Journal of King Saud University – Computer and Information Sciences 34 (2022) 7988–7999

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

DCIV: Decentralized cross-chain data integrity verification with blockchain

Jiajia Jiang [a], Yushu Zhang [a,*], Youwen Zhu [a], Xuewen Dong [b], Liangmin Wang [c], Yong Xiang [d]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[b] School of Computer Science and Technology, Xidian University, Xi'an, China
[c] School of Cyber Science and Engineering, Southeast University, Nanjing, China
[d] School of Information Technology, Deakin University, Melbourne, Australia

ARTICLE INFO

Article history:
Received 10 May 2022
Revised 14 June 2022
Accepted 19 July 2022
Available online 20 July 2022

Keywords:
Governing the chain by chain
Data integrity verification
Blockchain-enabled auditing
Cross chain
Smart contract

ABSTRACT

In recent years, blockchain holds promise to impact a wide range of application areas, but it still suffers from technical challenges such as security and scalability. The increase in the number of transactions puts blockchains under data storage pressure. The emergence of cross-chain technologies connects different blockchains and relieves the data storage pressure. However, the existing research generally focuses on the technical realization of cross chain, lacking in-depth research on consistency issues like data integrity verification of cross-chain interaction. In this paper, we propose a decentralized cross-chain data integrity verification scheme (DCIV) from the point of view of *governing the chain by chain*. We adopt *supervision chain* to audit the integrity of data in cross-chain interaction. We preprocess the off-chain original data in the form of Merkle tree. Before cross-chain interaction, we process the data with KZG polynomial commitment. During the auditing period, the supervision chain generates a challenge and verifies the integrity of cross-chain data. In particular, we add *audit digests* into the structure of transactions in blockchain to reduce the storage burden during auditing. Theoretical and experimental analyses demonstrate that DCIV can verify the integrity of data in cross-chain interaction, achieving secure and accurate cross-chain data sharing.

1. Introduction

Blockchain is gradually extending from small-scale applications to multiple fields, showing a bright development prospect. The increase in the number of transactions on the blockchain brings about pressure on blockchain data storage, which affects the efficiency of on-chain queries and calculations. To relieve the pressure of data storage, we can store data on different blockchains. When the blockchain needs to use data stored on other chains, the data can be called across chains.

Currently, the mainstream cross-chain technologies include notary schemes (Hope Bailie and Thomas, 2016), sidechains/relays (Back et al., 2014), and hash-locking (Deng et al., 2018). By connecting independent blockchain networks, cross chain realizes the interaction between chains, alleviating the data storage pressure of blockchains. However, most applications of these cross-chain technologies focus on asset transfer rather than information call. Since each blockchain has its internal security mechanism and does not participate in the consensus process of other blockchains, it is not easy for blockchains to judge the integrity of the received data. Therefore, measures should be taken to audit the integrity of data in cross-chain interaction.

Currently, verification methods for data integrity are mostly appeared in cloud storage (Wang et al., 2009; Wang et al., 2011; Wang et al., 2010; Ateniese et al., 2007; Juels and Kaliski, 2007). To relieve the burden of data storage management, users outsource local data to the cloud and access data remotely through the Internet (Yu et al., 2018). Users can verify whether the outsourced data are well preserved by the cloud without obtaining complete data information (Wang et al., 2009; Wang et al., 2011). However, the traditional verification schemes audit the integrity of the data stored by the data receiver, which does not apply to cross-chain scenarios. In cross-chain scenarios, we need to audit the behavior of the chain which transfers data in cross-chain interaction. As a

* Corresponding author.

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.07.015

result, traditional integrity verification methods cannot be directly applied to cross-chain scenarios in this paper.

Previous studies about integrity verification usually introduced third-party auditors to check the data integrity on the untrusted cloud (Wang et al., 2010; Wang et al., 2013; Barsoum and Hasan, 2015). But the use of third-party auditors in cross-chain scenarios weakens the decentralization of the blockchain system, and there is also the possibility that the auditors collude with the cloud to generate a biased verification result to deceive users. Therefore, many researchers have utilized blockchain for auditing (Zhang et al., 2021; Du et al., 2021). With the help of smart contracts and consensus mechanisms on blockchains, the laws and contracts for auditing blockchains can be transformed into simple and deterministic code-based rules, which are automatically executed by the underlying blockchain network, ensuring that the blockchain always outputs a fair audit result, which maintains the decentralization of the entire scheme in cross-chain interaction.

In this paper, we propose a decentralized cross-chain data integrity verification scheme (DCIV) by adopting a blockchain called *supervision chain* under the thought of *governing the chain by chain* to audit the process of cross-chain data interaction. BeDIV realizes decentralized and pluggable data integrity verification in cross-chain interaction. It processes the data stored on the chain and the data transmitted across the chain respectively. During the audit, the supervision chain completes the data integrity verification by comparing the received proof information, and always honestly returns unbiased audit results to the participants of the cross-chain interaction. In particular, we add *audit digests* into the structure of transactions in blockchain to reduce the storage burden during auditing. Our main contributions are summarized as follows:

(1) We propose a decentralized cross-chain data integrity verification scheme (DCIV) to audit the process of cross-chain data interaction, which enables an accurate and reliable audit of cross-chain calculation.
(2) We process the data stored on the chain and the data transmitted across the chain respectively to generate audit proof for integrity verification. The supervision chain can audit the integrity of data in cross-chain interaction without obtaining original data information.
(3) We add audit digests to modify the original structure of transactions in the blockchain. It makes the supervision chain independent of the cross-chain interaction, which increases the efficiency and feasibility of integrity verification.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 gives a brief introduction to the preliminaries covered in this paper. Section 4 defines the system model, threat model, and design goals. Section 5 presents the details of the proposed scheme. Section 6 conducts the security analysis. Section 7 evaluates the performance of the proposed cross-chain data integrity verification scheme. Section 8 concludes this paper and points out the future work.

2. Related Work

In this part, we summarize the existing cross-chain technologies and blockchain-enabled data integrity verification technologies.

2.1. Cross-chain Technology

The disconnection between blockchains greatly restricts the application and development of blockchains. The research on the interaction between chains is getting more and more attention.

Currently, the mainstream cross-chain technologies include notary schemes (Hope Bailie and Thomas, 2016), sidechains/relays (Back et al., 2014), and hash-locking (Deng et al., 2018). However, most applications of these cross-chain technologies focus on asset transfer rather than information call.

In 2016, Tsai et al. (Tsai et al., 2016) proposed a new scheme *Beihang Chain*. Through the Sharding strategy based on Account-Blockchain and TradingBlockchain, the scalability needs are realized. However, Beihang Chain is mainly aimed at financial business and does not involve legal verification support for complex transactions of diversified digital assets. In 2017, cross-chain projects Polkadot[1] and Cosmos[2] were proposed to build cross-chain basic platforms. Polkadot is designed to enable arbitrary message passing between parachains. Cosmos, on the other hand, focuses on cross-chain asset transfer, and its protocol is relatively simple. WeCross[3] proposed in 2019 supports multi-dimensional cross-chain interactions, aiming to promote cross-chain trust transferring and business cooperation across industries, institutions, and regions. Developed by *AntChain* in 2019, open data access trusted service (ODATS)[4] is an enterprise-level application-oriented service that supports trusted data interaction between homogeneous or heterogeneous blockchains. However, the methods above achieve cross-chain communication but do not consider data integrity issues of cross-chain interaction.

2.2. Data Integrity Verification Technologies

Currently, verification methods for data integrity are mostly appeared in cloud storage (Wang et al., 2009; Wang et al., 2011; Wang et al., 2010; Ateniese et al., 2007; Juels and Kaliski, 2007; Jalil et al., 2021). In 2007, Ateniese et al. proposed provable data possession (PDP) which uses the homomorphic verification tags of the RSA signature to verify the data integrity of the file (Ateniese et al., 2007). Although PDP can identify whether remote data are damaged, it cannot ensure availability. In the same year, Juels et al. considered how to recover damaged data and proposed sentinel-based proof of retrievability (POR), which can not only verify the data integrity but also recover damaged data to a certain extent (Juels and Kaliski, 2007). In 2009, Wang et al. introduced the public verification scheme, using a third-party auditor to verify the data integrity (Wang et al., 2009). In 2010, Wang et al. considered the issue of privacy protection by leveraging the random mask technology to effectively hide the data information in the proof returned by the cloud server. In this way, auditors cannot detect any details of the data (Wang et al., 2010). In 2018, Nayak et al. proposed a privacy-preserving provable data possession that supports multiple owners, data dynamics and batch verification (Nayak and Tripathy, 2021).

The basic idea of the above researches is based on the *challenge-response* mechanism. The cloud is required to answer the challenge of users or auditors. These schemes are effective in theoretical design, but there may be security threats such as data leakage or collusive attacks with the cloud (Ristenpart et al., 2009). Besides, third-party auditors may be malicious and collude with participants to give wrong audit results, making the audit results less credible.

