Enhancing Cloud Data Integrity Verification Scheme with User Legitimacy Check

Dong Wu[1(✉)], Chao Peng[1], Meilan Zheng[1], Chen Fu[1], Hua Wang[1], Hua Zhong[1], Qiuming Liu[2,3], He Xiao[2,3], and Siwen Dong[3]

[1] Information and Communication Branch of Jiangxi Electric Power Co., Ltd., Nanchang, China
972452554@qq.com

[2] Nanchang Key Laboratory of Virtual Digital Factory and Cultural Communications, Nanchang 330013, People's Republic of China

[3] School of Software Engineering, Jiangxi University of Science and Technology, Ganzhou, China
liuqiuming@jxust.edu.cn

Abstract. Users employ cloud servers to store data and depend on third-party audits to guarantee data integrity. However, this auditing system poses certain risks, as it may have vulnerabilities that attackers can exploit for intrusion. To address these concerns and achieve decentralization, a cloud data integrity verification scheme is proposed. This scheme is based on a verifiable random function and aims to eliminate the need for third-party auditing. Before performing data integrity verification, a blockchain smart contract is employed to calculate bilinear pairs, serving the purpose of verifying the user's legitimacy. If the user successfully completes this verification, the integrity of the cloud data is then verified using the verifiable random function. The simulation results demonstrate that this scheme is effective in detecting the legitimacy of users and significantly reduces the computational and communication overhead associated with verifying data integrity.

Keywords: User verification · Data integrity · Smart contract · Verifiable random functions

1 Introduction

With the rapid advancement of information technology and manufacturing industry, various high-performance storage and computing devices have been developed. Additionally, computer networks have made significant progress [1]. Service providers have started merging the benefits of large storage and computing devices with high-speed networks to offer cloud computing services to users, such As AliCloud (Alibaba Cloud) and Amazon AWS. These cloud computing services have gained popularity due to their on-demand service delivery, easy network access, extensive storage resources, and high flexibility. As a result, more and more organizations and individual users are drawn to these services, considering them as one of the most influential innovations [2].

C. Wu et al. (Eds.): MONAMI 2023, LNICST 559, pp. 55–69, 2024.
https://doi.org/10.1007/978-3-031-55471-1_5

Cloud computing allows users to save a substantial amount of storage and computing resources by migrating data storage to cloud servers or delegating computing tasks to cloud service providers. Moreover, cloud computing has found widespread applications in various fields, including e-commerce, automotive networking, and medical care, where large volumes of data and demanding computational tasks are involved [3]. Hence, it is crucial to ensure the completeness and reliability of cloud data, as well as to verify the legitimacy of users, ensuring that only authorized individuals with no malicious intent can access the data [4].

The traditional method involves the utilization of a trusted Third Party Audit (TPA) to serve as an intermediary that verifies the legitimacy of accessing users and the integrity of the data [5]. However, relying on TPAs comes with its own set of drawbacks. There are inherent risks associated with TPAs. Firstly, the TPA's system might have undiscovered vulnerabilities or weaknesses that can be exploited by attackers to gain unauthorized access. This can arise from faulty software, misconfiguration, or a failure to timely update the system. Secondly, attackers may employ social engineering techniques to manipulate employees of the audit organization into revealing sensitive information or exploiting their privileges. Additionally, Distributed Denial of Service (DDoS) attacks can target the network infrastructure of TPAs, resulting in the unavailability of their systems. An attacker executes a DDoS attack by overwhelming the target system with a flood of requests, surpassing its processing capacity. Blockchain, being a decentralized ledger, offers the capability to store, compute, verify, and share data through consensus mechanisms and other technologies [6]. By leveraging the strengths of blockchain technology, it is possible to achieve greater efficiency and security compared to other data integrity verification methods. Previous studies [5, 13, 14] have explored the use of blockchain to protect data integrity, addressing the issue of single points of failure in TPAs and implementing Measurement Hash Trees (MHT) to ensure data completeness. However, none of these studies have considered the verification of unauthorized and malicious users.

We propose a new approach to improve upon the traditional method by utilizing zero-knowledge proofs. Zero-knowledge proofs allow for the verification of a claim without revealing any additional information to the verifier. In this approach, the data owner (DO) stores their data in a cloud storage server (CSP) to save local storage space. When the DO needs to retrieve the data from the CSP, it is verified using a Verifiable Random Function (VRF). If the verification is successful, the data is downloaded from the CSP. Essentially, this method allows users to encrypt their data and store it in the cloud, while also providing a verification mechanism to ensure data integrity. This scheme combines the functions of encryption and integrity verification. Users can encrypt their data using an encryption key and generate authentication tags. When data integrity needs to be verified, the user can provide the key and data, and the cloud performs decryption and verification label calculations to confirm the data integrity.

