Double-Layer Blockchain-Based Decentralized Integrity Verification for Multi-chain Cross-Chain Data

Weiwei Wei, Yuqian Zhou(✉), Dan Li(✉), and Xina Hong

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
zhouyuqian@nuaa.edu.cn, lidansusu007@163.com

Abstract. With the development of blockchain technology, issues like storage, throughput, and latency emerge. Multi-chain solutions are devised to enable data sharing across blockchains, but in complex cross-chain scenarios, data integrity faces risks. Due to the decentralized nature of blockchain, centralized verification schemes are not feasible, making decentralized cross-chain data integrity verification a critical and challenging problem. In this paper, based on the ideas of "*governing the chain by chain*" and "*double layer blockchain*", we propose a double-layer blockchain-based decentralized integrity verification scheme. We construct a supervision-chain by selecting representative nodes from multiple blockchains, which is responsible for cross-chain data integrity verification and recording results. Specifically, our scheme relies on two consensus phases: integrity consensus for verification and block consensus for result recording. We also integrate a reputation system and an election algorithm within the supervision-chain. Through security analysis and performance evaluation, we demonstrate the security and effectiveness of our proposed scheme.

Keywords: Data integrity · Double-Layer blockchain · Cross chain · Decentralized verification · Multi-chain architecture

1 Introduction

Blockchain technology has gained significant attention across industries in recent years. Initially introduced as the underlying technology for cryptocurrency, blockchain has evolved into a disruptive innovation with the potential to revolutionize numerous sectors, including finance, supply chain management, healthcare, and more [14,18]. However, the widespread adoption of blockchain faces the scalability issue, which arises from the inherent design of traditional blockchain networks. As the number of participants and transactions increases, the single-chain architecture encounters limitations in terms of throughput, latency, and storage requirements.

To solve these problems, some researchers have introduced innovative solutions such as multi-chain architecture and cross-chain technology [6]. For example, Kang et al. propose a multi-chain federated learning (FL) framework, in

B. Luo et al. (Eds.): ICONIP 2023, LNCS 14452, pp. 264–279, 2024.
https://doi.org/10.1007/978-981-99-8076-5_19

which multiple blockchains are customized for specific FL tasks and individually perform learning tasks for privacy protection [7]. Multiple blockchains interact and collaborate with each other through cross-chain techniques, enabling a scalable, flexible, and communication-efficient decentralized FL system.

These multi-chain architectures all require data sharing among multiple block-chains. Existing researches [4,16,17] focus only on how to achieve cross-chain interaction and collaboration, without solving the problem of data integrity in multi-chain data sharing. Only when the integrity of the cross-chain data is confirmed can cross-chain applications effectively engage in data exchange and collaboration, thus achieving the objectives and advantages of a multi-chain architecture. Therefore, research is needed to ensure cross-chain data integrity among multiple chains.

The data integrity problem in the cloud storage environment has been extensively studied. The provable data possession (PDP) scheme, along with its various iterations [1,10,19], is widely used to address the data integrity problem in the cloud. However, the integrity of data sharing in a multi-chain environment is fundamentally different as: 1) In a multi-chain architecture, it is necessary to verify the integrity of cross-chain data distributed among multiple receiving blockchains for a specific piece of data. In contrast, in cloud storage scenario, the integrity verification focuses on data within a single cloud. 2) A multi-chain architecture consists of multiple decentralized blockchains, lacking the centralized control found in cloud storage scenario.

Additionally, to alleviate the heavy computational burden on users, various research studies employ third-party auditors (TPA) to check the data integrity on the untrusted cloud [5,11]. However, the centralized TPA, which can never be fully trusted, will weaken the decentralized nature of the blockchain.

In this paper, we propose a decentralized scheme based on the idea of double-layer blockchain [2] to ensure cross-chain data integrity across multiple blockchains. We utilize representative nodes from each blockchain in the multi-chain architecture to construct a supervision-chain, which is responsible for cross-chain data integrity verification. Additionally, based on the lightweight sampling method and the Boneh-Lynn-Shacham (BLS) signature, we provide a probabilistic integrity guarantee. Furthermore, we also provide detailed descriptions of the reputation system, node election algorithm, and block consensus process for the proposed supervision-chain. The main contributions of this paper are summarized as follows:

- This paper is the first to apply the idea of a double-layer blockchain to the field of cross-chain data integrity verification. Representative nodes are extracted from a multi-chain architecture to construct a supervision-chain, which is responsible for verifying the integrity of cross-chain data and recording the verification results.
- A cross-chain data integrity verification process is designed within the supervision-chain, where nodes collaborate with each other for decentralized verification. Additionally, leveraging lightweight sampling algorithms and BLS signature, we achieve an efficient verification process.

