A New Data Integrity Verification Scheme with Low Communication Cost

Fanglin An[1], *Jun Ye*[1,2*], *Zhen Guo*[1,2]

[1] *School of Cyberspace Security, Hainan University, China*
[2] *Key Laboratory of Internet Information Retrieval of Hainan Province, China*
22110839000008@hainanu.edu.cn, yejun@hainanu.edu.cn, guozhen@hainanu.edu.cn

Abstract

The development of cloud storage is indispensable for today's Internet information technology. The superiority of cloud storage is that the ownership of the storage medium and the ownership of the stored data are divided and held by two parties with respective interests - cloud server providers and data users. However, the transmission, storage and utilization of data constantly caused the violation of users' rights and interests. Data integrity as a countermeasure has been paid growing attention to by researchers. Traditional data integrity protection schemes are based on the cloud server-client architecture, which requires high computational overhead and communication. To address this scenario, this paper proposes an improved data integrity verification scheme that enables local users to perform rapid and efficient data integrity verification of data stored on cloud servers, reduces the volume of communication data during the verification process, and effectively prevents the loss or tampering of data stored in the cloud server.

Keywords: Tamper-Proof, Integrity verification, Communication overhead optimization

1 Introduction

With the advance of Internet information technology penetrating into all aspects of people's lives, users are more closely connected to each other, resulting in a proliferation of user data. However, the mobile storage technology, in recent years, has not made great progress in the upper ratio. Therefore, the user's local terminal device has not been able to meet the basic storage requirement. At the same time, due to the data volatility of the storage units in current mobile storage devices (solid-state drives themselves have design flaws that lead to data loss over long periods of power on and off), the demand for <u>important/significant/crucial/privacy data</u> backup is also becoming increasingly prominent. In order to face the above-mentioned problems, cloud storage products came into being, at the same time, how to ensure the data security and data integrity of users in the cloud storage environment has become a new urgent problem. In the traditional data integrity verification scheme, users need to save the hash value of each file data for hashing operation, then develop a hash table containing the hash values of all stored file data, and store the hash table in the local grant environment. When the cloud file needs to be updated, the hash value, corresponding to this file, in the hash table that is stored in the credit environment needs to be updated synchronously. When the integrity of the file data in the cloud storage requires verification, the file data in the cloud storage entail transmission to the user's local terminal, and applying the hashing operation on it, then matching it with the original hash value in the hash table and determining whether the data integrity is damaged based on the matching result.

The traditional method of data integrity verification calculates the hash value of each file and saves the hash value through the local authorization mechanism, which has the following shortcomings: each verification of file data needs to transfer the file data stored in the cloud to the local terminal in order to match, which occupies a large amount of local data bandwidth; with the gradual increase of file data, the number of hash values to be stored in the local authorization mechanism increases proportionally, and the complexity of managing hash data increases unequally. Because the above two key problems are not solved perfectly, the traditional data integrity verification is inefficient, resulting in the data integrity technology not being widely used in practice.

There is an urgent need to study an efficient and easy-to-manage data integrity scheme for cloud storage data that can efficiently complete data integrity verification requests and also ensure the correctness and non-tamperability of data verification results. At the same time, it can ensure that the proliferation of file data does not affect the efficiency of data integrity verification by users. These two aspects are the focus of data integrity verification and are of great importance to the management of user data.

With the advent of the big data era, a large amount of relevant business data generated in various fields, which are usually put in cloud servers for easy data management. However, these data may be tampered with by dishonest cloud servers, thus verifying data integrity is a necessity for secure data storage. In order to perform data integrity verification more efficiently, this paper proposes a trusted and communication data reduced integrity verification scheme in the cloud, and a special binomial tree data structure based on Merkle tree is proposed, which uses a top-down approach to compute arbitrary data nodes. In addition, for integrity verification, a new integrity checking approach is introduced by unifying the processing of database records and Merkle Tree nodes. The proposed scheme has less communication

*Corresponding Author: Jun Ye; E-mail: yejun@hainanu.edu.cn
DOI: 10.53106/160792642023032402007

overhead than existing ones.

