International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Block-chain based data provenance and integrity verification

Vaghani Divyeshkumar *

Master's in business administration with concentration in business analytics, Gannon University, 109 University Square, Erie, PA 16541, United States.

International Journal of Science and Research Archive, 2024, 12(01), 2516–2533

Publication history: Received on 03 May 2024; revised on 10 June 2024; accepted on 13 June 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.12.1.1076

Abstract

Blockchain applications face major scalability challenges, hindering their ability to support services with large-scale and frequent transactions, such as the computational and communication overhead involved in integrity verification for large-scale IoT data. To tackle the problem, we propose a Blockchain and Bilinear mapping-based Data Integrity Scheme (BB-DIS) tailored for large-scale IoT data in cloud storage. The paper introduces a blockchain-based framework for data integrity verification of large-scale IoT data which includes a series of protocols, verification algorithms, and detailed performance analysis; develops a prototype system incorporating an edge computing processor near the IoT devices to preprocess the large-scale IoT data to significantly reduce communication costs and computational burdens; and performs multiple simulation experiments on Hyperledger Fabric to provide a comparative analysis of computational and communication overhead between BB-DIS and other baseline schemes. Experimental results demonstrate that the proposed BB-BIS surpasses existing blockchain-based methods in computational cost and communication overhead for large-scale IoT data.

Keywords: Blockchain; Cloud storage; Data; Integrity verification IoT; Scalability

1. Introduction

With the widespread adoption of Internet of Things (IoT) technologies like smart cities, autonomous vehicles, and smart grids, the number of devices connected to the Internet is skyrocketing. Gartner predicts a 42% increase in IoT connections and $20 billion in spending from 2018 to 2020. As such, securely collecting [1], processing, storing, and analyzing these massive IoT datasets has become a critical issue for the continued development of IoT applications [2]. Traditional distributed database systems cannot meet the data management needs in the IoT landscape, leading to the rise of Cloud Storage Services (CSSs).

By storing data externally, the combination of IoT and cloud technology removes the burden of local storage and management. However, cloud service providers can potentially gain control over users' data, posing significant security risks. Therefore, verifying the integrity of IoT data is crucial for effective cloud storage. Current data integrity verification methods for cloud storage typically rely on hash functions [3], asymmetric cryptographic algorithms [4], and erasure codes [5]. These methods can be categorized into provable data possession (PDP) mechanisms [4] and proofs of retrievability (POR) mechanisms [6] based on their ability to correct erroneous data post-verification. Traditional methods often depend on trusted Third Party Auditors (TPAs) for auditing, which reduces user burdens during verification. For instance, in Wise Information Technology of 120 (WIT120), large volumes of electronic health records (EHR) collected by wearable devices are stored in the cloud, and service providers generally delegate validation tasks to TPAs to ensure data integrity. However, TPAs are not entirely trustworthy in real-world scenarios. Even encryption methods [7], which can prevent user privacy breaches, rely on the TPA's credibility for quality and effectiveness.

* Corresponding author: Vaghani Divyeshkumar

Blockchain technology has recently gained significant attention for its transparency, immutability, security, and decentralization. Researchers are exploring the execution of integrity verification services within decentralized blockchain networks, where transactions can occur without a trusted TPA. Ethereum and Hyperledger Fabric are two popular frameworks for implementing blockchain networks [8]. However, blockchain applications face major scalability challenges, hindering their ability to support services with large-scale and frequent transactions, such as the computational and communication overhead involved in integrity verification for large-scale IoT data [9][10][11]. Additionally, the dynamic nature of IoT data [12][13] has rarely been addressed by most existing blockchain-based data integrity methods.

To tackle the aforementioned issues, we propose a Blockchain and Bilinear mapping-based Data Integrity Scheme (BB-DIS) tailored for large-scale IoT data in cloud storage. The main contributions of this paper are summarized as follows:

We introduce a blockchain-based framework for data integrity verification of large-scale IoT data. This includes a series of protocols, verification algorithms, and detailed performance analysis.

We develop a prototype system incorporating an edge computing processor near the IoT devices to preprocess the large-scale IoT data, significantly reducing communication costs and computational burdens.

We perform multiple simulation experiments on Hyperledger Fabric, providing a comparative analysis of computational and communication overhead between BB-DIS and other baseline schemes. Various sampling strategies are explored, leading to the recommendation of an optimized sampling verification scheme.

2. Literature Review

2.1. Data Provenance

Data provenance is crucial for cloud computing system administrators to diagnose system or network intrusions. Cloud environments often involve data transfers between various system and network components, either within a data center or across multiple federated data centers. Due to multiple data copies and diverse transfer paths designed for resilience, tracking the exact path and origin of attacks becomes complex. This complexity makes it difficult for administrators to pinpoint the source of an attack, identify the software or hardware components involved, and assess the impact. Identifying security breaches with high precision is necessary, and data provenance can help achieve this.

Modern provenance systems in the cloud use logging and auditing technologies to support these tasks. However, these technologies are often ineffective in the complex nature of cloud computing systems, which involve multiple layers of interacting software and hardware spread across different geographic and organizational boundaries. Tracing the origin, cause, and impact of security violations in cloud infrastructures requires collecting forensics and logs from diverse sources, which is a challenging task. Logs only provide a sequential history of actions related to each application, whereas provenance data offers detailed histories of changes to data objects, including the components that processed the data and the users who interacted with it, thus providing stronger assurance requirements.

Several data provenance efforts have been presented by researchers. PASS was the first scheme to address the collection and maintenance of provenance data at the operating system level [8]. A file provenance system [9] intercepts file system calls below the virtual file system, requiring changes to the operating system. For cloud data provenance, S2Logger [10] was developed as an end-to-end data tracking tool providing both file-level and block-level provenance in kernel space. In addition to data provenance techniques and tools, the security of provenance data and user privacy have also been explored. Asghar et al. [11] proposed a secure data provenance solution for the cloud using a twofold encryption method to enhance privacy, though it comes with higher computational costs. SPROVE [12] ensures the confidentiality and integrity of provenance data using encryption and digital signatures but lacks data querying capabilities. Progger [13] is a kernel-level logging tool providing tamper-evidence at the cost of user privacy. Provenance data is also used for managing cloud environments, such as discovering usage patterns for cloud resources, promoting resource reuse, and managing faults [14].

2.2. Blockchain

Blockchain technology has garnered significant interest from a variety of sectors including finance, healthcare, utilities, real estate, and government agencies. Blockchains are shared, distributed, and fault-tolerant databases that all participants in the network can access, but no single entity can control. This technology is designed to function in highly contested environments against adversaries aiming to compromise the network. Blockchains anticipate the presence of

adversaries and counter their strategies by leveraging the computational power of honest nodes, ensuring that the exchanged information is resistant to manipulation and destruction. The reconciliation process between entities is accelerated due to the absence of a trusted central authority or intermediary. Tampering with blockchains is extremely challenging because of the use of a cryptographic data structure and the lack of reliance on secrets. Blockchain networks are fault-tolerant, allowing nodes to remove compromised nodes. Despite this robustness, several vulnerabilities exist [15], which could potentially affect the integrity of the blockchain. However, exploiting these vulnerabilities would require a malicious node to have immense computational power, making such attacks cost-prohibitive.

The decentralization and security features of blockchain have led researchers to develop various applications such as smart contracts, distributed DNS, and identity management. In addition to Bitcoin, Ethereum [16] is built on a public blockchain designed for the simple and rapid development of decentralized applications using a per-address transaction model. Multichain [17] offers an open-source permissioned blockchain network, allowing developers to host their blockchain on a private cloud architecture. Multichain uses a per-output transaction model and can handle high throughput [18]. Tierion [19] provides a platform for uploading and publishing data records into the blockchain network. With available public APIs, Tierion is convenient for integrating applications that require blockchain technology. Developers can post metadata using HTTP requests to the Tierion data store and fetch record information.

Each data record has a record ID used to retrieve the blockchain receipt generated from the blockchain transactions. The blockchain receipt includes the transaction ID, which is used to locate a transaction and the block that contains it. This ensures that the posted data record on the blockchain cannot be tampered with, thus assuring its integrity.

