Data Integrity for Dynamic Big Data in Cloud Storage: A Comprehensive Review and Critical Issues

Shamiel H. Ibrahim(✉), Maheyzah Md Sirat, and Widad M. M. Elbakri

University Technology Malaysia, 81310 Sukudi, JB, Malaysia
{hebshamiel2,mmewidad2}@graduate.utm.my, maheyzah@utm.my

Abstract. Cloud storage services provide vast storage space to solve the bottleneck of the data generated by different big data applications. However, the nature of big data in terms of its massive volume and rapid velocity, needs to be considered when designing data integrity schemes to provide security assurance for data stored in the cloud. The state of the art of data integrity in the cloud includes two primary schemes: (i) Proof of Retrievability (POR) and (ii) Provable Data Possession. Both techniques are designed to achieve the same goal in ensuring data integrity of outsourced data in cloud storage; However, PoR varies from PDP by error-correcting feature to retrieve the damaged outsourced data. This paper focuses on the proof of data retrievability technique (POR) for dynamic data. Dynamic data is defined as data under different update operations. The paper surveys the state of the art data integrity techniques for cloud storage (CS) and previous work on basic requirements for an effective data integrity technique for big data applications. Methods used to provide dynamic PoR are discussed before summarizing the classification of the POR state-of-the-art. The recently proposed techniques and their limitations are also discussed with issues to consider for future POR scheme design.

Keywords: Cloud computing · Cloud storage · Data integrity · Dynamic data update · Proof of data possession · Proof of data retrievability

1 Introduction

The Cloud computing (CC) model has led a new era of internet-based computing models that provide flexible, on-demand, and elastic capabilities. The most recent cloud survey shows that 94% of large enterprises utilize at least one cloud service [1]. This is largely due to the evolutionary change that cloud computing brings to data storage concepts; specifically shift from traditional server-attached storage to network-based distributed storage. In addition, the covid-19 pandemic has also contributed to accelerating cloud usage; surpassing expectation in cloud adoption [1]. Applications that are provided as services via the Internet as well as the hardware and software used in data centers to render such services are referred to as Cloud Computing [2]. NIST defined cloud

M. H. Miraz et al. (Eds.): iCETiC 2022, LNICST 463, pp. 67–81, 2023.
https://doi.org/10.1007/978-3-031-25161-0_5

computing model as a model for providing on-demand network access to a shared pool of customizable computing resources (e.g. Services, networks, servers, storage, and applications) that are supplied and released rapidly with minimal management effort or service provider engagement. NIST cloud computing model is composed of three service models (Software as a service (SaaS), Platform as a service (PaaS), and Infrastructure as a service (IaaS), four deployment models (Public, Private, Hybrid, and community cloud) and five essential characteristics (on-demand self-service, ubiquitous network access, measured services, rapid elasticity and location independent resource pooling) as depicted in Fig. 1.