In this paper, we propose a trusted data integrity verification scheme in the cloud that reduces the amount of lower communication. A special Merkle tree data structure is introduced to achieve the integrity verification of data in the cloud. And it is also very efficient to update the Merkle tree when some data changes frequently. The main contributions are as follows.

(1) An improved cloud data integrity verification scheme is proposed that can reduce the communication overhead between cloud servers and users.

(2) By using a special Merkle tree-based encryption technique, the user can verify whether a certain data is tampered or not with less communication overhead and computation overhead.

(3) In the data upload or update phase, the user only needs to send the data and the value of its parent node to the cloud server, and the update work is done by the cloud server.

2 Related Work

Nicolescu et al. [1] proposed that the value held by IoT services and products throughout their lifecycle in creating and maintaining the value of IoT services, while the work in this paper is in maintaining the value of IoT services. Deb et al. [2] proposed a new image cryptosystem based on chaos theory when using a statistical approach to test the data and the efficiency of the system, which shows the need to protect the operational efficiency of the system while safeguarding the integrity of the data. Ji et al. [3] on the optimization and allocation of communication resources for high throughput in UAV-ground communication scenarios, giving us a research focus on the need for a system that should be as efficient as possible for communication, is the latest research trend.

For the work on data integrity verification, Zhou et al. [4] made a summary of the need to accomplish functionality, security and efficiency. Changalvala and Malik [5] developed a data hiding technique based on 3-dimensional quantized index modulation for data integrity verification in the transmission protocol used for data transmission in autonomous driving scenarios, but the scheme did not consider data security. K. Hao et al. [6] also highlight the problem of untrustworthy storage environment caused by outsourcing data to cloud servers for storage. Ding et al. [7] propose a data integrity verification scheme with smaller computational overhead and latency for fog node servers whose computational overhead is not easily but large, but the scheme does not consider the weak computational power of the client. Xu et al. [8] address the data integrity. The unique tag replacement algorithm was proposed for the problem of reliability of tags generated by the proof, but the scheme did not consider the excessive computational overhead required for the verification process. Cui et al. [9] also proposed a solution for the data integrity verification problem in edge computing scenario, but the scheme used a large number of modulo multiplication operations which slowed down the data processing time. Khadse et al. [10] data hiding technique uses the data structure of Merkle tree for data integrity verification, but the scheme is only suitable for scenarios of large size like images because of the communication overhead of Merkle tree, which can be seen in the later scheme. Ji et al. [11] use the idea of continuous modulo multiplication for data label aggregation, which can better reduce the communication overhead, but the data computation of the scheme increases a lot. Arasu et al. [12] proposed to fix the key value of each node of the Merkle tree for the purpose of localization, which can reduce the computational and communication overheads to some extent, and provides an idea for the scheme in this paper.

Zhu et al. [13] proposed a data integrity verification scheme based on short signature algorithm with a rigorous proof of formula derivation, but the scheme used a large number of bilinear mappings, modulo power operations and modulo multiplication operations, but could not be applied to client devices with less computational power. Xu et al. [14] proposed a data integrity verification model with privacy protection based on healthcare system, and the Shen et al. [15] proposed an innovative data integrity verification scheme based on transportation systems, but the scheme relies too much on data transmission, and the scheme has too high overhead for data communication, which is not conducive to mobile in-vehicle data transmission.

In addition to the privacy protection work, there is the combination of authentication and data integrity verification work, Sahu et al. [16] added trusted third parties to challenge the cloud server, using Bloom Filter, lattice based signature construction and matrix computation to improve the search efficiency, in fact, the computation task is moved to trusted third parties to complete, the construction of the scheme has some innovative significance The construction of this scheme is innovative, but it cannot be applied to scenarios with relatively low data overhead requirements in the IoT environment. Considering the collusion attack problem brought by using trusted third parties to reduce the computational burden, Bai et al. [17] proposed to use blockchain for data integrity verification, which can well prevent the collusion attack problem caused by collusion between third parties and cloud servers, but putting IoT devices on the chain is not a good solution. In response to blockchain references, Cherupally et al. [18] turned to using distributed ledger technology to implement a data integrity verification scheme that optimizes the P2P network to reduce the computational overhead, but this undoubtedly increases the communication overhead between devices.