Blockstack Labs from Princeton University proposed a decentralized PKI service on top of Namecoin and a blockchain-based naming and storage system [20]. Blockchain applications in information-centric networks for name-based security of content distribution have also been proposed [21]. Enigma is a decentralized computation platform with guaranteed privacy, using the blockchain network to manage access control and identity, create tamper-proof event logs, and control the network [22]. Guardtime offers industrial-scale blockchain services using Keyless Signature Infrastructure (KSI) and secure one-way hash functions, which are quantum-immune compared to RSA [23]. Guardtime has also proposed a blockchain standard for digital identity and a protocol for authentication and digital signatures, providing a simplified mechanism for revocation management and long-term validity [24].

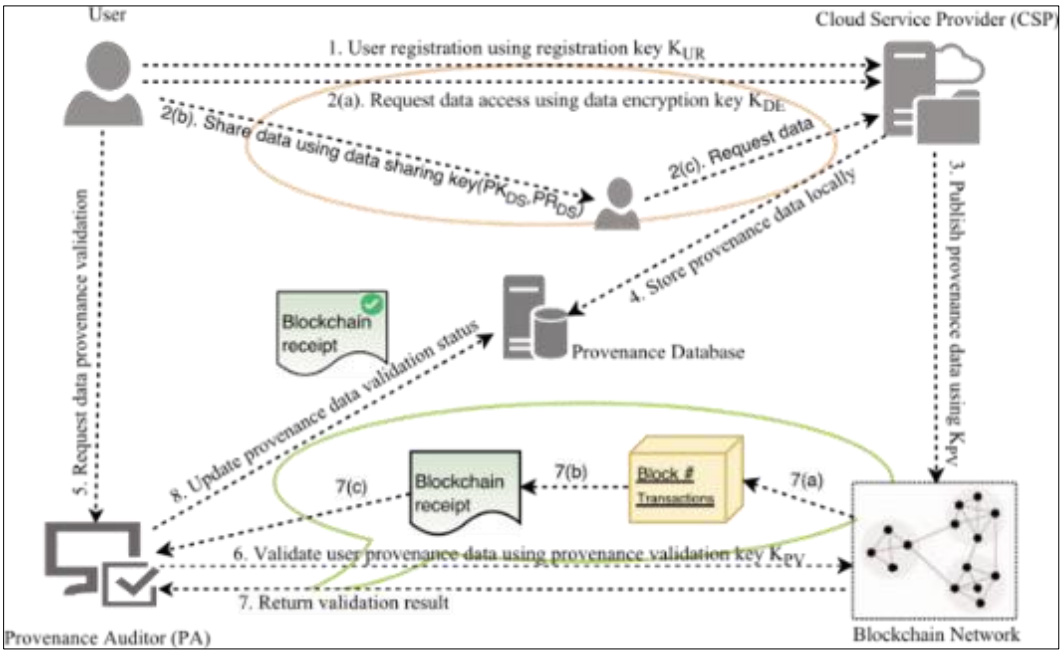


Figure 1 ProvChain System Interaction

2.3. Empirical Studies

2.3.1. Traditional Work on Data Integrity

The concept of data integrity verification was first introduced by Deswarte et al. [3] in 2004. They proposed two solutions involving the calculation and comparison of message authentication code (MAC) values to determine the

completeness of data on remote nodes. However, these solutions incurred significant communication overhead and computational costs. To address this, Sebé et al. [14] employed a method of blocking original data files to reduce computational expenses. Subsequently, Ateniese et al. [4] introduced the PDP scheme, which utilized Rivest-Shamir-Adleman (RSA) signatures. This model generated probabilistic proofs of possession by sampling random sets of blocks from the server, significantly lowering I/O costs. The client maintained a constant amount of metadata to verify the proof. However, the protocol was designed for static files and could not manage dynamic data storage without security risks. This issue was addressed in [15], although fully dynamic data operations were not supported.

Wang et al. [16] proposed a new challenge-response protocol using the Merkle hash tree to ensure the correctness of data blocks, introducing an independent TPA to perform the verification operation, thus alleviating the user's burden. Juels and Kaliski [6] first introduced a sentinel-based POR model, which randomly added "sentinel" data blocks to the stored data and used erasure codes to detect and downgrade distorted data to storage with undefined quality of service. Building on Ateniese et al.'s research, Shacham and Waters [17] utilized the Boneh-Lynn-Shacham (BLS) signature mechanism to create homomorphic verifiable tags (HVTs), reducing communication overhead while supporting public auditing, though it did not ensure user data privacy. Wang et al. [5] leveraged the linear characteristics of erasure codes to enable partial dynamic operations. Chen and Curtmola [18] employed Cauchy Reed-Solomon linear coding to preprocess data, improving the recovery speed of erroneous data, but the computational costs remained high. To prevent TPA from leaking private data [19], Wang et al. [7] proposed a data integrity verification mechanism using public key-based homomorphic authenticator and random mask, achieving privacy protection in public cloud systems. Othman et al. [21] aimed to reduce energy consumption in wireless sensor networks (WSNs) [20] by adopting symmetric-key homomorphic encryption to protect data privacy, combined with homomorphic signatures to verify aggregated data integrity. Zhu et al. [22] decreased the computational overhead of hash functions in the signature process [23] and utilized random masking techniques to preserve data privacy.

Considering the specificity and complexity of graph databases, Arshad et al. [24] proposed two security concepts based on hash message authentication code (HMAC) for verifying the integrity of graph data and query results. Reina et al. [25] took into account the direction of edges, calculating a new hash value, such as a chained hash, from the concatenation of the current node.

2.3.2. Applications of Blockchain

With its fundamental attributes of decentralization, persistence, and auditability [26], blockchain technology has emerged as a transformative and enabling technology, gradually being adopted across various industry verticals. Blockchain can play a crucial role in addressing numerous IoT security issues [2]. Suliman et al. [27] demonstrated a blockchain solution using Ethereum smart contracts for monetizing IoT data with automated payments, eliminating the need for intermediaries. Albreiki et al. [28] developed a blockchain-based system utilizing Ethereum smart contracts to manage access control policies for IoT data access in a decentralized manner without relying on a trusted third party. Despite these advancements, the integrity issues of IoT data still require solutions. Chaer et al. [29] examined the system integration architecture and sequence flow diagrams to show how blockchain can support and enhance 5G networks. Salah et al. [30] explored how integrating artificial intelligence (AI) with blockchain can foster the development of a new decentralized economy and identified open research challenges in utilizing blockchain for future AI applications. To combat fake digital content, Hasan and Salah [31] proposed a solution using Ethereum smart contracts to trace and track the provenance and history of video content back to its original source, even if the content has been copied multiple times.

2.3.3. Blockchain-Based Work on Data Integrity

Building on previous research in cloud storage architecture and data integrity, Liu et al. [9] introduced a blockchain-based method for ensuring the integrity of IoT data. This approach allows for integrity verification without relying on TPAs in a dynamic IoT environment. However, improvements are needed in the speed of uploading IoT data and the size of the verified data. Yue et al. [10] developed a blockchain-based P2P cloud storage data integrity verification framework, utilizing Merkle trees for verification and analyzing system performance with different Merkle tree structures. Liang et al. [32] proposed a decentralized and trusted cloud data provenance system to ensure data security, with the provenance auditor verifying data through blockchain information. Wang et al. [11] introduced a decentralized model to address the single point of trust issue in traditional data auditing by leveraging collective trust. This protocol enables users to trace the history of their data.

In summary, most existing blockchain-based data integrity verification methods primarily address trust issues rather than data size. A significant consideration is that IoT data stored in the cloud must be updated in real-time to meet the

latest requirements of various applications. Thus, it is essential to develop a blockchain-based dynamic solution focused on data renewal for data integrity verification.

3. Our Scheme BB-DIS

3.1. Preliminary

3.1.1. Blockchain Technology

Blockchain technology enables decentralized peer-to-peer transactions, coordination, and collaboration without the need for trust, utilizing data encryption, time stamping, and distributed consensus. It effectively addresses the issues of high cost, inefficiency, and insecure data storage inherent in centralized systems. The first generation of blockchain, introduced by Bitcoin, serves as a public ledger for digital currency transactions. While much research on blockchain focuses on Bitcoin, the technology has applications far beyond it [33]. The second generation of blockchain offers a flexible programmable platform, introducing smart contracts as autonomous programs that operate within blockchain networks. A smart contract is a digital protocol designed to establish agreements between parties based on predefined rules without needing a trusted third party [34]. These contracts can represent triggers, constraints, and even complete business processes. Ethereum, known for its smart contracts, is a prominent second-generation blockchain. Transactions in Ethereum are signed messages initiated by an external account, transmitted via the Ethereum network, and recorded on the Ethereum blockchain [35]. Ethereum transactions fall into three categories: transferring transactions, creating smart contracts, and executing smart contracts. Hyperledger Fabric, a permissioned blockchain, offers various consensus mechanisms. Versions 0.6 and 1.0 of Hyperledger Fabric provide PBFT and Kafka consensus mechanisms, respectively, and it is widely used in developing decentralized applications.

