Hindawi
Scientific Programming
Volume 2022, Article ID 4756899, 11 pages
https://doi.org/10.1155/2022/4756899

*Research Article*

# Cloud Data Integrity Verification Algorithm Based on Data Mining and Accounting Informatization

**Junli Wang,[1] Xiqian Yang,[2] and Zhi Li[3]**

[1]*Academy of Arts and Business, Xi'an Siyuan University, Xi'an 710038, Shaanxi, China*
[2]*Human Resources Management, Xi'an Siyuan University, Xi'an 710038, Shaanxi, China*
[3]*Academy of Tourism and Media, Xi'an Siyuan University, Xi'an 710038, Shaanxi, China*

Correspondence should be addressed to Zhi Li; 2192526167@stu.xaut.edu.cn

Received 8 July 2022; Revised 27 July 2022; Accepted 9 August 2022; Published 9 September 2022

Academic Editor: Juan Vicente Capella Hernandez

Data integrity verification means that the data in the cloud are uploaded by the user. In addition to the user's own update of the data, any external factors including the cloud service provider's data are destroyed, tampered with, and lost, and the data are not updated in a timely manner. Any inconsistencies in the actual data required can be detected by the user. This article aims to study cloud data integrity verification algorithms based on data mining and accounting informatization. This article proposes data mining technology, accounting information to help business managers assist management work. The article uses Company *H* as an example to illustrate the accounting information system and proposes a new information management strategy based on it. The CBF algorithm and data integrity verification algorithm are used to study the cloud storage data integrity verification protocol, the cloud data integrity verification model is constructed, the data program flow design is analyzed, and the time-consuming operation of file data insertion is analyzed. The experiment in this paper uses 16 standard mathematical calculation formulas to strengthen the analysis. The results show that the study of cloud data integrity verification algorithms based on data mining and accounting information is beneficial to the integrity and protection of data. When the number of documents added increases from 0 to 400, the document agreement shows an upward trend, and the agreement in this paper basically fluctuates between 10 and 80.

## 1. Introduction

When the client uses the cloud storage service function provided by the cloud service provider, the data on the cloud server are very important because the customer does not save a copy of the data locally. Customers hope and require that their stored data are not damaged or lost on the server side and that the data can be accessed at any time. This involves the problem of data integrity verification. The most traditional data integrity verification method is to use all data must be downloaded to the local hard disk for verification. However, because the number of users in the cloud environment is huge, and the data stored in the cloud are also very large, if the original method is adopted, it will bring a lot of overhead to all aspects. Because the data are managed by a cloud server provided by a third-party cloud service provider, the user loses control over the data. Data information security and privacy protection are completely dependent on cloud service providers.

The purpose of integrity verification is to ensure that the user's data stored in the cloud server are complete and has not been damaged or tampered with. For the incomplete data found during the verification process, data recovery can be performed on the damaged data. By optimizing resource allocation, computing resources are provided to users in the form of services, and a shared resource pool with network access and elastic expansion is realized. The ability is to access data in time. A verification mechanism to verify the integrity of the data in the cloud environment, making it independent of storage services and applicable to the existing basic service architecture, will bring a huge impetus to the smooth deployment and development of the cloud platform. The solution must not only ensure the high reliability of data verification but also not impose an excessive burden on users and cloud servers. At the same time, the privacy of user data should be protected during the verification process without affecting other data blocks, thereby reducing the computational overhead after the data are updated.

Real-world optimization problems usually involve multiple goals optimized for multiple variables at the same time under multiple constraints. Although multi-objective optimization itself may be a challenging task, it is equally difficult to understand the solution obtained. In this two-part paper, Ren Y discusses data mining methods that can be used to extract knowledge about multi-objective optimization problems from the solutions generated in the optimization process. In addition to assisting the optimization process in future design iterations through expert systems, this knowledge is expected to provide decision makers with deeper insights into the issues. The current paper investigates several existing data mining methods and categorizes them according to the method and the type of knowledge discovered. Most of these methods come from the field of exploratory data analysis and can be applied to any multivariate data [1]. In the cloud storage framework, once customers store their data remotely on the cloud storage provider, they will lose their physical knowledge of outsourced data control. The risk of unauthorized access to data has increased dramatically. One of the most serious problems of cloud storage is to ensure the correctness of outsourced data. Specifically, we need to protect these data from unauthorized operations; we also need to detect and restore user data after accidental changes. We propose a publicly verifiable scheme to protect the integrity of cloud data and support dynamic maintenance. The scheme is based on a location-aware Merkle tree. We use triples to define the nodes of the new Merkle tree. The node records the location of the corresponding node so that users can directly calculate the root value to verify the consistency of the challenge-response block without retrieving the entire Merkle tree [2]. Cloud computing is the latest trend in the IT field. It transfers computing and data from desktops and handheld devices to large processing centers and data centers, respectively. It has been proposed as an effective solution for data outsourcing and on-demand computing to control the rising cost of enterprise IT setup and management. However, with the cloud platform, the user's data are moved to remote storage, so that the user loses control of his data. This unique function of the cloud is facing many security and privacy challenges that need to be clearly understood and resolved. One of the important issues to be solved is to provide proof of data integrity, that is, the correctness of user data stored in cloud storage. Users cannot physically access the data in Clouds. Therefore, a mechanism is needed to allow users to check whether the integrity of their valuable data is maintained or destroyed. These methods use additional storage space by maintaining multiple copies of data or requiring the presence of a third-party verifier. A solution is proposed to solve the problem of proving data integrity in cloud computing, through which users can check the integrity of their data stored in the cloud [3].

Data integrity refers to the fact that data have not been tampered with or destroyed without authorization, to ensure that the data exist in a complete and true manner according to the wishes of the data owner. This article introduces data mining technology to mine potential information levels to meet different needs and different levels of learners. The requirements also provide users with decision-making support and reduce the probability of risk occurrence and accounting informatization to help corporate managers assist in management. The CBF algorithm and data integrity verification algorithm are used to study the cloud storage data integrity verification protocol, the cloud data integrity verification model is constructed, the data program flow design is analyzed, and the time-consuming analysis of file data insertion operations is based on data mining. Research on cloud data integrity verification algorithm is under the conditions of accounting informatization.

## 2. Cloud Data Integrity Verification Algorithm Based on Data Mining and Accounting Informatization

*2.1. Data Mining Technology.* Data mining usually refers to the discovery of inherent laws and valuable information from massive, seemingly irregular, and unsystematic data, and is generally combined with statistical software or modern computer technology. The application of data mining is wider and wider, and the potential level of information to be mined is deeper. It meets the requirements of learners with different needs and different levels. It also provides users with decision-making support and reduces the probability of risk [4].

The process of data mining generally needs to go through the four stages of obtaining data, preparing data, mining data, and expressing and explaining mining results, as shown in Figure 1.

