Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7830508, 13 pages
https://doi.org/10.1155/2022/7830508

Research Article

A Scalable Blockchain-Based Integrity Verification Scheme

Zequan Zhou,[1] Xiling Luo,[1,2] Yi Bai,[1,2] Xiaochao Wang,[1,2] Feng Liu,[1] Gang Liu,[3] and Yifu Xu[1]

[1]*School of Electronic and Information Engineering, BeiHang University, Beijing, China*
[2]*Beihang Hangzhou Innovation Institute, Hangzhou, Zhejiang, China*
[3]*Zhejiang Scientific Research Institute of Transport, Hangzhou, Zhejiang, China*

Correspondence should be addressed to Xiling Luo; luoxiling@buaa.edu.cn

Received 2 February 2022; Revised 3 March 2022; Accepted 4 April 2022; Published 10 May 2022

Academic Editor: Yingjie Wang

Ensuring the integrity of remote data is the prerequisite for implementing cloud-edge computing. Traditional data integrity verification schemes make users spend a lot of time regularly checking their data, which is not suitable for large-scale IoT (Internet of Things) data. On the other hand, the introduction of a third-party auditor (TPA) may bring about greater privacy and security issues. We use blockchain to address the problem of TPA. However, implementing dynamic integrity verification with blockchain is a bigger challenge due to the low throughput and poor scalability of blockchain. More importantly, whether there is a security problem with blockchain-based integrity verification is not yet known. In this paper, we propose a scalable blockchain-based integrity verification scheme that implements fully dynamic operations and blockless verification. The scheme builds scalable homomorphic verification tags based on ZSS (Zhang-Safavi-Susilo) short signatures. We exploit smart contract technology to replace TPA for integrity verification tasks, which not only eliminates the risk of privacy leakage but also resists collusion attacks. Furthermore, we formally define a blockchain-based security model and prove that our scheme is secure under the security assumption of cryptographic primitives. Finally, the mathematical analysis of our scheme shows that both the communication complexity and the communication complexity of an audit are $O(c)$, in which c is the number of challenge blocks. We compare our scheme with other schemes, and the results show that our scheme has the lowest time consumption to complete an audit.

1. Introduction

The rapid development of the Internet of Things (IoT) brings huge amounts of data. IoT devices store data on the cloud for cloud-edge computing. However, ensuring the integrity of remote data is the prerequisite for implementing cloud-edge computing [1].

Traditional cloud data integrity verification schemes [2, 3] rely on techniques such as message authentication codes and hash functions to let users know the status of their data. Nonetheless, these heuristics have large computation and communication overheads since users need to retrieve all data. Some schemes [4, 5] reduce the verification overhead of the integrity verification system by constructing homomorphic verification tags. While these schemes enable quick auditing of data, users still need to spend a lot of time auditing their data periodically. To reduce the auditing burden on users, third-party auditors (TPAs) [6] are introduced to perform auditing tasks on cloud data. However, in real-world scenarios, TPAs are not completely trustworthy, and there are two threats [7–9]. First, a malicious TPA may extract data privacy by auditing the same data blocks over and over again. Second, a malicious TPA may collude with cloud servers to produce fake audit results.

Fortunately, blockchain smart contract technology [10, 11] makes it possible to address these issues simultaneously. Smart contracts are encapsulated scripts that can be automated for execution. Therefore, we can use smart contracts to perform auditing tasks instead of TPAs. However, the low throughput and poor scalability of blockchain make it difficult for blockchain to be used in dynamic cloud storage. Therefore, it is a huge challenge to address the scalability of

integrity verification schemes in blockchain network environments. More importantly, whether the security of integrity verification schemes is affected in the open network environment of blockchain should be noticed. To the best of our knowledge, there is no scheme that gives formal security proof. Therefore, it is essential to give proof of security for blockchain-based integrity verification schemes.

In this paper, we propose a scalable blockchain-based integrity verification scheme that enables fully dynamic actions such as insertion, deletion, and modification to address the issues raised above. We create a scalable homomorphic verification tag based on the ZSS (Zhang-Safavi-Susilo) short signature, which uses basic cryptographic hash functions such as SHA-1 or MD5 and does not require expensive specific hash algorithms to accomplish scalability. The scheme supports blockless verification that allows users to audit their data without retrieving all of it. In addition, we use blockchain smart contract technology instead of TPA for the task of integrity verification, which not only eliminates the risk of privacy leakage but also protects against collusion attacks. To evaluate the level of security of our scheme in a blockchain environment, we formally define a blockchain-based security model and demonstrate that the scheme is secure against adaptive chosen message attacks under the security assumption of cryptographic primitives.

1.1. Contributions. The following are the main contributions of this paper:

(1) We propose a scalable blockchain-based integrity verification (SBB-IV) scheme that implements fully dynamic operations and blockless verification. The scheme achieves scalability under blockchain networks by building scalable homomorphic verification tags (HVTs) based on ZSS short signatures, which use general cryptographic hash functions and do not require expensive special hash functions

(2) We exploit smart contract technology to replace TPA for integrity verification tasks, which not only eliminates the risk of privacy leakage but also resists collusion attacks. Furthermore, we formally define a blockchain-based security model that captures the semantic security of adaptive chosen message attacks (CMA). We show that the SBB-IV scheme is secure against adaptive CMA under the security assumption of the q-CAA problem

(3) The mathematical analysis of our scheme shows that both the communication complexity and the communication complexity of the scheme are $O(c)$, in which c is the number of challenge blocks. In addition, we do a series of tests on Hyperledger Fabric V2.2 and compare our scheme to the current state-of-the-art. Our technique is more efficient, as it takes only 2.3 seconds to conduct an audit when 1% of the data blocks are faulty

1.2. Paper Organization. The remainder of this work is arranged in the following manner. We provide an overview of related works in Section 2. Preliminaries are shown in Section 3. The network, threat, framework, protocol, and security model are all shown in Section 4. The detailed algorithms are presented in Section 5. We examine the correctness, dynamic, and security in Section 6. The mathematical analysis and experimental results are presented in Section 7. The paper comes to a close with Section 8.