- To improve the efficiency of the block consensus process, this paper introduces a block consensus committee. Moreover, the reputation system of the supervision-chain and election algorithm for the block consensus committee are meticulously designed.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to the preliminaries covered in this paper. Section 3 defines the system model, threat models, and design goals. Section 4 presents the proposed scheme. Section 5 conducts the security analysis. Section 6 evaluates the performance of the proposed cross-chain data integrity verification scheme. Section 7 concludes this paper and points out the future work.

2 Preliminaries

2.1 Bilinear Pairing

Bilinear pairing [12] is based on cryptography, which relies on a difficult hypothesis similar to the elliptic curve discrete logarithm problem, which can often be used to reduce the problem in one group to an easier problem in another group. Let G and G_T be two multiplicative cyclic groups of large prime order q. A map function $e : G \times G \rightarrow G_T$ is a bilinear pairing only when it satisfies three properties below:

- Bilinear: For $u, v \in G$ and $a, b \in Z_q^*$, $e\left(u^a, v^b\right) = e(u, v)^{ab}$;
- Non-Degeneracy: $e(g, g)$ is a generator of G_T;
- Computability: For $\forall u, v \in G$, there exists efficient algorithms to compute $e(u, v)$.

2.2 BLS Signature

The Boneh-Lynn-Shacham (BLS) signature [9] is a cryptographic scheme widely used to help senders certificate their messages. It works on top of elliptic curves and employs bilinear pairing to perform verification. Assume that a sender is equipped with a public/private key pair (pk, sk) $\left(sk \in \mathbb{Z}_q^* \text{ and } pk = g^{sk}\right)$. To generate a signature sig for a given message mes, the sender maps mes to the elliptic curve with a secure hash function $hash()$. Then, it generates signature sig from its private key, i.e., $sig = hash(mes)^{sk}$. A receiver can verify mes with the bilinear mapping function $e()$ mentioned in Sect. 2.1 based on the sender's public key pk and message signature sig. If Eq. (1) holds, the received message mes is correct.

$$e(pk, hash(mes)) \overset{?}{=} e(g, sig) \tag{1}$$

BLS signature's security is ensured by the hardness of solving the Computational Diffie-Hellman (CDH) problem. In our scheme, each node in the supervision-chain has a randomly chosen unique sk as its private key. Then, its corresponding public key pk is generated by g^{sk}.

2.3 Verifiable Random Function

Verifiable random function (VRF) is a cryptographic function that provides pseudo-random and publicly verifiable values, e.g., the one introduced in [3] based on bilinear pairing. Specifically, given a random seed x, a user u equipped with a public/private key pair (pk, sk) can generate a random value $f_{sk}(x)$ by Eq. (2) and a tag $\pi_{sk}(x)$ by Eq. (3).

$$f_{sk}(x) = e(g, g)^{1/(x+sk)} \tag{2}$$

$$\pi_{sk}(x) = g^{1/(x+sk)} \tag{3}$$

Tag $\pi_{sk}(x)$ is used to prove the correctness of $f_{sk}(x)$. With both $f_{sk}(x)$ and $\pi_{sk}(x)$, a receiver can verify the correctness of $f_{sk}(x)$ based on u's public key pk. If both Eq. (4) and Eq. (5) hold, the random value $f_{sk}(x)$ is correctly generated by u.

$$e\left(g^x \times pk, \pi_{sk}(x)\right) \overset{?}{=} e(g, g) \tag{4}$$

$$e\left(g, \pi_{sk}(x)\right) \overset{?}{=} f_{sk}(x) \tag{5}$$

3 Problem Statement

3.1 System Model

In a multi-chain architecture with n consortium blockchains, the problem is that the blockchain which possesses the original data d intends to verify the integrity of the cross-chain data stored in the receiving blockchains. These representative nodes are selected from each blockchain to form a supervision-chain, which also is a consortium blockchain. We refer to the original blockchains in the multi-chain architecture as the sub-layer and the supervision-chain as the main layer, forming a double-layer framework [2]. In the supervision-chain, each node can read data from the blockchain within the sub-layer it belongs to. Therefore, cross-chain data integrity verification from each blockchain in the sub-layer can be done within the supervision-chain. There are n nodes in supervision-chain, denoted as $Node = \{node_i | 1 \le i \le n\}$. Each node $node_i \in Node$ has a public/private key pair $(pk_{node_i}, sk_{node_i})$ and is identified by its public key pk_{node_i}. Here, $sk_{node_i} = x_i \in Z_p^*, pk_{node_i} = g^{x_i} \in G$.

