Engineering Innovations
ISSN: 2813-1002, Vol. 14, pp 117-125
doi:10.4028/p-A50unQ

Submitted: 2024-08-29
Accepted: 2024-10-09
Online: 2025-04-04

An Improved Data Integrity Verification Model in Cloud Computing

Nomsa L. Khumalo[1,a], Topside E. Mathonsi[1,b], Sunday O. Ojo[2,c]

[1]Department of Information Technology, Tshwane University of Technology, Pretoria, South Africa

[2]Department of Information Technology, Durban University of Technology, Durban, South Africa

[a]lebokhumalo22@gmail.com, [b]MathonsiTE@tut.ac.za, [c]sundayo1@dut.ac.za

Keywords: Cloud computing, Data integrity, Cryptographic accumulator provable data possession−merkle hash tree algorithm, Data verification.

Abstract. Cloud computing eliminates the need for expensive hardware and software expenditures by revolutionising access to computing resources through internet-based utility services. However, Data Integrity (DI) in this paradigm faces a variety of challenging issues, including complexity, security, privacy, control limits, fallibility of human beings, and financial limitations. The shortcomings of current DI solutions in terms of guaranteeing data verification, preventing replay attacks, and controlling computational overhead have led to an increasing need for access to cloud infrastructures by third-party verifiers. The suggested Cryptographic Accumulator Provable Data Possession with Merkle Hash Tree (CAPDP-MHT) scheme demonstrates significantly improved performance over Provable Data Possession (PDP) and Rivest Shamir Adleman (RSA) algorithms in various domains, as demonstrated by thorough simulation and MATLAB-based evaluation. In particular, CAPDP-MHT outperforms PDP and RSA with an average data verification success rate of 25%, compared to their respective rates of 10% and 5%. Moreover, it identifies replay attacks in about 30 seconds, compared to 45 and 70 seconds for PDP and RSA, respectively. Furthermore, the computational overhead of CAPDP-MHT is about 27 seconds, while that of PDP and RSA is 45 and 60 seconds, respectively. Therefore, as compared to PDP and RSA-based systems, CAPDP-MHT not only exhibits exceptional computing efficiency but also outperforms in reliability.

Introduction

A concept known as "cloud computing" makes it possible to distribute computer resources - like storage, processing power, and software - over the Internet [1]. Users don't need local hardware or physical infrastructure to access and use these resources on demand [2]. This technology, which offers scalability, flexibility, and cost-efficiency, has completely changed how people and corporations store, process, and manage data [3]. Data generation is accelerating in the modern era, outpacing the capacity of conventional devices like PCs and mobile phones [4]. Due to their limited resources, these devices find it difficult to manage the increasing volume of data that is stored locally [5]. Because of this, cloud storage has gained popularity as a result of its practically infinite storage space and adaptable pay-as-you-go pricing model [6]. However, outsourcing data to cloud storage raises a number of security issues. Three key components of data security are availability, confidentiality, and data integrity (DI). DI verification is essential to guaranteeing the reliability of data stored on remote servers [7]. Provable Data Possession (PDP) systems, for example, enable a client to confirm the accuracy of its data that is outsourced without having to get the complete dataset [8].

The topic of DI, which includes safeguarding data from unauthorised change or alteration and making sure it remains valid and consistent throughout its lifecycle, is the main subject of this study. When using cloud storage, consumers upload their data with the expectation that it won't be changed or removed. But there are possible hazards involved. To generate profit, cloud storage services may remove user data that is not commonly accessed in order to free up storage space and distribute it to other customers. Furthermore, user data deletions can happen by unintentionally, in which case the server would hide the information to preserve its good name.

According to a survey of the literature, researchers have previously put out a number of DI verification systems in an effort to solve this problem [9]. Scholars have predominantly directed their attention towards the domain of cloud computing. Although current algorithms have demonstrated efficacy in managing DI verification, they frequently encounter obstacles such replay assaults and elevated computational overhead [10]. These restrictions may have a major effect on the network's overall performance and quality of service. This work suggests integrating the Merkle Hash Tree (MHT) and Cryptographic Accumulator Provable Data Possession (CAPDP) techniques to address these issues. The CAPDP-MHT is the name of the planned scheme.