With the emergence of blockchain, the decentralization and immutable characteristics of blockchain provide new solutions for the verification of data integrity. The decentralized characteristic allows the blockchain always provide fair and correct audit results based on the received audit proof, thereby enhancing the security of the system. In 2017, Sutton and Samvi described a blockchain-based verification approach to audit trails (Sutton and

[1] https://polkadot.network/
[2] https://cosmos.network/
[3] https://fintech.webank.com/wecross/
[4] https://www.aliyun.com/product/blockchainasaservice/baasodats

Samavi, 2017). They pay special attention to privacy, emphasizing the non-repudiation of privacy audit logs. In the same year, Liang et al. proposed a cloud environment security framework ProveChain, which audits the integrity of data through blockchain to improve the security of the cloud storage environment (Liang et al., 2017). In 2019, Zhang et al. proposed a certificateless public verification scheme against procrastinating auditors by the use of blockchain, which enables users to check whether auditors perform the verifications at the prescribed time (Zhang et al., 2021). In 2020, Wang et al. designed a blockchain-based private PDP scheme by leveraging RSA and quadratic residue group modulo the big composite number N (Wang et al., 2021). In 2021, Du et al. proposed a zero-knowledge storage auditing scheme with an on-chain check mechanism to address the storage freeriding security problem in the settings of rational actors (Du et al., 2021). However, these blockchain-based audits mostly consider the data integrity in cloud storage and do not consider the data integrity during cross-chain interaction.Table 1 summarizes the comparison of features for different existing auditing schemes and DCIV. Zhang et al. (Zhang et al., 2021) and Wang et al. (Wang et al., 2021) introduced blockchain to store audit information, but blockchain did not participate in the process of data auditing. Duan et al. (Duan et al., 2022) audited the integrity of data with blockchain but did not consider the auditing of data in cross-chain interaction. He et al. (He et al., 2021) realized the secure transmission of data in cross-chain interaction but did not give a specific cross-chain data audit method.

3. Priliminaries

3.1. Bilinear Pairing

A bilinear map (Menezes, 2005) is a map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{1}}$, where $\mathbb{G}$ and $\mathbb{G}_{\mathbb{1}}$ are two cyclic addictive groups generated with the same prime order p. A bilinear map has the following properties:

(1) Bilinearity: For all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}_{\mathbb{p}}$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
(2) Non-degeneracy: For $g \in \mathbb{G}$, $e(g, g) \neq 1$, where 1 is the identity element of group $\mathbb{G}_{\mathbb{1}}$.
(3) Computability: There exists an efficient algorithm that can compute e.

A bilinear mapping group that satisfies these properties is called a bilinear pair. Pairing based on cryptography 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. Classical applications such as the Boneh-Lynn-Shacham (BLS) (Boneh et al., 2001) signature scheme illustrates the use of bilinear pairs. The model established in this paper is based on bilinear pair-based cryptography verification technology, and mapping processing on the group, so it is necessary to introduce bilinear pairing here, which will be frequently used in the follow-up research work.

3.2. Polynomial Commitment

The polynomial commitment (Kate et al., 2010) is widely applied in many new cryptographic primitives (Lai and Malavolta, 2019). It can bind multiple data blocks to a single commitment, which allows validation to finish with fewer points challenged.

In this paper, we choose the polynomial commitment proposed by Kate et al. (KZG) to verify the integrity of data stored by the receiver in cross-chain interaction. KZG uses a degree n polynomial $f(X)$ to commit a vector, based on the *q-BSDH* assumption (Boneh and Boyen, 2004; Goyal, 2007). Compared to other commitment schemes, KZG generates a succinct evaluation proof of $f(a)$ for any a, which makes it easy to verify due to the polynomial remainder theorem, and meanwhile, $f(a)$ encapsulates a vector of the message. The only overhead associated with this feature is a set of system parameters g^{τ^i} generated during the setup phase, where $i \in [0, n], g$ is a group generator, and τ is a trapdoor. KZG commitment consists of three procedures as follows.

- **Committing.** Let $f(X)$ be a polynomial of degree n comprised of coeffifficients $c_0, c_1, \ldots, c_n \in \mathbb{Z}_{\mathbb{p}}$. Compute $cm = \prod_{i=0}^{n} \left(g^{\tau^i}\right)^{c_i} = g^{f(\tau)}$.
- **Witness creation.** To create a witness for the commitment, the prover can compute a evaluation proof such that $f(X) = y$. Given the polynomial remainder theorem, the sufficient and necessary condition for $f(X) = y$ is that $\exists h(X)$ such that $f(X) - y = h(X)(X - a)$, the prover can simply use the polynomial commitment to compute $h(X) : \pi = g^{h(\tau)}$.
- **Evaluation verification.** To verify the evaluation proof π, check whether the following bilinear pairing relation holds: $e(cm \cdot g^{-y}, g) = e(\pi, g^{\tau - a})$.

When the sender transfers data to a receiver, we will leverage the **committing** phase for data preprocessing, the **witness creation** phase for proof generation while being challenged, and the **evaluation verification** phase for proof verification.

4. Models

4.1. System Model

In this paper, we propose DCIV which considers the data integrity verification in cross-chain interaction. We apply Hyperledger Fabric to build three consortium blockchains: an access chain (*AC*), a receiving chain (*RC*), and a supervision chain (*SC*). As is shown in Fig. 1, a data owner (*DO*) participates in the model to preprocess data off-chain to generate audit information.

Data owner (*DO*) has the original data F which will be written into *AC* for storage. After preprocessing, it generates audit metadata of F and honestly sends F and audit metadata to *AC* for storage.

Access chain (*AC*) is the data storage platform that receives and honestly stores data F from *DO*. When cross-chain interaction occurs, it generates tags and commitments of the data to be transferred $\widetilde{F}$ and sends commitments and data $\widetilde{F}$ to *RC*. When *SC* initiates an audit challenge, it generates an audit proof based on the transferred data $\widetilde{F}$.

Receiving chain (*RC*) is the data receiving platform of cross-chain interaction that receives data $\widetilde{F}$ and audit metadata from *AC*. When *SC* initiates an audit challenge, *RC* generates an audit proof based on the received data.

Supervision chain (*SC*) is an auditing platform for the integrity of data in cross-chain interaction, which is jointly established by national regulatory authorities and centralized institutions in various fields. When it initiates an audit challenge, *RC* and *AC* generate proofs based on the challenge and send proofs to *SC* faithfully. After receiving proofs, *SC* completes the integrity verification to produce a reasonable judgment of this cross-chain interaction.

4.2. Threat Model

Off-chain *DO* has the original data, which are used to compare and verify the integrity of data transmitted by *AC*. Since we mainly study the data integrity verification of cross-chain interaction, we assume that *DO* is completely honest in this scheme.

Table 1
Comparisons with Existing Related Work.

Schemes	Zhang et al., 2021	Wang et al., 2021	Duan et al., 2022	He et al., 2021	DCIV
Transferring Integrity	No	No	No	Yes	Yes
Storage Integrity	Yes	Yes	Yes	No	Yes
Cross-Chain Interaction	No	No	No	Yes	Yes
Decentralized Auditing	No	No	Yes	No	Yes
Blockless Verification	Yes	Yes	Yes	No	Yes
Scalability	No	Yes	No	No	Yes
Unforgeability	Yes	Yes	Yes	No	Yes
Privacy Preserving	Yes	Yes	Yes	Yes	Yes

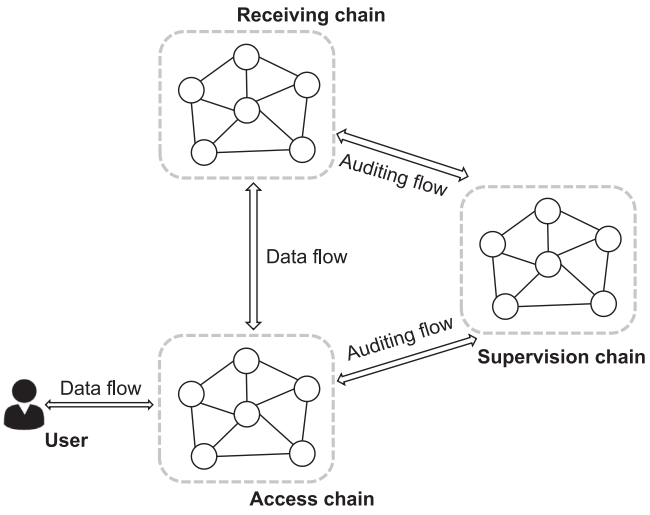


Fig. 1. System model.

(1) *AC* may provide *RC* with incomplete data (not the data sent by *DO*) for saving communication cost or cheating *RC*, trying to mislead *RC* that the data it received are integrate.
(2) *RC* may store the data from *AC* incompletely for profits and try to deceive *SC* that the data provided by *AC* are incomplete.
(3) Information leakage may occur when *RC* sends the proof to *SC*. Off-chain adversaries may be able to extract some privacy information from the proof *RC* generates.