3.1.2. Bilinear Mapping

We assume $G"$ is a Gap Diffie-Hellman (GDH) group, P is the generator of group $G"$, and $G\$$ is another multiplicative cyclic group of prime order q. The mapping e: $G" \times G" \rightarrow G\$$ is known as bilinear pairing and possesses the following properties:

Computability: For any $a, b \in G"$, there exists an efficient algorithm to compute (a, b).

Bilinear: For any $a, b, c \in G"$, the following holds:

$$e(a, bc) = e(a, b)e(a, c)$$

$$e(a, b) = e(b, a)$$

Non-degenerate: P is non-degenerate if $(P, P) \neq 1$.

3.1.3. Short Signature

RSA, Digital Signature Algorithm (DSA), and BLS signatures comprise 1024 bits, 320 bits, and 160 bits, respectively, under equivalent security conditions. The computational security of the RSA algorithm depends on the challenge of factoring large integers, resulting in excessive computational overhead. While the DSA signature is a variant of the RSA signature, it cannot encrypt data files. On the other hand, the BLS signature can function in any bilinear cryptographic context and remains unforgeable in the random oracle model. However, the BLS-based scheme necessitates the adoption of a specific hash function, which exhibits efficiency concerns with large-scale data. In this context, this paper introduces a secure hash function H: $\{0,1\}^* \rightarrow Z3^*$, which can be a general cryptographic hash function such as SHA-1 or MD5.

The ZSS short signature, proposed by Zhang et al. [23], is based on bilinear pairing and incurs less overhead than the BLS signature. We assume, employing the properties of bilinear mapping, where $(P5, P6) = (P, P)(56)$. This signature system primarily comprises three functions:

- **KeyGen**: The data owner selects a random integer $\alpha \leftarrow Z3^*$ as the private key sk and αP as the public key pk. The private key α cannot be derived from pk.
- **Sign**: The signature of the message m is Sig = G (H")IJ P.

- **Verify**: A verifier possesses (pk), m, Sig' and needs to verify Sig' = G (H")IJ P. This entails calculating (P, P) and ((m)P + αP, Sig') and determining their equality. If they are equal, the signature is verified to be generated by the holder of the private key α. Verification is achieved through the following equations:

$$e(H(m)P+\alpha P,Sig)=e\mathrm{N}(H(m)+\alpha)P,P\mathrm{O}H(m)+\alpha=e(P,P)(\mathrm{G(H)IJ})\cdot\mathrm{G(H")IJ}=e(P,P)\ldots\ldots\ldots(1)$$

3.2. Framework of Our Scheme

Table I gives a list of main symbols that will be covered in our scheme.

Table I Symbols in the Framework

Symbol	Definition
DOD	Data Owner Device
DCD	Data Consumer Device
CSP	Cloud Service Provider
HSSC	HVTs Storage Smart Contract
CRSC	Challenge Receiving Smart Contract
IVSC	Integrity Verification Smart Contract

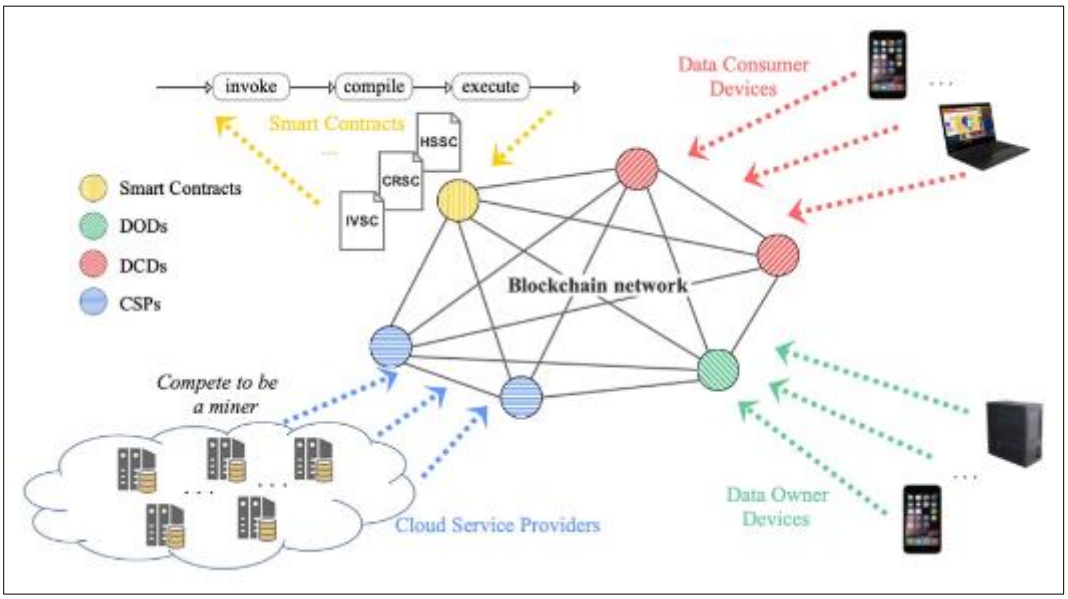


Figure 2 Framework of BB-DIS

The BB-DIS framework is illustrated in Figure 2, comprising four primary entities: Smart Contracts, Data Owner Devices (DODs), Data Consumer Devices (DCDs), and Cloud Service Providers (CSPs). To fulfill various functions, three types of smart contracts exist: HVTs Storage Smart Contract (HSSC), Challenge Receiving Smart Contract (CRSC), and Integrity Verification Smart Contract (IVSC). All entities can function as blockchain nodes within the blockchain network. In practice, data integrity verification involves multiple data owners and consumers, executed by smart contracts within the blockchain system. Users with integrity requirements can deploy blockchain clients on their node devices or exit the blockchain network. Additionally, CSPs act as nodes in the blockchain network, dispersing nodes entirely and enhancing integrity verification efficiency.

DODs and DCDs must be incorporated into the blockchain network during system initialization to create a key pair. The data owner is responsible for covering the costs of interacting with smart contracts and cloud storage services. CSPs can operate as miner nodes in the blockchain network, enabling them to offer services through mining and earn

corresponding rewards. Data consumers request access to data stored on cloud servers and pay the associated expenses. Unlike Hyperledger Fabric, each node account in the Ethereum network must have sufficient gas to ensure successful transactions. During each transaction, DODs pay promissory gas for data storage services to the respective CSP.

Each cloud service provider offers cloud storage solutions, such as Amazon S3, IBM Bluemix, Microsoft Azure, and Smart Ocean. Within this framework, CSPs furnish a standard data storage service for data owners, while non-cloud data can be transferred via an inter-node P2P network.

Once deployed, modifying smart contracts becomes challenging; therefore, if any security vulnerabilities exist in smart contracts, preventing attacks by hackers becomes arduous. Hence, it is imperative to thoroughly test smart contract code and employ essential security analysis tools to eradicate any security loopholes [36]. The Chaincode Scanner serves as a security analyzer for Hyperledger Fabric smart contracts, aiding in the detection of security vulnerabilities within the smart contracts.

For the blockchain network's security and efficiency, two primary concerns in this domain, we make the following two assumptions. Firstly, the likelihood of 51% attacks and selfish mining is minimal if all participating nodes prioritize their own benefit. Bitcoin's blockchain serves as the most evident proof; instead of initiating a 51% attack, a malicious attacker would prefer using computational power for mining. In reality, such attacks are rare. Secondly, achieving blockchain consensus within a short timeframe is feasible. In the Ethereum blockchain, transactions can be validated within an average duration of 12 seconds, while Hyperledger Fabric aims for consensus in less than 1 second. Generally, Hyperledger Fabric proves more suitable for enterprise-level blockchain applications compared to other blockchain platforms.