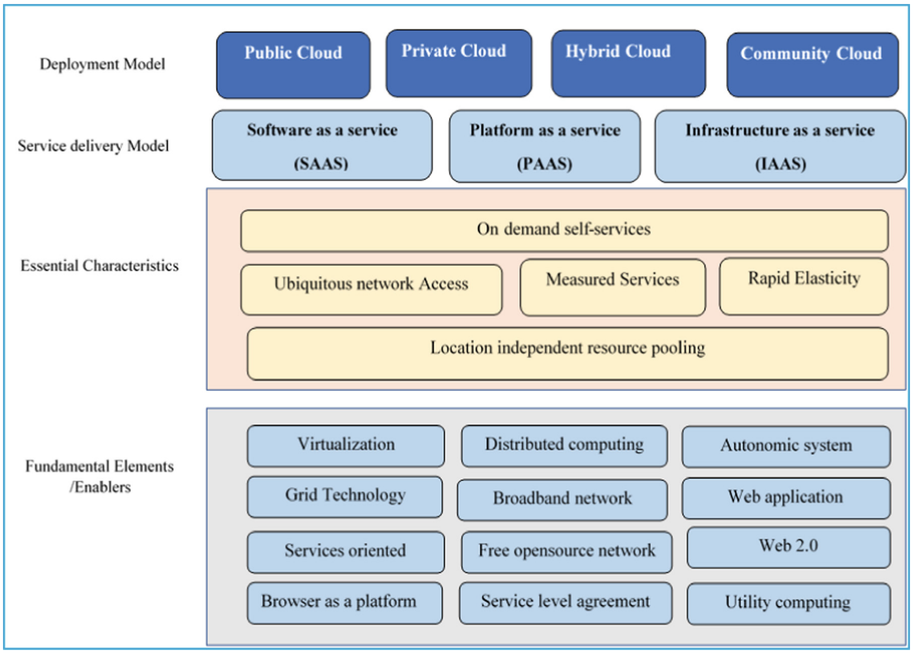

Fig. 1. NIST cloud computing model

2 Data Security Issues in Cloud Storage

Cloud computing provides advantages and benefits over the traditional computing model in terms of convenient commuting, data storage and backup, disaster recovery, and cost-effectiveness [3]. These attractive features support the growing demand for cloud storage services. The storage in the cloud consist of thousands of servers and storage devices joined together with supplied and distributed systems and other middleware to enable cloud service providers to offer cloud storage services to the end users. By adopting cloud storage services, enterprises can improve and expedite operating mode by providing on-demand and elastic data storage resources [4]. Despite its valuable characteristics, not all businesses favor shifting to the cloud, given that the adoption of cloud computing is restrained by several issues such as vendor lock-in, interoperability, reliability, data deletion assurance, and most important, data security and privacy [5]. When considering the shift to cloud storage enterprises‘ data security is the primary concern. The absence of efficient security practices can lead to unforeseen data breaches such as data leakage due to unauthorized access to cloud data storage. Due to concerns with risk of data leaks in cloud computing, pertinent studies have established relationship between the use of cloud computing and the number of data breach incidents perpetrated by attackers [6]. Figure 2, summarizes the security issues in cloud computing platform [7].

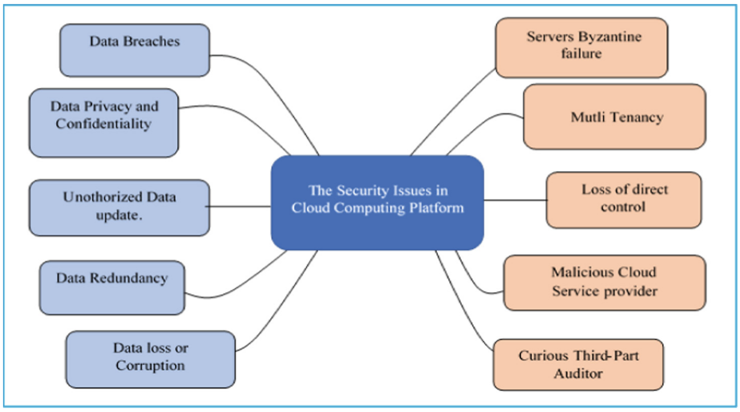
The Security Issues in Cloud Computing Platform
Data Breaches
Data Privacy and Confidentiality
Unothorized Data update.
Data Redundancy
Data loss or Corruption
Servers Byzantine failure
Mutli Tenancy
Loss of direct control
Malicious Cloud Service provider
Curious Third-Part Auditor

Fig. 2. Cloud computing security issues

Data security in cloud storage has gained considerable attention in academia and in the industry in recent years. So far, concerted scientific research efforts have been aimed at providing methods to guarantee the security and safety of data stored in the cloud [8].

NIST defines three major security requirements for information and information systems: confidentiality, Integrity, and Availability [9]. Data integrity is defined by NIST as an assurance that the data and programs are updated only with authentication and in an authorized manner. Therefore, data integrity techniques protect against unauthorized data alteration or deletion by assuring non-repudiation and validity. In contrast, loss of integrity is defined as an illegal alteration or deletion of data [10].

Data breach happens when sensitive information is exposed to an unauthorized party intentionally or unintentionally [6]. Several incidents of data breach in cloud computing have happened in recent history. For instance, a security breach at Dropbox (www.dropbox.com) was reported to have caused leakage of 68 million user accounts. Likewise, a database failure at Salesforce was reported to result in permanent data loss [11]. Another incident exposed more than 49 million user accounts in the AWS database. Instagram “influencers” and celebrities with large followers were targeted in the attack. Phone numbers, email addresses, profile images, and country locations were obtained during the incident. When the breach was found, it was revealed that the database had been accessible for at least 72 h without a password [12].

Theft of data from a cloud customer's system may occur when an employee acts maliciously; exploiting a configuration error to steal the login credentials of susceptible accounts and access their cloud infrastructure. Approximately 106 million credit card applications, including the names, addresses, and phone numbers of the applicants, were leaked by a former Amazon employee in 2019 [13]. Equifax, one of the three leading credit reporting companies in the US, revealed a compromise in September 2017; leading to a data breach that compromised the personal information of over 148 million Americans whose sensitive and confidential data such as names, birth dates, SSNs, residences, phone numbers, and driver's license numbers. In addition, over 209,000 credit cards were revealed. The magnitude the breach and severity were unparalleled [6].

Data loss may occur due to different reasons. For instance, cloud servers can lose clients' data due to internal reasons such as administrative errors, malicious insiders and hackers' invasions, or external reasons like natural disasters, power failure, and media damage. A typical example is an occurrence on August 31, 2019, when an Amazon AWS US-EAST-1 data center in North Virginia experienced power failure, on restoring the power, some Amazon Elastic compute cloud (EC2) instances and Amazon Elastic Block Store (EBS) volumes incurred hardware damages that led to inability to restore data [14]. The cloud servers can maliciously remove stored data to save storage space or gain economic benefits and generate a valid proof of data safety by reserving previously valid proof or intermediate proof to preserve its quality-of-service reputation.