*2.2. Accounting Informatization.* The definition of accounting informatization can be viewed from two perspectives. Broadly speaking, accounting informatization refers to all tasks involving accounting informatization. From the selection and customization of accounting information systems, what software to use and how to use them, to managers' views on accounting informatization and the continuing education and training of relevant accounting personnel, all belong to the scope of accounting informatization. In a narrow sense, accounting informatization refers to the combination of modern information technology and accounting information and presenting it in a new form of system to help business managers assist management. Accounting information system, we call it AIS for short, it can complete the collection, storage, processing, and accounting of accounting information, and then systematically conduct accounting management analysis and decision making [5]. *H* Company's current accounting information system serves the company's management. After years of exploration and application, it now includes the following relevant business modules. Through this concept,

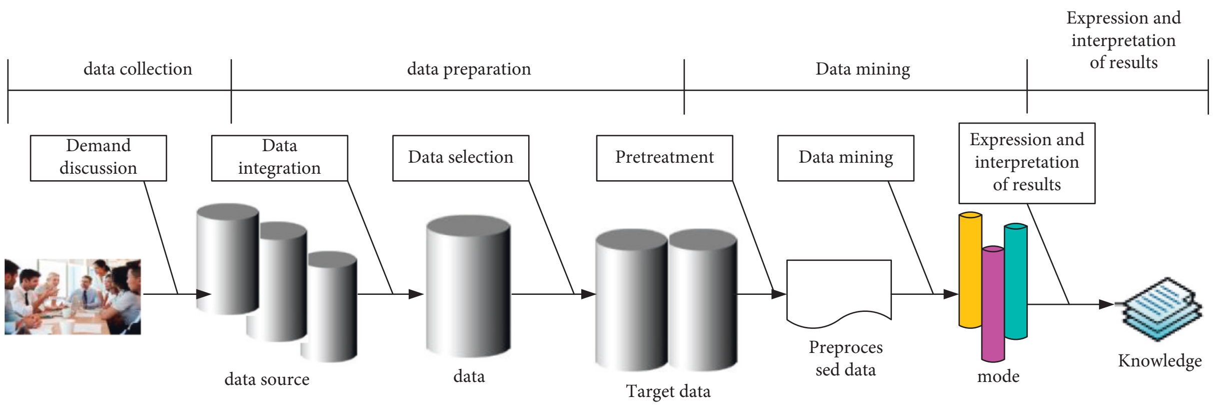

Figure 1: Data mining process.

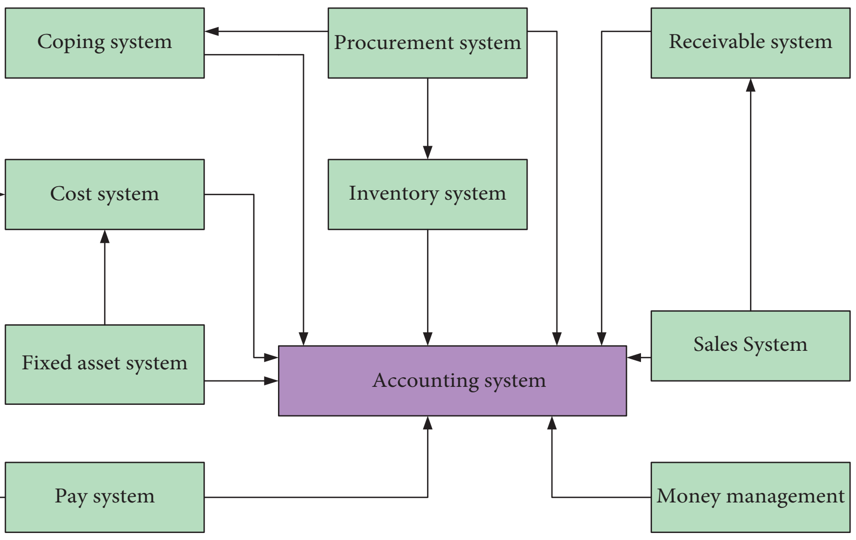

Figure 2: *H* company's accounting information system framework diagram.

we can see that accounting informatization includes scientific information technology, which integrates accounting information resources and uses them horizontally and vertically in order to establish an efficient system that promotes corporate management [6]. The architecture diagram of *H* company's accounting information system is shown in Figure 2.

*2.3. Data Integrity Verification Algorithm.* First, we introduce the basic process of the data integrity algorithm. According to the different running time and execution tasks, it can be divided into two sub-phases: the initialization phase and the challenge-response phase [7]. The frame diagram is shown in Figure 3.

Initialization stage: first, the client requests related to services from the cloud service provider CSP and trusted third-party TPA, respectively, and the two parties conduct interactive authentication to determine the client ID, data block, and check element storage server address and other information. Then, the client divides the file to be stored into blocks, then calls the Setup algorithm and the TagGen algorithm to generate checksums, and then uploads all data blocks and checksums to the CSP and TPA. After the TPA receives the check element, it registers it in its check task table [8]. After all steps are successfully returned, the client registers the verification information of the file in its management table. Finally, after confirming that the file has been stored correctly in the CSP, the local copy can be deleted. Challenge-response stage: TPA periodically initiates verification according to the protocol or performs verification after receiving the request sent by the client, queries the verification element management table, obtains the client ID and related verification element information, and passes them to the verification module. The verification module calls the Challenge algorithm according to the verification meta-information to initiate a challenge to the CSP cloud storage server. After receiving the challenge, the CSP invokes the Response algorithm according to the challenge keyword, and generates a response. After receiving the response message returned by the CSP, the TPA verification module executes the algorithm in the verification element and returns the result "Success" or "Failure." Finally, TPA returns the data verification result according to