2 Prerequisite

2.1 Bilinear Pair

A bilinear pair is defined as follows: $e : G \times G \rightarrow G_T$ where G, G_T are multiplicative cyclic groups, and the bilinear map should have the following properties:

(1) Bilinear $\forall \mathbf{x}, \mathbf{y} \in \mathbf{G} \Rightarrow \mathbf{e}\left(\mathbf{x^a}, \mathbf{y^b}\right) = \mathbf{e}(\mathbf{x}, \mathbf{y})^{\mathbf{ab}}$
(2) Non-degeneracy: $\forall \mathbf{x} \in \mathbf{G}, \mathbf{x} \neq 0 \Rightarrow \mathbf{e}(\mathbf{x}, \mathbf{x}) \neq 1$.
(3) Computability: e exists and is efficiently computable.

2.2 Elliptic Curve Encryption Algorithm

Elliptic Curve Cryptography (ECC) is based on the concept of utilizing operations between points on an elliptic curve to create a secure public key cryptosystem. ECC involves a finite field and an elliptic curve defined over it, where the set of points on the curve serves as the key space. The operations performed on these points on the elliptic curve adhere to properties such as the law of exchange, the law of union, and the law of distribution. These properties are utilized to carry out encryption and decryption operations.

The main steps of ECC include:

1. Parameter selection: Selection of appropriate elliptic curves and related parameters, which include the equations defining the curves, the size of the finite domain, the base point (generating element), etc.
2. Key generation: Generate public and private key pairs. The private key is a randomly chosen value and the public key is another point on the curve obtained by multiplying the base point by the private key.
3. Encryption: the plaintext message is converted into points on the curve and combined with the other party's public key to perform point operations to generate the ciphertext.
4. Decryption: Use your own private key to perform point operations with the ciphertext sent by the other party to restore the points on the curve and convert it to a plaintext message.

2.3 Homomorphic Hash Algorithm

For a given base g, exponent r, modulo N, the corresponding modulo power is g^rmod N. This is shown below:

$$H(m) = g^m modN \tag{1}$$

The function hash satisfies the following laws, as follows:

$$\begin{aligned} H\left(m_i + m_j\right) &= g^{m_i + m_j} modN \\ &= \left(g^{m_i} modN\right) \times \left(g^{m_j} modN\right) \\ &= H(m_i) \times H\left(m_j\right) \end{aligned} \tag{2}$$

H Similarly, it can be obtained:

$$\begin{aligned} H\left(a_i m_i + a_j m_j\right) &= g^{a_i m_i + a_j m_j} modN \\ &= H(m_i)^{a_i} \times H\left(m_j\right)^{a_j} \end{aligned} \tag{3}$$

2.4 Verifiable Random Functions

A Verifiable Random Function (VRF) is a cryptographic construction that combines a random function with verifiability properties. It allows a generator to produce a pseudo-random function that can be verified by a verifier without disclosing the generator's internal secrets.

The VRF is composed of three algorithms: generation, proof, and verification.

1. Generation Algorithm: This algorithm takes a key pair (public and private keys) as input and generates a pseudo-random function output based on random numbers and messages. Only the generator with the private key can compute the function, ensuring its pseudo-random nature.
2. Proof Algorithm: Given the output produced by the generation algorithm, along with the related input message and private key, the proof algorithm generates an additional proof value. This proof value can be used by the verifier to check the correctness and unpredictability of the output.
3. Verification Algorithm: The verification algorithm takes the public key, the message, the output generated by the generation algorithm, and the proof value generated by the proof algorithm. It verifies the correctness of the proof and determines whether the generated output was indeed produced by the generating algorithm.

3 System Modeling

The system model in the paper comprises three entities: data holder (DO), blockchain platform (BC), and cloud storage server (CSP). Before conducting data integrity verification, the user stores the data in the cloud server, while the blockchain stores homomorphic digital tags containing user information and data digest information.

To verify the data integrity of the cloud server, DO must first undergo identity legitimacy testing and send the identity verification information to BC. Once the verification is passed, BC sends a data integrity query to the cloud server. The cloud server returns the data integrity proof to BC, which then checks it and sends the verification result to DO. This process is made more reliable and trustworthy than TSP due to the use of a reliable blockchain platform and smart contracts, which document every verification. As a result, DO and CSP can trust each other completely, unlike the unreliable and untrustworthy TPA (Fig. 1).

