,(2007)	Homomorphic verifiabletas	N/A	Causes high computational expenses because of using RSA numbering.
Erway *et al*.,(2009)	Homomorphic verifiable tags	Using Authenticated rank based skip list	High computational cost and does not provide block less verification.
Ateniese *et al*.,(2008)	Cryptographic hash function and symmetric key cryptography	Token based list manipulation	High computational cost because of no derebalancing after dynamic insertion, deletion.
Wang *et al*.,(2012)	BLS Homomorphic authenticator	Merkle hash tree	After every updates function needs to calculate the root which incurs computational cost.
Yang & Jia,(2013)	Homomorphic verifiable tags	ITable	High computational cost because of nodere balancing in ITable after dynamic insertion, deletion.
Presented	Approach	Dynamic Updates approach	Weakness
Yu et al.,(2016)	RSA based	N/A	Does not support dynamic data updates.
Chen(2013)	Algebraic signature	N/A	Does not support dynamic data updates.
Sookhak etal.,(2007)	Algebraic signature	Divide and conquer table	Searching is not efficient and also creates bottle neck because of shifting of DC Ten tries.

Conclusion

Due to the fact that the proprietor of the data has less control over the data, which is accumulating daily in the cloud, the data is susceptible to a variety of attacks. This article examines the data integrity and organization strategies for outsourced data on cloud storage.

References

1. P. Jain, M. Gyanchandani, and N. Khare. (2016). Big data Privacy: A Technological Perspective and Review.
2. J. Big Data,3(1), 25.
3. S. Singh, Y. S. Jeong, and J. H. Park. (2016). A Survey on Cloud Computing Security: Issues, Threats, and Solutions. J. Net w. Comput. Appl.75,200-222.

4. M. Sookhak, A. Gani, M. K. Khan, and R. Buyya. (2017). Dynamic Remote Data Auditing for Securing Big Data Storage in Cloud Computing. Inf. Sci. (Ny), 380,101-116.
5. M. Ali, S. U. Khan, and A. V. Vasilakos. (2015). Security in Cloud Computing: Opportunities and Challenges. Inf. Sci.(Ny),305, 357-383.
6. D. A. B. Fernandes, L. F. B. Soares, J. V Gomes, M. M.Freire, and P. R. M. Inácio. (2014). Security Issues inCloud Environments: A Survey. Int. J. Inf. Secu., 13(2),113-170.
7. S. Subashini and V. Kavitha. (2011). A Survey on Security Issues in Service Delivery Models of Cloud Computing. J. Netw. Comput.Appl.,34(1), 1-11.
8. Q. Wang, S. Member, C. Wang, S. Member, and K. Ren.(2012). Enabling Public Auditability and Data Dynamicin Cloud Computing. IEEE Trans. Parallel Distrib. Syst.,22(5), 847-859.
9. K. Zeng.(2008). Publicly Verifiable Remote Data Integrity. 419-434.
10. Y. Yu, Y. Zhang, J. Ni, M. H. Au, L. Chen, and H. Liu. (2015). Remote Data Possession Checking with Enhanced Security for Cloud. Futur. Gener. Comput. Syst., 52,77-85.
11. L. Chen, S. Zhou, X. Huang, and L. Xu. (2013). Data Dynamics for Remote Data Possession Checking in Cloud Storage. Comput. Electr. Eng., 39(7):2413-2424.
12. F. Zafar et al. (2017). A Survey of Cloud Computing Data Integrity Schemes: Design Challenges, Taxonomy and Future Trends. Comput. Secur., 65,29-49.
13. C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia. (2009). Dynamic Provable Data Possession. CCS'09Proc. 16th ACM Conf. Comput. Commun. Secur.,213-222.
14. M. Sookhak, H. Talebian, E. Ahmed, A. Gani, and M. K.Khan. (2014). A Review on Remote Data Auditing in Single Cloud Server: Taxonomy and Open Issues. J. Netw.Comput.Appl.,43,121-141.
15. G. Ateniese et al. (2007). Provable Data Possession at Untrusted Stores. Proc. 14th ACM Conf. Comput. Commun. Secur.-CCS'07,1,598.
16. C. Papamanthou, R. Tamassia, and N. Triandopoulos. Authenticated Hash Tables Categories and Subject Descriptors. 437-448.
17. K. Yang and X. Jia. (2013). An Efficient and SecureDynamic Auditing Protocol for Data Storage in Cloud Computing. IEEE Trans. Parallel Distrib. Syst., 24(9),1717-1726.
18. Y. Yu et al. (2016). Cloud Data Integrity Checking withan Identity-based Auditing Mechanism from RSA. Futur. Gener. Comput. Syst., 62, 85-91.
19. G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik.(2008). Scalable and Efficient Provable Data Possession. Proc. 4th Int. Conf. Secur. Priv. Commun. Netowrks -Secur.'08, 1.
20. G. Ateniese. (2009). Proofs of Storage from Homomorphic Identification Protocols,1-14.
21. R. C. Merkle. (1979). Ralph C. Merkle ELX Si International, pp. 122-134.
22. S. J. Thomas Schwarz and E. L. Miller. (2006). Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage. Proc.-Int. Conf. Distrib. Comput. Syst.
23. L. Chen. (2013). Using Algebraic Signatures to Check Data Possession in Cloud Storage. Futur. Gener. Comput. Syst., 29(7),1709-1715.