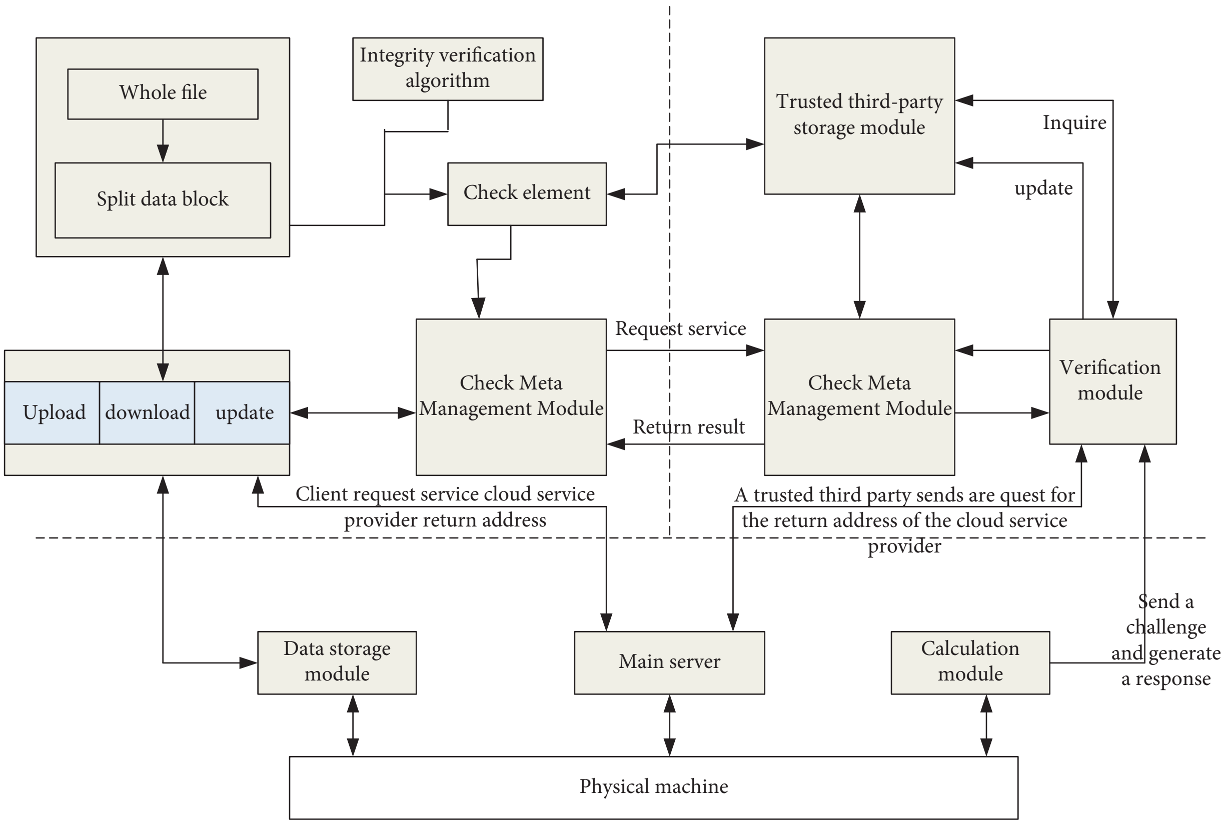
Whole file
Split data block
Integrity verification algorithm
Check element
Trusted third-party storage module
Inquire
update
Upload
download
update
Check Meta Management Module
Request service
Return result
Check Meta Management Module
Verification module
Client request service cloud service provider return address
A trusted third party sends are quest for the return address of the cloud service provider
Send a challenge and generate a response
Data storage module
Main server
Calculation module
Physical machine

Figure 3: Integrity verification algorithm framework diagram.

the protocol [9]. If it is "Failure," TPA returns the index number ID of the failed block to the client and reports that the data block is damaged [10].

### 2.4. CBF Algorithm

*2.4.1. The Misjudgment Rate of the Check Element.* CBF has a certain false positive and misjudgment rate. The meaning in this article is that there is a certain probability that a damaged data block can pass the verification [11]. Obviously, this probability should be very low. For this reason, it is necessary to evaluate its false positive rate. Consider in a test process, the check element CBF bit array of a single data block is $V$; its length is $L$; the hash function is $h1, h2, \ldots, hk$; the number of verification rounds is $t$; and its false positive and misjudgment rate are Peer [12]. Assuming that $k$ hash functions are independently and randomly obeyed uniform distribution, that is to say, the probability of any hash function mapping to a certain position is $1/L$. When the $t$ round check value is generated, a total of $k \times \mathrm{t}$ values are inserted into the CBF. At this time, the expected probability that a bit in the CBF array $V$ is still 0 is $p_0$:

$$p_0 = \left(1 - \frac{1}{L}\right)^{kt} \approx e^{-kt/L}. \tag{1}$$

For a data block $x$ to pass the check, it must have $V\,[hi\,(x)] > 0, \forall\, \mathrm{i} \in [1, k]$; that is, the $k$ position values are all greater than zero.

Well, sometimes there is a misjudgment

$$p_{eer} = (1 - p_0)^k \approx \left(1 - e^{(-kt/L)}\right)^k = \exp\left(k\ln 1 - e^{(-kt/L)}\right). \tag{2}$$

Let $p\prime$ represent the proportion of 0 in the actual bit array V, then its mathematical expectation is E $(p\prime) = p$ 0. Mitzenmacher proved in that the distribution of $p\prime$ is very close to $p_0$. Therefore, Peer can be used as the false-positive rate of the check element. To select the appropriate number of hash functions $k$ to minimize the misjudgment rate Peer, let $g = k\ln(1 - e - kt/L)$. From (2), when $g$ reaches the minimum, the misjudgment rate Peer is the smallest [13]. Take the derivative of $g$ with respect to $k$, and let $\mathrm{d}g/\mathrm{d}k = 0$, then

$$k = \ln 2\left(\frac{L}{t}\right), \tag{3}$$

$$\begin{aligned} p_0 &= \frac{1}{2}, \\ p_{\mathrm{eer}} &= \left(\frac{1}{2}\right)^k \\ &= \left(\frac{1}{2}\right)^{\ln 2\,(L/t)} \\ &= (0.618)^{(L/t)}. \end{aligned} \tag{4}$$

*2.4.2. The Minimum Space Length L and the Number of Hash Functions k.* The number of hash functions affects the amount of calculation of CBF, but because the hash function calculation speed is very fast, if the *H*3 hash function is used, its speed can reach the nanosecond level. The length of the check element affects storage and communication costs. Relative to the space occupied by the check element, the number of hash functions can be used as a secondary parameter. In other words, the problem now is to control the false-positive rate Peer under a certain threshold [14]. The optimal number of hash functions $k$ is obtained to minimize the storage space $L$ of the check element, which is more advantageous in communication. According to formula (2), we can obtain the derivative of $k$, simplifying

$$k = -\frac{\ln p_{\mathrm{eer}}}{\ln 2}. \tag{5}$$

Then, substituting the formula (2) is to obtain the minimum value of the length $L$ of the check element CBF at this time:

$$L \geq t \ \log_2 \ e * \ \log_2\left(\frac{1}{p_{\mathrm{eer}}}\right). \tag{6}$$

*2.4.3. Number of Counter Bits.* The number of bits in the counter increases by one for each round of check value added to the check element. When the digit of the counter is $W$, it can count up to $2w - 1$. Then, what needs to be considered is what value is set for the number of bits W of the counter so as not to overflow [15]. Li Fan et al. pointed out in the literature that $W = 4$ is sufficient for most applications. Let $c\,(u)$ be the value of the $u$-th counter, then the probability that the counter value is $j$ is

$$pr\,(c\,(u) = j) = C_{tk}^{j}\left(\frac{1}{L}\right)^{j}\left(1 - \frac{1}{L}\right)^{tk-j}. \tag{7}$$

Then, the probability that the value of the $u$-th counter is greater than $j$ is

$$pr\,(c\,(u) \geq j) \leq C_{tk}^{j}\left(\frac{1}{l}\right)^{j}. \tag{8}$$

A total of $L$ counters are the probability that the largest counter is greater than $j$

$$pr\,(\max\,(c\,(u)) \geq j) \leq L \cdot C_{kt}^{j}\left(\frac{1}{L}\right)^{j} \leq L \cdot \left(\frac{etk}{jL}\right)^{j}. \tag{9}$$

Also known from formula (8), $k \leq (L/t)\ \ln 2 \leq$ , then we can get

$$pr\,(\max\,(c\,(u)) \geq j) \leq L\left(\frac{e\ \ln\ 2}{j}\right)^{j}. \tag{10}$$

When $W = 4$, $j = 2$,·and $w = 16$, there are

$$pr\,(\max\,(c\,(u)) \geq 16) \leq 1.37 \times 10^{-15} \times L. \tag{11}$$

At this time, the overflow probability is already very small, which means that most of the time a 4-bit counter is sufficient.