2. Related Works

2.1. Traditional Data Integrity Verification. Provable data possession (PDP) [12] and proofs of retrievability (POR) [13] are two types of data integrity verification models. The PDP model was formally specified by Ateniese et al. [12], who presented an HVT based on RSA (Rivest-Shamir-Adleman) signatures. They separated the data into blocks and calculated the HVTs for each one. The user then chose a fixed number of blocks for verification at random. Although the sampling approach decreases the computing cost from linear to constant, the scheme is not capable of dynamic operations due to the fixed index of blocks. Juels et al. [13] presented a sentinel-based POR technique in which data segments (sentinels) were randomly inserted into the full data encoded using error correction codes. Due to the limited number of sentinels, it can only undertake limited auditing. BLS (Boneh-Lynn-Shacham) signatures were utilized by Shacham et al. [14] to create HVTs, which reduces communication overhead because the BLS signature is shorter than the RSA signature. Wang et al. [8] described how to build a dynamic PDP system using Merkle tree, an authenticated data structure. Similarly, Erway et al. [15, 16] proposed a skip-list-based dynamic-PDP (DPDP) system. Instead of using a fixed index, these data structures indicate block positions in terms of the order of leaf nodes, allowing blocks to be dynamically inserted at varied locations. However, because these data structures require supplementary information to validate the leaf node placements, they have a computational and communication complexity of $O(\log n)$, making them unsuitable for large-scale data. By first encrypting the data and then providing some precomputed hashes of the encrypted data to the TPA, Shah et al. [6, 17, 18] introduced a TPA to audit the data. The TPA, on the other hand, will be unable to continue auditing after the hashes run out. Furthermore, a hostile TPA may collect information by auditing the same data blocks over and over again. Although random mask approaches [5, 9, 19, 20] have been devised to obscure the linear combination of data and prevent the TPA from extracting it, they are still ineffective in preventing collusion attempts.

2.2. Blockchain-Based Data Integrity Verification. By replacing the integrity management service of centralized nodes with a fully decentralized data integrity service, Liu et al. [10] proposed a blockchain-based Internet of Things (IoT) data integrity service framework that eliminates TPA. However, as they only implemented the proposed protocol's basic features, the efficiency of building smart contracts for IoT devices is insufficient for large-scale IoT data. To assure data availability and privacy, Liang et al. [23] suggested a

Table 1: A comparison between our scheme and the state of art. (Comp. indicates computational complexity and Comm. indicates communication complexity; n indicates the number of all blocks, and c indicates the number of blocks to be audited; – indicates that the scheme does not involve the item.).

(a)

	Wang [8]	Erway [15]	Wang [7]	Hao [21]	Liu [10]	Yue [22]	Our scheme
With help of TPA	Yes	No	Yes	No	No	No	No
Public auditability	Yes	No	Yes	Yes	Yes	Yes	Yes
Privacy protection	No	—	Yes	Yes	Yes	Yes	Yes
Data dynamics	Yes	No	No	Yes	No	Yes	Yes
Support for sampling	Yes	Yes	Yes	No	Yes	Yes	Yes
Blockless	Yes	Yes	Yes	Yes	Yes	No	Yes

(b)

Comm.	$O(c \cdot \log n)$	$O(\log n)$	$O(c)$	$O(1)$	$O(c)$	$O(c \cdot \log n)$	$O(c)$
CSP comp.	$O(c \cdot \log n)$		$O(c)$	$O(n)$	$O(1)$	$O(c \cdot \log n)$	$O(c)$
Audit comp.	$O(c \cdot \log n)$		$O(c)$	$O(n)$	$O(1)$	$O(c \cdot \log n)$	$O(c)$

decentralized and dependable cloud data source protection architecture. The architecture used tamper-proof blockchain records and embedded data provenance in blockchain transactions, with auditors verifying the data's origins based on the information in the blocks. Paying the blockchain miners, on the other hand, would be prohibitively expensive for cloud customers. To address the problem of unreliability in traditional verification procedures, Yue et al. [22, 24] presented a blockchain-based P2P cloud storage data integrity verification methodology. The approach used the Merkle-tree to verify data integrity and examined system performance using various Merkle-tree architectures. Wang et al. [25] proposed a decentralized architecture to tackle the traditional paradigm's single-point trust problem through communal trust. The architecture built a public protocol that maintains the data state under public scrutiny and prevents storage parties from engaging in fraudulent activities. For large-scale IoT data, Wang et al. [11] developed a blockchain-based data integrity verification system. They constructed a prototype system of edge computing processors near IoT devices to preprocess large-scale IoT data and performed data integrity verification in the form of transactions. None of the aforementioned approaches provide formal proof of security, and the security of integrity verification in a blockchain network setting remains an open question. We compared our scheme with the state-of-art, as shown in Table 1.

3. Preliminaries

3.1. Smart Contract. A blockchain is a distributed database that uses encryption, hashing, timestamping, consensus mechanisms, and other techniques [26]. All operations (transactions) are recorded on the blockchain, which is a chained data structure with tamper-proof features. A smart contract is a blockchain-based event-driven program [27]. It is contained within a virtual node that allows automated script execution and data processing in response to event triggers. Smart contracts, like transactions on the blockchain, offer distributed storage and tamper-proof characteristics. Being different from traditional executable programs, smart contracts are distributed and run according to preset rules that create communication protocols between communicating parties [28]. As a consequence, smart contracts enable traceable and irreversible activities without the involvement of a third party.

3.2. ZSS Signature. Let g be the generator of the group $\mathbb{G}$ which is a cyclic additive group with the large prime order p. Allow $\mathbb{G}_T$ to be a cyclic multiplicative cyclic group of order p. Let $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be a bilinear pairing if it satisfies the following properties:

(1) Bilinear: $\forall P, Q \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p$, the equation $e(aP, bQ) = e(P, Q)^{ab}$ holds

(2) Computability: $\forall P, Q \in \mathbb{G}$, there is an effective algorithm to calculate $e(P, Q)$

(3) Nondegenerate: $\exists P, Q \in \mathbb{G}$, such that $e(P, Q) \neq 1$, which means that the map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$. The ZSS signature [29] includes three algorithms: *KeyGen*, *Sign*, and *Verify*. Let $H : \{0,1\}^* \longrightarrow \{0,1\}^\lambda$ be a secure hash function.