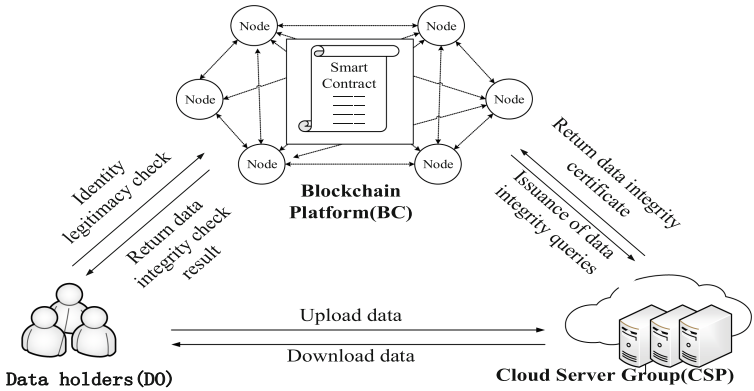

Fig. 1. System model

3.1 Data Integrity Threat Model

(1) Intentional data tampering by unauthorized users is a potential risk. Malicious individuals may manipulate validation results to gain false claims during the data integrity verification process.
(2) Cloud server failure can stem from different factors like hardware issues, software glitches, and network attacks. Such failures can lead to data corruption on the cloud server. Notably, corruption of validation information can impact both data validation and recovery.
(3) To mitigate replay attacks and minimize storage requirements, cloud servers may opt to store only data tags for integrity verification purposes.

4 Data Integrity Protection Program

Scheme overview: firstly, we design elliptic curve cryptographic verifiable random function, and then combine it with smart contract for data integrity verification, which mainly includes (1) data uploading; (2) user legitimacy checking; (3) data integrity verification; and (4) data recovery.

4.1 Elliptic Curve Verifiable Random Function Design

The verifiable random function algorithm based on elliptic curves consists of four main algorithms:

System initialization algorithm: given the system security parameter π, the algorithm produces the public parameters.

Key generation algorithm: The input includes the public parameter params, and the output comprises the public key pk and private key sk.

Random Number and Proof Generation Algorithm:The input consists of the public parameter params, message m, and private key sk. The output includes a random number value and its corresponding proof.

Verification algorithm: inputs are public parameters params, message m, proof and public key pk, output is result of verification.

The system initializes the algorithm with the following parameters: GF(q): a finite field of order q. q: a large prime number of π bits. E: an elliptic curve defined on GF(q). g: a base point on the elliptic curve.

Key generation algorithm:

A1: Choose a random number $x \in [1, q-1]$.
A2: Generate a pair of elliptic curve keys, where the private key is x and the public key is Y = xG.

Random number and proof generation algorithm: message m, private key x. Output: random number v, proof PROOF.
B1: Choose random number $k \in [1, q-1]$.
B2: Compute h = H1(m) using the hash function H1 to map message m to a point H on the elliptic curve;
B3: compute kH, kG;
B4: encode the input into an integer $c = H3((kH, kG)$; using function H3.
B5: Calculate $s = (k - c * x) mod n$;;
B6: Compute$\Gamma = xH$;
B7: Use the function H2 to encode the points on the elliptic curve as an integer to obtain the random number $\Gamma = xH$, proving that proof is (Γ, c, s).

Verification algorithm: input: message *m*', proof proof'. Output: legitimacy (*valid or invalid*).
C1: Map the message *m*' to a point *H*' on the elliptic curve using the hash function $H1$;
C2: Compute $U = c'Y + s'G$; $V = c'\Gamma + s'H'$;
C3: Calculate $c' = H3(U, V)$;
C4: If $c = c$', then it indicates that the random number is valid and the verification passes, and outputs *valid*; otherwise it indicates that the random number is invalid and the verification fails, and outputs *in valid*.

4.2 Data Upload

The user-side operation:In the user operation, the user initially applies the ECDSA algorithm to obtain the private key (sk) and public key (pk). Following this, the data is segmented into chunks for integrity verification purposes, allowing each block to be verified individually. Each data block generates a corresponding value.

Concurrently, corrective code encoding is executed to produce a recovery block. This recovered block is then combined with the original data block and uploaded to the cloud server. The set of data blocks is denoted as $D = \{d_i\}(i \in (1, n))$.

Each data block d_i is processed using the homomorphic hash algorithm and public key pk, as outlined in the preparatory knowledge, to create the label$T_i = (H(d_i))^e$. A hash operation is then performed on d_i to generate the data summary informationH_i, $H_i = H_1(d_i)$, The label T_i, H_i, and index i are uploaded to the blockchain.

Each data block also has a corresponding VALUE. The data owner uses the private key SK to calculate this VALUE of the data and stores it in the cloud along with the data.

Subsequently, the user carries out integrity verification. If the result is inconsistent, it indicates improper data storage, and distinct server is selected. If the results are consistent, both parties are in agreement.