## 3. Data Integrity Verification Experiment

*3.1. Research on Cloud Storage Data Integrity Verification Protocol.* There are three main participants in the cloud storage model, public cloud storage, private cloud storage, and hybrid cloud storage: the data owner is also called the user, who uploads the data to the cloud storage server for storage, and can customize the cloud storage service; a trusted third party, because it needs to pass Network access to data that has been previously stored to the cloud storage server by the data owner, and has audit capabilities that users do not have, and can help users audit data files stored in the cloud, thereby reducing the user's calculations during the verification phase; cloud storage servers are the "places" where cloud storage service providers build storage services, and they have super powerful computing and storage capabilities [16, 17]. From the perspective of data security, the cloud storage server is not completely reliable (it may accidently erase stored data, or maliciously delete infrequently used data, privately reduce the number of backup data, etc.) [18]. Therefore, in order to ensure the integrity of remote data, users will develop a reliable mechanism to entrust a trusted third party to query whether the data stored on the server are complete. Generally, before storing the data, the user first preprocesses the data to be stored to prepare for the integrity verification in the verification phase. It should be noted here that in some applications, the data owner and the trusted third party may be the same individual or belong to the same group. In some environments, there may be more than one user, and there may be multiple authorized users who can access the data stored in the cloud server. Data files are accessed and updated [19]. The specific secure cloud storage model is shown in Figure 4.

Data integrity detection here is an operation to be performed regularly, because when the data are damaged, only timely remedial measures can be taken to repair it. If the data are damaged and not detected in time, it may cause data destructiveness. Increasing it will even cause the overall unavailability of data, which will bring huge loses to users [20]. Moreover, due to the limitation of computing resources, users do not have a lot of energy and time to verify the integrity of remote data. Therefore, in general, users will entrust data integrity testing to an experienced and trusted third party, and will not disclose it. The data file information is given to a trusted third party, which better realizes the user's privacy protection. The specific data integrity detection protocol is generally divided into the following three stages:

(1) Setup stage: the data owner first runs the key generation and label generation algorithms to preprocess the file, save the key pair information, and then transfer the processed data file to the cloud server for storage.

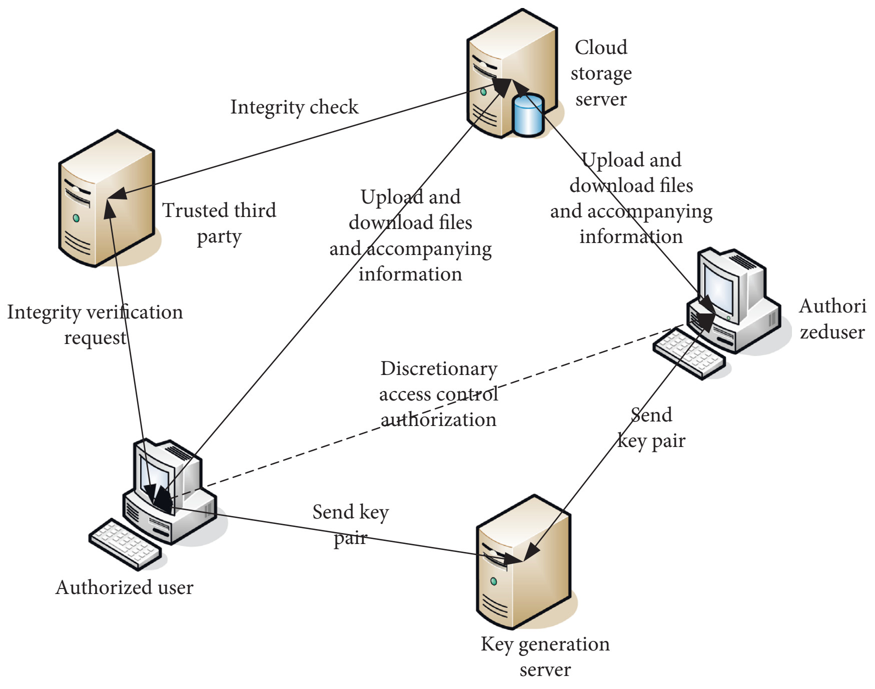
Cloud storage server
Integrity check
Trusted third party
Upload and download files and accompanying information
Upload and download files and accompanying information
Authori zeduser
Integrity verification request
Discretionary access control authorization
Send key pair
Send key pair
Authorized user
Key generation server

Figure 4: Secure cloud storage model.

(2) Challenge stage: the validator runs the challenge generation algorithm to generate challenge information and sends it to the cloud server.

*3.2. Cloud Data Integrity Verification Model.* Based on the data integrity verification protocol, we also refer to the paper *A position-aware Merkle tree for dynamic cloud data integrity verification*. In that paper, the authors propose a publicly verifiable scheme to protect the integrity of cloud data and support dynamic maintenance [21, 22]. Based on this, the following model is successfully designed in this paper with reference to the location-aware Merkle tree-based model. According to the number of participants, the cloud data integrity verification model is divided into a two-party verification model and a three-party verification model that supports public auditing [23]. As shown in Figure 5, the two-party model consists of users and CSPs. Users store data in the cloud and retain metadata information necessary for integrity verification. When data need to be used, a request is made to the cloud server, and the cloud returns user data. When the user wants to verify the integrity of his data, he uses the "challenge-response" mode to verify whether the data are verified through calculations based on the data block evidence provided by the CSP complete.

However, in the two-party model, neither the user nor the CSP is suitable for performing integrity verification, because neither can guarantee to provide fair and credible verification results. The user and the cloud server do not trust each other. In addition, the user, the client, requires certain computing and storage capabilities, so most integrity verification protocols introduce third-party audits to communicate the interaction between users and CSPs, improve the efficiency of cloud data integrity verification, and reduce

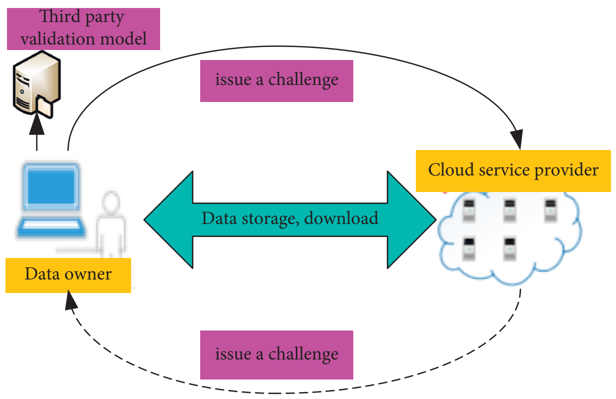

Figure 5: Two-party verification model.

the computing and storage overhead of the client [24] as shown in Figure 6.