4.3. Desirable Goals

Based on the analysis of the proposed cross-chain interaction model, we propose the following goals to ensure the integrity of data in cross-chain interaction.

- **The integrity of data.** *AC* can pass the data integrity verification only if it sends complete data to *RC*.
- **Pluggable supervision.** *SC* is independent of *AC* and *RC* in a pluggable form. The cross-chain interaction does not require the involvement of *SC*. Only when the cross-chain interaction needs to be audited, *SC* involves and conducts the verification of the interaction.
- **Low-overhead supervision.** *SC* gets audit proofs from *AC* and *RC*. It does not store any audit information of *AC* or *RC* in the procedure of cross-chain interaction.
- **Public auditability.** Any third party (whether in the form of a third-party auditor or blockchain) can perform the data integrity verification between chains without obtaining the complete data.
- **Data privacy.** Off-chain participants outside the system cannot extract any privacy information from the proof of *RC*.

5. The Proposed Scheme

In this section, we describe the details of DCIV about how to verify the integrity of data in cross-chain interaction. To start with, we give an overview of the verification framework. Next, we describe a new audit information storage policy that leads to low cost and pluggable auditing of *SC*. Then, we present methods about how to process and verify the integrity of the data transferred by *AC* and the data stored by *RC*. Finally, we describe the whole procedure of DCIV.

5.1. Overview

DCIV focuses on auditing the data transferred by *AC* and the data stored in *RC*. To audit the integrity of the data transferred by AC, DO preprocesses data F and generates a Merkle tree before storing the data into *AC*. To audit the integrity of the data stored in RC, AC generates a KZG polynomial commitment for data $\widetilde{F}$ before transferring them. The generated audit metadata are stored in the audit digests of transactions in the blockchain. When *SC* initiates an audit challenge, *AC* and *RC* generate a proof respectively and send them to *SC*. *SC* verifies the integrity of the proof according to the determined smart contract *S* installed on *SC*.

We install smart contracts for auditing on three blockchains. *AC* installs the smart contract S_a, which can process the data transferred across the chain and generate the integrity audit proof. *RC* installs the smart contract S_r that can generate proof for integrity audit. *SC* installs the smart contract *S*, which is responsible for generating an audit challenge and verifying the integrity of the received proofs. The overview of DCIV is shown in Fig. 2. The main notations used in our scheme are displayed in Table 2 with brief explanations attached for ease of presentation.

5.2. Audit Information Storage Policy

Traditional data integrity verification schemes for cloud storage mostly rely on third-party auditors. When data are stored in the cloud, audit metadata generated from data are sent to auditors for storage. The storage cost on auditors increases linearly with the size of audit metadata.

In blockchain systems, transactions that generate at regular intervals are packaged into a block. If we leverage the same strategy in DCIV, the storage and transfer of data on-chain are always accompanied by a transaction on SC for storing audit metadata. However, *SC* audits cross-chain interactions among multiple blockchains at the same time. If the audit metadata are written into *SC* every time the cross-chain interaction happens, the length of *SC* may be much longer than that of *AC* and *RC*, bringing excessive storage pressure on *SC*.

In this paper, we modify the data structure of transactions in the blockchain by adding audit digests, which store the audit metadata of the original data. The new structure of the transaction is shown in Fig. 3. The introduction of audit digests makes *SC* inde-

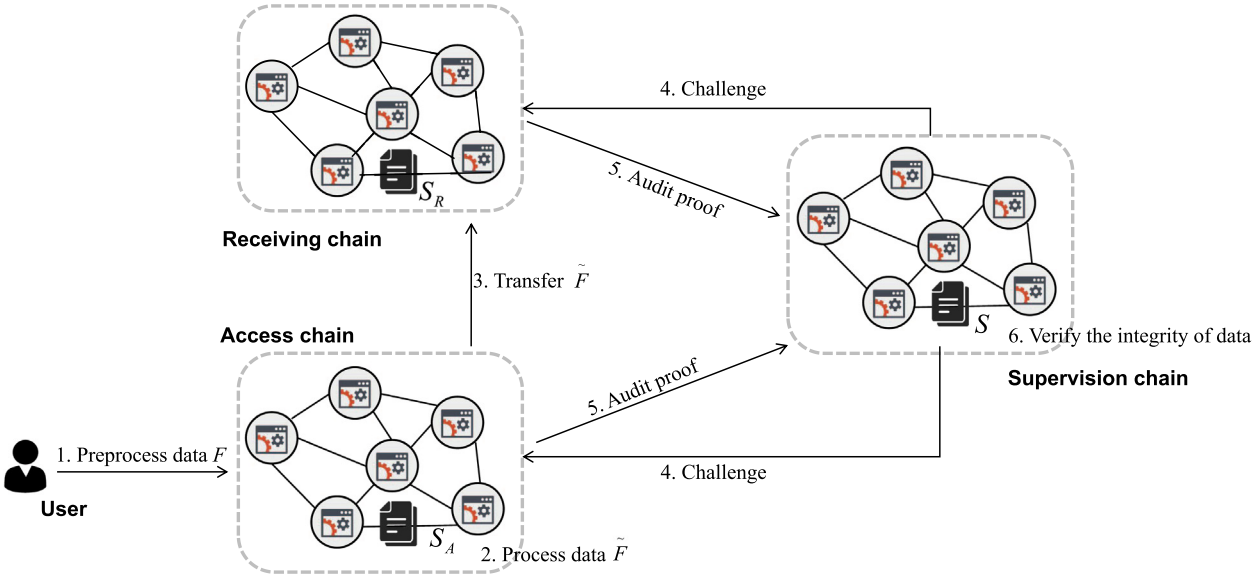


Fig. 2. Overview of the proposed scheme.

Table 2
Notations and Descriptions.

Notations	Descriptions
F	the data stored on *AC*
$\tilde{F}$	the data transferred to *RC*
B	the number of blocks in data
S	the number of segments in a block
C	the number of challenged segments
$d_i, d_{i,j}$	the ith block and the jth segment of ith block
κ	the index of node in the Merkle tree
s, n	the members of sk
λ, η	the members of pk
g_1, g_2	the group generators of $\mathbb{G}$
σ	the aggregation of challenged segments
y	the polynomial result of the challenge
ω	the witness created by polynomial remainder
M	the parameter for encrypting y
y'	the encrypted result of y

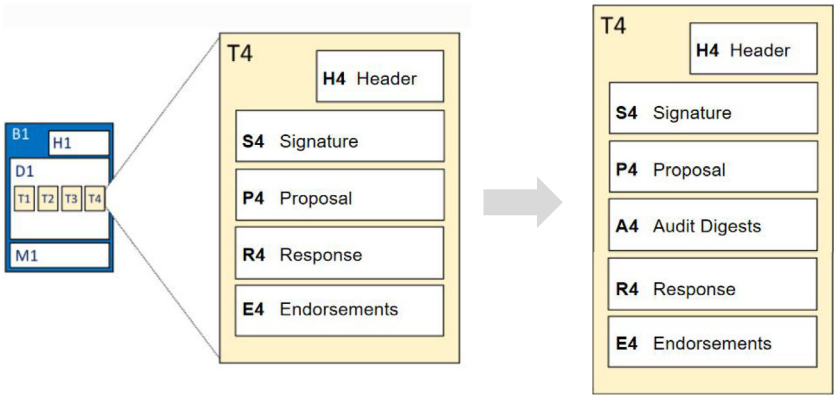
pendent of cross-chain interaction, which leads to a flexible and efficient audit procedure. The audit metadata of data are stored into audit digests of the corresponding transaction and then packed into the block. During the procedure of audit, *SC* randomly generates a challenge and sends it to *AC* and *RC*. *AC* and *RC* generate proofs based on audit digests stored in their respective transactions and return them to *SC*. *SC* receives proofs from *AC* and *RC* and verifies the integrity of data without any redundant data storage operations.

5.3. The Integrity of AC

To verify the integrity of data in cross-chain interaction, it is necessary to ensure that the data transferred by *AC* are complete, that is, consistent with the original data provided by *DO*.

We divide data F into B blocks and each block is then divided into S segments, where the size of block is 4 KB or 6 KB, $\{d_{ij}\} \in \mathbb{Z}_p, \mathbb{i} \in [\mathbb{0}, \mathbb{B}-\mathbb{1}], \mathbb{j} \in [\mathbb{0}, \mathbb{S}-\mathbb{1}]$. Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p and H be a SHA256 (N.I. of Standards and Technology, 2002) cryptographic hash function.

Off-chain preprocessing. *BuildMT()*. *DO* preprocesses data F in the form of a Merkle tree. The jth segment of each block is aggregated as a tag to generate a leaf node n_k. Instead of generating nodes by the original data directly, we generate tags of the original data and apply tags to compute leaf nodes to make the verification



(a) Original transaction (b) Proposed transaction

Fig. 3. Change of transaction data structure.