3 Data Integrity Schemes in Cloud Storage- the State of the Art

The cloud storage model includes two main entities. The data owners (clients) who want to store their data in the cloud, and the cloud storage servers (CSS) or cloud servers (CS), which represent cloud service providers that own broad storage capabilities that are offered to the clients as paid services according to usage [15]. Data in cloud storage could be subjected to different types of security attacks due to resource sharing in the cloud storage model. Hence, data owners need to perform regular integrity checks without downloading the whole data or revealing its content since they do not completely trust the service providers. Several criteria must be considered in the integrity check techniques. In general, the integrity verification method must have a low communication complexity since the primary purpose is to avoid downloading a large chunk of the file to test its extractability. Besides, the protocol's storage overhead must be reasonable since significant server overhead would result in high-cost. Lastly, the integrity verification procedure must have minimal computational cost both for the data owner (who is likely to possess a lightweight device) and the server (whose computation work could also be expensive for the Data owner).

In providing data integrity in the cloud computing model, researchers have introduced two major techniques: Proof of Data Possession (PDP) [16] and Proof of Data Retrievability (POR) [17]. In contrast to the traditional methods that require the possession of entire data files to check the integrity of data, these techniques were established to check the integrity of data without downloading the whole data files from remote servers. Basically, the integrity check protocol executed between the Data owner (Verifier) and the Cloud storage server (Prover) as depicted in Fig. 3, is composed of three main phases:

- **Setup phase**: Before outsourcing the data files, the data owner (Verifier) establishes the auditing protocol by specifying its parameters and pre-processes the files to produce a piece of authentication data called tags or sentinels that are saved stored locally.
- **Challenge/Response Phase:** The data owner (Verifier) generates a challenge request and sends it to the cloud storage server (Prover), asking it to prove the integrity of the remotely stored data files. In turn, the cloud server (Prover) responds by generating the required proof.
- **Verification phase:** This process requires the verifier to check and verify the provided proof without holding the data files (blockless verification). Moreover, the cloud storage server (Prover) should be able to generate the proof without interfering with the content of the original data files (privacy-preserving).

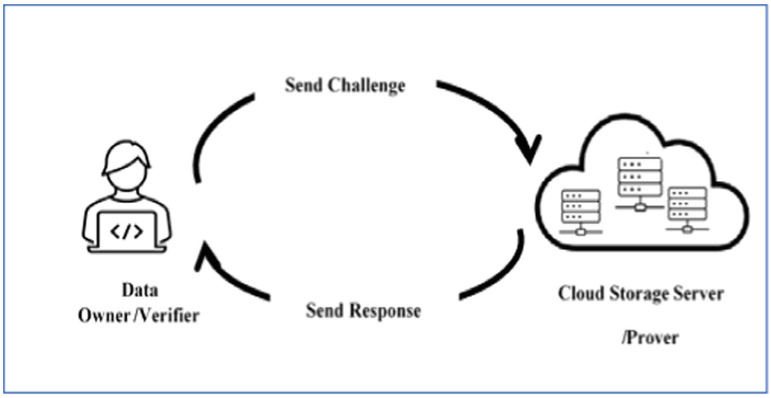

Fig. 3. Data integrity checking- Basic protocol

3.1 Proof of Data Possession-PDP

Many researchers have investigated integrity verification issues in the era of outsourced data storage, providing different schemes. Proposed the proof of data possession (PDP) technique. The scheme is established in the pre-processing phase, where the data owner divides the file into blocks of size 4KB each and generates an RSA alike tag for each file block. After utilizing the homomorphic property of RSA to generate a homomorphic verifiable tag (HVT), the data owner can combine the generated tags into a single value stored locally before sending the data file to the cloud server. Later, the client challenges the server to check data file availability by sending a random challenge against a randomly selected block index. The server responds to the challenge and provides proof of possession for the queried blocks and the corresponding tags. PDP is composed of two stages, setup and challenge phases, described as follows:

- **Setup Phase.** The client runs $KeyGen(1^k)$ to generate both public and private keys (pk, ask) as well as $TagBlock(pk, sk, m)$ to generate block tag Tm $= h\big((v||i).g^m)^d modN\big)$. The clients outsource the data file F along with sk and (T_{m1} …… T_{mn}) to the cloud server.
- **Challenge Phase.** The client generates a challenge chal and requests the server to prove data possession for a subset of blocks in the file accordingly; the server runs $GenProof(pk, F, chal)$ operation to generate the proof of data possession V; lastly, the client verifies the correctness of the provided proof by running $CheckProof(pk, sk, chal, V)$ operation, to output Success or Failure.