In the three-party verification model, the user uploads data to the cloud. When the user needs integrity verification, the user authorizes a third-party audit to challenge the CSP. The CSP obtains the corresponding data and integrity evidence according to the challenge request and returns it to the third-party audit. Finally, after a third-party audit and verification, the results are notified to users. However, verification is replaced by a third-party public audit, and there is a potential threat of colluding with CSP to deceive users or false verification [25].

*3.3. Data Sampling Mechanism.* Generally, the data that need to be integrity-checked in the cloud storage server are large data files. If all data blocks in the file are checked every time, the overhead will be very large. The random sampling

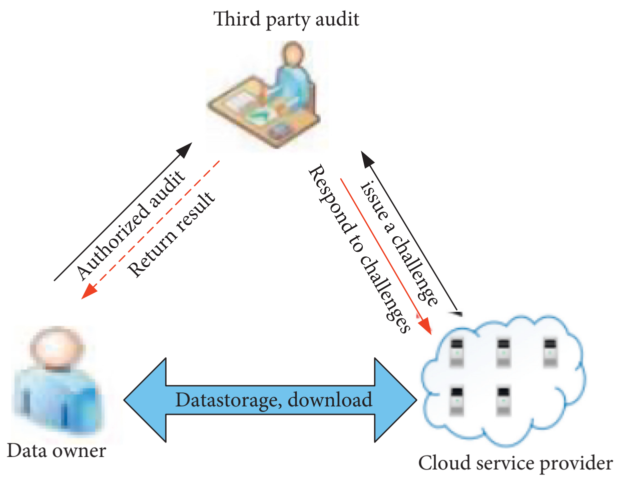

Figure 6: Three-party verification model.

mechanism will greatly reduce the number of blocks to be verified. Therefore, a sampling mechanism is also used in this algorithm. The question that arises from this is that for a file with $N$ data blocks, how many blocks are extracted each time is more appropriate for verification. We consider the general verification process. Assuming that the total number of file data blocks is $N$, a is the file block damage rate (i.e., $N \times$ a blocks Damaged), and $c$ is the number of blocks extracted in one check. $X$ is the total number of damaged data blocks found in the spot check. It is the probability that the file is found to be damaged, which is defined as the spot check confidence; then,

$$c * \left(\mathrm{time}_z + \mathrm{tume}_e + \mathrm{time}_{h(|m|)}\right)|m|, \tag{12}$$

$$\frac{N - N * a - i}{N - i} \geq \frac{N - N * a - i - 1}{N - i - 1}, \tag{13}$$

$$1 - \left(\frac{N - N * a}{N}\right)^c \leq p_{\mathrm{find}} \leq 1 - \left(\frac{N - N * a - c + 1}{N - c - 1}\right)^c. \tag{14}$$

From $C \leq N$, we get

$$p_{\mathrm{find}} \approx 1 - (1 - a)^c, \tag{15}$$

$$c \approx \frac{\ln\left(1 - p_{\mathrm{find}}\right)}{\ln(1 - a)}. \tag{16}$$

That is to say, the confidence of the spot check is only related to the file damage rate a and the number of blocks $C$ spot checked, and has nothing to do with the total file size. Obviously, as the number of sampled blocks increases, the credibility will increase, and $P$ find is consistent with the larger the number of blocks in each spot check, the greater the probability of damage is found.

## 4. Data Time-Consuming: Program Flow Analysis

*4.1. Data Program Flow Design Analysis.* The main functions of the system include users moving files to the cloud for online hosting, checking the integrity of files in the cloud when users obtain them, and performing online operations on files in the cloud. Sequence diagram shows the dynamic cooperation among multiple objects by describing the time sequence of sending messages between objects. The following is an introduction to the program flow involved in the above-mentioned main functions [26]. The interactive process of user transferring files to the cloud and online hosting is shown in Figure 7. The description of the process is as follows:

(1) User sends the file upload request to CSS, and CSS obtains the available Storage address from Tracker and returns it to User.

(2) User processes files locally, including dividing blocks, generating ACSL files, signing the nodeHash of the root node of ACSL, and so on.

(3) User accesses Storage, calls the file upload interface to transfer the file and ACSL authentication structure, signature value, etc. to Storage, and receives the file mapping name returned from Storage.

(4) User transfers the file mapping name to CSS, and CSS stores the key-value pair "original file name-file mapping name" in the file name mapping table [27].

When User accesses files stored in the cloud, the interaction flow of each component is shown in Figure 8. The description of the process is as follows:

(1) User sends the file name to CSS, and CSS finds the corresponding mapping name from the mapping table according to the received file name.

(2) The CSS sends the mapping name to the Tracker, and the Tracker returns the available Storage address where the file is stored to the CSS. The CSS accesses the Storage, obtains the ACSL authentication structure and the file owner's signature according to the mapping name, and generates it according to the ACSL authentication structure to complete evidence.

(3) CSS returns the integrity evidence, the signature of the file owner, the storage address, and the file mapping name to the User.

(4) User calls the file access interface to obtain the file according to the Storage address and file mapping name, and uses the integrity evidence and the owner's signature to verify its integrity [28].

When the user needs to operate files in the cloud online, the interaction flow of each component is shown in Figure 9. The description of the process is as follows:

(1) User sends the file name and dynamic operation type to CSS, and CSS finds the corresponding mapping name from the mapping table according to the file name.

(2) The CSS sends the mapping name to the Tracker, and the Tracker returns the available Storage address where the file is stored to the CSS. The CSS accesses the Storage and obtains the ACSL authentication structure and the signature of the file owner

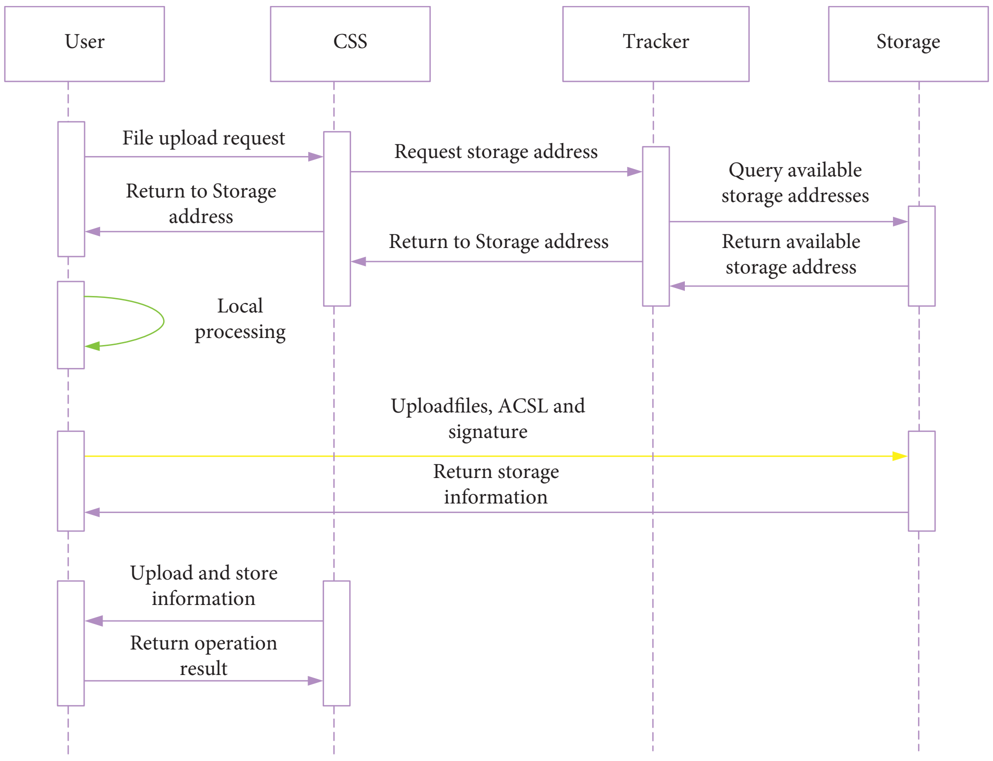
User
CSS
Tracker
Storage
File upload request
Request storage address
Query available storage addresses
Return to Storage address
Return to Storage address
Return available storage address
Local processing
Uploadfiles, ACSL and signature
Return storage information
Upload and store information
Return operation result