Assume that a blockchain in the sub-layer has sent data d by cross-chain method to each of $k (k < n)$ received blockchains, and the corresponding nodes in the supervision-chain denoted as $Node_d \subseteq Node$. The system model consists of four parts, including the sending-chain, receiving-chains, supervision-chain and other blockchains in multi-chain architecture. The system model is shown in Fig. 1.

Sending-Chain: The original data owner, who sends the data d to the receiving-chains using cross-chain methods, intends to verify the integrity of the cross-chain data d.

Receiving-Chain: Receiving-chain is the recipient of the cross-chain data, and it is subject to data integrity verification conducted by the supervision-chain.
Other Blockchain: The other blockchain in the multi-chain architecture act as participant in the construction of the supervision-chain.
Supervision-Chain: The supervision-chain is constructed by representative nodes from the blockchains in the multi-chain architecture and is responsible for verifying the integrity of cross-chain data in receiving-chains.

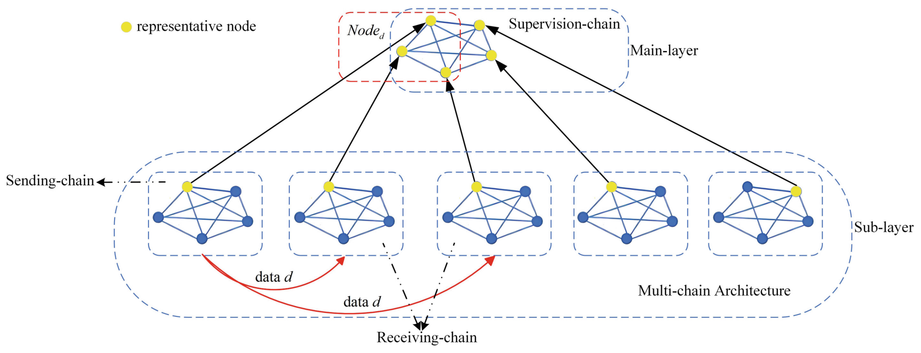

Fig. 1. System model

3.2 Threat Models

Assume that the representative node of the blockchain sending the data d is $node_s \in Node_d$, and the representative node of the blockchain receiving the data d is $node_r \in Node_d$. During the process of data integrity verification in the supervision-chain, the following threats exist:

- **Unexpected Failures.** Faults such as hardware failures, software exceptions and cyber attacks may cause cross-chain data to be corrupted.
- **Modification Attack.** The cross-chain data may be modified by the receiving-chain before being stored on the chain.
- **Freeriding Attack.** A $node_r$ may reuse an integrity proof message from another honest $node_r'$ to pass the integrity verification.
- **Prediction Attack.** If these nodes participating in the block consensus in the supervision-chain are easily predictable in advance, external adversaries can easily attack the block consensus process.

3.3 Design Goals

Under the above system model and threat model, our scheme should meet the following three goals.

- **Decentralized Verification.** In the context of cross-chain scenario, using centralized entities for data integrity verification will weaken the decentralization of blockchain systems. By distributing the verification process across multiple nodes, the overall system becomes more resilient and resistant to attacks or manipulations.
- **Correctness.** The proposed scheme should ensure that the supervision-chain can correctly verify the integrity of cross-chain data by integrity verification process.
- **Security.** The proposed scheme should prevent from modification attacks, freeriding attacks, and prediction attacks.

4 Double-Layer Blockchain-Based Decentralized Integrity Verification Scheme

4.1 Overview

In our scheme, we use the supervision-chain to verify cross-chain data integrity in blockchains of sub-layer and record the results. In order to achieve these goals, we employ two consensus protocols, one for integrity consensus and the other for block consensus [9]. The integrity consensus aims to achieve consensus on the verification result of a given cross-chain data d. The block consensus is utilized within the system to achieve consensus on the blocks that will be recorded on the blockchain. We assume that in the sub-layer, a blockchain possessing the original data d aims to verify the integrity of cross-chain data stored on the receiving blockchains. We refer to the representative node of the blockchain which sent the original data d as $node_s$ and refer to the representative nodes of the blockchains which received cross-chain data as $node_r \in Node_d$.

In summary, a complete scheme consists of two phases: integrity consensus and block consensus. In the first phase, the system reaches a consensus on the verification result. If enough results has generated in the first phase, the second phase starts. In the second phase, the system reaches a consensus on the transaction information to record it on the blockchain.

Next, we provide a detailed explanation of the integrity consensus process and the block consensus process. For simplicity, we give some notations in Table 1.

4.2 Integrity Consensus

Firstly, we describe the sampling algorithm used in the integrity verification process. Then we give the detailed integrity consensus process.

4.2.1 Sampling Algorithm Inspired by the sampling algorithm proposed in [15], $node_s$ generate sampling parameters sp_r for each node $node_r \in Node_d$ by the following steps. Assume that the number of $Node_d$ is $k + 1$. $Per(x, y)$ is a pseudo-random permutation function, where x is a random number and y is the total number of data blocks to be permuted.