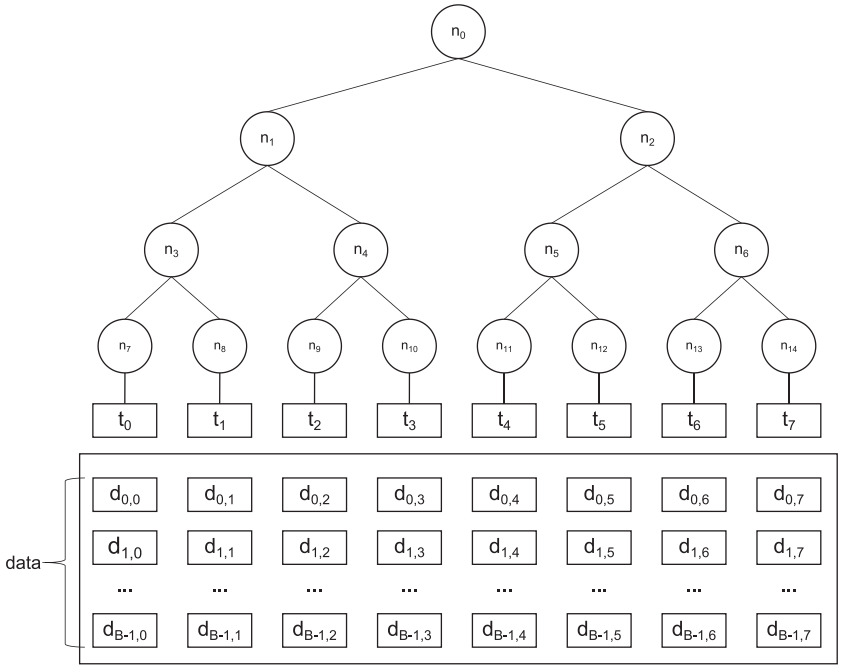


Fig. 4. The rule of building Merkle tree.

convenient. The Merkle tree is generated by S leaf nodes. Fig. 4 shows the rule of how to build the Merkle tree in DCIV.

(1) Randomly select a generator $g_1 \in \mathbb{G}$.
(2) For jth segment in B blocks, compute tags $t_j = \prod_{i=0}^{B-1} g_1^{id_{ij}}$, where $j \in [0, S-1]$.
(3) Build a Merkle tree MT with S leaf nodes $n_\kappa = (l_\kappa, h_\kappa)$, where κ is the index of nodes in MT, l_κ is the number of leaf nodes that can be reached from the κth node, and h_κ is a hash value. For the jth leaf node n_{S+i-1}, set $l_{S+j-1} = 1$, $h_{S+j-1} = H(t_j)$. For each non-leaf node n_κ, set $l_\kappa = l_{2\kappa} + l_{2\kappa+1}, h_\kappa = H(n_{2\kappa} \| n_{2\kappa+1})$
(4) Send data F, Merkle tree MT, and generator g_1 to AC.

On-chain operation. *AC.GenMeta*(). AC honestly receives data F from DO. Since the database used in Hyperledger Fabric is Level DB or Couch DB, both of which are non-relational databases based on key-value pairs,AC stores F in the format of $(key = H(F), value = F)$. Audit metadata (MT, g_1) are written into the audit digests of data storage transaction *Tx 1* of AC and packed into the block to achieve immutable storage. For the convenience of distinction, we define the data transferred by AC as $\widetilde{F}$, which is the data stored on AC. Before performing cross-chain interaction, AC needs to process data $\widetilde{F}$. The consistency of $\widetilde{F}$ and F means that AC has completely transferred all the data to RC. The details of the processing of $\widetilde{F}$ are as follows:

(1) Divide $\widetilde{F}$ into B blocks and each block is then divided into S segments, which is the same way as for F.
(2) Compute $T_j = \prod_{i=0}^{B-1} \widetilde{g_1^{id_{ij}}}$ with $\widetilde{F}$, where $\left\{\tilde{d}_{ij}\right\}$ is the segment of $\widetilde{F}$. The rule of computing tags is the same as t_j mentioned in *BuildMT()*.
(3) Store $\{T_j\}_{j=0}^{S-1}$ in the audit digests of data processing transaction *Tx 2* of AC.

Proof generation. *AC.GenProof*(). SC sends challenge $chal = \{(j_c, r_c)\}_{c=0}^{C-1}$ to AC, where j_c indicates the index of challenged segment, $r_c \in \mathbb{Z}_{\mathbb{p}}$ is a coefficient randomly generated by RC, and C is the number of segments to be challenged. The details of this procedure are as follows.

(1) For each challenged segment, select the siblings from the leaf node to the root node as θ_c of the path which are stored in the audit digests of *Tx 1* in AC.
(2) Select tags $\{T_{j_c}\}_{c=0}^{C-1}$ from *Tx*2.
(3) Send $\pi = \left(\{T_{j_c}\}_{c=0}^{C-1}, v1, \{\theta_c\}_{c=0}^{C-1}\right)$ as proof to SC, where *v1* is the root node of MT.

Integrity verification. *SC.VerifyAC()*. SC receives the proof π from AC and parses it as $\{T_{j_c}\}_{c=0}^{C-1}, v1, \{\theta_c\}_{c=0}^{C-1}$. Then SC performs the verification procedure with the smart contract C installed on SC. The details of this procedure are as follows.

(1) For each T_{j_c}, compute $h_{j_c} = H(T_{j_c})$.
(2) Compute the new root with h_{j_c} and θ_c as the building algorithm of the Merkle tree in Fig. 4. Return 0 if the new root is not equal to *v1*. Return 1 if all challenged nodes pass the verification.

5.4. The Integrity of RC

RC receives and stores $\widetilde{F}$ transferred from AC. We can judge the integrity of cross-chain interaction by auditing whether the data on RC are consistent with the original data from DO. If RC does not store data faithfully, SC may consider the data provided by AC to be incomplete. Therefore, we give methods to verify the integrity of data stored on RC, ensuring that RC faithfully stores data transferred from AC.

Key generation. *AC.GenKey*(). Before transferring $\widetilde{F}$ to RC, AC proposes a transaction *Tx 3* by invoking smart contract S_a to generate the secret key sk and the public key pk for data processing. Since the data of the consortium chain are only accessible to the organizations in the consortium and their users, the security of the private key can be guaranteed.

(1) Randomly select a generator $g_2 \in \mathbb{G}$.

(2) Compute $sk = (s, n)$, $pk = (\lambda = g_2^s, \eta = g_2^{sn}, \left\{g_2^{n^j}\right\}_{j=0}^{S-1})$, where $s \in \mathbb{Z}_{\mathbb{p}}$ is the signature of $Tx1$ and $n \in \mathbb{Z}_{\mathbb{p}}$ is the nonce of $Tx1$, which are all private to other participants except AC itself.

(3) Store sk and pk in transaction $Tx\ 3$.

Tags generation. $AC.GenTags()$. AC generate tags based on KZG polynomial commitment with sk and $\{T_j\}_{j=0}^{S-1}$.

(1) Divide $\widetilde{F}$ into B blocks and each block is then divided into S segments, which is the same way as the rule mentioned in 5.3 for F.

(2) Compute $\sigma_j = \left(T_j \cdot g_2^{f_{\overrightarrow{\widetilde{d}_j}}(n)}\right)^s$ to generate S commitments, where $j \in [0, S-1]$ and $f_{\overrightarrow{\widetilde{d}_j}}$ is a polynomial comprised of $\left\{\widetilde{d}_{ij}\right\}_{i=0}^{B-1} \in \mathbb{Z}_{\mathbb{p}}$ as coefficients to commit the data $\widetilde{F}$.

(3) Generate tags by signing commitments with sk which is private so that others can verify tags with pk.

(4) Send data $\widetilde{F}, pk, g_2$ and $\{\sigma_j\}_{j=0}^{S-1}$ to RC.

Proof generation. $RC.GenProof()$. SC sends challenge $chal = \{(j_c, r_c, q)\}_{c=0}^{C-1}$ to AC and RC, where j_c indicates the index of challenged segment, $r_c \in \mathbb{Z}_{\mathbb{p}}$ is a coefficient randomly generated by SC, $q \in \mathbb{Z}_{\mathbb{p}}$ is a coefficient for commitment, and C is the number to be challenged. The details of this procedure are as follows:

(1) Compute $\sigma = \prod_{(j_c, r_c) \in chal} \sigma_{j_c}^{r_c}$.

(2) Compute $y = P(q) \in \mathbb{Z}_{\mathbb{p}}$ as a polynomial commitment, where $P(X) = \sum_{(j_c, r_c) \in chal} r_c f_{\overrightarrow{\widetilde{d}_j}}(X)$.

(3) Compute a witness $\omega = g_2^{Q(n)} \in \mathbb{G}$, where $Q(X) = \frac{P(X) - P(q)}{X - q}$.