Figure 7: File upload process.

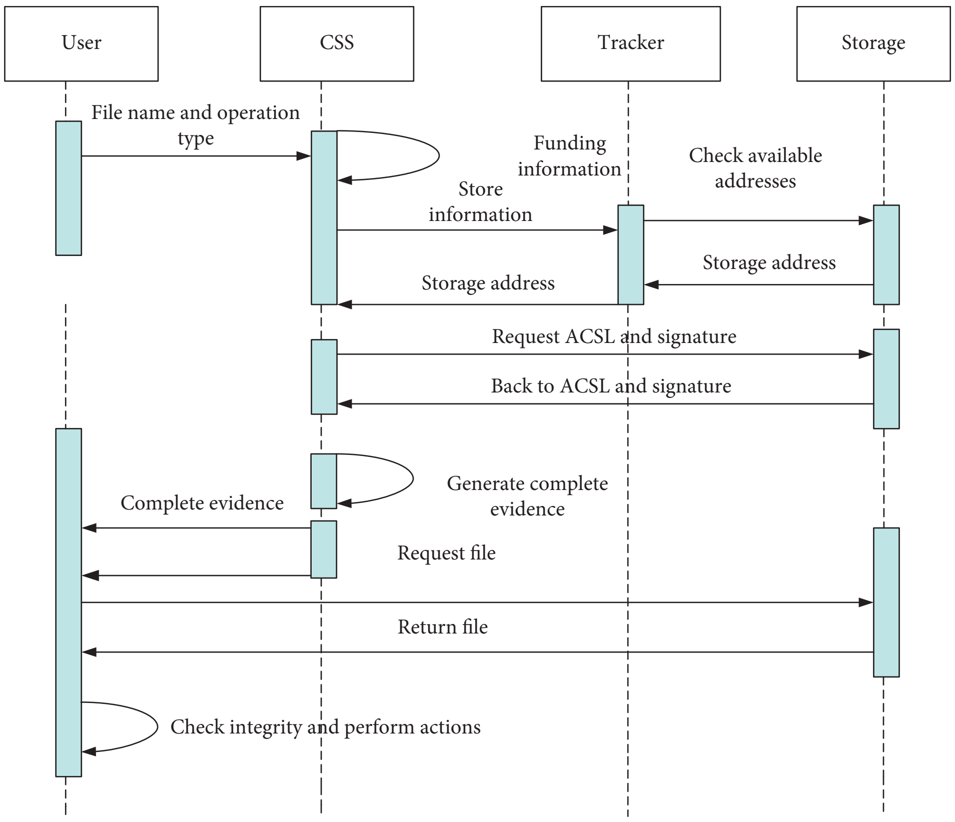
User
CSS
Tracker
Storage
File name and operation type
Funding information
Check available addresses
Store information
Storage address
Storage address
Request ACSL and signature
Back to ACSL and signature
Generate complete evidence
Complete evidence
Request file
Return file
Check integrity and perform actions

Figure 8: File access and integrity check.

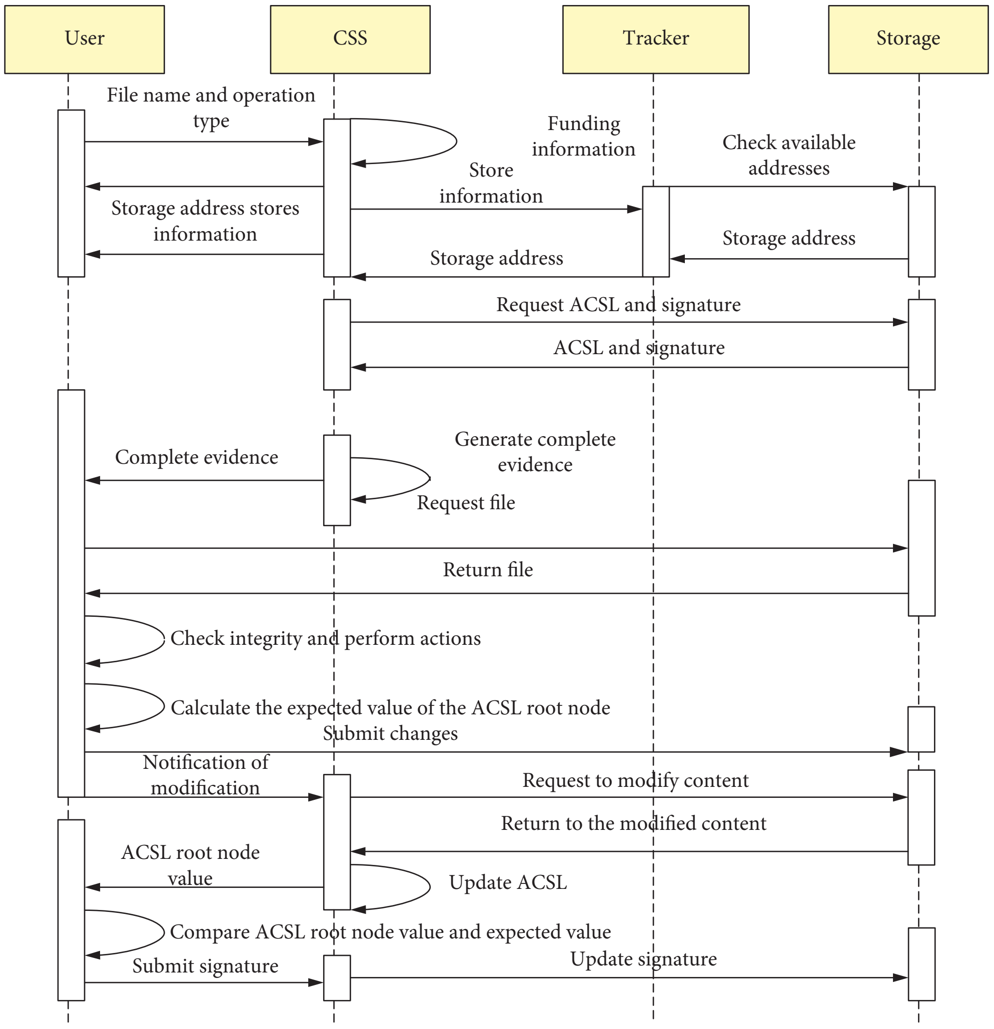
User
CSS
Tracker
Storage
File name and operation type
Store information
Funding information
Check available addresses
Storage address stores information
Storage address
Storage address
Request ACSL and signature
ACSL and signature
Complete evidence
Generate complete evidence
Request file
Return file
Check integrity and perform actions
Calculate the expected value of the ACSL root node
Submit changes
Notification of modification
Request to modify content
Return to the modified content
ACSL root node value
Update ACSL
Compare ACSL root node value and expected value
Submit signature
Update signature

Figure 9: Dynamic operation process.

according to the mapping name, and according to the dynamics submitted by the User Operation type, generating different integrity evidence.