(i) *KeyGen*. Randomly select an integer $\alpha \longleftarrow \mathbb{Z}_p^*$, and compute αg. The private key is $sk = \alpha$, and the public key is $pk = \alpha g$

(ii) *Sign*. Given a message $m \in \{0,1\}^*$, the signature is $Sig = \frac{1}{H(m) + \alpha} g$

(iii) *Verify*. Given a signature *Sig*, a public key *pk*, and a message m, compute $H(m)$, and verify the equation:

$$e(g, g) = e(H(m)g + pk, Sig). \quad (1)$$

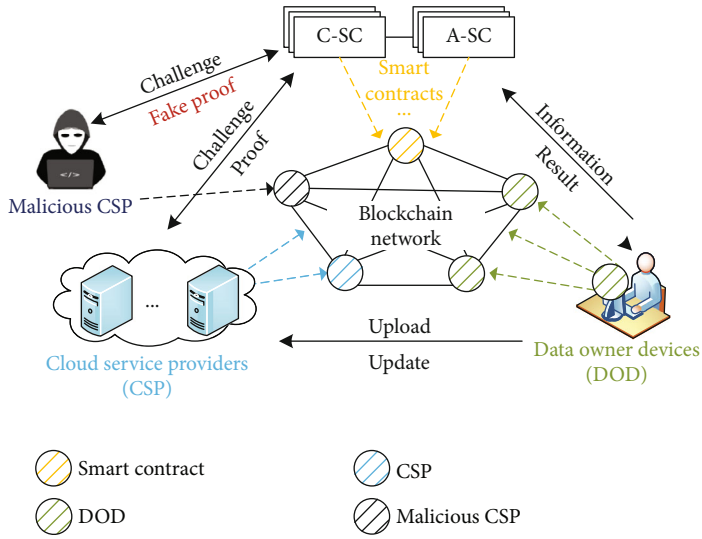

Figure 1: Network model.

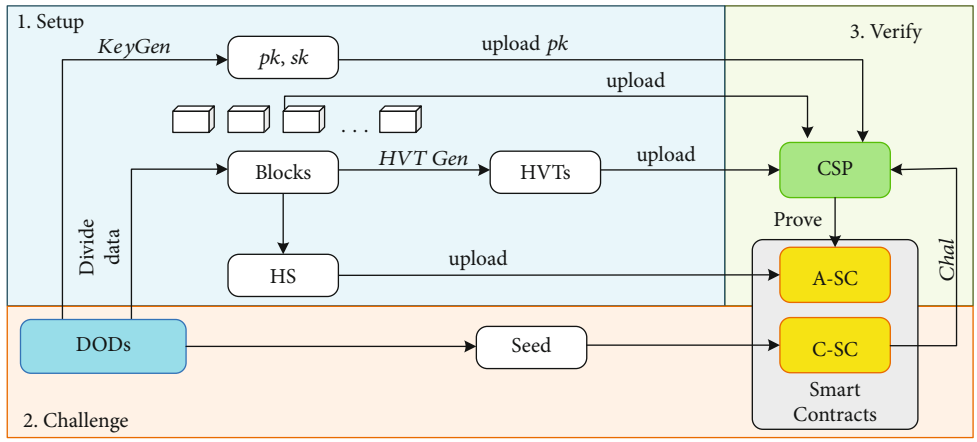

Figure 2: Verification protocol.

If the equation holds, the signature is valid; otherwise, the signature is invalid.

4. Approach Overview

4.1. Network and Threat Model. Figure 1 depicts the SBB-IV scheme's network model, which consists of three entities: data owner devices (DODs), cloud service providers (CSPs), and smart contracts.

(1) DODs: DODs act as nodes on the blockchain network, outsourcing users' data to CSPs and paying for the execution with smart contracts

(2) CSPs: Data storage and maintenance services are provided by CSPs, which are connected to the blockchain network as nodes

(3) Smart contracts: Smart contracts are virtual nodes that contain automated scripts. They cannot be destroyed or modified by any enemy

DODs outsource users' data to CSPs and pay for the execution through smart contracts on the blockchain network. Smart contracts issue a challenge to audit cloud data integrity. Based on the proof created by CSPs, smart contracts use the verification algorithm to check the proof's validity and deliver the outcomes to DODs. Finally, the blockchain keeps track of everything. The TPA collusive attack is avoided in this process because smart contracts are automated execution scripts. As a result, only the threat model described below is considered in this paper.

4.2. Malicious CSP. The malicious CSP knows the data and the public information; the purpose of the malicious CSP is to cheat smart contracts. That is, the malicious CSP owns the knowledge $<Data, public\ information>$ and wants to find fake proof to pass the verification of smart contracts.

4.3. Protocol. The SBB-IV scheme is a collection of five polynomial-time algorithms: $KenGen$, $HVTGen$, $Challenge$, $Response$, and $VerifyProof$ (see detailed algorithm in Section 5-B). Based on the scheme, we create an integrity

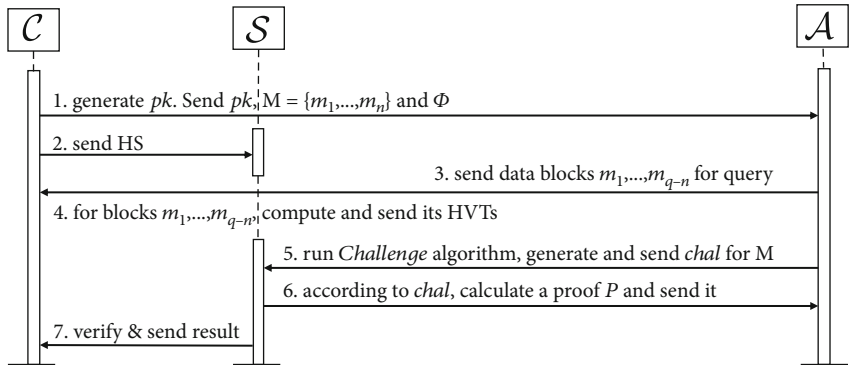

Figure 3: Blockchain-based security model.

verification protocol. Setup, Challenge, and Verify are the three stages of the protocol, as shown in Figure 2.