The goals of the CAPDP-MHT scheme's creation were to address the issues with replay attacks, DI verification, and computational overhead in cloud computing. In cloud computing, CAPDP is a useful tool for DI verification. It utilises an adapted cryptographic accumulator based on RSA to allow users to collect data block hash values [11]. Without having to retrieve the complete dataset, this accumulator makes data integrity verification easier. Furthermore, CAPDP provides some defence against replay attacks due to its reliance on an accumulator. The current accumulator state, which dynamically changes as new data is added or modified, makes it difficult for attackers to replay previous data and serves as a partial deterrent to replay attacks [12]. However, MHT is particularly good at handling the computational overhead involved in data integrity checking. Compared to rehashing the full dataset, it offers efficiency and lowers processing needs by just recalculating a tiny section of the tree when data changes [13]. The purpose of integrating these techniques into CAPDP-MHT was to improve DI verification while lowering the chance of replay attacks and minimising computational complexity.

The subsequent sections of this paper are organized as follows: Section II offers an extensive examination of previous studies related to the DI verification schemes. In Section III, we introduce the proposed CAPDP-MHT algorithm. Section IV presents the simulation results obtained by applying the algorithm. Finally, in Section V, we outline future work and conclude the paper with remarks on the findings.

Related Works

Significant attempts have been made in the last few years to improve DI in cloud computing. Researchers have created methods, algorithms, mathematical models, protocols, and other tools to confirm the accuracy of data kept in cloud storage

A simple deterministic DI check procedure based on homomorphic hash functions and RSA cryptographic public-key encryption was presented by Tsaloli et al. [14]. Their method did not support data dynamics, even though it produced adequate findings with an infinite number of verifications. In this work, we offer a modified RSA-based deterministic technique that concurrently supports data dynamics and enables users to check data integrity an infinite number of times.

A probabilistic DI approach for Provable Data Possession (PDP) was presented by Zhou [15]. Through this innovative approach, clients may now independently confirm the accuracy of data stored on untrusted servers without having to retrieve the complete dataset. Instead of reading the entire file, a random sampling of data blocks was used in this approach. It was found through experimentation that disc input/output (I/O) operations had a greater impact on PDP performance than cryptographic calculations. Performance showed decreasing returns as file size rose.

Our study, on the other hand, uses a proved data-possession technique to guarantee the integrity of data that is outsourced. Rather than depending on a subset for verification, our deterministic technique seeks to check the integrity of every data block in a dataset.

An integrity-checking approach based on message authentication codes (MAC) was presented by Yi et al. [16] and does not require a third party's involvement as a requirement. Thanks to the method developed by these researchers, stored data can be validated locally with an exceptionally high degree of accuracy. Users send a verification tag - such as block numbers - when verification is required, and the cloud server verifies it. This technique resists replay and man-in-the-middle attacks and efficiently verifies the integrity of the data. The MAC approach does, however, have the disadvantage of storing redundant data, which raises the overhead costs associated with computation and storage.

In this work, we propose a substitute verification technique that provides protection against replay assaults as well as ensuring the integrity of data saved in the cloud. In addition, we use an adapted RSA-based accumulator to confirm the accuracy of data that is outsourced, which lowers the overhead associated with computation and storage.

A symmetric cryptographic approach was presented by Ferretti et al. [17] in order to validate data DI in cloud databases. These researchers' method finds unapproved changes to outsourced data by using many bloom filters. Although this system can identify replay attacks, it is unable to stop them.

Cryptographic Accumulator Provable Data Possession - Merkle Hash Tree (CAPDP-MHT) Scheme

Cryptographic-Accumulator Provable Data Possession (CAPDP)-Merkle hash tree (MHT) (CAPDP-MHT) scheme was designed by integrating CAPDP and MHT in order to improve data integrity in cloud computing.