Table 1. The notations in our scheme.

Notation	Description
d	The cross-chain data
n	The total number of nodes in the supervision-chain
$Node_d$	The corresponding nodes which possess data d
$node_s$	The representative node of the sending-chain in the supervision-chain
$node_r$	The representative node of the receiving-chain in the supervision-chain
$node_b$	The block consensus committee
k	The total number of representative nodes of the receiving-chains
m	The number of nodes in the block consensus committee
$Per(x, y)$	A pseudo-random permutation function
$hash()$	A hash function
f	The number of malicious nodes that the blockchain system can tolerate

Step 1. For data d, which is divided into n data blocks, $node_s$ uses $Per(x, y)$ to process the index array $\{1, 2, ..., n\}$ and get a randomly sorted index array $SIA = \{index_1, index_2, ..., index_n\}$.

Step 2. $node_s$ divide SIA into k subsets, i.e. $SIA = \{C_1, C_2, ..., C_k\}$, satisfying the intersection of k subsets is empty and the union is $\{1, 2, ..., n\}$. These k subsets are used as the challenged index sets for $Node_d$. Generally, each subset has about $\lfloor n/k \rfloor$ elements, where $\lfloor n/k \rfloor$ represents the integer part of n/k. In particular, the last subset has $n - (k - 1) \lfloor n/k \rfloor$ elements.

4.2.2 Integrity Consensus Process Assume that $node_s$ would like to verify the integrity of cross-chain data d in $node_r \in Node_d$. The integrity consensus process goes through four steps, as shown in Fig. 2.

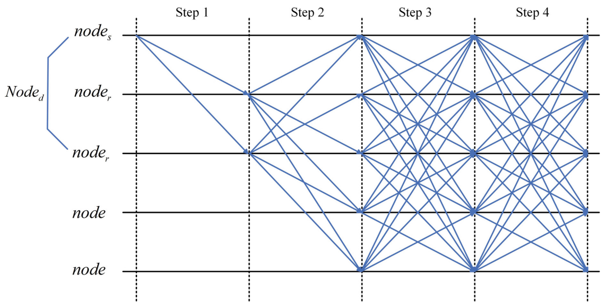

Fig. 2. Integrity consensus process

Step 1 Verification Request. $node_s$ use the sampling algorithm (see Sect. 4.2.1 Sampling Algorithm) to generate sampling parameters sp =

$\{C_1, C_2, ..., C_k\}$. Then it samples data blocks from d according to the sampling parameter $C_i \in sp$. Then it generates a Merkle Hash Tree(MHT) with a hash function $hash()$ based on the sampled data blocks. Next, $node_s$ calculate the BLS signature of the root node of MHT as the reference integrity proof for $node_r$, denoted as sig_r.

$$sig_r = hash(root)^{sk_{node_s}} \tag{6}$$

Finally, $node_s$ send verification request verification request $vr =< pk_s, sp_r, d_{id}, sig_r >$ to each $node_r \in Node_d$.

Here, pk_s represents the unique identifier of $node_s$, sp_r denotes the sampling parameters for $node_r$, d_{id} is the the unique identifier of cross-chain data d and sig_r signifies the reference integrity proof for $node_r$.

Step 2 Integrity Proof Generation. Upon the receipt of a verification request, $node_r$ generates a MHT from its own cross-chain data d based on the specified sampling parameters sp_r and use the root of MHT to calculate its own integrity proof sig'_r.

$$sig'_r = hash(root')^{sk_{node_r}} \tag{7}$$

Then $node_r$ broadcasts the integrity proof message $ipm =< pk_s, pk_r, sig_r, sig'_r >$ in the supervision-chain. Here, pk_r represents the unique identifier of $node_r$.

Step 3 Verification Response. Upon the receipt of an integrity proof message ipm from $node_r$, a node $node_i$ checks if Eq. (8) holds.

$$e(pk_s, sig'_r) \overset{?}{=} e(pk_r, sig_r) \tag{8}$$

If Eq. (8) holds, the ipm is valid and the cross-chain data in the blockchain of sub-layer is intact. Otherwise, the integrity of cross-chain data is corrupted. After validating all received ipm, $node_i$ broadcasts the verification response $vo = <d.id, result, list>$, where $result$ is the summary of the verification results and $list$ is a set of pk belonging to the nodes for which the integrity proof is invalid. Specially, if all equations holds, $result$ is true and $list$ is empty.