Changalvala [19] proposed low complexity data hiding techniques that are easy to implement and do not require a lot of processing power or memory, and used them to protect sensor data transmission in resource-constrained and real-time environments (autonomous driving), which is an alternative idea that can guarantee data integrity verification. Ren et al. [20] proposed a data integrity verification scheme at the data block level and proposed to be able to resist attacks such as label forgery, data alteration, etc. and suggested that a good scheme needs to support dynamic manipulation of data. In addition to considering data integrity, data trust in wireless sensor networks also needs to be considered. Daniel et al. [21] proposed a trust-based data aggregation protocol that can successfully check the correctness of data, but the scheme does not take into account that this consumes too much communication overhead for the devices. Yu et al. [22] addressed the problem of frequent checking of data integrity that wastes resources and proposed a scheme that uses periodic checking and arbitration algorithm for data integrity verification strategy, which is more reasonable and fair compared to similar schemes. In order to guarantee the supervision of pharmaceutical production line data, Leal et al. [23] proposed a data supervision scheme based on blockchain and smart contracts and using industrial sensors for uploading data, but the scheme does not pay much attention to the privacy protection of data and is only applicable in this scenario. In order to make the scheme applicable to more scenarios, Abhishek et al. [24] proposed a low latency data integrity verification scheme based on vehicular networking, which is capable of low data latency, meaning that it does not require excessive communication overhead, which provides ideas for our scheme.

2.1 Preliminaries

2.1.1 Hash Functions

Hash function is an irreversible function, common Hash algorithms are MD5 and SHA series, currently MD5 and SHA1 have been cracked, this paper recommends the use of SHA-256 algorithm.

SHA-256 operates in a similar process to a normal hash function, where the following two steps are first performed before the hash is computed:

a) Complementary processing of the message, with the final length being a multiple of 512 bits.

b) Chunking the message in 512-bit units to M_1, M_2, ..., M_n.

c) Process the message chunks: starting with an initial hash H_0, iteratively compute: $H_i = H_{i-1} \oplus CM_i\left(H_{i-1}\right)$

where C is the SHA256 compression function, $\oplus$ is the mode 2^{32} addition, H_n is the hash value of the message block.

2.1.2 Demand Analysis

A practical data integrity verification scheme needs to be designed to meet the following security objectives.

(1) The integrity attestation cannot be falsified by the cloud server.

(2) The data verification can manipulate over single data or multiple data concurrently.

(3) Availability: Resistant to collusive attacks by attackers and cloud servers.

(4) Correctness and accuracy: Any errors in the data can be identified by the user.

3 New Data Integrity Verification Scheme

3.1 System Model

In our hypothesis, the two parties involved are: the user and a semi-trusted cloud server. The user has absolute ownership of the data, uploading, updating, reading and verifying the integrity of the data. The data is required to be stored in a specified location in the cloud. The specific steps are as follows.

a) Upload data: The user sends a command to send the data to the cloud location, and at the same time, the value of the corresponding parent node in the Merkle tree is calculated and uploaded together with the data, which is done when the cloud data is first opened.

b) Update data: The user sends a data update command to replace the data at the cloud location, and at the same time recalculates the value of the corresponding parent node in the Merkle tree and uploads it together with the data.

c) Get (download) data: The user sends a data download command, and the cloud server returns the data at the cloud location to the user along with the value of the corresponding parent node in the Merkle tree.

d) Data integrity verification: After obtaining the values returned from the cloud and their corresponding parent nodes in the Merkle tree, the user verifies them according to the rules of the Merkle tree designed by us.

If the data integrity verification passes, the user accepts the data as true and complete, otherwise the user considers the cloud data unreliable. The system model is shown in Figure 1.