As the data tags are stored on the blockchain, any node can access and notarize this data. This process ensures that data deletion is agreed upon and does not result in data loss.

4.3 User Legitimacy Check

Users who want to operate on the data should first verify the legitimacy of the user, and prohibit subsequent operations if they are unauthorized users, and similarly prohibit further operations if they are malicious users. An unauthorized user is a user who has not been given a sk and a malicious user is a user who intentionally denies data integrity and uses a g_s that does not match x, resulting in an error in the server-side data integrity check, and thus begging for compensation from the data repository.

The user first sends x and g_s to the blockchain legitimacy checking smart contract (SC_L).

$$x = d \times s \tag{4}$$

$$g_s = g^s \tag{5}$$

d comes from the user's private key $sk = (d, N)$, $s \in Z_p^*$, g is the generator of G_1.x is used to hide the private key sk, and g_s is used to check the user's legitimacy and data integrity.

When (SC_L) receives x and g_s from the user, it can determine whether the user is legitimate or not by verifying whether the following equation holds:

$$e\left(H^e, g^x\right) mod N = e(H, g_s) \tag{6}$$

according to the curve $h : \{0, 1\}* \rightarrow G_1$ where $H = h(v)$,$\forall v \in R$.

If the condition stated above is satisfied, it proves that the user is authorized and is a normal user. A set of random coefficients $A = \{a_i\}$, $i \in [1, n]$ is generated using (SC_L) for data integrity verification.

4.4 Data Integrity Validation

The user verifies the *VRF* value of the downloaded data block using the public key pk and a verifiable random function. The public key is used to verify the correctness of the VRF value and ensure that the data has not been tampered with. During the verification process, the generated proof value is compared with the downloaded *VRF* value Based on the result of verification of *VRF* value, the integrity of the data block is determined.

A successful match indicates that the data block remains intact and unaltered. If the *VRF* values do not match, it indicates that the data block may have been tampered with or corrupted.

4.5 Data Recovery

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \\ \boldsymbol{d}_4 \\ \boldsymbol{d}_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \boldsymbol{B}_{11} & \boldsymbol{B}_{12} & \boldsymbol{B}_{13} & \boldsymbol{B}_{14} & \boldsymbol{B}_{15} \\ \boldsymbol{B}_{31} & \boldsymbol{B}_{32} & \boldsymbol{B}_{33} & \boldsymbol{B}_{34} & \boldsymbol{B}_{35} \end{bmatrix}^{-1} \times \begin{bmatrix} \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \\ \boldsymbol{d}_5 \\ \boldsymbol{c}_1 \\ \boldsymbol{c}_3 \end{bmatrix} \tag{7}$$

B is the invertible Cauchy matrix, and C is the array of recovered blocks. The above method reveals that only the correct block can restore the original data. Initially, the server identifies and excludes the incorrect block by locating the index of the erroneous block in the scattered data storage. By comparing the summary information of the blockchain storage, which consists of non-modifiable data uploaded in the blockchain H_i the server can determine the serial number index of the data block where the error occurred. This summary information serves as a reliable reference for data recovery.

To begin the recovery process, the server performs a Hash operation on the original data it stores:$H_i^* = H_1(d_i)$. The server then transmits the index, user public key, and hash value of the stored data to the data recovery smart contract (SC_D). SC_D utilizes the index and public key to locate the summary information H_i of the data block stored on the blockchain. It compares this information with the received H_i^* and returns the index of the incorrect block to the server.

In the event that $H_i^* = H_i$, it indicates that the i-th block of the original data remains intact. Conversely, if they do not match, the i-th block of data is identified as the incorrect block.

5 Security Analysis

5.1 User Legitimacy

According to the proposed scheme, smart contracts with verifiable random functions are unforgeable and unpredictable, and only authorized and non-malicious users satisfy Eq:

$$e\left(H^e, g^x\right) modN = e(H, g_s) \tag{8}$$

Proof:

$e(H^e, g^x) modN = e(H, g)^{ex} modN = e(H, g)^{eds} modN = \left(e(H, g)^s\right)^{ed} modN$ By Euler's theorem:

$$m^{\varphi(n)} \equiv 1(modn) \tag{9}$$

$$m^{k\varphi(n)+1} \equiv m(modn) \tag{10}$$

$$ed \equiv 1(mod\,(\varphi(n))) \tag{11}$$

$ed = k\varphi(n) + 1$, with k an arbitrary constant.

$$\begin{aligned} &(e(H, g)^s)^{ed} modN \\ &= (e(H, g)^s)^{k\varphi(N)+1} modN \\ &= e(H, g)^s \end{aligned} \tag{12}$$

By the bilinear pair mapping property, it follows that:

$$e(H, g_s) = e(H, g)^s \tag{13}$$

so that the equation holds for

$$e\left(H^e, g^x\right) mod N = e(H, g_s) \tag{14}$$

If a non-authorized user is agnostic, Euler's theorem will not be satisfied and the above equation will not hold.