3.3. Verification Protocol

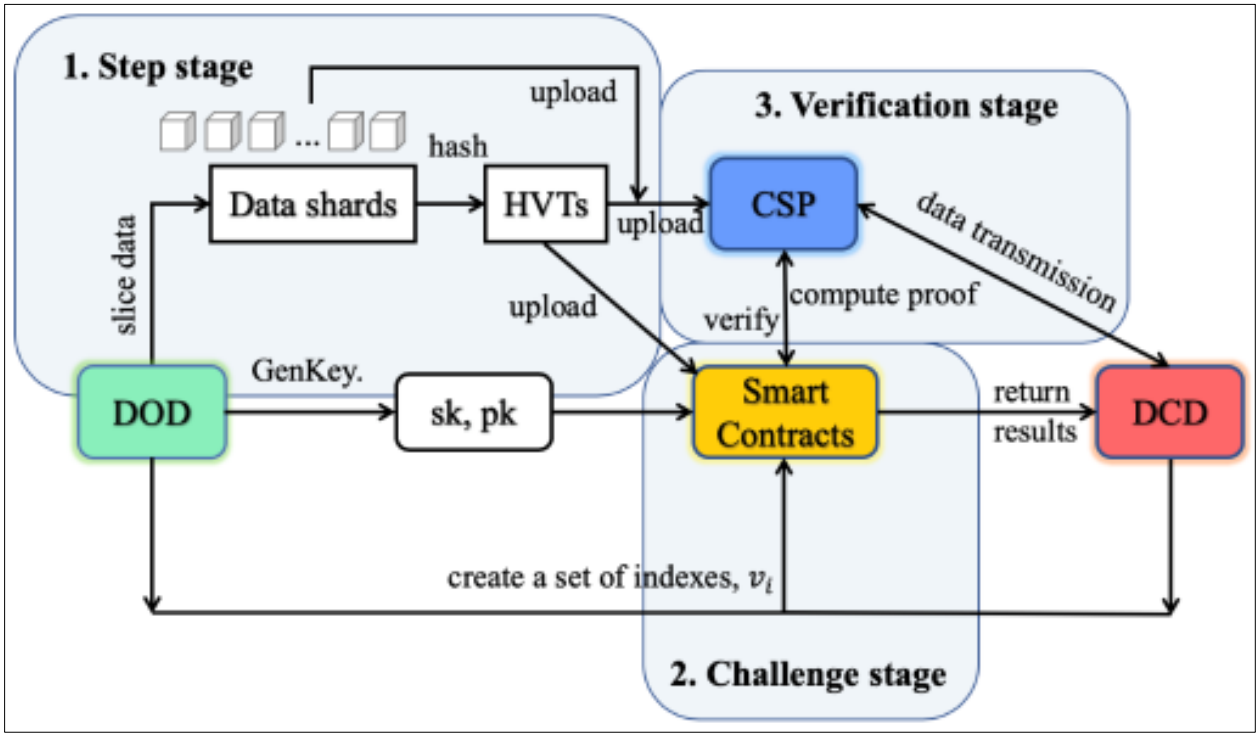


Figure 3 Verification protocol

The verification process of this plan is depicted in Figure 3, while the transactions among various smart contracts and all participants are meticulously detailed in Table II. The protocol unfolds in three phases: the step stage, challenge stage, and verification stage, with smart contracts and CSP assuming the roles of verifier and proof provider, respectively.

Table II Protocol: Data Integrity Verification using Blockchain

	Step Entities	**Operation**
	Step stage	
1	DOD	Generate sk, pk (for short signature)
2	DOD	Slice data into shards, generate hvts
3	DOD → CSP	Upload data shards and hvts
4	DOD → HSSC	Upload hvts
	Challenge Stage	
1	DOD	Create a set of indexes I = {sR}, vR
2	DOD → CSP	$chal$ = {(i, vR)}
3	DOD → CRSC	$chal$ = {(i, vR)}
	Verification Stage	
1	HSSC → IVSC	Send hvts
2	CRSC → IVSC	Send $chal$
3	CSP	Compute proof {R, μ, η}
4	CSP → IVSC	Send proof {R, μ, η}
5	IVSC	Verify
6	IVSC → DOD	Return the verification result

In the step stage, the DOD initiates a bilinear mapping, selects a short signature hash function, randomly chooses a private key, and computes a corresponding public key from the private key. Subsequently, the DOD divides a data file into a set of equally sized data shards, computes the HVT of each data shard post-hashing, generating an authentication metadata set to reduce communication overhead while supporting public auditing. The DOD uploads the data shard set and metadata set to the cloud storage server and transmits the metadata set to the HSSC via the blockchain network in a transaction format before locally deleting the data file.

During the challenge stage, the DOD selects c elements from HSSC to construct a data shard index set randomly and forwards a series of random values, alongside the data shard index set, to the CSP and CRSC in the form of a challenge request.

In the verification stage, IVSC obtains HVTs and the challenge request from HSSC and CRSC, respectively. Upon receiving the challenge request, CSP computes the proof {R, μ, η} and dispatches it to the IVSC, which verifies the proof's accuracy. If deemed correct, indicating data integrity in the cloud, the IVSC conveys the outcome to the DOD.

Additionally, a DCD can also instigate a verification request for stored data. In such scenarios, a DOD remains imperative for preliminary tasks in the step stage. A DCD seeking integrity verification service can participate in the challenge and verification stages. Upon confirming data integrity, the CSP directly dispatches the data from servers to the corresponding client via the P2P network.

3.4. SC-Verification Algorithm

In the verification process, we utilize smart contracts and devise the verification algorithm in accordance with the verification protocol to authenticate metadata. The application of the SC-Verification algorithm unfolds through the step stage, challenge stage, and verification stage as outlined below:

3.4.1. Step stage:

G" represents a q-order cyclic additive group, where P denotes one of its generators, and $G$$ signifies a q-order cyclic multiplicative group. Z3 denotes the integer ring of mod q.

To begin, the DOD must establish a bilinear mapping: e: $G" \times G" \rightarrow G\$$ and a security hash function for short signatures: H: {0,1}∗ → {0,1}Y Considering ϕ as a pseudorandom function where and $|q| \geq \lambda \geq 160$.

Next, the DOD randomly selects a private key α, yielding the corresponding public key as $Y = \alpha P$. The public key is denoted as Y, and the private key as α, ensuring the impossibility of computing the private key from the public key.

Subsequently, the DOD partitions the data file F into equal-length data shards: {m", m\$, mf, … , mg}, and computes an HVT for each data shard mR:

$$\delta R = (mR) + \alpha P \quad \text{.....(2)}$$

This results in a metadata collection: $\Phi = \{\delta", \delta\$, \dots, \delta g\}$

Finally, the DOD uploads the data shard set to the cloud storage server and transmits the metadata set Φ to HSSC. Following this, the DOD locally deletes the data file. The procedure is depicted in Algorithm 1:

Algorithm 1: ***Step***

Input: {Data File F, Hash Function H}

Output: $\{\delta_R\}$

$F = \{m", m\$, m\xi, \dots, m\xi\}$, $\delta_R = 0$

for $i = 1$ to n **do**

$\delta_R = {}^{P}\mathrm{m}H(m_R) + \alpha$

end for

return δ_R

3.4.2. Challenge stage

The DOD selects c elements at random to create a data shard index set $I = \{s", s\$, \dots, sn\}$, and generates a pseudo-random number for each $i \in I$. The DOD then sends these random values along with the data shard index set to the CSP and CRSC as part of the challenge request $chal = \{(i, vR)\}$, where $s" < i < sn$.