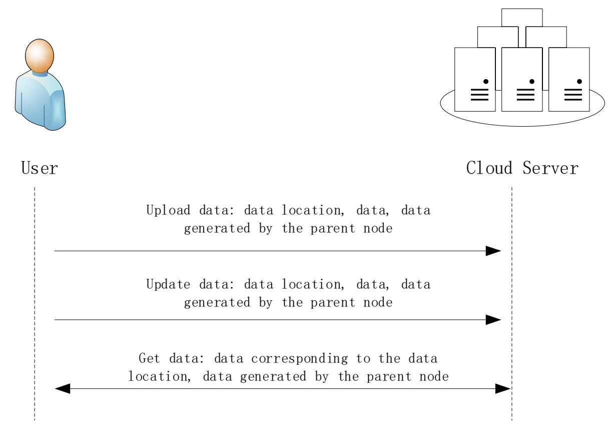
User
Cloud Server
Upload data: data location, data, data generated by the parent node
Update data: data location, data, data generated by the parent node
Get data: data corresponding to the data location, data generated by the parent node

Figure 1. System model

3.2 Security Goal

In addition to safeguarding the data integrity, the data integrity verification scheme needs to consider the complexity of the cloud environment and the different attack directions of the attackers, and needs to satisfy the following assumptions proposed by the security model.

a) The scheme should be able to resist forgery attacks, where an attacker forges false data and false integrity verification parameters that the user can identify during the verification process.

b) The scheme should be able to guarantee the privacy of user keys and user data, where an attacker cannot decrypt the user's private information on the parameters exposed on the public channel and data in the cloud server.

c) The user can have the ability to have unique proof of ownership of the data stored in the cloud, i.e., an attacker cannot make changes to the ownership of the data.

The attacker is a malicious user who is able to listen to the content of the user's conversation with the cloud server on a public channel, and a capable attacker could potentially conspire with the cloud server to modify the actual user's data.

3.3 Our Scheme

A data integrity verification scheme with a faster verification rate is proposed here:

First, first introduce a binary tree data structure: as shown in Figure 2, define the key value of the root node of the binary tree as ϕ , the left child node of each node on the binary tree (including the root node ϕ) is the key value of its parent node followed by 0, the right child node is the key value of its parent node followed by 1, up to a certain layer t. Here is an example of $t = 257$, the number of leaf nodes at the lowest level covered by the binary tree is $2^{257/-1} \approx 1.1579208923732 \times 10^{77}$, here the bottom level of the binary tree is used The leaf nodes represent the key code of the data, and the other nodes are used as the key code of the Merkle tree. By analogy with the difficulty of cracking the hash function above, this value has long exceeded the number of atoms in the known universe, so for the cloud server, this value is far enough to mark (number) all the stored data.

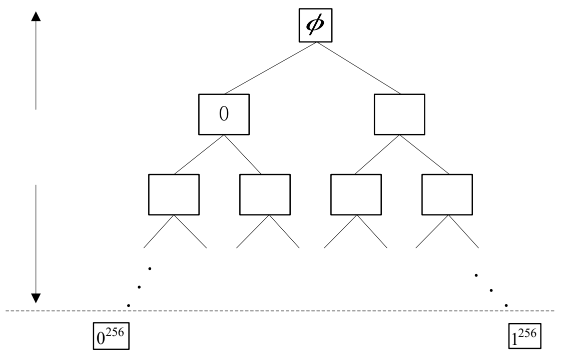

Figure 2. Relationship between Merkle tree and data key values

In the traditional bottom-up Merkle tree-based data integrity verification, the data communication overhead involved is large, and the cloud needs to transmit the verified data and each layer of ancestor nodes involved in the data to the user for hash verification, which is obviously too heavy for a larger database because the amount of communication required for one data integrity verification is $2(t - 2) + 1 = 2t - 3$, for example , as assumed here above of $t = 257$, in order to verify the integrity of 1 data, the cloud needs to transmit data up to 511 node data in Merkle tree, this result is too heavy for the load of the network. Therefore, this paper proposes a top-down data integrity verification scheme to reduce the communication overhead of data in the integrity verification process.

A Merkle tree-based data structure is defined as shown in Figure 3. A new data integrity verification scheme using this data structure is described below.

Setup: The cloud server selects a non-collision hash function $h(\cdot)$ and publishes it to the user as a global parameter.

Special Merkle tree construction: the cloud server generates two random numbers r_1, r_2, which are used to construct the values of the corresponding second-level nodes in the data structure of Figure 3: $r_1 \rightarrow h_0$, $r_2 \rightarrow h_1$, and sends the values of h_0 and h_1 to the user.

The user generates the private key mk locally to prevent the data from being falsified by the cloud data and generates the root node in Figure 3: $root_\phi = h(mk, h_0, h_1)$, the user only saves the value of the root node $root_\phi$.