If the user has malicious intent, the following equation will not be satisfied:

$$g_s \neq g^{\frac{x}{d}} \tag{15}$$

Ultimately, only authorized and legitimate users with no malicious intent can be satisfied; In all other cases, the condition does not apply.

5.2 Data Integrity

Users in the data download stage, due to hardware and software failure or network delay and other factors, the downloaded data may be lost or damaged during transmission, or the server may be subjected to hacking attacks, etc., resulting in some data being tampered with, and all of these factors make the data less than complete.

According to the integrity of verifiable random function: if the inputs to the generation algorithm and the proof algorithm are legitimate, The verification algorithm will consistently accept and validate the accuracy of the proof. In the original scheme, the client will cross-verify all received data results, and only when they are all identical, the verification process will succeed Therefore, even if the attacker successfully attacks only one data node, the client will fail to validate the proof because of the inconsistency of the received data results. Even if the determination strategy is changed to more than half of the data results are consistent, the endorsement result is valid, in this case, when the number of malicious nodes in the data nodes is more than half of the number of malicious nodes, the attacker can completely control the result of the inconsistency of the data. In the optimization scheme, the client no longer needs to get all the results of the data nodes, so that the attacker affects the probability of the transaction is reduced. The same is the success of the attack on a candidate data node, if the adversary tries to influence the data, It is essential to endorse only one node from the candidate set of nodes, specifically the node that was successfully targeted by the adversary, to achieve a probability of success.

$$Pr = \frac{1}{n} C(1-\gamma)\gamma^{(n-1)} = (1-\gamma)\gamma^{(n-1)} \tag{16}$$

Prevent the attack from succeeding with a very low probability.

5.3 Preventing Replay Attacks

Once the cloud server has computed the data labels, it deletes the stored raw data and stores only the data labels g^{m_i}. The case is also not likely to pass validation because the g_s sent to the cloud server is randomized each time. There is no way for the cloud server to $g_s \rightarrow s$, which at this point converts to a discrete logarithm problem.

Discrete Logarithm (DL) problem: Let G_1 be a cyclic multiplicative group of order q on an elliptic curve with large prime q, g is a generator of G_1, $s \in Z_q$, and it is known that $g, g, g^s \in G_1$ and computing s is difficult. Namely

$$Pr\left(A_{dl}(g, g_s) \rightarrow a \in Z_q, s.t. g_s = g^s\right) \leq \varepsilon \tag{17}$$

Therefore, it is impossible for the server to give a correct verification proof with only the data labels saved.

If $P = STmodN$, it means that the cloud server saves the complete data.

Proof.

$$\begin{aligned} P &= g_s^{a_1 m_1 + a_2 m_2 + \ldots + a_n m_n} \\ &= g_s^{a_1 m_1} \times g_s^{a_1 m_1} \cdots \times g_s^{a_n m_n} \\ &= \prod_{i=1}^{n} g^{a_i m_i s} \end{aligned} \tag{18}$$

$$\begin{aligned} ST &= T_1^{a_1 x} \times T_2^{a_2 x} \times \ldots \times T_n^{a_n x} \\ &= \prod_{i=1}^{n} T_i^{a_i x} = \prod_{i=1}^{n} g^{m_i e a_i d s} \end{aligned} \tag{19}$$

According to Eqs. (13) and (14), it can be obtained:

$$\begin{aligned} STmod\ N &= \prod_{i=1}^{n} g^{m_i e a_i d s} mod\ N \\ &= \prod_{i=1}^{n} g^{m_i a_i s} \\ &= p \end{aligned} \tag{20}$$

Therefore,$P = STmodN$ holds in the case where the cloud server stores complete data.

It is also impossible for the cloud server to compute the labels on the blockchain ahead of time because s is already hidden by $x = d \times s$.

6 Performance Analysis and Experimental Evaluation

The cloud data integrity verification scheme proposed in this experiment becomes VRF-Y. The comparative literature for this experiment [5] is based on MHT to verify the integrity and is called MHT-V; and literature [13] is based on a variant of MHT and BLS and is called MB. The experiment uses elliptic curve verifiable random function and bilinear pair nature for data block signing and verification, and the hash function is SHA-256, and the hash function is SHA-256, and the hash function is SHA-256, on a A PC configured with Intel Core Duo, i7-7200U CPU with 2.50 GHz and 8 GB of RAM, calling CryptGenRandom library and OpenSSL library for key generation.