3.4.3. Verification stage

Upon receiving the challenge request, the CSP, acting as the proof provider, computes the following:

$$R=\sum i\in IviY \text{.....(3)}$$

$$\mu=\sum i\in IviH(mi)P \text{ .....(4)}$$

$$\eta=P-\sum i\in I\delta i \text{.....(5)}$$

The CSP sends $\{R, \mu, \eta\}$ as the proof to the IVSC. Upon receiving this proof, the IVSC checks the integrity of the data stored in the cloud by verifying:

$$(\eta,P)\cdot(\mu+R,P)=e(P,P) \text{.....(6)}$$

If the equation holds true, the data is considered intact. The smart contract then returns the verification result to the requester. This procedure is outlined in Algorithm 2:

Algorithm 2: *Challenge and Verification*

Input: {Data Block Index I, Φ, Hash Function H, P, Y}

Output: {Proof R, μ, η}

$I = \{s_1, s_2, \ldots, s_n\}$

$R = 0$

$\mu = 0$

$\eta = P$

for $i = s_1$ to s_n **do** 6 $\quad v_R = \phi(k_0, i)$

$proof1 = v_R Y$

$proof2 = v_R H(m_R)P$

$proof3 = v_R m\delta_R$ 10 $\quad R = R + proof1$

$\mu = \mu + proof2$

$\eta = \eta - P^{\$} \cdot proof3$

end for

return R, μ, η

3.5. Performance Analysis of SC-Verification Algorithm

3.5.1. Feasibility

Based on the proposed scheme, if the CSP maintains the integrity of the stored data, the proof provided by CSP will be accurate. The correctness of our scheme is demonstrated by the following calculation:

$$e(\eta,P)\cdot e(\mu+R,P)=e((P-\sum_{i\in I}vi\delta i),P)\cdot e((\sum_{i\in I}viH(mi)P+\sum_{i\in I}viY),P)e(\eta, P)$$

$$=e((P-\sum_{i\in I}viH(mi)P+\alpha P),P)\cdot e((\sum_{i\in I}viH(mi+\alpha)P),P)$$

$$=e(-\sum_{i\in I}viH(mi+\alpha)P,P)\cdot e(P,P)\cdot e((\sum_{i\in I}viH(mi+\alpha)P),P)$$

$$=(P,P)\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

From the deduction of equation (7), it is clear that our SC-verification algorithm is feasible.

3.5.2. Security:

We assume there are attackers or malicious servers attempting to tamper with the data stored by data owners in the cloud. To pass the verification of the smart contract, they would need to construct a signature $\delta*=\sum vi*P$ such that:

$$\mu*=\sum_{i\in I}viH(mi)P+vH(m*)P\ldots\ldots\ldots\ldots(8)$$

And:

$$\eta*=P-\sum_{i\in I}vi\delta i-v\delta*\ldots\ldots\ldots\ldots\ldots\ldots(9)$$

$$(\eta*,P)\cdot(\mu*+R,P)=e(P,P)\quad\ldots\ldots\ldots\ldots\ldots(10)$$

However, without knowledge of the private key α, it is impossible to forge $m*$ that satisfies:

$$1/H(m*)+\alpha = 1/H(m)+\alpha \quad \text{.....(11)}$$

Thus, the proof cannot be altered.

If attackers or malicious servers delete the data mmm in the cloud storage, similar to the above analysis, they cannot forge a valid $m*$ without the private key α.

From this analysis, we conclude that our SC-verification algorithm effectively counters malicious attacks.

3.5.3. Dynamicity

Our scheme supports dynamic data updates through the update request algorithm UpdateReq() and the update execution algorithm UpdateExec(). These operations include appending, modifying, and deleting data shards.

- **UpdateReq()**: This algorithm is executed on the DOD, generating an update request for the outsourced file copy stored at the remote CSP. The output is an update request formatted as $\langle BlockOp, Ind, mi, \delta i \rangle$, where $BlockOp$ indicates the operation type, and Ind, mi, and δi denote the index of the updated data shard, the updated data shard, and the updated metadata, respectively. The DOD sends this request to the cloud.
- **UpdateExec()**: Executed on the CSP server, this algorithm takes the update request from the DOD as input and produces a new file copy F and new metadata δi as output. After each update, to ensure the update's correctness, the DOD will run the challenge agreement.
- **Appending Operation**: The DOD inserts a new data shard at position j. Initially, if there are n data shards, the number of shards will increase to $n + 1$ after appending. Even if the generated challenge request includes the new data block mn+1, verification remains valid since the metadata set is updated.
- **Deletion Operation**: When a data shard is deleted, all subsequent shards shift one position forward. If the data shard at index j is to be deleted, the DOD sends a delete request $\langle Delete, j, null, null \rangle$ to the CSP. Upon receiving this request, the CSP removes the data shard at index j from the backups.
- **Modification Operation**: Similar to the appending operation, after modifying, there will still be n data shards.

4. Prototype System of BB-DIS

Figure 4 illustrates a prototype system based on the BB-DIS framework. The system is structured into four layers from bottom to top: 1) IoT devices, 2) edge computing devices and clients, 3) cloud storage service, and 4) data consumer devices.

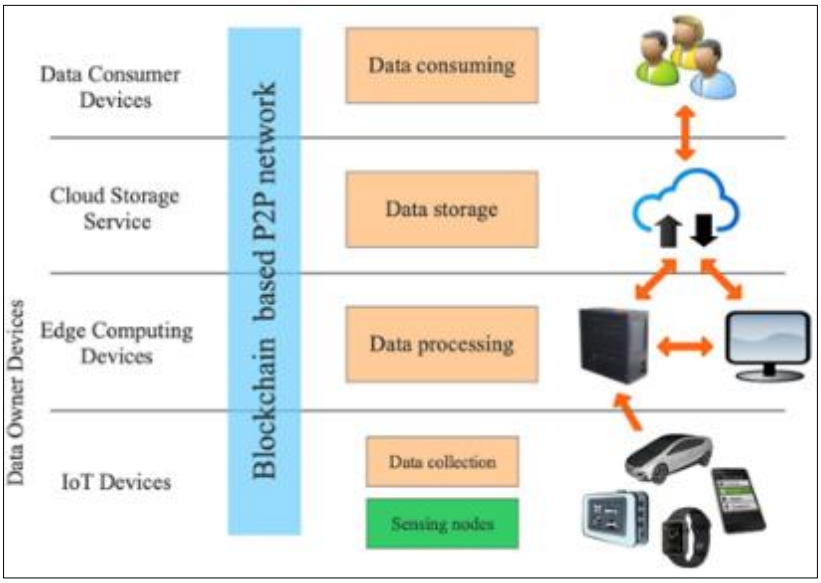


Figure 4 Prototype system

As depicted in Figure 4, each component functions as a node in both the blockchain and P2P network. IoT devices handle data generation. Clients and edge devices act as data owners, with edge computing managing the local processing and

transmission of source data. The processed IoT data is then stored in cloud servers or smart contracts. Data owner clients initiate verification requests and send challenge requests through the blockchain network. Data consumer clients, operating on PCs or in the cloud, can send data consumption requests or access stored IoT data.

4.1. Edge Computing

Edge computing features a decentralized architecture, utilizing any node with computing and network resources between the data generation source and the cloud center as an edge node. This architecture moves application operations, data resources, and services from the central node to the network's edge nodes. Consequently, it accelerates data processing and transmission, reduces delays, and enhances the efficiency of handling large volumes of data. With the IoT's rapid growth and the expansion of cloud computing services, edge computing offers significant benefits in areas like cloud offloading, video analytics, smart homes, and smart cities . Its low latency and high data processing capabilities greatly improve our daily lives.

In the proposed blockchain-based data integrity scheme, edge devices play a crucial role. As shown in Figure 4, they not only facilitate the transmission of IoT device messages and transactions but also aid in managing data storage and performing computations . The functions of edge devices in our prototype system include:

- **Identifying IoT Devices**: The edge server maintains an identity record of all nearby IoT devices and assists each device in generating data shards and HVTs.
- **Creating Transactions for IoT Devices**: A valid blockchain transaction requires the IoT device's signature or verification from other nodes. Edge servers help mitigate any deficiencies in IoT devices.
- **Collecting and Transferring Data to the Blockchain Network**: The edge server continuously gathers data from nearby IoT devices. It identifies cloud server addresses for data storage and sends data blocks to them.

4.2. Blockchain based P2P File System

The P2P solution builds upon the client-server model suited for small distributed environments where the server possesses significant processing power. The hallmark of P2P networks is symmetric communication between peer nodes, where each node can function as both a client and a server. This system addresses the bandwidth challenge of sharing files from server to client. Peers can share files among themselves through different segments, eliminating the need to request all files from the server simultaneously. This greatly improves the scalability and efficiency of file sharing.