Step 4 Agreement. When a node $node_i$ receives $\lceil 2n/3 \rceil$ same verification response vo, the final verification conclusion is made. In detail, if $node_i$ receives $\lceil 2n/3 \rceil$ $result = true$, it thinks all nodes in $Node_d$ possess intact cross-chain data. Otherwise, $node_i$ thinks these nodes appearing $\lceil 2n/3 \rceil$ times in $list$ do not possess intact cross-chain data. After confirming the final result, $node_i$ broadcasts a integrity consensus commit in the supervision-chain. When a node receives $\lceil 2n/3 \rceil$ integrity consensus commits, it ends the integrity consensus process.

4.3 Block Consensus

In this subsection, we describe the whole block consensus process. Firstly, the reputation system and the election algorithm used in the block consensus process will be introduced. Then the detailed block consensus process will be given.

4.3.1 Reputation System In the supervision-chain, there are two types of rewards as incentives to motivate node to participate in integrity verification process.

Transaction Reward. A transaction reward is provided by the $node_s$ and allocated to those nodes that honestly verify integrity proof message *ipm*.

Block Reward. This reward is produced by the system to encourage nodes to participate in the maintenance of the supervision-chain, similar to most blockchain systems. For a node to gain block reward, it needs to satisfy the following two conditions: First, it is within the block consensus committee. Second, it actively and honestly participates in the block consensus process. The rewards gained by each node in the past are recorded on the blockchain, and the reputation of each node is calculated based on its historical rewards. In the election process of the consensus committee, nodes with a higher reputation score will be given prioritization.

In the reputation system, a node's reputation is determined by three factors: 1) The reputation score is higher when the node actively engages in a substantial number of integrity verification. Merely being honest without significant participation will not yield a high reputation score. 2) The reputation decreases when there are more occurrences of concealing inappropriate behavior. Each instance of such behavior results in a deduction of rewards, thereby lowering the reputation. 3) Recent behavior holds considerable weight in shaping the reputation score.

To meet the mentioned factors, we use an exponential moving average algorithm with bias correction to calculate the reputation of a node.

$$r^t_{node_i} = \begin{cases} 0 & t = 0 \\ \frac{\rho \times p^t_{node_i} + (1-\rho) \times r^{t-1}_{node_i}}{1-\rho^t} & t > 0 \end{cases} \tag{9}$$

Here, $p^t_{node_i}$ is the total amount of rewards from the recent transactions that have not been packed in block, $\rho \in (0,1)$ is a weighting factor, and a larger ρ indicates a higher weight on the impact of recent transactions on reputation. $1-\rho^t$ is a bias correction factor that ensures the stability of the reputation score over time.

4.3.2 Election Algorithm During the block consensus process, we first elect a block consensus committee. The block consensus is then carried out within this committee, and the consensus-reaching block is subsequently synchronized to other nodes. Using a block consensus committee to achieve block consensus has the advantage of improving the efficiency of block consensus. A drawback is that it reduces the number of malicious nodes (denoted as f) that the supervision-chain system can tolerate. Given that these nodes in the supervision-chain are diligently chosen as representatives from the sub-layer consortium blockchain, the incidence of malicious nodes within the supervision-chain is notably low. Thus, this drawback is acceptable. If the block consensus committee (m nodes) can be predicted based on information available on the blockchain, adversaries can disrupt the consensus process by attacking $\lceil 2m/3 \rceil$ nodes in the consensus committee. Therefore, it is unsafe to always select the top m nodes with the highest reputation scores to form the block consensus committee. Based on the reputation system(see Sect. 4.3.1 Reputation System), we use VRF (Verifiable Random Function) [9,13] to elect the consensus committee, ensuring that only nodes with reputation score surpassing a certain threshold have the chance to be selected for the block consensus committee. The block consensus committee election goes through two steps, as follows:

Step 1 Candidate preparation. Only nodes with reputation score exceeding a certain threshold are qualified to become candidates. By referring to their own reputation records stored on the blockchain, a node can easily determine whether it meets the criteria necessary to become a candidate. If eligible, $node_i$ generates a competition request $cr_i =< pk_{node_i}, f_{sk_{node_i}}(x), \pi_{sk_{node_i}}(x) >$ based on a random seed x and its private key sk_{node_i}. Here, $f_{sk_{node_i}}(x)$ and $\pi_{sk_{node_i}}(x)$ are calculated as follows:

$$f_{sk_{node_i}}(x) = e(g, g)^{1/(x+sk_{node_i})} \tag{10}$$

$$\pi_{sk_{node_i}}(x) = g^{1/(x+sk_{node_i})} \tag{11}$$

Then it broadcasts competition request cr_i in the supervision-chain.