An RSA-based cryptographic accumulator is used by CAPDP, a useful tool for cloud computing DI verification, to streamline DI testing. It lessens the requirement to retrieve the complete dataset by enabling users to collect hash values of data blocks. Because the accumulator state dynamically alters in response to updates in the data, CAPDP also provides security against replay attacks. However, MHT excels at lowering computational overhead by improving efficiency by just recalculating a small portion of the tree when data changes. By using these algorithms, CAPDP-MHT improves DI verification while reducing computing needs and replay assaults.

Three important players are shown in Fig. 1's cloud computing architecture: the data owner (DO), the third-party auditor (TPA), and the data storage service (DSS), which is made up of four entities. Cloud service providers (CSPs) provide storage services with great computational capabilities, and the DO has a large amount of data for cloud storage. TPAs ensure security and integrity by managing and supervising the data that DOs outsource. The outsourced data is accessible to authorised applications (AAs). TPA protects data, DO uploads data to the cloud, CSP keeps resources updated, and AAs use a range of cloud-based apps. In cloud computing, this structure makes data storage, security, and accessibility easier.

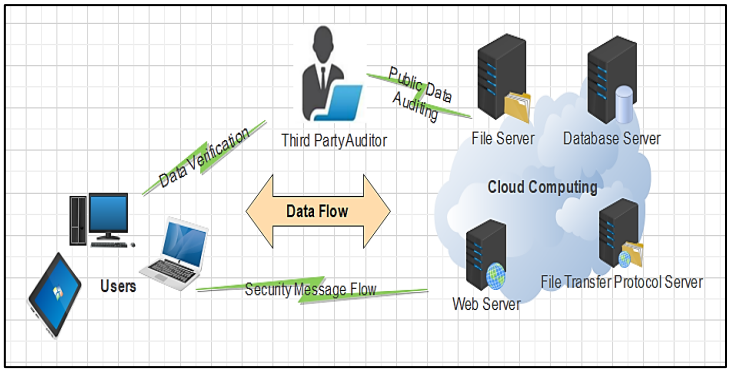

Fig. 1. Cloud computing architecture.

Cryptographic-Accumulator Provable Data Possession. The CAPDP makes DI checks easier. It lessens the requirement to retrieve the complete dataset by enabling users to collect hash values of data blocks [11]. Since the accumulator state dynamically changes in response to updates in the data, CAPDP also provides security against replay attacks.

Verification remains unaffected by the volume of data blocks; and can be carried out on low-power devices. Additionally, the scheme functions as a safeguard against a range of attacks, including tag forgery, data deletion, replacement, and leakage, while also detecting replay attacks.

Equation 1: Generating two symmetrical numbers

Step2: w, z = generate_two_numbers

In this equation, the algorithm generates two symmetrical numbers, w, and z. The specific method or process used to generate these numbers is not mentioned in the algorithm and would depend on the implementation.

Equation 2: Calculating m Step 3: m = w * z

This equation calculates the value of m by multiplying the two symmetrical numbers, w, and z, obtained in Equation 1. The resulting value, m, will be used in subsequent steps of the algorithm.

Equation 3: Calculating y Step 4: y = (g - 1) * (z - 1)

In this equation, the algorithm calculates the value of y. The equation uses the previously defined variable g (which is not explicitly defined *in* the algorithm) and subtracts 1 from this variable. Similarly, the algorithm subtracts 1 from the symmetrical number z. It then multiplies these two values to obtain y. The resulting value of y will be used in Step 5.

Equation 4: Selecting an integer approximately prime to y. Step 5: b = select_approximately_prime_integer(y)

In this equation, the algorithm selects an integer b that is approximately prime to the value of y obtained in Equation 3. The specific method for selecting such an integer is not provided in the algorithm. The selected value of b will be used in subsequent steps.

Equation 5: Calculating the public and private keys Step 7: Public Key = {g, a} Step 8: Private Key = {c, a}

In these equations, the algorithm defines both the public and private keys. The public key consists of the value of g (which is not explicitly defined in the algorithm) and another variable a. The private key consists of the value of c and the same variable a. The specific values assigned to a, g, and c are not specified in the algorithm and would depend on the implementation.