Since a polynomial can eventually be recovered through the *lagrange polynomial interpolation*, malicious adversaries may extract private data contents from the polynomial commitments within the audit trails. Given this, we encrypt polynomial $P(X)$ by leveraging the random masking technique mentioned in (Wang et al., 2010) to prevent privacy disclosure in the procedure of proof generation. The details are as follows.

(4) Generate a random parameter $\varepsilon \in \mathbb{Z}_{\mathbb{p}}$ and select a random oracle $H() : \mathbb{G} \to \mathbb{Z}_{\mathbb{p}}$.

(5) Compute $M = \lambda^{\varepsilon}$ and get a hiding parameter $\chi = H(M)$.

(6) Compute new $y' = \chi y + \varepsilon \in \mathbb{Z}_{\mathbb{p}}$ based on the parameters above.

(7) Sends $\Phi = (M, \sigma, y', \omega)$ instead of (σ, y, ω) as proof to SC.

Integrity verification. $SC.VerifyRC()$. SC receives Φ from RC and verifies the integrity of data stored in RC. The details of this procedure are as follows:

(1) AC computes $\rho = \prod_{(j_c, r_c) \in chal} T_{j_c}^{r_c}$ and sends it to SC.

(2) Combined with the polynomial evaluation verification and the BLS signature verification, S on SC automatically runs the verification as follows:

$$e(M \cdot \sigma^{\chi}, g_2) \cdot e\left(g_2^{-y'}, \lambda\right) \stackrel{?}{=} e(\rho^{\chi}, \lambda) \cdot e(\omega^{\chi}, \eta \cdot \lambda^{-q}) \quad (1)$$

The above verification equation requires S to perform four times bilinear pairing before deriving a final verdict.

If (1) holds, it means that the data stored on RC are consistent with the data transferred by AC; otherwise, the data stored on RC are incomplete.

5.5. The Integrity of Cross-chain Interaction

Now, we give a detailed description of the integrity verification of data in cross-chain interaction, which consists of **off-chain preprocessing**, **on-*AC* processing**, **challenge generation**, **proof generation**, and **integrity verification**. We summarize the four transactions involved in DCIV, as shown in Table 3. The interaction among four participants is depicted in Fig. 5.

Off-chain preprocessing. Before sending data F to AC, DO preprocesses F to generate the audit metadata for integrity verification. The details of this procedure are as follows:

(1) Divide F into $\{d_{ij}\}$, where $i \in [0, B-1], j \in [0, S-1]$.

(2) Call the data preprocessing method *BuildMT()* to build a Merkle tree of F.

(3) Send element g_1, Merkle tree MT, and data F to AC.

On-*AC* processing. AC receives and honestly stores data F from DO. Before performing cross-chain interaction, AC processes $\widetilde{F}$, so as to verify the integrity of the data provided by AC. The details of this procedure are as follows:

(1) Initiate a transaction $Tx\ 1$ to store F in the format of $(key = H(F), value = F)$ and write (g_1, MT) to the audit digests of $Tx\ 1$.

(2) Call $AC.GenKey()$ to obtain keys (sk, pk).

(3) Call $AC.GenTags()$ to obtain tags $\{\sigma_j\}_{j=0}^{S-1}$ of $\widetilde{F}$ before sending $\widetilde{F}$ to RC.

(4) Call $AC.GenMeta()$ to compute $\{T_j\}_{j=0}^{S-1}$ based on $\widetilde{F}$.

(5) Write $\{T_j\}_{j=0}^{S-1}$ to the audit digests of $Tx\ 2$ of AC and (sk, pk) to the audit digests of $Tx\ 3$.

(6) Send $\widetilde{F}, g_2$, and $\{\sigma_j\}_{j=0}^{S-1}$ to RC.

(7) Write g_2 and $\{\sigma_j\}_{j=0}^{S-1}$ to the audit digests of transaction $Tx\ 4$ of RC.

Challenge generation. When SC wants to conduct the auditing operation, it invokes smart contract S to randomly generate a challenge $chal$ and sends it to AC and RC, waiting for AC and RC to return proofs for verification. The details of this procedure are as follows.

(1) Generate a challenge $chal$ in which C segments will be challenged, where $C \leqslant S$. Randomly choose C indexes $\{j_c\}_{c=0}^{C-1}$, C elements $\{r_c\}_{c=0}^{C-1}$ where $r_c \in \mathbb{Z}_{\mathbb{p}}$, and an element $q \in \mathbb{Z}_{\mathbb{p}}$.

(2) Send challenge $chal$=$(\{j_c\}_{c=0}^{C-1}, \{r_c\}_{c=0}^{C-1}, q)$ to AC and RC.

Proof generation. AC and RC generate proofs based on the challenge $chal$ received from SC.

(1) AC calls $AC.GenProof()$ to generate proof $\pi = (\{T_{j_c}\}_{c=0}^{C-1}, v1, \{\theta_c\}_{c=0}^{C-1})$ and sends π to SC.

(2) RC calls $RC.GenProof()$ to generate proof $\Phi = (M, \sigma, y', \omega)$ and sends Φ to SC.

(3) Send π, Φ, and pk to SC.

Integrity verification. SC receives proofs π and Φ and parses them as $\{T_{j_c}\}_{c=0}^{C-1}$, $v1$, $\{\theta_c\}_{c=0}^{C-1}$, M, σ, y', ω for verification.

Table 3
Notations of Transactions.

Name	Description	Audit digests
Tx 1	store data *F* into *AC*	g_1, *MT*
Tx 2	compute T_j with $\widetilde{F}$	$\{T_j\}_{j=0}^{S-1}$
Tx 3	generate keys for $\widetilde{F}$	*sk*, *pk*
Tx 4	process $\widetilde{F}$	g_2, $\{\sigma_j\}_{j=0}^{S-1}$

(1) Call *SC.VerifyAC()* to check the proof π, return 0 if the reconstructed root is not equal to *v1*, which means data $\widetilde{F}$ transferred to *RC* are not consistent with the original data *F* of *DO*.
(2) Call *SC.VerifyRC()* to check the proof Φ, return 0 if the equation is false, which means *RC* does not stored the data from *AC* faithfully.

If all verifications return 1, *SC* outputs a result that the data of the interaction between *AC* and *RC* are integrate.

6. Security Analysis

In this section, we briefly evaluate the security of DCIV to ensure the integrity and privacy of DCIV. The integrity of DCIV is guaranteed if *AC* indeed transfers complete data to *RC* and *RC* faithfully stores data from *AC*. In addition, DCIV needs to ensure that the audit results on *SC* are accurate if the data transferred are incomplete.

6.1. The Integrity of Data on AC

The integrity of data on *AC* realizes if *AC* can generate a correct proof of the challenged tags, and the contract *S* installed on *SC* can rebuild the correct path from the challenged leaf nodes to the root node of the Merkle tree. *AC* cannot generate a proof that can pass the verification on *SC* when it transfers incomplete data to *RC*.

Theorem 1. *DCIV guarantees the data integrity of cross-chain interaction if the SHA256 hash function is collision-resistant and the discrete logarithm problem is hard in* $\mathbb{G}$.

Proof: Since the SHA256 hash function is collision-resistant, adversaries could not generate a new T_1^* where $H(T_1^*) = H(T_1)$; otherwise, a collision is found immediately. Similarly, adversaries could not find two values n_2^* and n_3^* that make $H(n_2^* \| n_3^*) = h_1$. Besides, l_κ in node n_κ marks the position of the node in the Merkle tree, which is difficult for adversaries to forge wrong positions.

6.2. The Integrity of Data on RC

The integrity of data on *RC* realizes if *RC* can generate a correct proof of the challenged segments of data, and the contract *S* installed on *SC* can always pass (1). By simplifying (1), it can be seen that the contract *S* always makes a correct integrity verification. The details are as follows:

$$\begin{aligned} LHS &= e\left(g_2^{s\varepsilon}\cdot\sigma^{\chi},g_2\right)\cdot e\left(g_2^{-(\chi y+\varepsilon)},g_2^{s}\right)\\ &= e(g_2,g_2)^{-\chi y s}\cdot e(\sigma^{\chi},g_2)\\ &= e(g_2,g_2)^{-\chi y s}\cdot e\left(\prod_{chal}T_{j_c}^{r_c s\chi},g_2\right)\cdot e\left(\prod_{chal}g^{s\chi r_c f_{\overrightarrow{d_j}}^{(n)}},g_2\right)\\ &= e(g_2,g_2)^{-\chi y s}\cdot e\left(\prod_{chal}T_{j_c}^{r_c s\chi},g_2\right)\cdot e\left(g^{s\chi\sum_{chal}r_c f_{\overrightarrow{d_j}}^{(n)}},g_2\right)\\ &= e(g_2,g_2)^{\chi s[P(n)-P(q)]}\cdot e\left(\prod_{chal}T_{j_c}^{r_c s\chi},g_2\right)\\ &= RHS \end{aligned} \tag{2}$$