6.1 Fuel Consumption of Smart Contracts

We will assess the fuel consumption (gas) of the smart contract SC_L in the integrity verification protocol, where the test platform of the smart contract is Remix, and its programming language is Solidity of version 0.4.19. In this scheme, the inputs of the smart contract SC_L are mainly in the form of the seed value Q, the current timestamp QN, and the identity Bid of the current database. In particular, the integer type of the seed value Q selected in the experiment is int64 data type, which varies from 10 to 10^15 (Fig. 2).

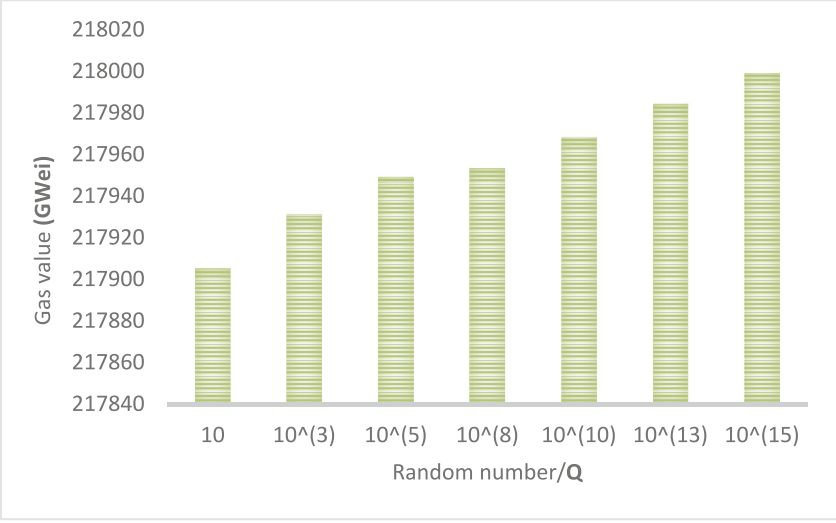

Fig. 2. Fuel value required for smart contract deployment

6.2 Comparison of Time Required to Generate Labels

MHT-V, on the other hand, does not employ labels, and thus, there is no comparison of its label generation time. The experimental results are shown in Fig. 4, which shows that VRF-Y is significantly faster than MB generation. While both utilize the modulo power operation, MB's label generation includes additional curve hash operation and MHT tree root generation, whereas VRF-Y only involves extra hash operations. As a result, VRF-Y is more efficient (Fig. 3).

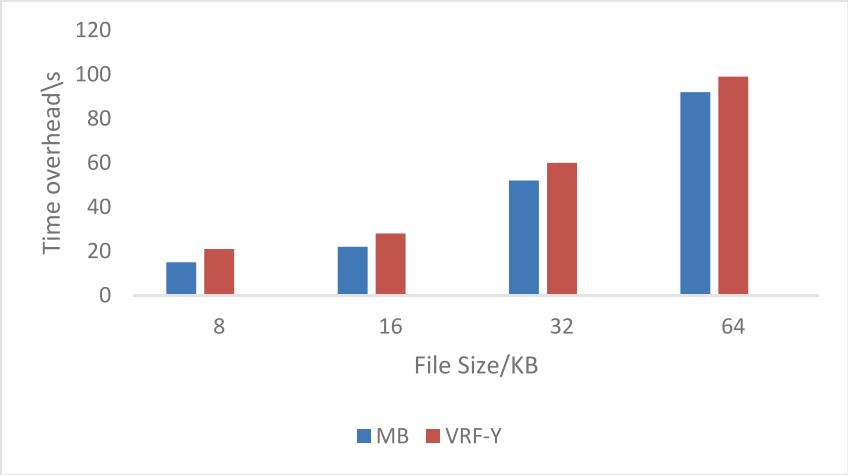

Fig. 3. Time overhead of label generation for different file sizes

6.3 Comparison of Time Spent Verifying Integrity

In VRF-Y, ECDSA algorithm is used, compared with RSA algorithm used in MB and VRF-Y Firstly, under the same security length, ECDSA has shorter key length and signature length than RSA digital signature algorithm, and in a distributed system such as blockchain, the shorter signature length can effectively reduce the content of the data transmission and lower the cost of system communication. Secondly, ECDSA is also better than RSA in terms of signature generation speed. Key nodes, as the generators of proofs and random numbers, need to perform a large number of signature operations, using ECDSA can reduce the algorithm running time and lower the impact of VRF algorithm on system throughput. Among the three methods, MHT-V demonstrates the fastest processing speed but incurs the highest communication overhead, whereas MB is the slowest with comparatively lower communication overhead, and VRF-T is in between with the lowest communication overhead. MHT-V is the fastest because it mainly performs hash operation, but it has to send the raw data to the blockchain every time, but the storage of the blockchain is limited after all. MB not only involves the modulo-power operation but also requires hash and MHT generation operations, making the entire process the most time-consuming. On the other hand, VRF-T only performs the modulo-power and hash operations, resulting in shorter processing time. Since it only needs to transmit the modulo-power calculation result to the blockchain, VRF-T incurs the least communication overhead. As shown in Fig. 5:

6.4 Error Finding

The experimental results illustrating the time overhead of error block identification under various file sizes are presented in Fig. 6. MHT-V is an error block identification method based on the MHT tree, which requires passing auxiliary information and a data block each time. It continually hashes upwards until obtaining a new root, which

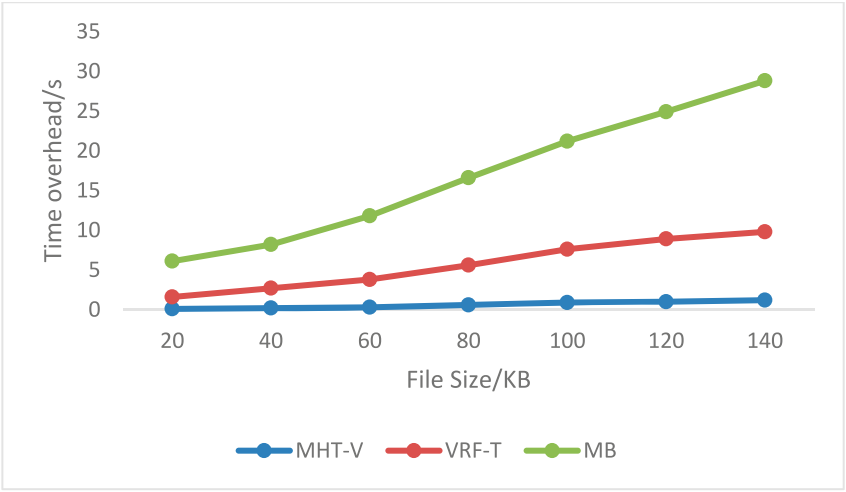

Fig. 4. Time overhead of verifying integrity with different file sizes

is then compared with the old root on the blockchain to determine block correctness. In contrast, VRF-T directly hashes the data block and compares it with the corresponding block label on the blockchain, making it a simple and efficient approach.

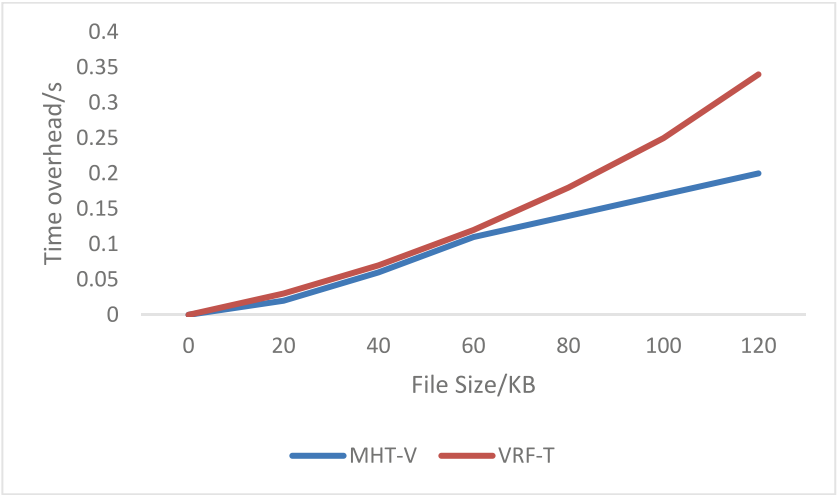

Fig. 5. Time overhead of error block lookup for different file sizes

7 Concluding Remarks

In this paper, we propose a cloud data integrity verification scheme based on verifiable random functions. The stored data is encoded with corrective deletion code to enhance data recovery capability. At the same time, the verifiable random function is used in combination with a smart contract instead of the traditional third-party auditor to realize the idea of blockchain decentralization. Before the user's access to the data, the user's identity is first checked for legitimacy. After the integrity of cloud data is verified, this scheme can help the cloud server to find the error block to recover the data. Security analysis shows that VRF-T can make users and cloud storage servers trust each other. Replacing the RSA algorithm with ECDSA algorithm also reduces the integrity verification time overhead and the computational overhead of data recovery. Moving forward, we will continue to pay attention to the trend of edge computing and continue to study cloud computing and other related fields.