In the Setup stage, DOD uses the algorithm $(pk, sk) \longleftarrow KeyGen(1^{\kappa})$ to create the pair of keys. It runs the algorithm $(\mathbf{\Phi}, \mathbf{HS}) \longleftarrow HVTGen(sk, \mathbf{M})$ to compute a HVTs sequence and a hash sequence, then uploads the blocks $\mathbf{M} = \{m_1, m_2, \cdots, m_n\}$ and the HVTs sequence $\mathbf{\Phi}$ to CSP, and saves the hash sequence **HS** to audit smart contracts (A-SC).

In the Challenge stage, DOD sends a random *seed* to the challenge smart contract (C-SC). Then, using the algorithm $chal \longleftarrow Challenge(seed)$, C-SC produces a challenge and transmits it to CSP and A-SC.

In the Verify stage, CSP creates a proof using the procedure $P \longleftarrow Response(pk, chal, \mathbf{\Phi}, \mathbf{M})$ and delivers it to A-SC, according to *chal*. The proof is then verified by A-SC using the algorithm $VerifyProof(pk, chal, P, \mathbf{HS})$, and the result is sent to DOD.

4.4. Blockchain-Based Security Model. An interactive game between a challenger $\mathscr{C}$, a smart contract $\mathscr{S}$, and an adversary $\mathscr{A}$ defines the blockchain-based security model. In the Setup phase, we convey the challenged data **M** to the adversary to capture the semantic security of adaptive chosen message attack. As a result, in the Query phase, the adversary might adaptively pick multiple data blocks for the HVT query. The game is played in the following manner:

(i) Setup: $\mathscr{C}$ generates and sends a public key *pk* to $\mathscr{A}$. Then, $\mathscr{C}$ sends a data $\mathrm{M} = \{m_1, m_2, \cdots, m_n\}$ and its HVTs sequence Φ to $\mathscr{A}$. Finally, $\mathscr{C}$ sends the hash sequence HS to $\mathscr{S}$

(ii) Query: $\mathscr{A}$ adaptively makes queries; the selected $q - n$ blocks $m_1, \cdots, m_{q-n}$ is sent to $\mathscr{C}$. According to queries, $\mathscr{C}$ computes HVTs for all m_j, where $1 \le j \le (q - n)$, and returns the HVLs. Note that m_j can be a block of the data $M = \{m_1, m_2, \cdots, m_n\}$

(iii) Challenge: $\mathscr{S}$ generates *chal* for M by running the *Challenge* algorithm

(iv) Forge: According to the challenge *chal*, $\mathscr{A}$ calculates a proof P for M and sends it to $\mathscr{S}$

(v) Verify: $\mathscr{S}$ verifies the proof P by executing the algorithm $VerifyProof$. If P is valid, $\mathscr{A}$ wins the game

The game process can be referred to as Figure 3. The security definition of the scheme is as follows.

Definition 1. If any probabilistic polynomial-time adversary $\mathscr{A}$ cannot win the game with nonnegligible probability, the SBB-IV scheme is secured against the adaptive chosen messages attack when the integrity of the remote data **M** is violated.

5. Our Schemes

5.1. Notations. We employ a pseudorandom permutation function (PRP), $\pi : \{0,1\}^{log_2(n)} \times \{0,1\}^{\kappa} \longrightarrow \{0,1\}^{log_2(n)}$, and a pseudorandom function (PRF), $f : \{0,1\}^{*} \times \{0,1\}^{\kappa} \longrightarrow \mathbb{Z}_p$, in addition to the symbols specified in the preceding section. Aside from that, $H : \{0,1\}^{*} \longrightarrow \{0,1\}^{\lambda}$ is a secure generic hash function.

5.2. Scheme Detail. The SBB-IV scheme is described in full in this section:

(1) $KeyGen(1^{\kappa}) \longrightarrow (pk, sk)$. The algorithm selects a random number from the ring, $\alpha \longleftarrow \mathbb{Z}_p^*$, as a private key *sk*, and then computes $Y = \alpha g$ as a public key, according to the security parameter κ

(2) $HVTGen(sk, \mathbf{M}) \longrightarrow (\mathbf{\Phi}, \mathbf{HS})$. The algorithm firstly splits a data **M** into n equal length blocks; that is, $\mathbf{M} = \{m_1, m_2, \cdots, m_n\}$. Next, for $1 \le i \le n$, it calculates the hash value $H(m_i)$ for a block m_i and then computes the HVT as following equation:

$$\delta_i = \frac{1}{H(m_i) + m_i + \alpha} g. \tag{2}$$

Finally, it outputs a HVTs sequence, $\mathbf{\Phi} = \{\delta_1, \delta_2, \cdots, \delta_n\}$, and a hash sequence $\mathbf{HS} = \{H(m_1), H(m_2), \cdots, H(m_n)\}$.

(3) $Challenge(seed) \longrightarrow chal$. Based on the $seed = (c, \kappa)$, the algorithm chooses two random numbers (k_1, k_2), where $1 \le k_1 \le \kappa$, $1 \le k_2 \le \kappa$. For $1 \le j \le c$, it computes $i_j = \pi_{k_1}(j)$ and $v_j = f_{k_2}(j)$ by the PRR function and the PRF function. The output is $chal = \{(i_j, v_j)\}_{1<j<c}$

(4) $Response(pk, chal, \mathbf{\Phi}, \mathbf{M}) \longrightarrow P$. According to the challenge, the algorithm's calculation is as follows:

$$\begin{aligned} \theta &= \sum_{j=1}^{c} v_j Y, \\ u &= \sum_{j=1}^{c} v_j m_{i_j}, \\ \eta &= g - g^2 \sum_{j=1}^{c} \frac{v_j}{\delta_{i_j}}. \end{aligned} \tag{3}$$

Lastly, it outputs $P = \{\theta, u, \eta\}$.