Theorem 2. *Valid storage proof* $\Phi = (M, \sigma, y', \omega)$ *can only be generated when RC faithfully stores the data, if the signature scheme used for tags is existentially unforgeable, the t-SDH and the t-BSDH assumptions hold, and the CDH problem is hard.*

Proof: The proof of Theorem 2 is similar to the security proof in (Yuan and Yu, 2013). Suppose a probabilistic polynomial time adversary can generate a forged proof $(M,\sigma^*,y r^*,\omega^*),(M,\sigma^*,y r^*,\omega^*) \neq (M,\sigma,y',\omega)$ after receiving a challenge and try to pass the verification on *SC*. We can get the following two equations:

$$e(M\cdot\sigma^{\chi},g_2)\cdot e\left(g_2^{-y'},\lambda\right)=e(\rho^{\chi},\lambda)\cdot e(\omega^{\chi},\eta\cdot\lambda^{-q}) \tag{3}$$

$$e(M\cdot\sigma^{*\chi},g_2)\cdot e\left(g_2^{-y r^*},\lambda\right)=e(\rho^{\chi},\lambda)\cdot e(\omega^{*\chi},\eta\cdot\lambda^{-q}) \tag{4}$$

Dividing (3) with (4), we obtain:

$$\left(\frac{e(\sigma,g_2)}{e(\sigma^*,g_2)}\right)\cdot e(g_2,\lambda)^{y^*-y}=\left(\frac{e(\omega,g_2)}{e(\omega^*,g_2)}\right)^{s(n-q)} \tag{5}$$

Now we conduct a case analysis on whether $\sigma=\sigma^*$.

Case 1: $\sigma\neq\sigma^*$.

As $\left(\frac{e(\omega,g_2)}{e(\omega^*,g_2)}\right)^{s(n-q)}$, $e(g_2,\lambda)^{y^*-y}$, and $e(\sigma,g_2)$ are known to adversaries, we rewrite (5) as

$$e(\sigma^*,g_2)=e(\sigma,g_2)\cdot\Upsilon \tag{6}$$

$$e(\sigma^*,g_2)=e\left(\prod_{(j_c,r_c)\in chal}T_j^{sr_c},g_2\right)\cdot e\left(g_2^{sP(n)},g_2\right)\cdot\Upsilon \tag{7}$$

where $\Upsilon=e(g_2,\lambda)^{y^*-y}/\left(\frac{e(\omega,g_2)}{e(\omega^*,g_2)}\right)^{s(n-q)}$ as a known value to adversaries.

Since the *CDH* problem is hard to solve, if any probabilistic polynomial time adversary $\mathcal{A}$ can find σ^* with non-negligible probability and make (7) hold, we can construct a new algorithm

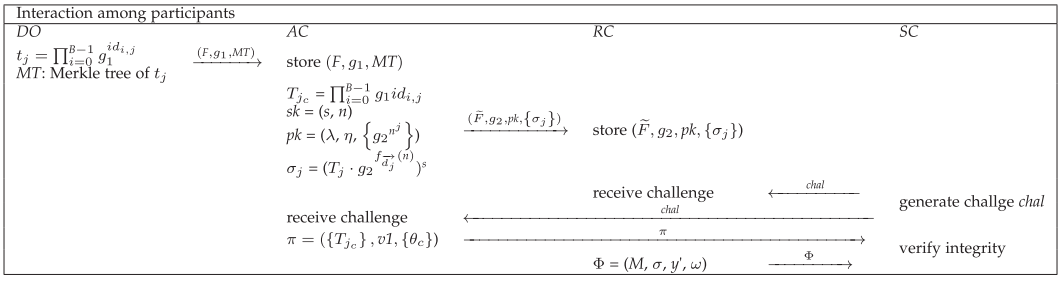


Fig. 5. The interaction among participants.

$\mathscr{B}$ to solve the *CDH* problem. Specifically, given $\sigma^* \neq \sigma$ provided by $\mathscr{A}$, which makes (7) hold, $\mathscr{B}$ can easily extract $g_2^{sP(n)}$. With the given information, $\mathscr{B}$ can get $g_2^s, g_2^{sP(n)}$ where s and $P(n)$ are unknown, and thus solve the *CDH* problem. As a result, there is no probabilistic polynomial time adversary that can find a valid forged proof $(M, \sigma^*, y^*, \omega^*) \neq (M, \sigma, y', \omega)$ and $\sigma^* \neq \sigma$ with non-negligible probability.

Case 2: $y \neq y^*$.

In this case, since $\sigma = \sigma^*$, we rewrite (5) as:

$$e(g_2, \lambda)^{s(y^*-y)} = \left(\frac{e(\omega, g_2)}{e(\omega^*, g_2)}\right)^{s(n-q)} \tag{8}$$

From (8) and $y \neq y^*$, we can infer that $n \neq q$. In this case, we show how to construct an algorithm $\mathscr{B}$ that can break the *t-BSDH* assumption with a valid solution $\left(-q, \left(\frac{e(\omega,\lambda)}{e(\omega^*,\lambda)}\right)^{\frac{1}{y^*-y}}\right)$.

We denote ω as $g_2^{Q(n)}$ and ω^* as $g_2^{Q(n)*}$, and then we can rewrite (8) as:

$$e(g_2, \lambda)^{s(y^*-y)} = \left(\frac{e\left(g_2^{Q(n)}, g_2\right)}{e\left(g_2^{Q(n)*}, g_2\right)}\right)^{s(n-q)} \tag{9}$$

$$(n-q)Q(n) + y = (n-q)Q(n) * + y^* \tag{10}$$

$$\frac{Q(n) - Q(n)*}{y^* - y} = \frac{1}{n-q} \tag{11}$$

Therefore, algorithm $\mathscr{B}$ can compute

$$\left(\frac{e(\omega, \lambda)}{e(\omega^*, \lambda)}\right)^{\frac{1}{y^*-y}} = e(g_2, \lambda)^{\frac{Q(n)-Q(n)*}{y^*-y}} = e(g_2, \lambda)^{\frac{1}{n-q}} \tag{12}$$

and return $(-q, e(g_2, \lambda)^{\frac{1}{n-q}})$ as a solution for *t-BSDH* problem. It is easy to see that the probability of success in solving the *t-BSDH* problem is the same as that of success in adversaries, and the time required is a small constant larger than that by adversaries by a small constant.

Case3 : $\omega \neq \omega^*$ In this case, since $\sigma = \sigma^*, y = y^*$, we rewrite (8) as:

$$\left(\frac{e(\omega, g_2)}{e(\omega^*, g_2)}\right)^{s(n-q)} = 1 \tag{13}$$

Since $\omega \neq \omega^*$, i.e., $\frac{e(\omega,g_2)}{e(\omega^*,g_2)} \neq 1$, and $s \neq 0$, we can infer $n = q$ from (13). In DCIV, q is known to adversaries. Thus, adversaries can find $sk = n$ and solve the *t-SDH* problem with solution $e(q, g_2^{\frac{1}{n+q}})$ by given system parameters. Therefore, no valid forged proof $(M, \sigma^*, y^*, \omega^*) \neq (M, \sigma, y', \omega)$ and $\omega \neq \omega^*$ can be found by the probabilistic polynomial time adversary $\mathscr{A}$ with non-negligible probability.

As a result, Theorem 2 is proved properly.

Theorem 3. Suppose the data are divided into n blocks and sent to *RC* for interaction. If d blocks are modified and c blocks are challenged, the modified blocks can be detected with the probability

$$P_X = 1 - \frac{(n-c)!(n-d)!}{n!(n-c-d)!} \tag{14}$$

Proof: Define X as a discrete random variable which is the number of blocks to be challenged that matches the modified blocks. Define P_X as the probability that at least one of the challenged blocks matches one of the modified blocks.

$$\begin{aligned} P_X &= P_X \geqslant 1 \\ &= 1 - P\{X = 0\} \\ &= 1 - \tfrac{n-d}{n}\tfrac{n-1-d}{n-1}\cdots\tfrac{n-c+1-d}{n-c+1} \\ &= 1 - \tfrac{(n-c)!(n-d)!}{n!(n-c-d)!} \end{aligned} \tag{15}$$

When the transferred data are incomplete or modified, it is crucial to specify the number of queried segments to guarantee a high verification confidence level (Ateniese et al., 2007). The most appropriate challenge number is 300 and 460, which can make verification assurance of 95% and 99% if only 1% of the entire data are modified.

7. Performance Evaluation

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

7.1. Implementation and Evaluation Overview

We implement DCIV based on the PBC Library (PBC-0.5.14)[5] which performs the mathematical operations underlying pairing-based cryptosystems. Considering the interaction between multiple chains in DCIV, we use the consortium blockchain to simulate the test environment. All the results of experiments are represented 100 trials on average.