(3) CSS returns the integrity evidence, the signature of the file owner, the storage address, and the file mapping name to the User.

(4) The User obtains the corresponding file content according to the dynamic operation type according to the Storage address and file mapping name, and calls the file access interface, and uses the integrity evidence and the owner's signature to check the integrity of the obtained content [29].

(5) User modifies the acquired file locally, calculates the modified ACSL root node nodeHash expected value, and submits the modified file content to Storage, notifying the CSS operation has been submitted.

(6) CSS obtains the submitted file content from Storage, updates the ACSL authentication structure, and returns the new ACSL root node nodeHash value to User.

(7) User confirms that the nodeHash value of the ACSL root node returned by CSS is valid, signs it with the private key, and submits the signature value to CSS [30].

(8) CSS verifies whether the signature submitted by the user is valid, and if it is valid, the result of the dynamic operation is persisted; otherwise, the operation fails.

*4.2. Time-Consuming Analysis of File Data Insertion Operation.* Test the time consumed by randomly inserting file blocks at any position in the file for authentication.

The test result is shown in Figure 10. From the experimental results, it can be seen that the protocol proposed in this paper and the protocol proposed in the literature have similar performance when inserting file blocks at random positions.

In theory, dynamic operation will make the authentication structure used by the protocol proposed in the literature appear unbalanced. The unbalanced authentication

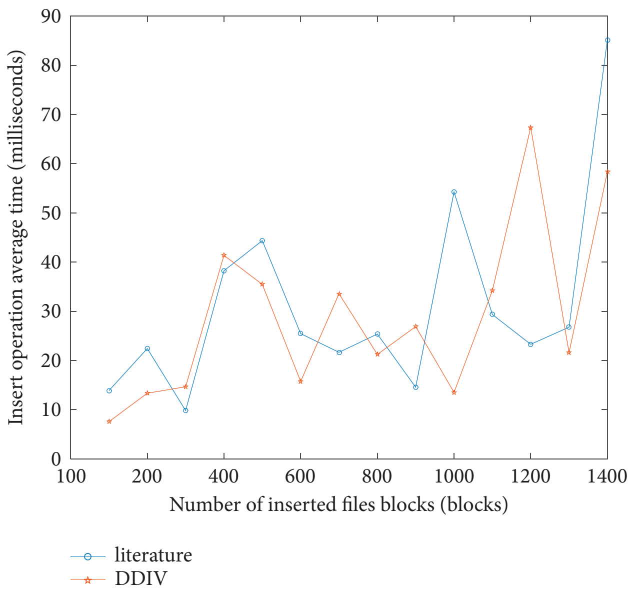

FIGURE 10: Insertion operation at random position takes time.

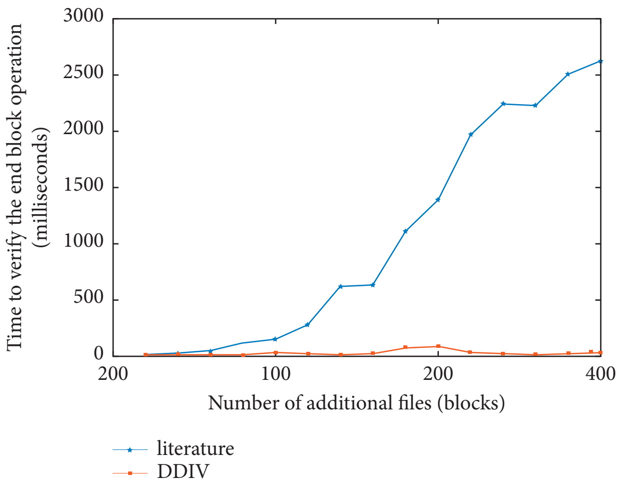

FIGURE 11: Verification time-consuming in an unbalanced state.

structure will cause the authentication path of some file blocks to be longer than other file blocks. For file blocks with too long paths, please check them. It will consume more time when it is complete. Below, in the case of imbalance, the test compares the time consumed by the protocol proposed in this article and the protocol proposed in the literature for integrity verification. The test results are shown in Figure 11.

From theoretical analysis, it can be seen that in the proposed integrity verification protocol, the authentication structure will not appear unbalanced. Therefore, the time consumed to verify the integrity of the last file block and the number of file blocks contained in the current file is statistically significant. It is a logarithmic relationship, and in an unbalanced state, the time required for the protocol in the literature to verify the integrity of the last file block has a linear relationship with the number of additional file blocks. It can be seen from the experimental results that the proposed protocol is better than the protocol proposed in the literature in this case.

## 5. Conclusions

People pay more and more attention to data security in the cloud storage environment, and integrity verification is the cornerstone of data security. The traditional integrity verification technology requires the verifier to hold a complete verification object. However, due to resource constraints, especially network resource constraints, the traditional integrity verification technology has relatively large deficiencies in the cloud storage environment. At the same time, because the cloud storage service provider is not completely credible, there is a possibility that user files may be damaged or lost due to hardware failure, network attack, or misoperation of the management personnel. In this case, the cloud storage service provider may act out of its own. It is chosen to conceal or even deceive users due to the consideration of interests. Therefore, in the cloud storage environment, the possibility of service providers actively launching attacks should also be considered when checking the integrity of files. This paper constructs a cloud data integrity verification model, analyzes the design of the data program flow, and analyzes the time-consuming operation of file data insertion, and studies the cloud data integrity verification algorithm based on data mining and accounting informatization. It is found through the comparison of experimental research results. The integrity verification protocol is better than the protocol proposed in the literature, which is conducive to the integrity verification and protection of the data. Due to the limitations of the authors' capabilities and the length of the study, the system protocol proposed in this paper was not experimented with running in a larger database. The database for the experiments can be expanded in future research with a view to the adaptability of the system proposed in this paper.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] Y. Ren, J. Shen, H C. Chao et al., "Efficient data integrity auditing for storage security in mobile health cloud," *Peer-to-Peer Networking and Applications*, vol. 9, no. 5, pp. 854–863, 2016.

[2] J. C. Saranya, V. Usha, and D. S. Alex, "Dynamic data integrity and checkpoint recovery using public auditing in cloud storage," *International Journal of Civil Engineering & Technology*, vol. 8, no. 9, pp. 692–700, 2017.