5. Simulation Results and Performance Evaluation

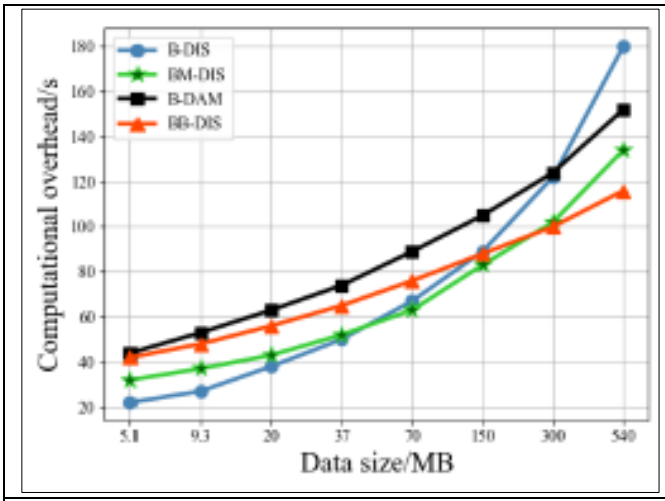
5.1. Experimental Preparations

Building on the prototype system, we conducted a series of experiments to evaluate the performance of our scheme. The server used was an Inspur Yingxin NF8465M4, while the PC was equipped with an Intel i7 quad-core processor at 3.30GHz and 16GB of memory, running a 64-bit operating system. We utilized Hyperledger Fabric 1.1.0 as the Blockchain platform. The algorithms in this study employed the pairing-based cryptography (PBC) library version 0.5.14, with a key size of 160 bits and random number size of 80 bits. The Raspberry Pi 3 B+ served as the IoT device, collecting data for integrity verification.

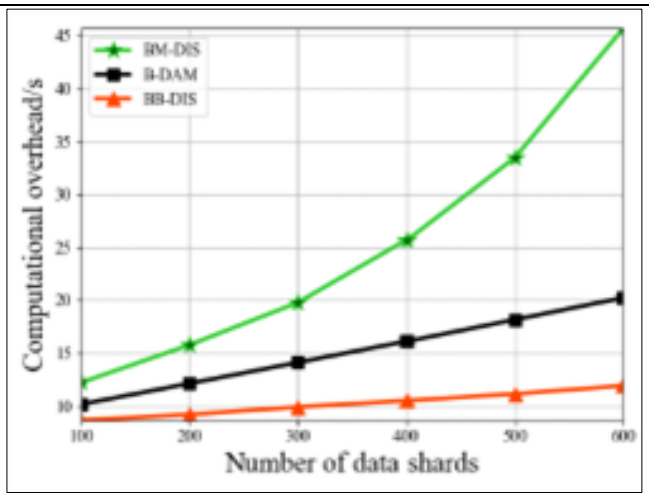
An edge device-based stream data processing structure was established near the data collection layer to process source data and generate the metadata set for IoT data collection. A blockchain network was created on Hyperledger Fabric to provide a trusted environment for data integrity verification. Hyperledger Fabric's capacity exceeds 4000 TPS, compared to Ethereum's 20-30 TPS. Unlike Ethereum and Bitcoin, in Hyperledger Fabric, smart contracts (Chaincodes) run on nodes instead of being stored in blocks, allowing for complex business logic implementation.

5.2. Performance Comparisons

To validate our data integrity scheme (BB-DIS), we conducted a comparative analysis with methods [9], [10], and [11]. Method [9] (B-DIS) uses the blockchain network to store hash results directly. Method [10] (BM-DIS) hashes data shards multiple times using a Merkle tree structure. Method [11] (B-DAM) proposes a Blockchain-based data audit mechanism. The experiment data was averaged over 30 tests.



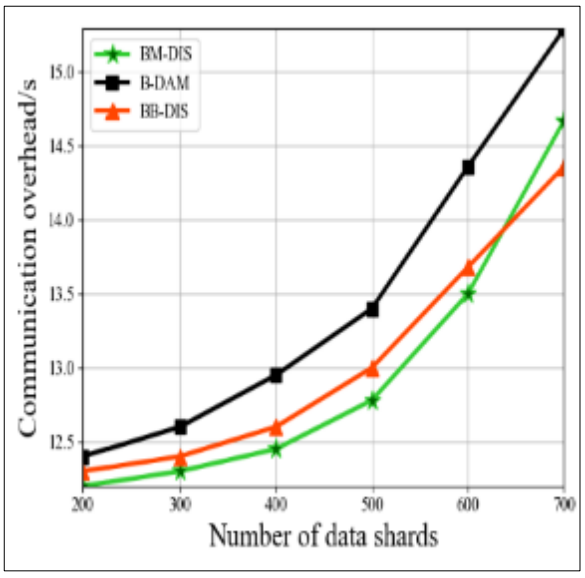
(a) Computational overhead with different



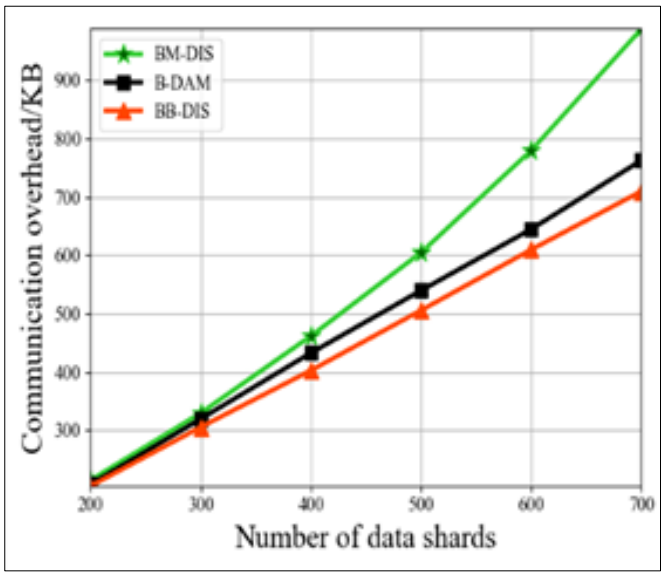
(b) Computational overhead with data size. different number of data shards

Figure 5 Comparison of computational overhead

Figure 5 illustrates the computational overhead for integrity verification across different IoT data scales. We maintained a constant number of shards and samples. In Figure 5(a), when the data size exceeds 150MB, our scheme demonstrates greater efficiency, significantly improving verification speed for large-scale data. Figure 5(b) shows that with a fixed data shard size (20KB), BB-DIS incurs less computational overhead compared to BM-DIS and B-DAM.



(a) Time cost.



(b) Space overhead.

Figure 6 Comparison of communication overhead

The communication overhead refers to the time cost or the amount of data generated during data transmission between each part in the verification process. The experimental results of three methods are shown in Figure 6. From (a), it is evident that our solution's time cost becomes the lowest when there are more than 700 data shards (50KB). As seen in (b), the space overhead increases linearly as the number of sample data shards (1KB) grows. Compared to the other two methods, our solution also has the lowest space overhead. Therefore, we can conclude that our solution is more advantageous for large sample sizes.

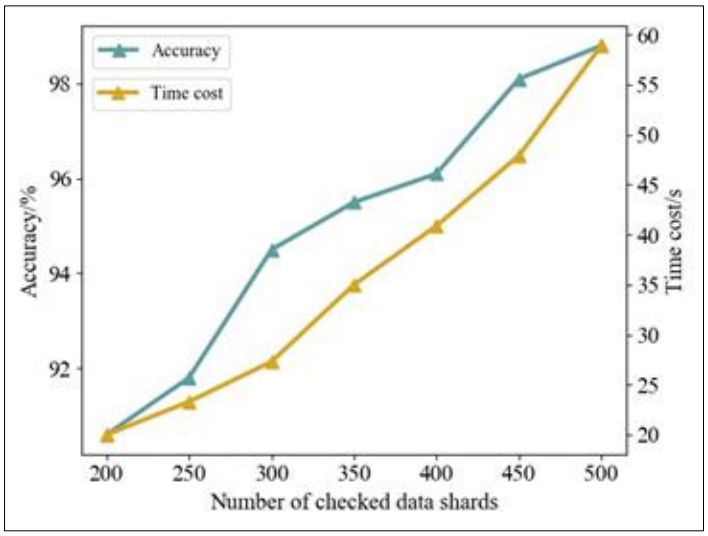
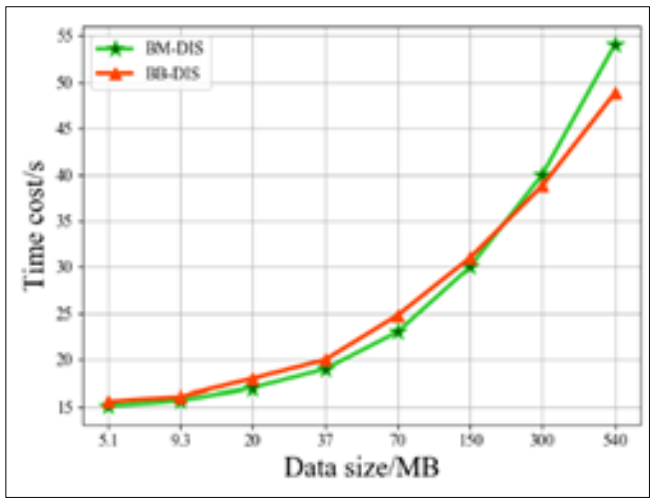