The user calculates locally the values of all nodes of the binary tree:

$$\begin{cases} h_k^{(i)} = h(mk, h_{\hat{k}}^{(i-1)}, k), 3 \le i \le t-1 \\ h_k^{(i)} = h(mk, h_{\hat{k}}^{(i-1)}, k, m_k), i = t. \end{cases} \quad (1)$$

Here k denotes the key value in the Merkle tree: $k \in \{\{0, 1\}^2, \{0, 1\}^3, ..., \{0, 1\}^{t-1}\}$; $\hat{k}$ denotes the key value of the parent node of k, and $h_k^{(i)}$ denotes the node value of the Merkle tree with the key value k at the i-th level.

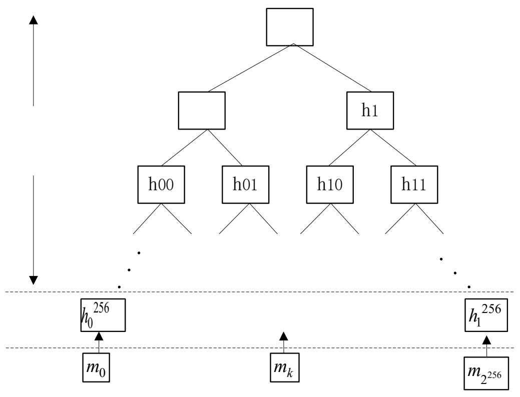

Figure 3. Special data structure based on Merkle tree

Validation: When the user requests integrity verification of the data m_k downloaded for access, the cloud server sends the values of two data h_0 and h_1 to the user, who first verifies that the values of h_0 and h_1 have not been tampered with using their root node.

$$\begin{cases} h_k^{(i)} = h(mk, h_{\hat{k}}^{(i-1)}, k), 3 \le i \le t-1 \\ h_k^{(i)} = h(mk, h_{\hat{k}}^{(i-1)}, k, m_k), i = t. \end{cases} \quad (2)$$

Note that here, unlike the previous construction step, instead of constructing the entire Merkle tree node, a top-down path is constructed from the root node ϕ to the bottom node $k \in \{0, 1\}^t$ of the Merkle, and no other nodes are computed. Once the above equation holds, it is proved that the data m_k is complete and has not been tampered with. Otherwise, the verification fails and m_k has been tampered.

4 Scheme Analysis

4.1 Correctness Analysis

The scheme guarantees that the data completes the data integrity proof as in Equation (1), and with the addition of a special Merkle tree, the user is able to reverse the construction of partial proof data in the same way, i.e., to reason from the root node to any of the leaf nodes, thus being able to accurately verify the authenticity of the corresponding data.

In addition, the solution is able to verify the authenticity of multiple data at the same time, achieving the goals required for integrity verification: bulk verification, correctness and accuracy.

4.2 Security Analysis

The proposed scheme in this paper can meet certain security requirements, firstly, for cloud servers, which cannot get any details of the Merkle tree from the data uploaded by users.

Theorem 1: The cloud server cannot forge a proof of data integrity to deceive the user.

Proof: In the scheme of this paper, the user uses a hashing algorithm for data processing and the Merkle tree is generated to hide the user data information. Even during the integrity verification process, the user information is hidden in the hashed data block. Therefore, the user key and hash function cover up the proof data construction details and the cloud server cannot know about it.

While meeting the above requirements, the solution also has the availability to resist various means of attack by cloud servers or attackers.

Theorem 2: An attacker cannot fake a new Merkle tree to deceive the user.

Proof: Here we first prove the probability that the parent node of the second level is faked at time $P_{\{0,1\}^{255}}$.

$$P_{\{0,1\}^{255}} = P(Adv(\{0,1\}^{255}))=P(Adv(mk))=\frac{1}{p}. \quad \mathbf{(3)}$$

And the probability of forging the whole Merkle tree also depends on whether the attacker can obtain the user key *mk*, so the probability of forging the whole Merkle tree event P_{merkle} is:

$$P_{merkle} = P(Adv(merkle)) = P(Adv(mk)) = \frac{1}{p}. \quad \mathbf{(4)}$$

And the original Merkle tree is constructed as:

$$\begin{cases} h_k^{(i)} = h(mk, h_{\hat{k}}^{(i-1)}, k), 3 \le i \le t-1 \\ h_k^{(i)} = h(mk, h_{\hat{k}}^{(i-1)}, k, m_k), i = t. \end{cases} \quad \mathbf{(5)}$$

By the nature of the non-colliding hash function, it is very difficult to find a number x that satisfies $h(x) = y$, given a value y in advance. Here, given the value $h_k^{(i)}$ of any tree node, it is also impossible to solve for the reverse mk.