Step 2 Leader and Member Determination. Upon receiving a competition request cr_i from node $node_i$, each node $node_j \in Node$ performs the necessary checks: 1) It checks that if node $node_i$ has sufficient reputation to qualify as a candidate. 2) It validates the correctness of the competition request cr by verifying the following two equations:

$$e(g^x \cdot pk, \pi_{sk}(x)) = e(g, g) \tag{12}$$

$$e(g, \pi_{sk}(x)) = f_{sk}(x) \tag{13}$$

If multiple candidates simultaneously possess valid competition requests, the node with the highest value of $f_{sk_{node}}(x)$ is selected as the leader. Next, select $3f$ nodes with higher values of $f_{sk_{node}}(x)$ from the candidates who were not successful in the election to become members of the block consensus committee.

4.3.3 Block Consensus Process When multiple integrity consensus processes are completed and there is enough transaction information to be packaged into a block, the block consensus process starts. A block consensus process consists of three steps, as shown in Fig. 3.

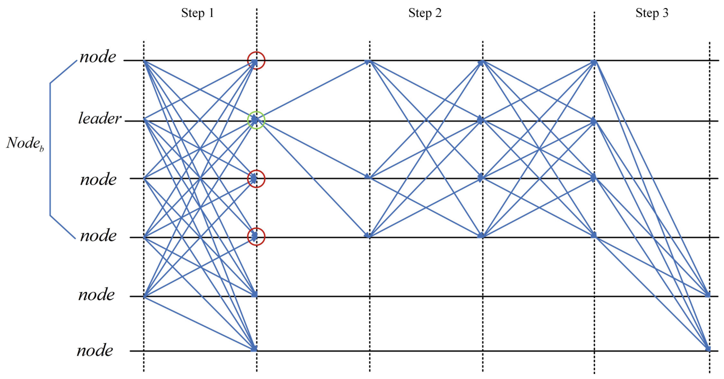

Fig. 3. Block consensus process

Step 1 Election. A block consensus committee $Node_b$ is elected (see Sect. 4.3.2 Election Algorithm).

Step 2 Consensus. The leader packs the transactions(several final consensus information reached during the integrity consensus phase) into a new block. Specifically, based on the messages received, it allocates the transaction rewards provided by $node_s$ to these nodes that verify the integrity proof message ipm honestly. Next, it allocates negative transaction rewards to these representative nodes of the sub-layer blockchain that do not honestly store cross-chain data d. Next, it updates the reputations of all nodes in the supervision-chain with Eq. (9). Specifically, the related rewards and reputations are also packed into the block. The leader then broadcasts the block in the block consensus committee $Node_b$ for validation. Upon receiving the block, the node $node_i \in Node_b$ checks correctness of current round information. If passed, $node_i$ broadcasts a prepare message to claim its ready state. Once $node_i$ obtains more than $\lceil 2m/3 \rceil$ prepares message, it begins to verify the content of the block based on the information obtained during the integrity consensus phase. If passed, $node_i$ broadcasts the commit message to other nodes in $Node_b$. Finally, if $node_i$ receives more than $\lceil 2m/3 \rceil$ commit messages, it will accept the new block and append it to the ledger.

Step 3 Synchronization. All nodes in the block consensus committee $Node_b$ respond to other nodes in the supervision-chain. A node will accept the new block if more than $f + 1$ same blocks are received, where f is the maximum number of malicious nodes that the blockchain system can tolerate. At the end, the leader receives the block reward, while the other nodes in $Node_b$ receive a block reward that is less than the leader's.

5 Security Analysis

In this section, we provide a brief evaluation of the correctness and security of the proposed scheme.

Theorem 1. *If a blockchain honestly stores cross-chain data d, its representative node can pass the integrity verification during the integrity consensus phase.*

Proof. The correctness of the Eq. (8) can be proved as follow:

$$e(pk_s, sig'_r) = e(g^{sk_s}, hash(root')^{sk_r}) = e(g^{sk_r}, hash(root)^{sk_s}) = e(pk_r, sig_r)$$

If a node $node_r$ have the intact data, it can generate a valid sig'_r. Therefore, it can pass other nodes' verification(Step 3 in 4.2.2 Integrity Consensus Process).

Theorem 2. *A node $node_i \in Node_d$ cannot reuse an integrity proof message from an honest node $node_r$ to attack the integrity consensus.*

Proof. For a integrity proof message $ipm =< pk_s, pk_r, sig_r, sig'_r >$, it contains only signatures sig_r and sig'_r but not the original hash tags. It is impossible for $node_i$ to forge a signature due to the hardness of solving the CDH problem [8]. Moreover, $node_i$ may reuse the existing signatures but change the identity of $node_r$ from pk_r to pk_i in ipm to forge an integrity proof message $ipm' =< pk_s, pk_i, sig_r, sig'_r >$. But this behavior can be easily detected by Eq. (8).