(5) $VerifyProof(pk, chal, P, \mathbf{HS}) \longrightarrow \{1, 0\}$. The algorithm accepts pk, P, $chal = \{(i_j, v_j)\}_{1<j<c}$ and **HS** as input and then calculates:

$$\begin{aligned} \nu &= \sum_{j=1}^{c} v_j H\left(m_{i_j}\right) g, \\ \mu &= \nu + ug. \end{aligned} \tag{4}$$

Finally, the output is depending on whether the following equation holds:

$$e(\eta, g) \cdot e(\mu + \theta, g) = e(g, g). \tag{5}$$

If the equation holds, the algorithm outputs 1; otherwise, it outputs 0.

5.3. Dynamic. Because the HVT built in the scheme (as shown in Equation (2)) is based solely on the block and excludes a fixed numerical index, it can enable completely dynamic operations such as modification, insertion, and deletion. The technique generates a hash sequence that records the location of each block. As a consequence, the following procedure is used to update the data:

Step 1: DOD delivers an request to A-SC, $Request = (ope, pos, con)$, where *ope* represents the updated operations, *pos* denotes the updated position, and *con* represents the updated content. Note that when a delete operation is performed, *con* is empty

Step2: According to the request, A-SC performs the corresponding update operation and records the modification on the blockchain

Step3: When the blockchain is recorded successfully, DOD sends the *Request* to CSP to complete the update

5.4. Implementation. In our scheme, we encapsulate the *Challenge* algorithm and the *VerifyProof* algorithm into smart contracts to perform the task of auditing instead of TPA. Users can trigger the execution of smart contracts by sending *seed*. This *Seed* not only includes the number of audit blocks and security parameters, but users can also set parameters such as the audit cycle time and the number of audits performed according to their needs. The parameter c is the number of randomly sampled blocks in one audit. The parameter c is larger, the higher the audit confidence and the higher the computational overhead. Therefore, users set different c according to their needs to make a trade-off between different confidence levels and computation overhead. The tamper-proof nature of smart contracts eliminates the possibility of privacy leakage and collusion attacks. Because to the collision resistance and one-way nature of the hash function, an attacker cannot access the data through the hash value, despite the fact that we have put the hash sequence on the public smart contract.

6. Scheme Analysis

6.1. Correctness. The PRP and PRF functions in the *Challenge* algorithm of the SBB-IV scheme ensure that the blocks are randomly picked for each audit, making it impossible for a malicious CSP to prepare proofs ahead of time. If the remote data is preserved, the proof P generated by the *Response* algorithm will always pass the *VerifyProof* algorithm's verification. The scheme is correct in the following ways:

$$\begin{aligned} &e(\eta, g) \cdot e(\mu + \theta, g) \\ &\quad = e\left(\sum_{j=1}^{c} v_j H\left(m_{i_j}\right) \cdot g + \sum_{j=1}^{c} v_j m_{i_j} g + \sum_{i=1}^{c} v_j Y, g\right) \cdot \\ &e\left(g - g^2 \sum_{j=1}^{c} \frac{v_j}{\delta_{i_j}}, g\right) \\ &\quad = e(g, g) \cdot e\left(-g \sum_{j=1}^{c} v_j \left(H\left(m_{i_j}\right) + m_{i_j} + \alpha\right), g\right) \cdot \\ &e\left(g \sum_{j=1}^{c} v_j \left(H\left(m_{i_j}\right) + m_{i_j} + \alpha\right), g\right) \\ &\quad = e(g, g) \cdot e(g, g)^{-\sum_{j=1}^{c} v_j \left(H\left(m_{i_j}\right) + m_{i_j} + \alpha\right)} \\ &\qquad \cdot e(g, g)^{\sum_{j=1}^{c} v_j \left(H\left(m_{i_j}\right) + m_{i_j} + \alpha\right)} = e(g, g). \end{aligned} \tag{6}$$

6.2. Security. We treat the hash function $H(\cdot)$ as a random oracle and reduce the security of the SBB-IV scheme to the q-CAA problem [21].

Definition 2 (q-CAA problem). For an integer q, and $\alpha \in_R \mathbb{Z}_p$, $g \in \mathbb{G}$, given

$$\left\{ g, Y = \alpha g, \frac{1}{w_1 + \alpha} g, \cdots, \frac{1}{w_q + \alpha} g \right\}, \tag{7}$$

where $w_1, \cdots, w_q \in_R \mathbb{Z}_p$, to compute $(1/w + \alpha)g$ for some $w \notin \{w_1, \cdots, w_q\}$.

q-CAA assumption. The q-CAA problem is (t, ε)-hard if for a t-time adversary $\mathscr{A}$, the advantage of $\mathscr{A}$ to solve the problem is negligible:

$$Adv_{\mathscr{A}} = \Pr\left[\mathscr{A}\left(g, \alpha g, \frac{1}{w_1 + \alpha} g, \cdots, \frac{1}{w_q + \alpha} g \right) = \frac{1}{w + \alpha} g \right] \leq \varepsilon, \tag{8}$$

where ε is a negligible probability, and $w_1, \cdots, w_q \in_R Z_p$.