To complete the experiments, we deploy three blockchains in Hyperledger Fabric 2.4[6]. All the on-chain procedures are leveraged by smart contract installed on *AC*, *RC*, and *SC*. *DO* is a laptop running Ubuntu 16.04 with an Intel 2.5 GHz CPU and 4 GB memory that performs the off-chain data preprocessing procedure.

Next, we will conduct experiments to evaluate the performance of DCIV, including the performance of off-chain data processing, on-chain data processing, data transFERRING, and THE integrity verification procedures.

7.2. Experiments and Analysis

First, we evaluate the performance of the data processing procedures in cross-chain interactions. It can be seen from Fig. 8 that the size of the data block affects the time OF data processing. When the size of blocks is set to 4 KB, it can be divided into more blocks than 6 KB of the same data *F* and the data processing at each procedure takes a relatively long time.

Off-chain data preprocessing. Before sending data to *AC*, *DO* needs to preprocess the data to generate corresponding audit metadata. As is shown in Fig. 8(a), the time of data preprocessing increases with the size of data *F*. Fig. 8(c) shows that the time of data preprocessing is dependent on the number of segments in a block. More segments mean more tags to be generated to build the Merkle tree, causing a large computation cost. It can be seen in Fig. 6(a) that the extra storage cost of the Merkle tree raises with the number of segments in a block. The number of segments in a block determines the number of leaf nodes in the Merkle tree. For example, when *S* is 1024, it means that there will be 1024 leaf nodes and the whole tree will have *2S-1* nodes in total. In addition, no matter how the size of a block is set, the storage cost of data processing is fixed. The storage cost is not affected by the number of blocks in the data and only has a relationship with the number of segments in a block.

The generated Merkle tree needs to be stored in the audit digests of transactions on *AC*. In Fabric2.4, the size of transactions packaged into a block is related to the maximum block size and the

[5] https://crypto.stanford.edu/pbc/howto.html
[6] https://hyperledger-fabric.readthedocs.io/en/release-2.4/

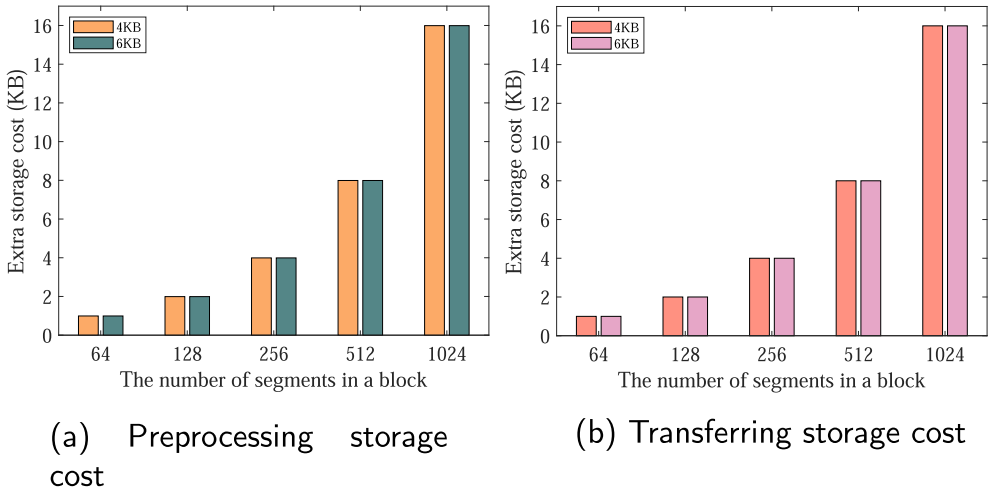


Fig. 6. Data processing storage cost.

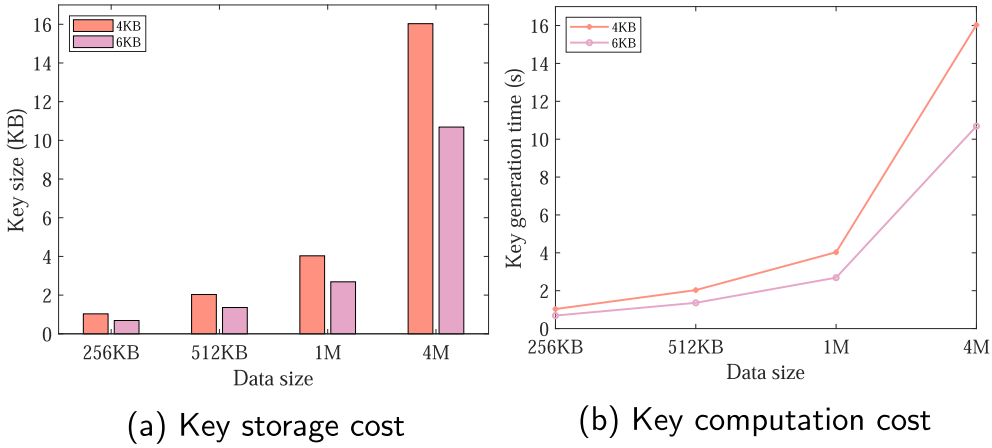


Fig. 7. Key generation cost.

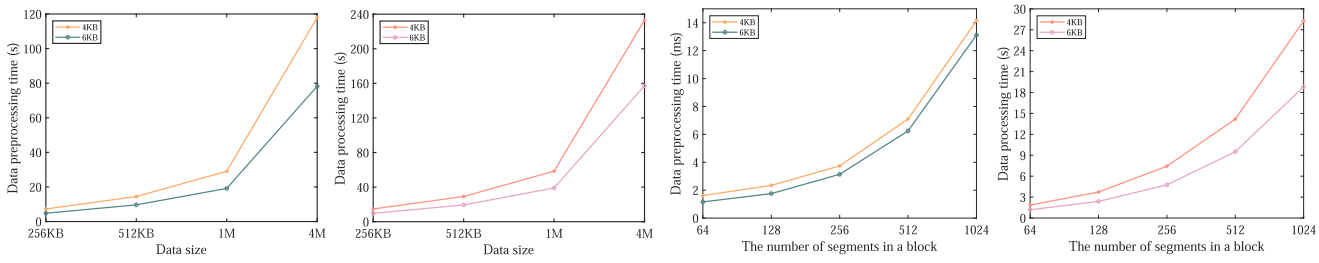


Fig. 8. Data processing computation cost.

maximum number of transactions that can be written to the block. *AbsoluteMaxBytes* represents the maximum size of a block. No block is allowed to exceed this size under any circumstances. If a transaction exceeds this size, it will be returned directly without the opportunity to participate in packaging. Therefore, the size of data and S should be set to a reasonable range to reduce the size of transactions.

Transferred data processing on *AC*. When cross-chain interaction occurs, *AC* needs to generate audit metadata of the data to be transferred for future integrity verification of data stored on *RC*. *AC* generates *sk* and *pk* for data signing and verifying. The main cost in the key generation is caused by the generation of *pk*. As is shown in Fig. 7, the storage and computation costs increase linearly with the number of blocks in data $\widetilde{F}$.

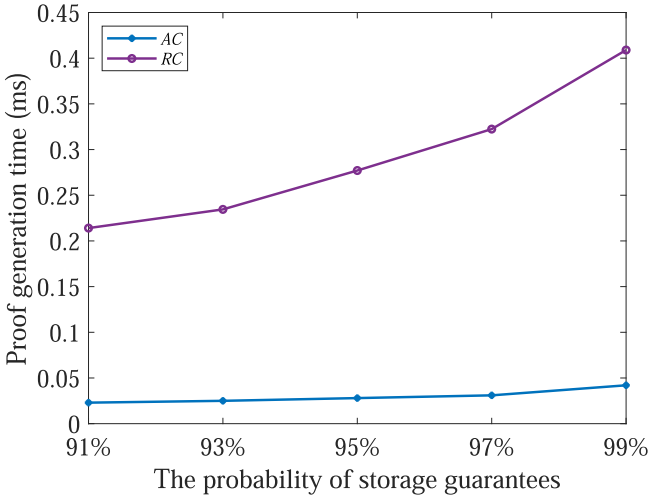


Fig. 9. Proof generation time.

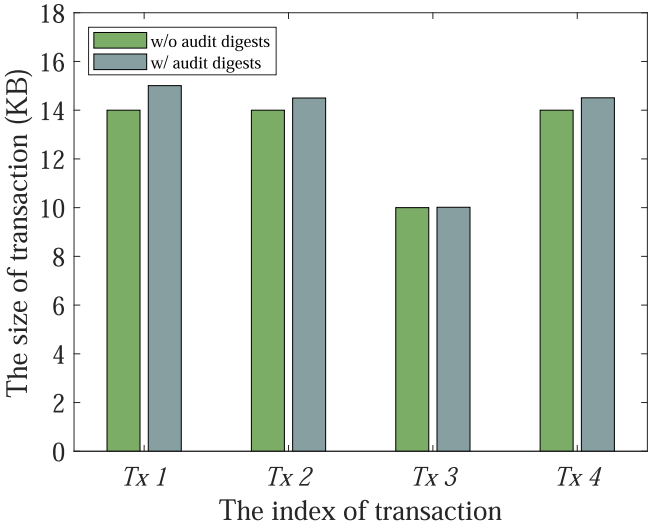


Fig. 10. The size of transaction.