Theorem 3. *If the cross-chain data, denoted as d, is divided into n blocks, the probability that the supervision-chain successfully detects a dishonest blockchain that does not store d accurately is at least $Pr = 1 - (\frac{n-c}{n})^t$. In this equation, c represents the number of altered data blocks within d, and t stands for the number of challenged blocks.*

Proof. Based on the sampling algorithm, the challenged blocks on k nodes in $Node_d$ are completely unrepeatable and their union is exactly all the data blocks of d. For a node $node_r \in Node_d$, the probability of finding d modified is equal to the probability of at least challenging one modified data block. In other words, we can calculate the probability that at least one modified block is challenged on $node_r$ during the integrity consensus phase as follows:

$$\Pr = 1 - \left(\frac{n-c}{n}\right)\left(\frac{n-c-1}{n-1}\right)\left(\frac{n-c-2}{n-2}\right)\cdots\left(\frac{n-c-t+1}{n-t+1}\right).$$

Given an arbitrary integer $i \leq n$, there is $\frac{n-c-i}{n-i} \geqslant \frac{n-c-i-1}{n-i-1}$. Hence, the following inequality holds:

$$Pr > 1 - (\frac{n-c}{n})^t.$$

Theorem 4. *During the election process of the block consensus committee, any node can verify the correctness of a competition request cr.*

Proof. For a competition request $cr =< pk, f_{sk}(x), \pi_{sk}(x) >$, the correctness of it can be verified as follows:

$$e(g^x \cdot pk, \pi_{sk}(x)) = e(g^x \cdot g^{sk}, g^{1/(x+sk)}) = e(g, g)$$

$$e(g, \pi_{sk}(x)) = e(g, g^{1/(x+sk)}) = e(g, g)^{1/(x+sk)} = f_{sk}(x)$$

If both of the above two equations hold true, then cr is generated by the node with the public key pk.

6 Performance Evaluation

In this section, we conduct a series of experiments to evaluate the performance of our scheme.

6.1 Experimental Settings

We implement the integrity verification process based on the Java Pairing-Based Cryptography Library (JPBC) version 2.0.0, which performs the mathematical operations underlying pairing-based cryptosystems. In our experiments, we choose the type A pairing parameters in JPBC library, which the group order is 160 bits and the base field order is 512 bits. And our experiments are conducted on a PC laptop which runs Windows 10 on an Intel Core i5 CPU at 2.50 GHz and 8 GB DDR4 RAM. To get more precise results, each experiment is conducted 100 trials.

6.2 Experimental Results and Analysis

We focus on the evaluation related to computation cost, communication cost and detection precision in the integrity verification process. The detailed analysis is as follows.

Computation Cost. We set the number of nodes required for data integrity verification from 2 to 22. For the same block size of 8 KB, we set the data size of d to 4 MB and 8 MB, respectively, and measure the time required for the integrity verification process. As Fig. 4, the larger the file, the longer it takes to complete the integrity verification process. This is due to the increase in the number of data blocks, which leads to a higher number of samples being taken from each node. Consequently, a larger number of MHT nodes need to be computed during the verification process, resulting in increased time consumption. Furthermore, we can also observe from Fig. 4 that as the number of nodes to be verified increases, the time consumption also increases, which aligns with our expectations and falls within an acceptable range.

Communication Cost. We set the total number of nodes in the system from 5 to 25, with 4 nodes required for data integrity verification. For the same block size of 8 KB, we set the data size of d to 4 MB and 8 MB, respectively, and measure the communication cost for the integrity verification process. From Fig. 5, it can be observed that as the data size increases, the communication overhead remains relatively constant. This is because the increase in data size results in a larger number of data blocks, but it only leads to an increase in the number of blocks sampled within the data d of each node. Given that the number of integrity consensus nodes remains unchanged, the number of integrity proofs that need to be sent (i.e., the MHT root of BLS signatures) also remains constant. As a result, the communication overhead during the integrity verification process remains approximately unchanged. Furthermore, as shown in Fig. 5, for the same data size, the communication overhead increases with the number of nodes in the system. This is attributed to the fact that a larger number of nodes necessitates increased communication to achieve integrity consensus.

Detection Precision. We set the number of data blocks to 45,000 and investigate the relationship between detection accuracy and the number of sampled blocks under different corruption rates. As shown in Fig. 6, for each different data corruption rate, our scheme achieves close to 100% detection accuracy with only a small number of sampled data blocks.

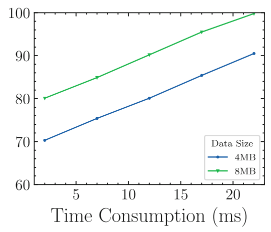

Fig. 4. The computation cost in the integrity verification process.