From this, it can be obtained that the attacker cannot forge a new Merkle tree to pass the user's verification and the theorem holds.

Theorem 3: The attacker cannot collude with the cloud server to achieve collusion attack.

Proof: Assuming that in the process of joint deception between the attacker and the cloud server, the user cannot be informed of the event in advance, then the attacker and the cloud server may create false data, but this possibility is proven by Theorem 1 to be unattainable because the scheme design itself has made the data owner's proof of data integrity construction to include a random number set by itself as the key, unless the attacker and the cloud Unless the attacker and the cloud server have access to the user key, the attacker and the cloud server cannot achieve collusion attack. This proves that the scheme in this paper can resist collusion attacks.

It can be seen that the scheme in this paper can better meet the requirements for verification of data integrity, and its availability cannot be compromised by attackers during the operational phase of the scheme.

5 Scheme Comparison

In this section, we compare the scheme of this paper with similar schemes [14] and [17] in terms of both computational and communication overheads. As shown in Table 1, the computational overhead and communication overhead of each scheme are presented. Where MI denotes modulo-inverse operation, ME denotes modulo-power operation, MM denotes modulo-multiplication operation, H denotes hash operation, and B denotes bilinear operation. The computational overhead here is the computational overhead required to verify n data, and it calculates only the consumption of additional verification data except for the data itself.

From the table, we can roughly see that our scheme has lower computation overhead and communication overhead compared to similar schemes [14] and [17], which exhibits the advantage of our scheme in addition to this paper, which will be more accurately verified in the experimental section.

Table 1. Scheme comparison

	Calculation overhead	Communication overhead
Scheme [14]	2n*MI; 10n+4*MM; n*H; 3*B	2n
Scheme [17]	4n+3*ME; 5n+1*MM; n*H; 2*B	3n
Our scheme	$2^n - 1$*H	n

6 Experimental Simulation

To compare various programs, data simulation experiments are simulated. The experimental environment runs on a laptop computer model Lenovo Erazer Z51-70, processor model: Intel(R) Core(TM) i5-5200U CPU @2.20GHz; running memory: 8.00 G RAM; storage space: 480G SSD; operating system: 64-bit Windows 10 Professional; programming language is JAVA, the toolkit is JPBC library, and the hash function used is SHA-256.

Figure 4 is the overhead experimental results of the three data integrity verification schemes, from which it can be seen that the scheme proposed in this paper has disadvantages in terms of computational overhead to a certain extent, but the difference is not very large, keeping the same magnitude of computational overhead.

The reason for this experimental phenomenon in Figure 4 is explained by the fact that our scheme only uses hashing operations, resulting in almost the same computational overhead as similar schemes in the finite domain, and even performing better in all cases.

The communication overhead of the three data integrity verification schemes is simulated and the experimental result is plotted in Figure 5, which illustrates that the scheme of this paper can meet practical requirements better than similar schemes, and the response delay is improved to a great extent. The reason for jumping points in the graph is that there are anomalies in the communication time overhead due to the instability of the communication environment, but this has no impact on the judgment of the overall trend.

The reason for the experimental phenomenon shown in Figure 5 is due to the fact that our scheme only requires the cloud server to return the nodes at both ends of the binomial tree to the user during the validation phase, without transmitting this Merkle tree to the user.