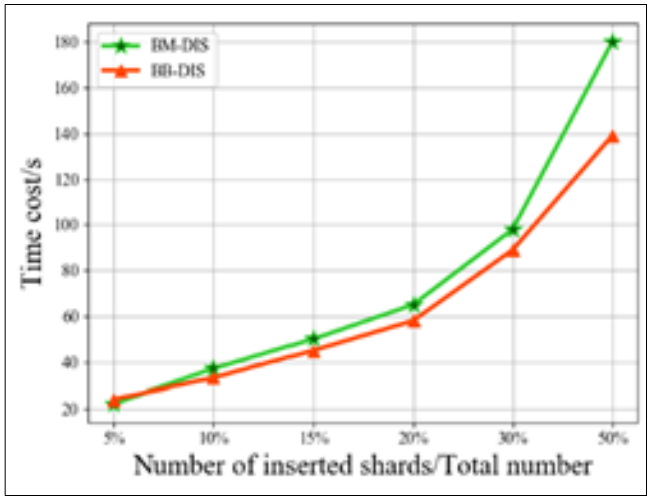
Figure 7 Time cost and accuracy of verification for BB-DIS

Figure 7 illustrates the time cost and accuracy with different numbers of samples for BB-DIS. The total data size is fixed at 10MB, and the time cost increases as the number of checked shards rises. From all the simulation results above, we can conclude that our scheme achieves higher verification efficiency when the data size exceeds 300MB. Additionally, our scheme attains higher accuracy, surpassing 95%, when the number of samples reaches 350.

To demonstrate the dynamic nature of our scheme, we conducted a series of related simulation experiments. Method [10] also exhibits dynamic properties, though the author did not detail them in the article. We used it for comparison and conducted simulation experiments on data appending and data modification operations, as shown in Figures 8 and 9, respectively.

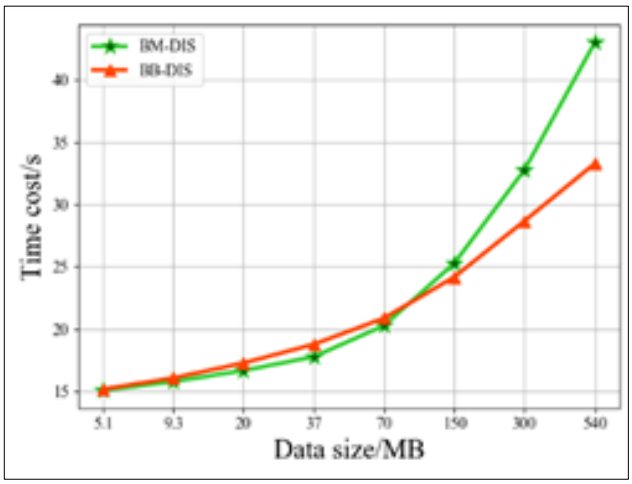


(a) Time cost with different data size.

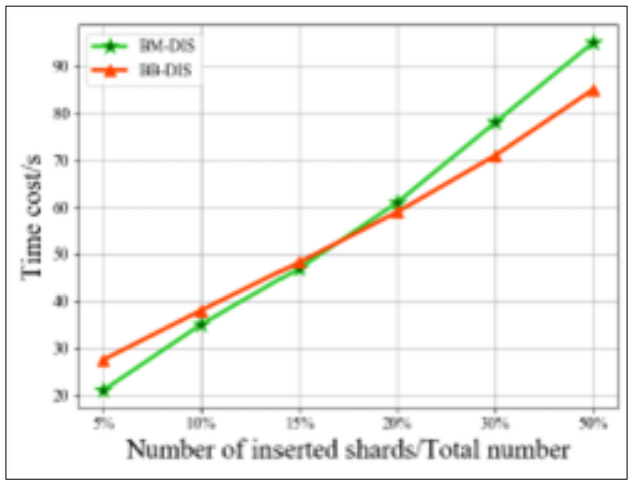


(b) Time cost with different number of inserted shards

Figure 8 Comparison of time cost for appending



(a) Time cost with different data size.



(b) Time cost with different number of inserted shards.

Figure 9 Comparison of time cost for modification

Figures 8 and 9 illustrate that the modification operation takes significantly less time than the appending operation. The time required for data modification depends on the time needed to generate the short signature of the data shards. In contrast, the appending operation time is contingent on the data writing speed to the disk. Clearly, as data size increases, our solution performs well in dynamic operations. Furthermore, the algorithm does not account for the time cost of deletion operations as they do not involve computational overhead.

Next, we compare BB-DIS with three existing blockchain-based data integrity methods: B-DIS, BM-DIS, and B-DAM in Table III. We also highlight the main advantages of our proposed BB-DIS in the following sections.

Table III Performance Comparison

	Dyn.	**Comm.**	**Comp.**	
			CSP	**IVSC**
B-DIS	No	$O(t)$	$O(1)$	$O(1)$
BM-DIS	Yes	$O(c \log n)$	$O(c \log n)$	$O(c \log n)$
B-DAM	Yes	$O(n)$	$O(n)$	$O(n)$
BB-DIS	Yes	$O(n + c)$	$O(c)$	$O(c)$

Note: t denotes the number of data files. n denotes the number of data shards in each file. c denotes the number of shards being challenged (samples). **Dyn.** denotes whether dynamic operations are supported. **Comm.** denotes communication cost. **Comp.** denotes computational complexity.

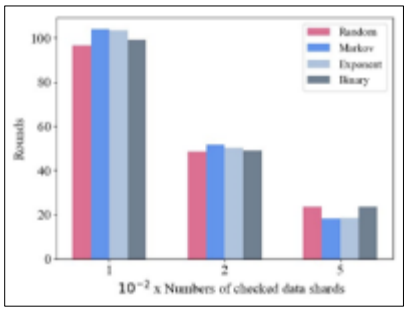
The main advantages of BB-DIS include:

- **Reduced communication overhead**: Unlike BM-DIS, BB-DIS does not require CSP to transmit auxiliary position information with data shards during verification.
- **Lower verification delay**: BM-DIS's verification is slowed by the Merkle tree structure, as calculating the root node requires multiple hashings of the data shards. The more layers the Merkle tree has, the greater the computational cost.
- **No need for a specific hash function**: With BB-DIS, the key size is only 160 bits. Although it requires some initial work, it performs exceptionally well with large data sizes.

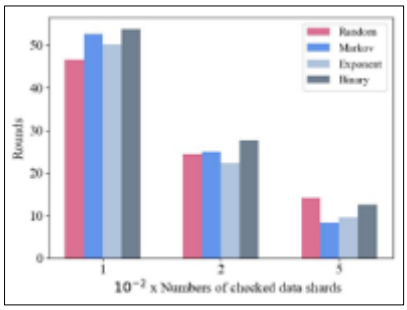
5.3. Sampling Algorithms

Currently, most data integrity methods use a simple random method for sampling and verification. The distribution function of the sample directly impacts the sampling outcome. Therefore, we need to establish an optimal sampling model for the proposed data integrity scheme. Reference [10] deliberately invalidates a sample and compares the effects

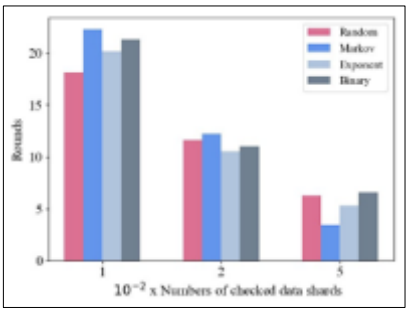
of several sampling models at different sample sizes. However, in reality, the corrupted data in the cloud server is likely to be more than just one sample. The original sampling model may not be effective when the amount of corrupted data increases.



a) Corruption rate d = 0.01%.



(b) Corruption rate d = 0.02%.



(c) Corruption rate d = 0.05%.

Figure 10 Comparison of different sampling algorithm

Figures 10(a), 9(b), and 9(c) show comparisons of different sampling algorithms at corruption rates of 0.01%, 0.02%, and 0.05%, respectively. The four methods compared are simple random distribution, Markov process sampling, exponential distribution sampling, and binomial distribution sampling. With n=10000 data shards, the ordinate indicates the number of rounds needed to detect data corruption. To avoid anomalies, we conducted 30 experiments and calculated the average results.