Theorem 3. *Suppose the (t, ε)-q-CAA assumption holds in the group $\mathbb{G}$, our scheme is (t, ε)-secure against adaptive chosen message attack under the random oracle model.*

Proof. If an adversary $\mathscr{A}$ can break the security of the SBB-IV scheme, we will show a challenger $\mathscr{C}$ how to use $\mathscr{A}$ to solve the q-CAA problem. The challenger $\mathscr{C}$ has known that $g \in \mathbb{G}, Y = \alpha g, w_1, w_2, \cdots, w_q$ and $\delta_1 = (1/w_1 + \alpha)g, \delta_2 = (1/w_2 + \alpha)g, \cdots, \delta_q = (1/w_q + \alpha)g$, and her goal is to calculate $(1/w + \alpha)g$ for some $w \notin \{w_1, w_2, \cdots, w_q\}$. Therefore, an interactive game between a challenger $\mathscr{C}$, a smart contract $\mathscr{S}$, and an adversary $\mathscr{A}$ as follows:

(i) Setup: The challenger $\mathscr{C}$ generates the public key pk, $Y = \alpha g$ and sends it to $\mathscr{A}$. Then, $\mathscr{C}$ selects a data $M = \{m_1, m_2, \cdots, m_n\}$ and constructs its HVTs sequence Φ and its hash sequence HS as follows. $\mathscr{C}$ maintains a list of tuples $<w_i, H_i, m_i>$. The list is initially empty. For a block m_i, $\mathscr{C}$ selects a $w_i \in \{w_1, \cdots, w_n\}$ and computes $H_i = w_i - m_i$; its HVT is $\delta_i = (1/w_i + \alpha)g = (1/H_i + m_i + \alpha)g$. Then $\mathscr{C}$ adds the tuple $<w_i, H_i, m_i>$ to the list and removes w_i from w-parameters $(w_1, \cdots, w_q)$. Finally, $\mathscr{C}$ sends the HVTs sequence to $\mathscr{A}$ and the hash sequence HS to $\mathscr{S}$

(ii) Query: The adversary $\mathscr{A}$ adaptively selects $q - n$ different blocks $m_1, m_2, \cdots, m_{q-n}$ and sends them to $\mathscr{C}$ for HVTs queries. Note that m_j $(1 \leq j \leq (q - n))$ can be a block of the data $M = \{m_1, m_2, \cdots, m_n\}$. At any time $\mathscr{A}$ can query hash value. When $\mathscr{A}$ queries at m_j, $\mathscr{C}$ responds as follows: [1)]

(1) Hash query. $\mathscr{C}$ firstly checks if the query m_j already exists in the list $<w_i, H_i, m_i>$. If so, $\mathscr{C}$ responds with H_i; otherwise, $\mathscr{C}$ randomly selects a w_j in the remaining w-parameters and responds with $H_j = w_j - m_j$ and adds the tuple $<w_j, H_j, m_j>$ to the list and removes w_j from w-parameters

(2) HVT query. $\mathscr{C}$ firstly checks if the query m_j already exists in the list $<w_i, H_i, m_i>$. If so, $\mathscr{C}$ responds with corresponding $\delta_j = (1/w_j + \alpha)g = (1/H_j + m_j + \alpha)g$; otherwise, $\mathscr{C}$ randomly selects a w_j in the remaining w-parameters and computes $H_j = w_j - m_j$ and responds with corresponding δ_j. Then, $\mathscr{C}$ adds the tuple $<w_j, H_j, m_j>$ to the list and removes w_j from w-parameters

(iii) Challenge: $\mathscr{S}$ generates *chal* for M by running the *Challenge* algorithm

(iv) Forge: According to the challenge *chal*, $\mathscr{A}$ calculates a proof P for M and delivers it to $\mathscr{S}$

(v) Verify: $\mathscr{S}$ verifies the proof P by executing the algorithm $VerifyProof$. If $VerifyProof$ outputs 1, $\mathscr{A}$ wins the game

When the block audited is corrupted, suppose that there is a block m_j corrupted and $\mathscr{A}$ can forge a fake proof that passes the verification with a nonnegligible probability.

We assume that the fake proof is $P^* = \{\theta^*, u^*, \eta^*\}$, where

$$\begin{aligned} \theta^* &= \sum_{i=1}^{c} v_i Y, \\ u^* &= \sum_{i=1, i \neq j}^{c} v_i m_i + v_j m_j^*, \\ \eta^* &= g - g^2 \sum_{i=1, i \neq j}^{c} \frac{v_i}{\delta_i} - g^2 \frac{v_j}{\delta_j^*}. \end{aligned} \tag{9}$$

When $\mathscr{S}$ verifies the proof P^* by executing the algorithm $VerifyProof$, it computes

$$\begin{aligned} \nu &= \sum_{i=1}^{c} v_i H(m_i) g, \\ \mu^* &= \nu + u^* g. \end{aligned} \tag{10}$$

Therefore, the process of verification is as follows:

$$\begin{aligned}
&e(\eta^*, g)\cdot e(\mu^*+\theta^*, g)\\
&= e\left(g\sum_{i=1}^{c} v_i H(m_i) + \sum_{i=1 i\neq j}^{c} v_i m_i g + v_j m_j^* g + \sum_{i=1}^{c} v_i Y, g\right)\\
&\quad \cdot e\left(g - g^2 \sum_{i=1 i\neq j}^{c} \frac{v_i}{\delta_i} - g^2 \frac{v_j}{\delta_j^*}, g\right)\\
&= e(g,g)\cdot e(g,g)^{-\sum_{i=1 j\neq j}^{c} v_i (H(m_i)+m_i+\alpha)} \cdot e(g,g)^{-g\left(v_j/\delta_j^*\right)}\\
&\quad \cdot e(g,g)^{\sum_{i=1 j\neq j}^{c} v_i (H(m_i)+m_i+\alpha)} \cdot e(g,g)^{v_j\left(H\left(m_j\right)+m_j^*+\alpha\right)}\\
&= e(g,g)\cdot e\left(-g^2\frac{v_j}{\delta_j^*}, g\right)\cdot e(g,g)^{v_j\left(H\left(m_j\right)+m_j^*+\alpha\right)}\\
&= e(g,g)\cdot e(g,g)^{-g\left(v_j/\delta_j^*\right)}\cdot e(g,g)^{v_j\left(H\left(m_j\right)+m_j^*+\alpha\right)}\\
&= e(g,g)\cdot e(g,g)^{-g\left(v_j/\delta_j^*\right)+v_j\left(H\left(m_j\right)+m_j^*+\alpha\right)}.
\end{aligned} \tag{11}$$