1. Key Generation Algorithm - CAPDP uses two types of keys: public keys and private keys. Some of the steps for key generation are as follows:

Algorithm 1: CAPDP Key Generation

Input: Generate two symmetrical numbers
Output: Public & private keys
Step1: Begin the process:{b,c,j,y,k}
Step2: Generate two numbers w & z
Step3: Calculate m=w*z
Step4: Calculate y=(g-1)*(z-1)
Step5: Select an integer approximately prime to y & call it b
Step6: Calculate g=(i*2)+1
Step7: The public key is {g,a)
Step8: The private key is {c,a}
End procedure

2. Encryption Algorithm – An encryption algorithm is used to turn data into a cipher text(unreadable) format only authorized parties can access. Some of the steps for encryption are as follows:

Algorithm 2: CAPDP Encryption

Input: public key, simple text or plaintext & modulus (absolute values)
Output: cipher text or unreadable text
Step 1: Begin procedure {g, m, a}
Step 2: Generate a public key (g, a)
Step 3: Assign a positive integer to the plaintext message
Step 4: Calculate the cipher text $C = m^{\left(\frac{g-1}{2}\right)} \bmod(a+1)$
Step 5: Send the cipher text or encrypted data
End process

3. Decryption Algorithm - Decryption is the process of converting cipher text or encrypted data back to its original form or readable data. Some of the steps for decryption are as follows:

Algorithm 3: CAPDP Decryption

Input: Public key, cipher text & modulus (absolute values)

Output: Plaintext or readable data

Step 1: Initiate the process.

Step 2: The user initiates a request to the service provider, asking for user authentication and the provision of encrypted data, denoted as ‘d’.

Step3: The user performs decryption through calculation, resulting in the determination of x= $C^d \bmod(a+1)$

Step 4: Return the plaintext from an integer representation of x.

End process

Merkle Hash Tree. MHT is a useful tool for data synchronisation and verification, allowing for quick and easy assurance that a group of pieces is still intact [18]. Implementing it reduces server processing time significantly. By merely recalculating a small portion of the tree when data changes, MHT excels at lowering computational overhead and improving performance.

Based on the provided description of the Merkle hash tree (MHT), here are five equations that represent the steps involved:

Equation 6: Leaf node hashing: H1 = Hash(file_block_1) H2 = Hash(file_block_2)

Equation 7: Combining and hashing intermediate nodes: H3 = Hash(H1 + H2)

Equation 8: Hashing and combining additional file blocks: H4 = Hash(file_block_3) H5 = Hash(file_block_4) H6 = Hash(H4 + H5)

Equation 9: Rehashing and obtaining the root node: H7 = Hash(H3 + H6)

Equation 10: Final expression of the root: Root = H7

These equations illustrate the process of generating an MHT for data verification and synchronization. The specific hash function used may vary based on the chosen cryptographic method, and the example provided is a simplified representation of the steps involved.

CAPDP-MHT Flowchart. The CAPDP-MHT flowchart depicted in Fig. 2 can be employed collaboratively to transmit a text message from Workplace 1 to Workplace 2.

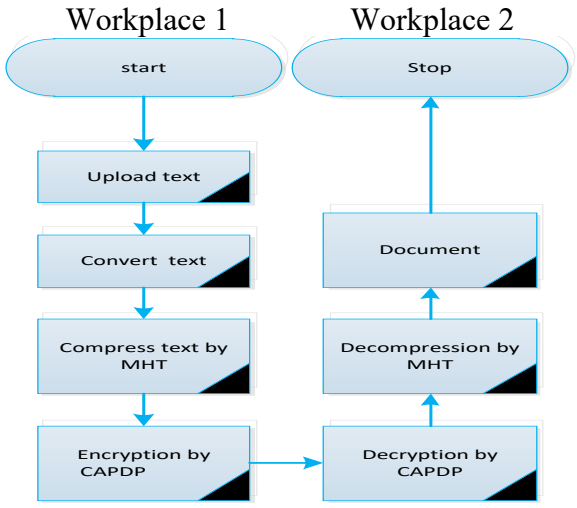

Fig. 2. CAPDP-MHT flowchart.

Simulation Results

The simulation studies carried out to assess how well the CAPDP method ensures DI and security are presented in this section.