References

1. Liang, H., Li, M., Chen, Y., et al.: Architectural protection of trusted system services for SGX enclaves in cloud computing. IEEE Trans. Cloud Comput. **9**(3), 910–922 (2019)
2. Zhang Guipeng. Research on cloud data security storage technology based on blockchain [D]. Guangdong:Guangdong University of Technology (2022(
3. Xu, Y., Ren, J., Zhang, Y., Zhang, C., Shen, B., Zhang, Y.: Blockchain empowered arbitrable data auditing scheme for network storage as a service. IEEE Trans. Serv. Comput. **13**(2), 289–300 (2020)
4. Yinhao, X., Yizhen, J., Chunchi, L.: Edge computing security: state of the art and challenges. Proc. IEEE **107**(8), 1608–1631 (2019)
5. Dongdong, Y., Ruixuan, L., Yan, Z.: Blockchain-based verification framework for data integrity in edge-cloud storage. J. Parallel Distrib. Comput. **146**, 1–14 (2020)
6. Zhang, Y.: Identity-based integrity detection scheme for cloud storage. Computer Engineering. **44**(3) (2018)
7. Xiuqing, L., Zhenkuan, P., Hequn, X.: An integrity verification scheme of cloud storage for internet-of-things mobile terminal devices. Comput. Secur. **92**, 101686 (2020)
8. Deswarte, Y., Quisquater, J., Saidane, A.: Remote integrity checking. In: The Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS) (2007)
9. Wutong, M., Zhang, D.: Hyperledger Fabric consensus mechanism optimization scheme. J. Autom. 47(8), 1885–1898 (2007)
10. Hovav, S., Brent, W.: Compact proofs of retrievability. J. Cryptol. **26**(3), 442–483 (2013)
11. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In: Backes, M., Ning, P. (eds) Computer Security – ESORICS 2009. ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009).https://doi.org/10.1007/978-3-642-04444-1_22
12. Zheng, Z., Xie, S., Dai, H.: An overview of blockchain technology: architecture, consensus, and future Trends, pp. 557–564 (2017)
13. Li, J., Wu, J., Jiang, G.: Blockchain-based public auditing for big data in cloud storage. Inf. Process. Manage. **57**(6), 102382 (2020)
14. Wang, Y.: Research on data integrity auditing in cloud storage[D]. Chongqing University (2021)

15. Vujičić, D,, Jagodić, D., Ranđić, S.: Blockchain technology,bitcoin,and ethereum:a brief overview. In: Proceedings of the 2018 17th International Symposium on Infoteh-Jahorina, pp. 1–6 (2018).https://doi.org/10.1109/INFOTEH.2018.8345547
16. Xiao, Y., Zhang, N., Lou, W.J., et al.: A survey of distributed consensus protocols for blockchain networks. IEEE Commun. Surv. Tutorials **22**(2), 1432–1465 (2020)
17. Zheng, M., Wang, H., Liu, H., et al.: Survey on consensus algorithms of blockchain. Netinfo Secur. **7**, 8–24 (2019)
18. Hai-yang, D., Zi-chen, L., Wei, B.: (k, n)halftone visual cryptography based on Shamir′s secret sharing. J. China Univ. Posts Telecommun. **25**(2), 60–76 (2018)
19. Qi-feng, S., Zhao, Z., Yan-chao, Z., et al.: Survery of enterprise blockchains. J. Softw. **30**(9), 2571–2592 (2019(
20. Yong, Y., Xiao-chun, N., Shuai, Z., et al.: Block consensus algorithms: the state of the art and future trends. Acta Automatica Sinica, **44**(11), 2011–2022 (2018)
21. Shao, Q.F., Jin, C.Q., Zhang, Z., et al.: Blockchain: architecture and research progress Chinese. J. Comput. **41**(5), 969–988 (2018)
22. Bo, Y.: A chameleon hash authentication tree optimisation audit for data storage security in cloud calculation. Int. J. Innovative Comput. Appl. **11**(2–3), 141–146 (2020)
23. Mehibel, N., Hamadouche, M.H.: Authenticated secret session key using elliptic curve digital signature algorithm. Secur. Priv. **4**(2), e148 (2021)
24. Khalili, M., Dakhilalian, M., Susilo, W.: Efficient chameleon hash functions in the enhanced collision resistant model. Inf. Sci. **510**, 155–164 (2020)
25. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) Computer Security – ESORICS 2009. ESORICS 2009. Lecture Notes in Computer Science, vol. 5789, pp. 355–370. Springer, Heidelberg (2009).https://doi.org/10.1007/978-3-642-04444-1_22
26. Zhang, Y., Xu, C.X., Yu, S., et al.: SCLPV: secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. IEEE Trans. Comput. Soc. Syst. **2**(4), 159–170 (2015)