PDP protocol gives a probabilistic assurance that the server possesses the clients' data by checking a random subset of stored data. However, the scheme employs RSA-based modular operations for tag generation and integrity verification which entails a considerable computation time at the client and the cloud server-side; a problem for large-sized data files. Besides, the scheme is introduced for static data in private verification mode only and does not consider privacy-preserving issue [17].

3.2 Proof of Data Retrievability- POR

Proof of data retrievability (POR) was introduced by Juels, Kaliski [17] as a cryptographic proof of knowledge [18]. A POR system is characterized by an interactive POR protocol between a storage server (The prover) and the data owner (The verifier). The client submits a series of queries, and the cloud storage provider responds with the appropriate responses. By adopting cryptographic techniques, Juels and Kaliski's protocol enables the data owners to verify that they can retrieve their data files intact. The protocol starts by encoding the data blocks with efficient error correcting code before using a symmetric key encryption to encrypt the encoded data blocks. A one-way hash function is then utilized to produce a set of random values called sentinels that are to be embedded into the encrypted file using pseudo permutation function. Afterwards, the data owner challenges the server by asking it to return a certain subset of sentinel in encrypted data. Since the sentinels are indistinguishable from the server, it will be difficult to return its value if the file is corrupted or modified. However, POR itself does not protect data from corruption or loss. It only reveals data corruption or tampering and works with file robustness techniques to strengthen file availability by utilizing error-correcting code. The initial POR scheme which is designed for static archived storage with a fixed number of verification challenges does not consider the case of dynamic data updates. Although, this scheme is secure, however, the permutation of the file blocks to secure sentinels' positions conflicts with I/O sequential performance since sequential blocks in the original file are not in the same place in the resulting file after permutation; thereby limiting the possibility of extending the scheme to support dynamic update operation [7].

4 Public Data Integrity Techniques

Public integrity verification refers to the process of conduct data integrity verification by a party other than the data owner. According to the public model of [18], the scheme involves three entities: Data owner (DO), Cloud service provider (CSP), and Third-Party Auditor (TPA). The integrity checking process is delegated to the TPA who owns the efficient facilities to conduct such verification on behalf of the data owner. Figure 4 depicts the processes followed in a public data integrity audit, consisting of the same phases as the basic auditing technique but with the challenge generated and verified by a third-party auditor rather than the data owner. The scheme proposed by Shacham and Waters [18] introduced the first public data integrity technique that utilizes BLS signature [19] to verify the cloud server response with bilinear pairing function. However, the scheme results in high computational time on the verifier side. [19] Introduced another POR scheme that supports public verifiability by utilizing the independent third-party auditor (TPA), who is well-equipped with the suitable computing resource to challenge CSP and conduct a regular integrity check. However, the scheme does not address the data privacy issue.

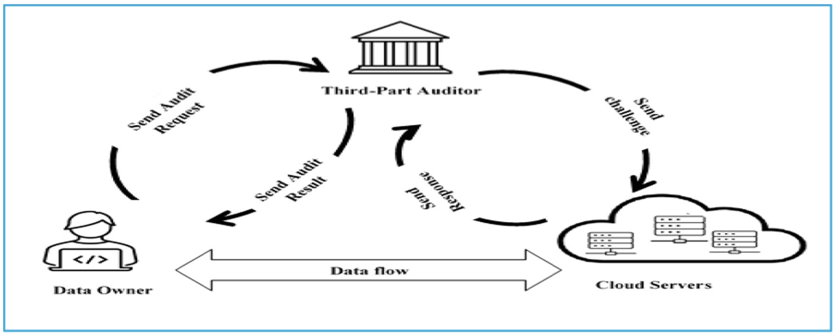

Fig. 4. Public data integrity

[20] Occupy a ring signature to generate an integrity verification token and hide the user's identity. However, since the scheme uses the bilinear pairing to verify the correctness of the signature, its computational cost is relatively high. Similarly, [21] proposed a public auditing method that avoids pairing operations by utilizing homomorphic MAC for tag generation. The scheme preserves the data privacy with blind file blocks with random values, however, the scheme provides for static data only. [22] Proposed a public data integrity technique using BLS-HVT to generate tags that ignores the privacy-preserving issues in the presence of TPA. Different public data integrity techniques abound in the literature using different advanced technique such as Blockchain [23], pairing free [24],and Homomorphic-based signature [25]. To design an efficient public data integrity protocol, preserving data privacy must be considered since both cloud service providers and TPA are not fully trusted despite utilizing different cryptographic techniques such as symmetric encryption [26], Homomorphic encryption [27], data blind techniques [28].