1. Data-verification experiment:

Dataset sizes: 1GB, 5GB, and 10GB

Verification process: Perform DI verification using the CAPDP-MHT scheme on each dataset.

Performance metrics:

Verification time: Measure the average time taken for the CAPDP-MHT scheme to verify the integrity of the datasets.

Resource utilization: Evaluate CPU usage and memory consumption during the verification process.

2. Replay attack experiment:

Types of replay attacks: Timestamp replay attacks and data-modification replay attacks.
Varying attack intensity: Introduce replay attacks at different frequencies, e.g., 10% and 20% of the dataset.
Performance metrics:
Detection rate: Calculate the percentage of successfully detected replay attacks of the total number of attack instances.
False-positive rate: Measure the rate of false-positive detections, indicating legitimate data being wrongly identified as replay attacks.

3. Computational overhead experiment:

Varying dataset complexity: Introduce different levels of data modification or corruption within the dataset.
Varying system configurations: Adjust the number of parallel processing threads (2 and 4 threads) and available memory (8GB and 16GB).
Performance metrics:
Processing time: Measure the time taken for the CAPDP-MHT scheme to perform DI verification for each dataset complexity and system configuration.
Memory usage: Evaluate the memory consumption during the verification process.
CPU utilization: Measure the percentage of CPU usage during the verification process.

Table 1. Simulation Parameters.

Experiment	Parameter	Value
Data Verification Experiment	Dataset sizes	Dataset 1: 1GB Dataset 2: 5GB Dataset 3: 10GB
	Verification time	Dataset 1: 25 seconds Dataset 2: 120 seconds Dataset 3: 250 seconds
	Resource utilization	CPU usage: 80% Memory consumption: 6GB
Replay Attack Experiment	Types of replay attack	Timestamp replay attacks, Data-modification replay attacks.
	Varying attack intensity	Replay attack frequency 1: 10% of the dataset Replay attack frequency 2: 20% of the dataset
	Detection rate	Replay attack frequency 1: 95% Replay attack frequency 2: 85%
	False-positive rate	Replay attack frequency 1: 2% Replay attack frequency 2: 5%
Computational Overheads Experiment	Varying dataset complexity	Dataset complexity 1: 10 seconds Dataset complexity 2: 45 seconds Dataset complexity 3: 80 seconds
	Varying system configurations	Parallel processing threads: 2 Threads Parallel processing threads: 4 Threads Available memory: 8GB Available memory: 16GB
	Processing time	Dataset complexity 1: 10 seconds Dataset complexity 2: 45 seconds Dataset complexity 3: 80 seconds
	Memory usage	Dataset complexity 1: 4gb Dataset complexity 2: 8gb Dataset complexity 3: 12gb
	CPU utilization	Dataset complexity 1: 60% Dataset complexity 2: 80% Dataset complexity 3: 90%

Table 1 provides an overview of the parameters and their corresponding values for each simulation experiment.

Data Verification. The average data-verification times for three different algorithms - with a particular emphasis on the number of challenged blocks ranging from 100 to 400 - are shown in Figure 3. Two-point multiplication operations have been added into this data-verification method. Nonetheless, the time variance is negligible - less than 1 ms - when compared to the PDP and RSA systems. This makes it essentially meaningless in situations involving actual data verification. As a result, the suggested strategy both improves security and keeps processing costs within an acceptable range.

The suggested CAPDP-MHT, the PDP model, and the RSA model show rates of 25%, 10%, and 5%, respectively, with regard to data-verification averages.

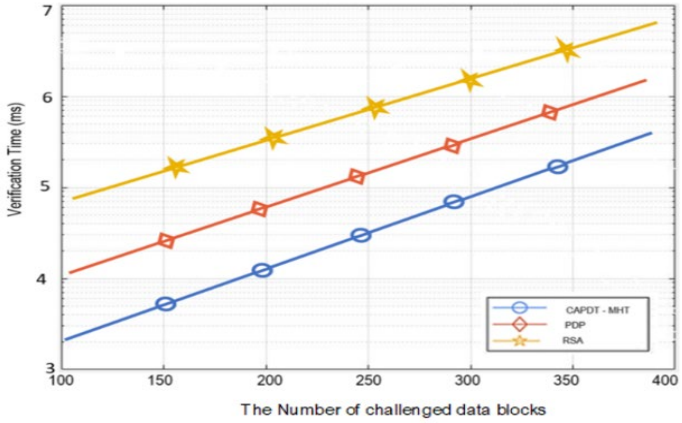

Fig. 3. Data verification.