If the fake proof passes the verification, we get $e(\eta^*, g)\cdot e(\mu^*+\theta^*, g) = e(g,g)$. Hence, from the above derivation, we get the following equation:

$$e(g,g)\cdot e(g,g)^{-g\left(v_j/\delta_j^*\right)+v_j\left(H\left(m_j\right)+m_j^*+\alpha\right)} = e(g,g). \tag{12}$$

where $-g(v_j/\delta_j^*) + v_j(H(m_j) + m_j^* + \alpha) = 0$. That is, $g(1/\delta_j^*) = H(m_j) + m_j^* + \alpha$. As a result, we get $\delta_j^* = (1/H(m_j) + m_j^* + \alpha)g$. Since we have assumed that $m_j \neq m_j^*$, we will discuss it in two cases.

(i) Case 1: $H(m_j) + m_j^* = H(m_j) + m_j$.

In this case, we get $m_j^* = m_j$, which contradicts our hypothesis. Therefore, this case proves that the block m_j is not corrupted when the adversary $\mathcal{A}$ wins the game.

(ii) Case 2: $H(m_j) + m_j^* \neq H(m_j) + m_j$.

This case shows that when the adversary $\mathcal{A}$ finds a fake HVT δ_j^* with a nonnegligible probability in a time t, the challenger $\mathcal{C}$ finds a $(1/w + \alpha)g$ for some $w \notin \{w_1, \cdots, w_q\}$ with same nonnegligible probability in a time t, which means $\mathcal{C}$ breaks the q-CAA problem.

In summary, suppose the (t, ε)-q-CAA assumption holds in the group $\mathbb{G}$, our scheme is (t, ε)-secure against adaptive chosen message attack under the random oracle model. □

6.3. Scalability. In an IoT data storage system, with the continuous increase of IoT data, cloud storage needs to have scalability. In the SBB-IV scheme, we divide large data into smaller blocks, which is beneficial to the fine-grained control of the data and enhances the scalability of the cloud storage system. In the meanwhile, the proposed scheme is fully dynamic which means node devices can insert, modify, and delete uploaded data according to their needs. Furthermore, the scheme can be compatible with more systems without compromising efficiency, since HVTs are computed using general cryptographic hash functions rather than expensive elliptic curve hash functions [30]. As a result, the scheme is suitable for the integrity verification of large-scale IoT data.

In addition to IoT systems, our scheme can also be applied to a blockchain-based P2P (peer-to-peer) file system. In this system, an edge device is a peer node, and each peer node can become a client or server. Our scheme solves the bandwidth problem of sharing files from a central server to clients. Files can be shared through different nodes without requesting all files from a central server. At the same time, due to the homomorphism of HVTs, the speed of nodes verifying file integrity is greatly improved. Therefore, the SSB-IV scheme greatly improves the scalability and efficiency of file sharing

7. Evaluation

To justify the performance of the SBB-IV scheme, we conduct mathematical analysis and a series of experiments in this part. The pairing-based cryptography library (PBC, http://crypto.stanford.edu/pbc/) is used in our experiments. The experiments are implemented in the GoLang programming language and run on an Intel(R) Core(TM) i7-10700 CPU with 16 GB of RAM. The blockchain platform is Hyperledger Fabric 2.2.0. The security level has been set to 80 bits, implying that the $|p| = 160$. We set each block's size to 8 KB and produce 1000, 5000, 10000, 50000, and 100000 blocks for the test. We present the average values across these 10 trials throughout the examination.

7.1. Mathematical Analysis. We calculate the computation complexity of the SBB-IV scheme. Users execute the algorithms *KeyGen* and *HVTGen*; smart contracts run the algorithms *Challenge* and *VerifyProof*; CSPs runs the algorithm *Response*. The complexity of each algorithm is shown in Table 2.

(i) *KeyGen*: This algorithm performs only one multiplication

(ii) *HVTGen*: In this algorithm, since each block is needed to compute a HVT, the overhead of the algorithm is $O(n)$

(iii) *Challenge*: According to the parameter c, C-SC needs to use the PRF function and the PRP function to calculate two groups of random numbers. Hence, the computation overhead is $O(c)$

(iv) *Response*: This algorithm needs to calculate $\theta = \sum_{j=1}^{c} v_j Y$, $u = \sum_{j=1}^{c} v_j m_{i_j}$, and $\eta = g - g^2 \sum_{j=1}^{c} (v_j/\delta_{i_j})$. The computation cost is $O(c)$

Table 2: Complexity analysis.

(a)

Algorithm	Computation	Complexity
KeyGen	$Mult_{\mathbb{G}}^{1}$	$O(1)$
HVTGen	For $m_1, m_2, \cdots, m_n$ computes $\delta_1, \delta_2, \cdots, \delta_n$	$O(n)$
Challenge	$PRF^c + PRP^c$	$O(c)$
Response	$2Mult_{Z_p}^{c} + 2Add_{Z_p}^{c-1} + Mult_{\mathbb{G}}^{c+2} + Add_{\mathbb{G}}^{c}$	$O(c)$
VerifyProof	$Mult_{\mathbb{G}}^{c+2} + Add_{\mathbb{G}}^{c} + BM_{\mathbb{G}}^{3}$	$O(c)$

(b)

Phase	Communication	Complexity
Setup	User sends pk, **M**, **Φ**, **HS**	$O(n)$
Challenge	A-SC sends $chal = \{(i, v_i)\}_{s_1 \le i \le s_c}$	$O(c)$
Verify	CSP sends $P = \{\theta, u, \eta\}$	$O(1)$