[3] W. Amol and V. Rastogi, "Data Data Integrity Auditing of Cloud Storagentegrity auditing of cloud storage," *International Journal of Computer Applications*, vol. 133, no. 17, pp. 17–21, 2016.

[4] M. E. Ghazouani, M. Kiram, and L. Er-Rajy, "Blockchain & multi-agent system: a new promising approach for cloud data integrity auditing with deduplication," *International Journal*

*of Communication Networks and Information Security*, vol. 11, no. 1, pp. 175–184, 2019.

[5] S. H. Abbdal, T. A. Kadhim, Z. A. Abduljabbar, and Z. A. Hussien, "Ensuring data integrity scheme based on digital signature and Iris features in cloud," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 2, no. 2, pp. 452–460, 2016.

[6] R. Hariharan, D. Madan Raja S, and M. Daniel, "AN extensive review on data integrity schemes and security issues in cloud paradigm," *International Journal of Advanced Research*, vol. 8, no. 6, pp. 1093–1100, 2020.

[7] S. Gokulakrishnan and J. Gnanasekar, "Data Data Integrity and Recovery Management Under Peer to Peer Convoluted Fault Recognition Cloud Systemsntegrity and recovery management under peer to peer convoluted fault recognition cloud systems," *Journal of Computational and Theoretical Nanoscience*, vol. 17, no. 5, pp. 2147–2150, 2020.

[8] J D. Jl and C. Maria, "Data integrity method for dynamic auditing in cloud environment," *Indian Journal of Computer Science and Engineering*, vol. 11, no. 6, pp. 843–850, 2020.

[9] B. Kang, L. Si, H. Jiang, C. Li, and M Xie, "ID-ID-Based Public Auditing Protocol for Cloud Data Integrity Checking with Privacy-Preserving and Effective Aggregation Verificationased public auditing protocol for cloud data integrity checking with privacy-preserving and effective aggregation verification," *Security and Communication Networks*, vol. 2018, no. 3, pp. 1–9, Article ID 3205898, 2018.

[10] R. J. Wang, F. L. Zhang, and X. Y. Wang, "A cloud data integrity verification protocol based on improved skip lists," *Journal of the University of Electronic Science and Technology of China*, vol. 47, no. 1, pp. 88–94, 2018.

[11] R. Jegadeesan and C. Sahithi, "A scalable mechanism of cloud storage for data integrity auditing without private key storage," *International Journal of Research*, vol. 10, no. 3, pp. 1–8, 2021.

[12] S. Kumar and L. Parthiban, "Cloud Cloud Data Integrity Auditing Over Dynamic Data for Multiple Usersata integrity auditing over dynamic data for multiple users," *International Journal of Intelligent Engineering and Systems*, vol. 10, no. 5, pp. 239–246, 2017.

[13] S. Kaushik, A. Tripathi, and P. P. Singh, "Review Review Paper on Data Integrity for Cloudaper on data integrity for cloud," *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, vol. 7, no. 5, pp. 1408–1411, 2019.

[14] M. Tian, L. Wang, H. Zhong, and J. Chen, "Attribute-based Attribute-based Data Integrity Checking for Cloud Storageata integrity checking for cloud storage," *Fundamenta Informaticae*, vol. 163, no. 4, pp. 395–411, 2018.

[15] R. C. Subashini, "Identity-Identity-Based Proxy-Oriented Data Uploading and Remote Data Integrity Checking in Public Cloudased proxy-oriented data uploading and remote data integrity checking in public cloud," *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, vol. 7, no. 4, pp. 400–405, 2019.

[16] A. Alia, M. A. Jusoh, H. J. Mohsin, and H. Yas, "The effect of e-accounting and mediated by internal control system on the performance of SME in Iraq," *American Journal of Business and Operations Research*, vol. 3, no. 1, pp. 5–38, 2021.

[17] N. El-Rashidy and N. Moustafa, "Mobile Mobile Cloud Database Security: Problems and Solutionsloud database security: problems and solutions," *Fusion: Practice and Applications*, vol. 7, no. 1, pp. 15–29, 2021.

[18] M. S. Krishna, D. Sravani, and B. A. Triveda, "Enhance data integrity for data storage in cloud computing," *International Journal of Engineering & Technology*, vol. 7, no. 2, pp. 68–70, 2018.

[19] S. Saxena and M. Sharma, "Secure Secure Technique to Achieve Data Privacy and Data Integrity in Cloud Computingechnique to achieve data privacy and data integrity in cloud computing," *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, vol. 6, no. 10, pp. 545–548, 2018.

[20] F Zhu, A. Kalra, T. Saif, Z. Yang, K. H. Yang, and A. King, "Parametric analysis of the biomechanical response of head subjected to the primary blast loading – a data mining approach," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 19, no. 10, pp. 1053–1059, 2016.

[21] R. Kanniga Devi, M. Gurusamy, and P. Vijayakumar, "An efficient cloud data center allocation to the source of requests," *Journal of Organizational and End User Computing*, vol. 32, no. 3, pp. 23–36, 2020.

[22] N. Baskaran and R. Eswari, "Efficient VM Efficient VM Selection Strategies in Cloud Datacenter Using Fuzzy Soft Setelection strategies in cloud datacenter using fuzzy soft set," *Journal of Organizational and End User Computing*, vol. 33, no. 5, pp. 153–179, 2021.

[23] H. Lu, R. Setiono, and H. Liu, "Effective data mining using neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 957–961, 1996.

[24] C. Helma, T. Cramer, S. Kramer, and L. D. Raedt, "Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds," *Journal of chemical information and computer sciences*, vol. 44, no. 4, pp. 1402–1411, 2004.

[25] D Adeniyi, Z. Wei and Y. Yongquan, Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method," *Applied Computing and Informatics*, vol. 12, no. 1, pp. 90–108, 2016.

[26] V. Chaurasia and S. Pal, "A novel approach for breast cancer detection using data mining techniques," *Social Science Electronic Publishing*, vol. 2, no. 1, pp. 2320–9801, 2014.

[27] L. Xu, C. Jiang, J. Wang, and J. Yuan, "Information Information Security in Big Data: Privacy and Data Miningecurity in big data: privacy and data mining," *IEEE Access*, vol. 2, no. 2, pp. 1149–1176, 2014.

[28] X S. Yan, L. S. Zheng, and L. Zheng, "Fundamental Fundamental Analysis and the Cross-Section of Stock Returns: A Data-Mining Approachnalysis and the cross-section of stock returns: a data-mining approach," *The Review of Financial Studies*, vol. 30, no. 4, pp. 1382–1423, 2017.

[29] T Jiang, X. Chen, J. Ma, and J. Ma, "Public Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocationntegrity auditing for shared dynamic cloud data with group user revocation," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2363–2373, 2016.

[30] J. Mao, Y. Zhang, J. Liu et al., "A position-aware merkle tree for dynamic cloud data integrity verification," *Soft Computing*, vol. 21, no. 8, pp. 2151–2164, 2017.