5 Data Integrity Schemes for Dynamic Data

Big data applications often require data upgrades. Given that the contents of many big data applications are dynamic and often updated, it is critical for a cloud security mechanism, such as a public auditing scheme, to support dynamic data effectively [29]. After sending the data to the cloud storage, the data owner may need to apply different update operations for many datasets like social networks, business transactions, and electronic health records. To propose data integrity techniques for dynamic data, the scheme needs to meet the same security guarantees (existence, consistency, and the ability to retrieve the data) for content that can undergo an unlimited number of (legal) alterations. Proofs of integrity techniques are mutual protocols that allow a verifier to validate the consistent existence and availability of data stored in an untrustworthy storage source like cloud storage. According to Juels and Kaliski, developing a PoR method that allows efficient updates is a central open problem [30]. The earliest data integrity schemes were proposed for static data [16] before they were extended to support dynamic data update operations [31]. Erway et al. were the first to define and construct dynamic PDP- (DPDP) [32], while [33] introduced the first dynamic POR (DPOR). The data owner or TPA can challenge the cloud server to verify the requested update after each update operation because it is important to ensure that the storage server keeps the updated data block and its related authentication metadata, to avoid replay and replace attack, and ensure data integrity after update operation, different scheme have been proposed in the literature. Generally, the scheme include additional algorithm to perfume the update operations:

- **RequestUpdate ()→Request.** This algorithm is executed by the data owner to specify the update type (Insert, Delete, and Modify).
- **Update ()→Updated Data Block.** This algorithm is executed by the Cloud server taking the encoded file and its related authentication data and the update request as input to produce the update file and its signature while showing the proof of the update as an output.
- **VerifyUpdate ()→True/False.** This algorithm runs by the TPA or the data owner. It takes the public key, the update operation, and the proof of update to output True or False.

To implement dynamic update operations, different authenticated data structures (ADS) are used to ensure the integrity and freshness of the data block after each update operation. The existing literature reveal different scheme that utilizes the authenticated data structure such as the Merkle Hash tree (MHT) [34], Skip list [35], oblivious RAM [36].

6 Proof of Data Possession for Dynamic Data

To support dynamic data updates operations [37] proposed an improvement for the scheme PDP to support integrity verification for dynamic data updates, reducing the computation cost by symmetric key operation for both the Setup and Verification steps, they enables the data owner to calculate a fixed number of possession verification tokens before outsourcing the data file using a pseudo-random function and pseudo-random permutation to generate a random token for randomly selected indexes. As an advanced PDP for dynamic data, different techniques utilize different methods to provide data integrity assurance in cloud storage. [27] used algebraic signature for authentication tag generation, however, less formal security analysis is performed on algebaric signature so far, whereas [38] used Boneh, Lynn, and Shacham (BLS) signature [39]. Also, [22] proposed a public auditing protocol based on BLS signature to generate a homomorphic verifiable authenticator (BLS-HVA). The scheme utilized doubly-linked info table to support dynamic data update operation. However, the scheme entails considerable computation due to pairing operation in verification phase. Besides, the scheme does

Table 1. Data integrity techniques- the state of the art

No	Scheme	Public	Dynamic	Privacy preserving	Tag	Limitations
1	[39]	No	Yes	Yes	Hash Function	• Restricted no of queries • Partially update operation • Probabilistic integrity assurance, because tags are generated to a subgroup of the tags
2	[43]	No	Yes	No	HVT-RSA	• Does not support privacy-preserving auditing on user's outsourced data
3	[44]	No	No	No	HVT -RSA	• For static data only
4	[24]	Yes	Yes	No	BLS-HVA	• The scheme does not provide privacy-preserving and recovery techniques
5	[42]	Yes	Yes	No	BLS	• The scheme does not provide privacy-preserving and recovery techniques

not consider keeping the privacy of data in the presence of TPA. Likewise, [40] proposed a public dynamic auditing scheme with fair arbitration for cloud storage. The scheme which considers the case of dishonest data owner utilizes BLS signature for tag generation (Table 1).

7 Proof of Data Retrievability for Dynamic Data

To provide proof of data retrievability for dynamic data, [43] introduced a dynamic POR based on BLS signature for tag generation and an improved authenticated data structure based on B+ tree of order three, Merkle Hash tree (MHT) and BLS called Merkle B+ tree (CMBT). The scheme achieves $O(logn)$ worst-case running time. [44] Suggest a practical dynamic POR scheme using the Merkle Hash tree to offer storage authentication since cn erasure-coded needs to be updated in any update block's operation. Besides, the scheme delays the updates of cn parity blocks upon writes. Rather, the client places the newly updated block into an erasure-coded log structure. The scheme, is however introduced for private auditing mode, and does not provide a privacy preserving technique. By using Oblivious Ram (ORAM) technique while utilizing minimum bandwidth regenerating codes (MBR), [45] proposed proof of retrievability with BLS signature for tag generation and bilinear pairing for verification. The scheme was introduced for static data and does not consider the data update operation. Sengupta and Ruj [46] proposed a public verifiable POR based on the storage structure of [44] that utilizes homomorphic hashing to generate another hierarchical storage to eliminate the cost of reading and writing at the data owner's side. The proposed scheme highlights the solution to the data update problem in POR data integrity model. To provide the proof of retrievability, data should be encoded with any of data recovery technique such as error correction code. In case of dynamic POR, the owner of data needs to apply different update operations over the already encoded data. Therefore, all codes related to the updated block should be updated. Besides, data authentication meta-data needs to be recalculated as well, which entails a considerable computational cost. To overcome this limitation, the encoded copy (C) is not updated for each write operation (insertion, deletion or modification). Instead, it is updated (or rebuilt) only after enough of the data blocks has been updated.