The experimental results show that as the corruption rate increases, it takes less time to detect the damaged data. In our integrity verification model, when c is small, simple random distribution performs better. When c reaches 500, Markov process sampling shows clear superiority.

6. Conclusion

This paper introduces a data verification integrity scheme leveraging blockchain and bilinear mapping. Initially, we integrate smart contracts with bilinear mapping to propose a new data integrity verification framework. We divide the data into shards and compute metadata for each shard, allowing smart contracts to perform verification. Based on this framework, we develop the corresponding data integrity verification protocol and algorithm. Additionally, we incorporate provable update mechanisms to handle the dynamic nature of IoT data. Furthermore, we present a prototype system utilizing edge computing to process IoT data. Experimental results demonstrate that the proposed BB-BIS surpasses existing blockchain-based methods in computational cost and communication overhead for large-scale IoT data. Future research will focus on extending our scheme to accommodate more complex data types, such as graph data, and addressing data recovery challenges in large-scale IoT environments.

References

[1] F. Xiao, G. Ge, L. Sun, and R. Wang, An energy-efficient data gathering method based on compressive sensing for pervasive sensor networks, *Pervasive Mob. Comput.*, vol. 41, pp. 343–353, Oct. 2017.

[2] M. A. Khan and K. Salah, IoT security: Review, blockchain solutions, and open challenges, *Futur. Gener. Comp. Syst.*, vol. 82, pp. 395-411, May. 2018.

[3] Y. Deswarte, J. J. Quisquater, and A. Saïdane, Remote integrity checking, in *Proc. 6th Working Conf. Integr. Internal Control Inf. Syst. (IICIS)*, 2004, pp. 1–11.

[4] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, Provable data possession at untrusted stores, in *Proc. CCS*, 2007, pp. 598–610.

[5] C. Wang, Q. Wang, K. Ren, and W. Lou, Ensuring data storage security in cloud computing, in *Proc. IEEE Int. Work. Qual. Serv. IWQoS*, 2009, pp. 1–9.

[6] A. Juels and B. S. Kaliski, Pors: Proofs of retrievability for large files, in *Proc. CCS*, 2007, pp. 584–597.

[7] C. Wang, Q. Wang, K. Ren, and W. Lou, Privacy-preserving public auditing for data storage security in cloud computing, in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[8] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks, *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[9] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, Blockchain Based Data Integrity Service Framework for IoT Data, in *Proc. ICWS,* 2017, pp. 468–475.

[10] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, Blockchain Based Data Integrity Verification in P2P Cloud Storage, in *Proc. ICPADS,* 2018, pp. 561–568.

[11] C. Wang, S. Chen, Z. Feng, Y. Jiang, and X. Xue, Block ChainBased Data Audit and Access Control Mechanism in Service Collaboration, in *Proc*. *ICWS,* 2019, pp. 214-218.

[12] F. Xiao, X. Xie, Z. Jiang, L. Sun, and R. Wang, Utility-aware data transmission scheme for delay tolerant networks, *Peer-to-Peer Netw. Appl.,* vol. 9, no. 5, pp. 936-944, Sep. 2016.

[13] H. Gao, Y. Duan, H. Miao, and Y. Yin An Approach to Data Consistency Checking for the Dynamic Replacement of Service Process, *IEEE Access*, vol. 5, pp. 11700–11711, 2017.

[14] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, and J. J. Quisquater, Efficient remote data possession checking in critical information infrastructures, *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, Aug. 2008.

[15] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Ysudik, Scalable and efficient provable data possession in *Proc. SecureComm*., 2008, pp. 1-10.

[16] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing, in *Proc. ESORICS*, 2009, pp. 355-370.

[17] H. Shacham and B. Waters, Compact proofs of retrievability, in *Proc*. *ASIACRYPT*, 2008, pp. 90–107.

[18] B. Chen and R. Curtmola, Robust dynamic remote data checking for public clouds, in *Proc. CCS*, pp. 1043-1045.

[19] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, Identity-based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage, *IEEE Trans. Inf. Forensic Secur.*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[20] F. Xiao, L. Chen, C. Sha, L. Sun, R. Wang, A. X. Liu, and F. Ahmed. Noise tolerant localization for sensor networks, *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp:1701–1714, Jul. 2018.

[21] S. Ben Othman, A. A. Bahattab, A. Trad, and H. Youssef, Confiden- tiality and integrity for data aggregation in WSN using homomorphic encryption, *Wireless Pers. Commun.*, vol. 80, no. 2, pp. 867-889, Jan. 2015.

[22] H. Zhu, Y. Yuan, Y. Chen, Y. Zha, W. Xi, B. Jia, and Y. Xin, A Secure and Efficient Data Integrity Verification Scheme for CloudIoT Based on Short Signature, *IEEE Access*, vol. 7, pp. 90036– 90044, 2019.

[23] F. Zhang, R. Safavi-Naini, and W. Susilo, ''An efficient signature scheme from bilinear pairings and its applications,'' in *Proc. Int. Workshop Public Key Cryptogr*. Berlin, Germany: Springer, 2004, pp. 277-290.

[24] M. U. Arshad, A. Kundu, E. Bertino, A. Ghafoor, and C. Kundu, Efficient and scalable integrity verification of data and query results for graph databases, *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 866-879, Nov. 2017.

[25] F. Reina, H. V. Netto, L. Rech, and A. F. Luiz, A Method to Verify Data Integrity in Graph Databases, in *Proc. IEEE Symp. Comput. Commun.*, 2018, pp. 220–223.

[26] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, ''An overview of blockchain technology: Architecture, consensus, and future trends,'' in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, 2017, pp. 557–564.

[27] A. Suliman, Z. Husain, M. Abououf, M. Alblooshi, and K. Salah, Monetization of IoT data using smart contracts, *IET Netw.*, vol. 8, no. 1, pp. 32–37, Jan. 2019.

[28] H. Albreiki, L. Alqassem, K. Salah, M. H. Rehman, and D. Svetinovi c. (2019). Decentralized Access Control for IoT Using Blockchain and Trusted Oracles. [Online]. Available: https://www.researchgate. net/publication/335258940_Decentralized_Access_Control_for_IoT_ Data_Using_Blockchain_and_Trusted_Oracles

[29] A. Chaer, K. Salah, C. Lima, P. P. Ray, and T. R. Sheltami. (2019). Blockchain for 5G: Opportunities and Challenges. [Online]. Availa ble: https://www.researchgate.net/publication/335518169_Blockchai n_for_5G_Opportunities_and_Challenges

[30] K. Salah, M. H. Ur Rehman, N. Nizamuddin, and A. Al-Fuqaha, ‘‘Blockchain for AI: Review and open research challenges,’’ *IEEE Access*, vol. 7, pp. 10127–10149, 2019.

[31] H. R. Hasan and K. Salah, Combating Deepfake Videos Using Blockchain and Smart Contracts, *IEEE Access*, vol. 7, pp. 4159641606, 2019.

[32] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, ‘‘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)*, Madrid, Spain, May 2017, pp. 468–477.

[33] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, ‘‘Blockchain challenges and opportunities: A survey,’’ *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, Oct. 2018.

[34] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, A User Authentication Scheme of IoT Devices using Blockchain-Enabled Fog Nodes, in *Proc. IEEE/ACS 15th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2018, pp. 1-8.

[35] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, ‘‘Blockstack: A global naming and storage system secured by blockchains,’’ in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, Denver, CO, USA, Jun. 2016, pp. 181–194.

[36] H. R. Hasan and K. Salah, Proof of delivery of digital assets using blockchain and smart contracts, *IEEE Access*, vol. 6, pp. 65439– 65448, 2018.

[37] J. J. Jung, Computational Collective Intelligence with Big Data: Challenges and Opportunities, *Futur. Gener. Comput. Syst.*, vol. 66, pp. 87–88, Jan. 2017.

[38] M. M. Rathore, A. Paul, A. Ahmad, and G. Jeon, ‘‘IoT-based big data: From smart city towards next generation super city planning,’’ *Int. J. Semantic Web Inf. Syst.*, vol. 13, no. 1, pp. 28–47, 2017.

[39] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, Edge Computing: Vision and Challenges, *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[40] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities, *IEEE Netw.*, vol. 33, no. 2, pp. 111- 117, Mar. 2019.