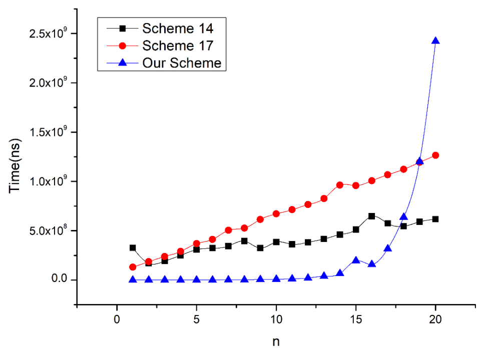

Figure 4. Experimental results of computational overhead simulation

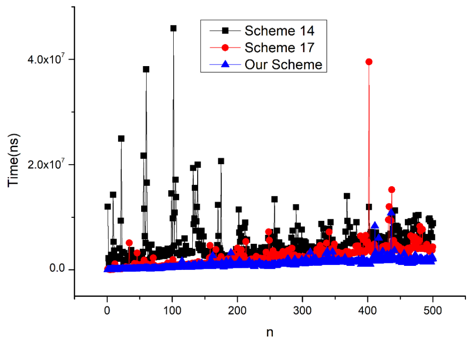

Figure 5. Communication overhead simulation results

7 Conclusion

In this paper, we propose an improved data integrity verification scheme to solve the problem of tampering with user data stored in the cloud server. The Merkle tree required for data integrity verification is constructed step by step starting from the root node. Within the same computational overhead, the communication overhead required for verification is greatly reduced because the cloud server and the user only need to transmit the values of each leaf node in the last layer during the verification process. The scheme security analysis shows that the proposed scheme can meet the requirements of the security model for resisting forgery attacks and resisting collusion attacks.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (62162020), the Science Project of Hainan University (KYQD (ZR) 20021).

References

[1] R. Nicolescu, M. Huth, P. Radanliev, D. D. Roure, Mapping the values of IoT, *Journal of Information Technology*, Vol. 33, No. 4, pp. 345-360, December, 2018.

[2] S. Deb, B. Bhuyan, N. Kar, K. S. Reddy, Colour image encryption using an improved version of stream cipher and chaos, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 41, No. 2, pp. 118-133, July, 2022.

[3] A. Ji, X. Guo, R. Zhang, J. Wu, X. Cheng, Joint HAP deployment and resource allocation for HAP-UAV-terrestrial integrated networks, *IET Communications*, pp. 1-12, October, 2022. https://doi.org/10.1049/cmu2.12512

[4] L. Zhou, A. Fu, S. Yu, M. Su, B. Kuang, Data integrity verification of the outsourced big data in the cloud environment: A survey, *Journal of Network and Computer Applications*, Vol. 122, pp. 1-15, November, 2018.

[5] R. Changalvala, H. Malik, LiDAR data integrity

verification for autonomous vehicle, *IEEE Access*, Vol. 7, pp. 138018-138031, September, 2019.

[6] K. Hao, J. Xin, Z. Wang, G. Wang, Outsourced data integrity verification based on blockchain in untrusted environment, *World Wide Web*, Vol. 23, No. 4, pp. 2215-2238, July, 2020.

[7] Y. Ding, Y. Li, W. Yang, K. Zhang, Edge data integrity verification scheme supporting data dynamics and batch auditing, *Journal of Systems Architecture*, Vol. 128, Article No. 102560, July, 2022.

[8] G. Xu, S. Han, Y. Bai, X. Feng, Y. Gan, Data tag replacement algorithm for data integrity verification in cloud storage, *Computers & Security*, Vol. 103, Article No. 102205, April, 2021.

[9] G. Cui, Q. He, B. Li, X. Xia, F. Chen, H. Jin, Y. Xiang, Y. Yang, Efficient verification of edge data integrity in edge computing environment, *IEEE Transactions on Services Computing*, pp. 1-1, June, 2021. DOI: 10.1109/TSC.2021.3090173

[10] D. B. Khadse, G. Swain, Data Hiding and Integrity Verification based on Quotient Value Differencing and Merkle Tree, *Arabian Journal for Science and Engineering*, pp. 1-13, July, 2022. https://doi.org/10.1007/s13369-022-06961-9

[11] Y. Ji, B. Shao, J. Chang, G. Bian, Flexible identity-based remote data integrity checking for cloud storage with privacy preserving property, *Cluster Computing*, Vol. 25, No. 1, pp. 337-349, February, 2022.