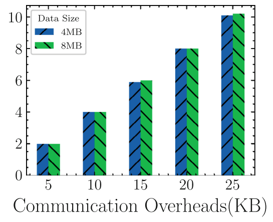

Fig. 5. The communication cost in the integrity verification process.

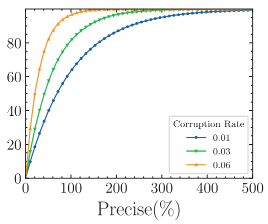

Fig. 6. The detection rate for data corruption.

7 Conclusion

To solve the problem of cross-chain data integrity in the multi-chain architecture, we propose a double-layer blockchain-based decentralized integrity verification scheme based on the ideas of "*governing the chain by chain*" and "*double-layer blockchain*". In detail, We construct a supervision-chain by selecting representative nodes from multiple blockchains. And the supervision-chain is responsible for integrity verification and recording the corresponding results. To achieve decentralized data integrity verification, we propose an integrity consensus protocol.

To record the verification results, we propose the block consensus protocol. And we utilize a block consensus committee to enhance the efficiency of the block consensus process. The security analysis and performance evaluation demonstrate the security and effectiveness of our proposed scheme.

In the future, we aim to investigate the verification scheme for ensuring the integrity of cross-chain data processing results in multi-chain architectures.

Acknowledgement. This work is supported by the Fundamental Research Funds for the Central Universities (Grant No. NS2023047), the National Key Research and Development Program of China (Grant No. 2020YFB1005500) and Postgraduate Research & Practice Innovation Program of NUAA (No. xcxjh20221616).

References

1. Ateniese, G., et al.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 598–609 (2007)
2. Ding, Q., Gao, S., Zhu, J., Yuan, C.: Permissioned blockchain-based double-layer framework for product traceability system. IEEE Access **8**, 6209–6225 (2019)
3. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30580-4_28
4. Jiang, Y., Wang, C., Wang, Y., Gao, L.: A cross-chain solution to integrating multiple blockchains for iot data management. Sensors **19**(9), 2042 (2019)
5. Jin, H., Jiang, H., Zhou, K.: Dynamic and public auditing with fair arbitration for cloud data. IEEE Trans. Cloud Comput. **6**(3), 680–693 (2016)
6. Kan, L., Wei, Y., Muhammad, A.H., Siyuan, W., Gao, L.C., Kai, H.: A multiple blockchains architecture on inter-blockchain communication. In: 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 139–145. IEEE (2018)
7. Kang, J., et al.: Communication-efficient and cross-chain empowered federated learning for artificial intelligence of things. IEEE Trans. Netw. Sci. Eng. **9**(5), 2966–2977 (2022)
8. Li, B., He, Q., Chen, F., Jin, H., Xiang, Y., Yang, Y.: Inspecting edge data integrity with aggregate signature in distributed edge computing environment. IEEE Trans. Cloud Comput. **10**(4), 2691–2703 (2021)
9. Li, B., He, Q., Yuan, L., Chen, F., Lyu, L., Yang, Y.: EdgeWatch: collaborative investigation of data integrity at the edge based on blockchain. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 3208–3218 (2022)
10. Li, J., Zhang, L., Liu, J.K., Qian, H., Dong, Z.: Privacy-preserving public auditing protocol for low-performance end devices in cloud. IEEE Trans. Inf. Forensics Secur. **11**(11), 2572–2583 (2016)
11. Liu, J., Huang, K., Rong, H., Wang, H., Xian, M.: Privacy-preserving public auditing for regenerating-code-based cloud storage. IEEE Trans. Inf. Forensics Secur. **10**(7), 1513–1528 (2015)
12. Menezes, A.: An introduction to pairing-based cryptography. Recent Trends Crypt. **477**, 47–65 (2009)

13. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th annual Symposium on Foundations of Computer Science (cat. No. 99CB37039), pp. 120–130. IEEE (1999)
14. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized Bus. Rev. 21260 (2008)
15. Qiao, L., Li, Y., Wang, F., Yang, B.: Lightweight integrity auditing of edge data for distributed edge computing scenarios. Ad Hoc Netw. **133**, 102906 (2022)
16. Wood, G.: Polkadot: vision for a heterogeneous multi-chain framework. White paper **21**(2327), 4662 (2016)
17. Xiao, X., Yu, Z., Xie, K., Guo, S., Xiong, A., Yan, Y.: A Multi-blockchain Architecture Supporting Cross-Blockchain Communication. In: Sun, X., Wang, J., Bertino, E. (eds.) ICAIS 2020. CCIS, vol. 1253, pp. 592–603. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-8086-4_56
18. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. arXiv preprint arXiv:1906.11078 (2019)
19. Yu, Y., et al.: Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans. Inf. Forensics Secur. **12**(4), 767–778 (2016)