8 Issues Proof of Data Retrievability

Generally, three major issues need to be addressed when proposing a POR scheme to ensure data integrity in cloud storage. First, the security of the tag generation algorithm; the proposed technique must provide a secure integrity scheme that resists replay, replace and tag forgery attacks. The second issue is data disclosure by the cloud service provider or TPA during the auditing process. Since a enormous volume of outsourced data and the data owner's constrained computing capabilities make it difficult to evaluate and check the security services in the cloud by the data owners in their own [47]. To introduce the third party to audit the integrity of the cloud services the following two major requirements must be satisfied in order to introduce a third party auditor (TPA) securely, first the third-party auditor (TPA) should be able to audit cloud data storage effectively without requesting a local copy of the data and shouldn't place an extra online strain on cloud users. Second the third-party auditing procedure should not result in any new privacy risks for users' data. The third issue is the increased computational cost of the data integrity process for data update operations; since big data and its applications in the cloud are constantly expanding and changing, the computational and communication cost should be in lower bound in a way that does not affect the security requirements of the proposed data integrity technique, the proposed technique should adopt signature techniques with small key and signature size to reduces the storage space at both data owner and cloud server side.

9 Conclusion

Because of the rise of cloud computing, many bigdata applications are migrating their large data from local to cloud in order to take advantage of the easy and flexible services offered by a CSS, the data owners on the other hand, do not totally trust the cloud storage since they lack direct control over their large data. Data integrity techniques are gaining popularity since it is a necessary prerequisite for ensuring that consumers can trust a CS. There has been a lot of study done on data integrity techniques thus far. Private auditing and public auditing are the two types of schemes, and the literature emphasizes on the latter since it relieves resource-constrained users of a hefty burden. The goal of public data integrity is to allow users or TPAs to verify the integrity of data in CSS with little burden. The idea and system model of data integrity techniques are introduced in this study. Furthermore, a clear discussions are provided from dynamic data integrity techniques and progressive categorization of the data integrity methods depending on the mode of integrity services (PDP) and (POR) is offered focusing on POR for dynamic data. Furthermore, certain major issues and the accompanying technological methods are discovered. Finally, we explore open difficulties and challenges in order to provide some useful suggestions for future research (Table 2).

Table 2. Public dynamic proof of data retrievability

No	Scheme	Data Recovery	Tag	ADS	Limitation
1	[21]	RS Code	BLS	MHT	- Using the classic MHT construction cause an efficiency problem
2	[50]	ECC	Hash-compress and sign (HCS)	Range based -Rb23 Tree	- Huge computation overhead on server-side to rebalance 23RB TREE

(*continued*)

Table 2. (*continued*)

No	Scheme	Data Recovery	Tag	ADS	Limitation
3	[51]	RS Code	BLS-HA	MHT	- High computation cost at the verifier side due to pairing operation in BLS
4	[46]	Erasure code	MAC-based tag	MHT	- Public auditing is not supports - High computation cost due to use of locally decodable codes and Oblivious RAM
5	[52]	RS Code	HA	MHT	- Does not provide privacy-preserving
6	[33]	NC	(ASBB) Based	rb23Tree	- Privacy-preserving not supported
7	[48]	Erasure code	Homomorphic Hash	multi-level hierarchical MHT	- Privacy-preserving is not supported
8	[54]	Information Dispersal Algorithm	BLS -HA	MPHT Multiple Hash tree	- Only modify update operation is supported
9	[52]	NA	BLS-HA	IHT	- High computation cost at the verifier side due to pairing operation in BLS. Not specify data recovery