The computation cost in this procedure is mainly the calculation of data $\widetilde{F}$ to generate and sign tags with *sk*. Fig. 8(b) shows that it takes about 15s to process data of 256 KB. Fig. 6(b) shows that the extra storage cost of transferred data processing increases with the number of segments in a block and is not affected by the size of the block which is similar to off-chain data preprocessing in Fig. 6(a).

When *SC* needs to audit the cross-chain interaction, a challenge is randomly generated and sent to *AC* and *RC* respectively. *AC* and *RC* generate corresponding proof according to the received challenge and send them to *SC* for integrity verification.

In Fig. 9, we compare the proof generation time under the influence of different standards (Ateniese et al., 2007) of tampered data detection. When the confidence level of the storage guarantees increases from 91% probability (the number of challenged segments equals 240) to 99% probability (the number of challenged segments equals 460), the time to generate a proof increases significantly for a given data tampering 1%. We can conclude that the time required for proof generation mainly depends on the number of challenged segments.

The time of proof generation on *RC* is higher than that on *AC*. The time of proof generation is more affected by the number of challenged segments. In the case of 460 challenged segments (detection confidence of 99% probability), the total proof generation time of the two parts is also less than 0.45*ms*.

```
"action": {
        "endorsements": [ {
                "endorser": "",     "signature": ""
        } ],
        "proposal_response_payload": {
            "extension": {
                "chaincode_id": { },   "events": { },     "response": { },     "results": { }
            },
            "proposal_hash": ""
        }
        "Audit Digest": {
                "metadata": {
                    "g": " ",
                    "MT": " "
                }
        }
},
```

Fig. 11. The data structure of transaction with audit digests.

Audit metadata storage. In Hyperledger Fabric, each step of data operation will be recorded into a transaction. In 5.2, we propose to add audit digests to the data structure of transactions in the blockchain to store audit metadata for integrity verification, aiming to reduce the data storage pressure of *SC*. In Fig. 10, we compare the change of the size of transactions mentioned in Table 3 before and after the introduction of audit digests. We set the size of data to 4 KB, the size of a block to 4 KB, and the number of segments in a block to 64. We find that with the introduction of audit digests, the size of the transaction slightly increases, which will not bring too much storage burden to the block. We show the details of the audit digests in the data structure of the transaction in Fig. 11, where the contents of the metadata vary depending on different transactions.

8. Conclusion and Future Work

In this paper, we have proposed a new cross-chain audit scheme under the thought of governing the chain by chain, which uses blockchain to verify the data integrity of cross-chain interaction. We have designed off-chain and on-chain data processing and integrity proof generating methods, which effectively ensure the data integrity of cross-chain interaction. Moreover, we have modified the data structure of the transaction in the blockchain and realized a flexible, pluggable, and low-overhead blockchain auditing scheme, which makes our verification scheme more universal and practical. In the future work, we will expand the existing scheme to the audit of cross-chain data modification operations, aiming to realize dynamic cross-chain data integrity verification technology. At the same time, we will study the audit method of the integrity of cross-chain data processing results.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2020YFB1005500) and the Basic Research Program of Jiangsu Province (No. BK20201290).

References

Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D., 2007. Provable data possession at untrusted stores. In: Proc. ACM Conf. Comput. Commun. Secur. (CCS), pp. 598–609.

Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A.K., Poelstra, A., Timón, J., Wuille, P., 2014. Enabling blockchain innovations with pegged sidechains. Sidechains protocol white paper.

Barsoum, A.F., Hasan, M.A., 2015. Provable multicopy dynamic data possession in cloud computing systems. IEEE Trans. Inf. Forensics Secur. 10 (3), 485–497.

Boneh, D., Boyen, X. 2004. Short signatures without random oracles. In Proc. 23th Int. Conf. Theory Appl. Cryptographic Tech. (EUROCRYPT), pages 56–73.

Boneh, D., Lynn, B., Shacham, H., 2001. Short signatures from the weil pairing. In Proc. 7th Int. Conf. Theory Appl. Crypt. Inf. Secur. (ASIACRYPT), pages 514–532.

Deng, L., Chen, H., Zeng, J., Zhang, L., 2018. Research on cross-chain technology based on sidechain and hash-locking. In Proc. Int. Conf. Edge Comput. EDGE, 144–151.

Du, Y., Duan, H., Zhou, A., Wang, C., Au, M.H., Wang, Q., 2021. Enabling secure and efficient decentralized storage auditing with blockchain. IEEE Trans. Dependable Secure Comput., 1–12

Duan, H., Du, Y., Zheng, L., Wang, C., Au, M.H., Wang, Q., 2022. Towards practical auditing of dynamic data in decentralized storage. In: IEEE Trans. Dependable Secure Comput., p. 1.

Goyal, V., 2007. Reducing trust in the pkg in identity based cryptosystems. In Proc. 27th Annual Int. Cryptology Conf. (CRYPTO), pages 430–447.

He, Y., Zhang, C., Wu, B., Yang, Y., Xiao, K., Li, H., 2021. A cross-chain trusted reputation scheme for a shared charging platform based on blockchain. IEEE Internet Things J., 1

Hope Bailie, A., Thomas, S., 2016. Interledger: Creating a standard for payments. In: Proc. 25th Int. Conf. Companion World Wide Web (WWW), pp. 281–282.

Jalil, B.A., Hasan, T.M., Mahmood, G.S., Noman Abed, H., 2021. A secure and efficient public auditing system of cloud storage based on bls signature and automatic blocker protocol. J. King Saud University - Comput. Inf. Sci. ISSN 1319–1578.

Juels, A., Kaliski, B.S., 2007. Pors: proofs of retrievability for large files. In Proc. ACM Conf. Comput. Commun. Secur. (CCS), pages 584–597.

Kate, A., Zaverucha, G.M., Goldberg, I., 2010. Constant-size commitments to polynomials and their applications. In Proc. 16th Int. Conf. Theory Appl. Crypt. Inf. Secur. (ASIACRYPT), pages 177–194.

Lai, R.W.F., Malavolta, G., 2019. Subvector commitments with application to succinct arguments. In Proc. 39th Annual Int. Cryptology Conf. (CRYPTO), pages 530–560.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., Njilla, L., 2017. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In Proc. 17th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGRID), pages 468–477.

Menezes, A., 2005. An introduction to pairing-based cryptography. URL http://www.math.uwaterloo.ca/∼ajmeneze/publications/pairi-ngs.pdf.

Nayak, S.K., Tripathy, S., 2021. Sepdp: Secure and efficient privacy preserving provable data possession in cloud storage. IEEE Trans. Serv. Comput. 14 (3), 876–888.

N.I. of Standards and Technology. Secure hash standard (shs). pages FIPS 180–2, Aug. 2002.

Ristenpart, T., Tromer, E., Shacham, E., Savage, S., 2009. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS), pages 199–212.

Sutton, A., Samavi, R., 2017. Blockchain enabled privacy audit logs. In: Proc. 16th Int. Semantic Web Conf. (ISWC), pp. 645–660.

Tsai, W.-T., Blower, R., Zhu, Y., Yu, L., 2016. A system view of financial blockchains. In Proc. IEEE Symp. on Service-Oriented Syst. Eng. (SOSE), pages 450–457.

Wang, B., Li, B., Li, H., Li, F., 2013. Certificateless public auditing for data integrity in the cloud. In Proc. IEEE Conf. Commun. Netw. Secur. (CNS), pages 136–144.

Wang, C., Wang, Q., Ren, K., Lou, W., 2010. Privacy-preserving public auditing for data storage security in cloud computing. In Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM), pages 1–9.

Wang, H., Wang, Q., He, D., 2021. Blockchain-based private provable data possession. IEEE Trans. Dependable Secure Comput. 18 (5), 2379–2389.

Wang, Q., Wang, C., Li, J., Ren, K., Lou, W., 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. In Proc. 14th Eur. Symp. Res. Comput. Secur. (ESORICS), pages 355–370.

Wang, Q., Wang, C., Ren, K., Lou, W., Li, J., 2011. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst., 22: 847–859.

Yu, J., Wang, K., Zeng, D., Zhu, C., Guo, S., 2018. Privacy-preserving data aggregation computing in cyber-physical social systems. ACM Trans. Cyber-Phys. Syst. 3 (1), Aug.

Yuan, J., Yu, S., 2013. Proofs of retrievability with public verifiability and constant communication cost in cloud. In Proc. Int. Workshop on Secur. Cloud Comput. (SCC), pages 19–26, May. 2013.

Zhang, Y., Xu, C., Lin, X., Shen, X., 2021. Blockchain-based public integrity verification for cloud storage against procrastinating auditors. IEEE Trans. Cloud Comput. 9 (3), 923–937.