[12] A. Arasu, B. Chandramouli, J. Gehrke, E. Ghosh, D. Kossmann, J. Protzenko, R. Ramamurthy, T. Ramananandro, A. Rastogi, S. Setty, N. Swamy, A. Renen, M. Xu, Fastver: Making data integrity a commodity, *Proceedings of the 2021 International Conference on Management of Data*, Xi'an, Shaanxi, China, 2021, pp. 89-101.

[13] H. Zhu, Y. Yuan, Y. Chen, Y. Zha, W. Xi, B. Jia, Y. Xin, A secure and efficient data integrity verification scheme for cloud-IoT based on short signature, *IEEE Access*, Vol. 7, pp. 90036-90044, June, 2019.

[14] J. Xu, L. Wei, W. Wu, A. Wang, Y. Zhang, F. Zhou, Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber–physical system, *Future Generation Computer Systems*, Vol. 108, pp. 1287-1296, July, 2020.

[15] X. Shen, Y. Lu, Y. Zhang, X. Liu, L. Zhang, An Innovative Data Integrity Verification Scheme in the Internet of Things assisted information exchange in transportation systems, *Cluster Computing*, Vol. 25, No. 3, pp. 1791-1803, June, 2022.

[16] I. K. Sahu, M. J. Nene, Identity-Based Integrity Verification (IBIV) protocol for cloud data storage, *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies*, Bhilai, India, 2021, pp. 1-6.

[17] Y. Bai, Z. Zhou, X. Luo, X. Wang, F. Liu, Y. Xu, A Cloud Data Integrity Verification Scheme Based on Blockchain, *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation*, Atlanta, GA, USA, 2021, pp. 357-363.

[18] S. R. Cherupally, S. Boga, P. Podili, K. Kataoka, Lightweight and Scalable DAG based distributed ledger for verifying IoT data integrity, *2021 International Conference on Information Networking*, Jeju Island, Korea, 2021, pp. 267-372.

[19] R. Changalvala, *Sensor Data Integrity Verification for Real-time and Resource Constrained Systems*, Ph.D, Thesis, University of Michigan-Dearborn, Michigan, US, 2021.

[20] Y. Ren, J. Qi, Y. Liu, J. Wang, G. J. Kim, Integrity verification mechanism of sensor data based on bilinear map accumulator, *ACM Transactions on Internet Technology*, Vol. 21, No. 1, pp. 1-19, February, 2021.

[21] A. Daniel, S. E. Roslin, Data validation and integrity verification for trust based data aggregation protocol in WSN, *Microprocessors and Microsystems*, Vol. 80, Article No. 103354, February, 2021.

[22] H. Yu, Q. Hu, Z. Yang, H. Liu, Efficient continuous big data integrity checking for decentralized storage, *IEEE Transactions on Network Science and Engineering*, Vol. 8, No. 2, pp. 1658-1673, April-June, 2021.

[23] F. Leal, A. E. Chis, S. Caton, H. González–Vélez, J. M. García–Gómez, M. Durá, A. Sánchez–García, C. Sáez, A. Karageorgos, V. C. Gerogiannis, A. Xenakis, E. Lallas, T Ntounas, E. Vasileiou, G. Mountzouris, B. Otti, P. Pucci, R. Papini, D. Cerrai, M. Mier, Smart pharmaceutical manufacturing: Ensuring end-to-end traceability and data integrity in medicine production, *Big Data Research*, Vol. 24, Article No. 100172, May, 2021.

[24] N. V. Abhishek, M. N. Aman, T. J. Lim, B. Sikdar, DRiVe: Detecting Malicious Roadside Units in the Internet of Vehicles with Low Latency Data Integrity, *IEEE Internet of Things Journal*, Vol. 9, No. 5, pp. 3270-3281, March, 2022.

Biographies

Fanglin An, received his M.S. degree from Hainan University in 2019, he is doctoral candidate at in Hainan University, and his research interests are information security, privacy protection and cloud computing.

Jun Ye, PhD, graduated from Xidian University. Now he is an associate professor and PhD supervisor in the School of Cyberspaces Security, Hainan University. His research interests are applied cryptography, privacy protection and cloud computing.

Zhen Guo, PhD, graduated from Xidian University, she is an associate professor at Hainan University. Her main research interests include network security and cryptography.