References

1. State of the Cloud Report | Flexera: State of the Cloud Report (2022). https://www.flexera.com/about-us/press-center/flexera-releases-2021-state-of-the-cloud-report#:~:text=61%20percent%20overall%20plan%20to,a%20centralized%20approach%20to%20cloud. Accessed 9 Mar 2021
2. Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., et al.: Above the clouds: a Berkeley view of cloud computing. Dept. Electrical Eng. Comput. Sci. Univ. California, Berkeley, Rep. UCB/EECS, **28**(13) (2009)
3. Liu, A., Yu, T.: Overview of cloud storage and architecture. Int. J. Sci. Technol. Res. (2018)
4. Marks EA, Lozano B. Executive's guide to cloud computing. John Wiley and Sons; 2010
5. Takabi, H., Joshi, J.B., Ahn, G.-J.: Security and privacy challenges in cloud computing environments. IEEE Secur. Priv. **6**, 24–31 (2010)

6. Sampson, D., Chowdhury, M.M.: The growing security concerns of cloud computing. In: 2021 IEEE International Conference on Electro Information Technology (EIT), pp. 050–5. IEEE (2021)
7. Tan, C.B., Hijazi, M.H.A., Lim, Y., Gani, A.: VIP2_A survey on proof of retrievability for cloud data integrity and availability: cloud storage state-of-the-art, issues, solutions and future trends. J. Netw. Comput. Appl. **110**, 75–86 (2018). https://doi.org/10.1016/j.jnca.2018.03.017
8. Taneja, D., Tyagi, S.: Information security in cloud computing: a systematic literature review and analysis. Int. J. Sci. Eng. Technol. **6**(1), 50–55 (2017)
9. FIPS P. 199 Standards for security categorization of federal information and information systems. Computer Security Division, NIST (2004)
10. Stallings, W., Brown, L., Bauer, M.D., Howard, M.: Computer Security: Principles and Practice. Pearson Upper Saddle River, Hoboken (2012)
11. Sharwood, S.: Salesforce.com crash caused DATA LOSS (2016). https://www.theregister.com/2016/05/13/salesforcecom_crash_caused_data_loss/. Accessed 02 Mar 2021
12. Chaturvedi, A., Bureau, E.: Instagram data breach trail leads to Chtrbox (2019). Accessed 31 May 2022
13. EPIC: Equifax Data Breach. https://archive.epic.org/privacy/data-breach/equifax/. Accessed 03 Apr 2022
14. Peng, S., Zhao, L., Al-Dubai, A.Y., Zomaya, A.Y., Hu, J., Min, G.Y., et al.: Secure lightweight stream data outsourcing for internet of things. IEEE Internet Things J. **8**(13), 10815–10829 (2021). https://doi.org/10.1109/jiot.2021.3050732
15. Odun-Ayo, I., Ajayi, O., Akanle, B., Ahuja ,R.: An overview of data storage in cloud computing. In: International Conference on Next Generation Computing and Information Systems (ICNGCIS). IEEE (2017)
16. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., et al.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, Virginia, USA, pp. 598–609 ACM (2007)
17. Juels, A., Kaliski, B.S.: VIP: PORs: Proofs of Retrievability for Large Files, pp. 584–97 (2007)
18. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_28
19. Shacham, H., Waters, B.: Compact proofs of retrievability. J. Cryptol. **26**(3), 442–483 (2012). https://doi.org/10.1007/s00145-012-9129-2
20. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004)
21. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04444-1_22
22. Wang, B., Li, B., Li, H.: Oruta: privacy-preserving public auditing for shared data in the cloud. IEEE Trans. Cloud Comput. **2**(1), 43–56 (2014)
23. Zhang, X., Xu, C., Zhang, X.: Efficient pairing-free privacy-preserving auditing scheme for cloud storage in distributed sensor networks. Int. J. Distrib. Sens. Netw. **11**(7), 593759 (2015)
24. Shen, J., Shen, J., Chen, X., Huang, X., Susilo, W.: An efficient public auditing protocol with novel dynamic structure for cloud data-pairing based. IEEE Trans. Inf. Forensics Secur. **12**(10), 2402–2415 (2017). https://doi.org/10.1109/tifs.2017.2705620
25. Wang, J., Peng, F., Tian, H., Chen, W., Lu, J.: Public auditing of log integrity for cloud storage systems via blockchain. In: Li, J., Liu, Z., Peng, H. (eds.) SPNCE 2019. LNICSSITE, vol. 284, pp. 378–387. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21373-2_29