Replay Attack. The examination of the three systems was part of the assessment, which also examined the average replay-attack incidence that occurs during cloud computing data access. We looked at the time required for attack detection in the PDP, RSA, and the proposed CAPDP-MHT scheme, as shown in Fig. 4. It is important to note that replay-attack detection takes longer with all systems, including the suggested and current approaches (PDP and RSA). The suggested CAPDP-MHT method, on the other hand, detected replay assaults in 30 seconds, whereas the PDP and RSA schemes required 45 and 70 seconds, respectively.

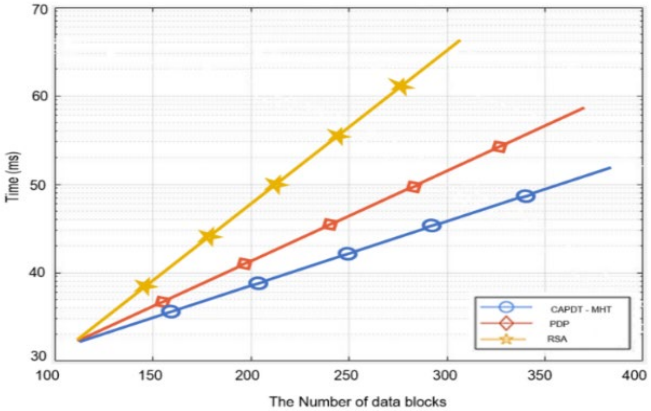

Fig. 4. Replay attack.

Computational Overhead. This study compares the average computational overhead of the PDP and RSA algorithms with the suggested CAPDP-MHT scheme, which is based on 10 simulations. Upon closer inspection, Fig. 5 clearly illustrates how much the suggested CAPDP-MHT method reduces the average computational cost in comparison to the other techniques. This decrease has been achieved by quickly and effectively certifying the integrity and unaltered state of a set of items using the MHT algorithm. Consequently, the MHT algorithm significantly reduces the amount of time needed for the data-verification procedure. Specifically, the average computational overhead for the CAPDP-MHT method was about 27 seconds, compared to 45 and 60 seconds for the PDP and RSA systems, respectively.

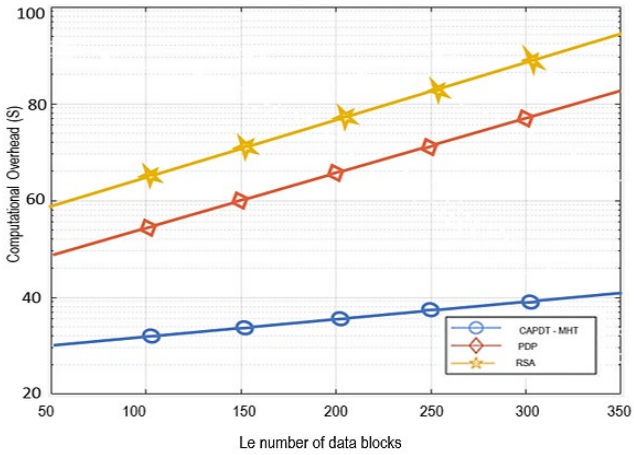

Fig. 5. Computational overhead.

Conclusion and Future Work

The goal of the CAPDP-MHT technique, which was presented in this paper, is to reduce computational overhead, lower the risk of replay assaults, and verify DI. We combine the MHT and CAPDP schemes to create the CAPDP-MHT scheme. The CAPDP-MHT is the name of the planned scheme. In order to ensure replay attack and computational overhead in cloud computing, we incorporated the CAPDP scheme and MHT scheme to secure DI verification. The reason CAPDP was chosen was its resistance to replay assaults. The MHT scheme was selected to serve as a useful tool for data verification and synchronisation, enabling quick and easy confirmation of the integrity and unchanged status of a group of elements. This is accomplished by continuously changing the accumulator state in response to data modifications, making it less vulnerable to such security breaches. The adoption of it considerably cuts down on server processing time. One notable feature of MHT is its ability to reduce computing overhead by just recalculating a portion of the tree when data changes, which improves overall performance. We verified the efficacy of the suggested CAPDP-MHT with comprehensive MATLAB computer simulations. The outcomes showed that it could successfully prevent replay attacks and minimise computing complexity while securing DO verification.

Future work employing machine learning to assess data accuracy in data integration should also examine a number of other crucial aspects. These variables include things like fault tolerance, resource efficiency, and scalability.

Acknowledgment

The Tshwane University of Technology is acknowledged by the authors for its financial help in carrying out this research. They also attest that there are no conflicts of interest associated with this paper's publishing

References

[1] Surbiryala, J. and C. Rong. Cloud computing: History and overview. in 2019 IEEE Cloud Summit. 2019. IEEE.

[2] Sehgal, N.K., P.C.P. Bhatt, and J.M. Acken, Cloud computing with security. Concepts and practices. Second edition. Switzerland: Springer, 2020.

[3] Huy, V.Q., The Value of Applying Cloud Computing in Education and Development Solutions. Onomázein, 2023(61 (2023): September): p. 178-185.

[4] Li, S., et al., Research on the application of information technology of Big Data in Chinese digital library. Library Management, 2019. 40(8/9): p. 518-531.

[5] Imteaj, A., et al., A survey on federated learning for resource-constrained IoT devices. IEEE Internet of Things Journal, 2021. 9(1): p. 1-24.

[6] Ibrahim, H., Using cloud computing services in the knowledge sharing process in Iraqi universities. 2022, Altınbaş Üniversitesi/Lisansüstü Eğitim Enstitüsü.

[7] Huang, P., et al., A collaborative auditing blockchain for trustworthy data integrity in cloud storage system. IEEE Access, 2020. 8: p. 94780-94794.

[8] Ibrahim, S.H., M.M. Sirat, and W.M. Elbakri. Data Integrity for Dynamic Big Data in Cloud Storage: A Comprehensive Review and Critical Issues. in International Conference for Emerging Technologies in Computing. 2022. Springer.

[9] Wei, P., et al., Blockchain data-based cloud data integrity protection mechanism. Future Generation Computer Systems, 2020. 102: p. 902-911.

[10] Tian, J., H. Wang, and M. Wang, Data integrity auditing for secure cloud storage using user behavior prediction. Computers & Security, 2021. 105: p. 102245.

[11] Khedr, W.I., H.M. Khater, and E.R. Mohamed, Cryptographic accumulator-based scheme for critical data integrity verification in cloud storage. IEEE Access, 2019. 7: p. 65635-65651.

[12] Grover, J., Security of Vehicular Ad Hoc Networks using blockchain: A comprehensive review. Vehicular Communications, 2022. 34: p. 100458.

[13] Ajayi, O.J., Blockchain-Based Architecture for Secured Cyberattack Signatures and Features Distribution. 2021, The City College of New York.

[14] Tsaloli, G., G. Banegas, and A. Mitrokotsa, Practical and provably secure distributed aggregation: Verifiable additive homomorphic secret sharing. Cryptography, 2020. 4(3): p. 25.

[15] Zhou, C., A certificate-based provable data possession scheme in the standard model. Security and Communication Networks, 2021. 2021: p. 1-12.

[16] Contiu, S., Applied Cryptographic Access Control for Untrusted Cloud Storage. 2019, Université de Bordeaux.

[17] Ferretti, L., et al., A symmetric cryptographic scheme for data integrity verification in cloud databases. Information Sciences, 2018. 422: p. 497-515.

[18] Peng, S., et al., Efficient, dynamic and identity-based remote data integrity checking for multiple replicas. Journal of network and computer applications, 2019. 134: p. 72-88.