26. Han, J., Li, Y., Chen, W.: VIP$_a lightweight and privacy-preserving public cloud auditing scheme without bilinear pairings in smart cities. Comput. Stand. Interfaces **62**, 84–97 (2019). https://doi.org/10.1016/j.csi.2018.08.004
27. Shao, B., Ji, Y.: Efficient TPA-based auditing scheme for secure cloud storage. Clust. Comput. **24**(3), 1989–2000 (2021). https://doi.org/10.1007/s10586-021-03239-x
28. Chen, D., Yuan, H., Hu, S., Wang, Q., Wang, C.: BOSSA: a decentralized system for proofs of data retrievability and replication. IEEE Trans. Parallel Distrib. Syst. **32**(4), 786–798 (2020)
29. ALmarwani, R., Zhang, N., Garside, J.: An effective, secure and efficient tagging method for integrity protection of outsourced data in a public cloud storage. Plos one **15**(11), e0241236 (2020)
30. Li, C., Wang, P., Sun, C., Zhou, K., Huang, P.: $WiBPA: an efficient data integrity auditing scheme without bilinear pairings. Comput. Mater. Continua. **58**(2), 319–333 (2019)
31. Liu, C., Chen, J., Yang, L.T., Zhang, X., Yang, C., Ranjan, R., et al.: Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. IEEE Trans. Parallel Distrib. Syst. **25**(9), 2234–2244 (2014). https://doi.org/10.1109/tpds.2013.191
32. Cash, D., Küpçü, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. J. Cryptol. **30**(1), 22–57 (2015). https://doi.org/10.1007/s00145-015-9216-2
33. Ren, Z., Wang, L., Wang, Q., Xu, M.: Dynamic proofs of retrievability for coded cloud storage systems. IEEE Trans. Serv. Comput. **11**(4), 685–698 (2018). https://doi.org/10.1109/tsc.2015.2481880
34. Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. ACM Trans. Inf. Syst. Secur. (TISSEC). **17**(4), 15 (2015)
35. Wang, C., Wang, Q., Ren, K., Cao, N., Lou, W.: Toward secure and dependable storage services in cloud computing. IEEE Trans. Serv. Comput. 5(2), 220-232 (2012)
36. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
37. Pugh, W.: Skip lists: a probabilistic alternative to balanced trees. Commun. ACM **33**(6), 668–676 (1990)
38. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM (JACM). **43**(3), 431–473 (1996)
39. Ateniese, G., Pietro, R., Mancini, L., Gene, T.: Scalable and efficient provable data possession. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (2008)
40. Gan, Q., Wang, X., Fang, X.: VIP22/6_Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud- Algebraic signature. Sci. Chin. Inf. Sci. **61**(12), 1–15 (2018)
41. Tian, H., Chen, Y., Chang, C.-C., Jiang, H., Huang, Y., Chen, Y., et al.: Dynamic-hash-table based public auditing for secure cloud storage. IEEE Trans. Serv. Comput. **10**(5), 701–714 (2015)
42. Jin, H., Jiang, H., Zhou, K.: Dynamic and public auditing with fair arbitration for cloud data. IEEE Trans. Cloud Comput. **6**(3), 680–693 (2018). https://doi.org/10.1109/TCC.2016.2525998
43. Erway, C.C.: Dynamic provable data possession_important. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, Ccs 2009 (2009)
44. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., et al.: Remote data checking using provable data possession. ACM Trans. Inf. Syst. Secur. **14**(1), 1–34 (2011)
45. Zhen, M., Yian, Z., Shigang, C.: A dynamic Proof of Retrievability (PoR) scheme with O(logn) complexity. In: 2012 IEEE International Conference on Communications (ICC), pp. 912–916 (2012)

46. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 325–336. ACM (2013)
47. Chen, J., Peng, Y., Du, R., Yuan, Q., Zheng, M.: Regenerating-codes-based efficient remote data checking and repairing in cloud storage. In: Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 143–150. TrustCom (2015)
48. Sengupta, B., Ruj, S.: Efficient proofs of retrievability with public verifiability for dynamic cloud storage. IEEE Trans. Cloud Comput. **8**(1), 138–151 (2020). https://doi.org/10.1109/tcc.2017.2767584
49. Razaque, A., Rizvi, S.S.: Privacy preserving model: a new scheme for auditing cloud stakeholders. J. Cloud Comput. **6**(1), 1–17 (2017). https://doi.org/10.1186/s13677-017-0076-1
50. Zheng, Q., Xu, S.: Fair and dynamic proofs of retrievability. In: Proceedings of the First ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, pp. 237–48. ACM (2011)
51. Wang, Q., Cong, W., Ren, K., Lou, W., Jin, L.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans.Parallel and Distrib. Syst. **22**(5), 847–859 (2011)
52. Li, J., Tan, X., Chen, X., Wong, D.S., Xhafa, F.: OPoR: Enabling Proof of Retrievability in Cloud Computing with Resource-Constrained Devices. IEEE Trans. Cloud Comput. **2**(3), 195–205 (2015)
53. Fu, A., Li, Y., Yu, S., Yu, Y., Zhang, G.: DIPOR: an IDA-based dynamic proof of retrievability scheme for cloud storage systems. J. Netw. Comput. Appl. **104**, 97–106 (2018)
54. Zhu, Y., Ahn, G.J., Hu, H., Yau, S.S., An, H.G., Hu, C.J.: Dynamic audit services for outsourced storages in clouds. IEEE Trans. Serv. Comput. **6**(2), 227–238 (2013). https://doi.org/10.1109/TSC.2011.51