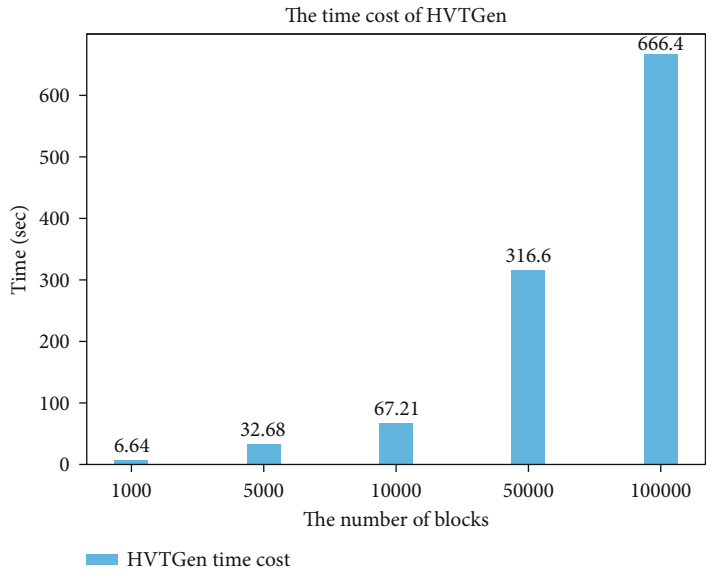

Figure 4: Comparison of computational overhead.

(v) *VerifyProof*: This algorithm needs to verify the equation $e(\eta, g) \cdot e(\mu + \theta, g) = e(g, g)$, where $\mu = \nu + ug$, $\nu = \sum_{j=1}^{c} v_j H(m_{i_j}) g$. The computation cost is $O(c)$

In the Setup phase, user sends pk, M, Φ, and HS, in which the communication complexity is $O(n)$. In the Challenge phase, C-SC sends challenge $chal = \{(i, v_i)\}_{s_1 \le i \le s_c}$ which is $2c|p|$ bits. In the Verify phase, CSP sends proof $P = \{\theta, u, \eta\}$ which is $3|p|$ bits. Therefore, the communication complexity of an audit is $O(c)$.

7.2. Experiments. In this section, we evaluate the actual performance of the scheme with a series of experiments.

7.2.1. Setup. In the Setup stage, the user's main computation overhead comes from the *HVTGen* algorithm. At the same time, the smart contract needs to store a hash sequence **HS** . In our experiments, we set the number of blocks to 1,000, 5,000, 10,000, 50,000, and 100,000, respectively. Because each block is 8 KB in size, 100,000 blocks represent 780 MB of data. As shown in Figure 4, the time consumption of the *HVTGen* algorithm grows linearly, but the algorithm only needs to be executed once. For 780 MB of data, the smart contract's storage consumption is only 15.04 MB, which is easily achievable for a distributed ledger (Figure 5).

7.2.2. Audit. As we discussed in Section 5, in the *Challenge* algorithm, the parameter c is larger, the higher the audit

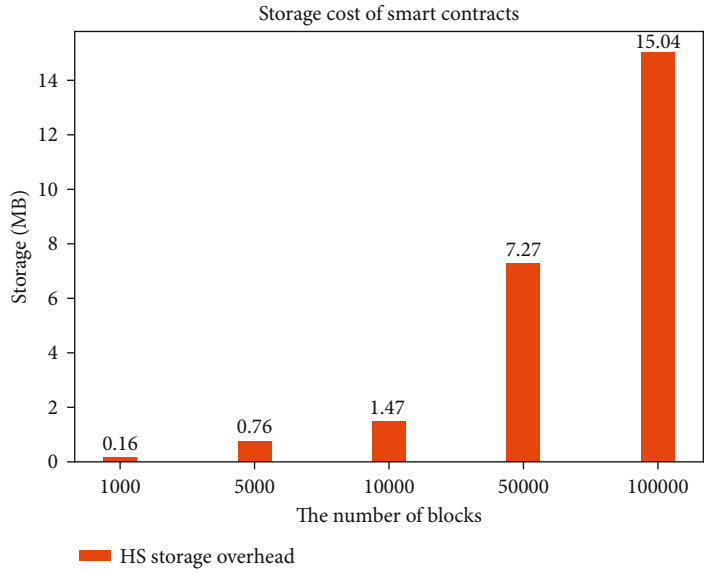

Figure 5: Storage overhead.

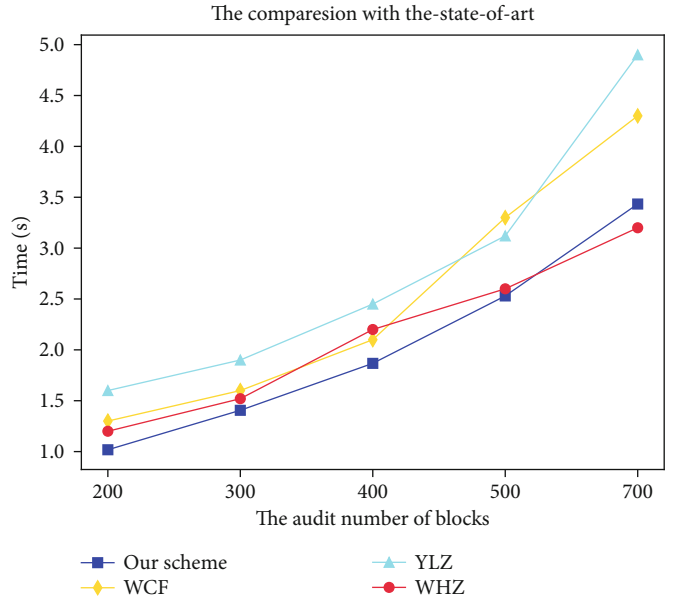

Figure 6: The time cost of an audit.

confidence, and the higher the computational overhead. Therefore, users set different c according to their needs to make a trade-off between different confidence levels and computation overhead.

To fully understand the time consumption of an audit, we locally tested the overall time consumption of three algorithms which include the *Challenge* algorithm, the *Response* algorithm, and the *VerifyProof* algorithm. We select other three blockchain-based schemes (YLZ-[22], WCF-[25], and WHZ-[11]) for comparison. In our experiments, we audit numbers c to 200, 300, 400, 500, and 700, respectively. The experimental results (as presented in Figure 6) show that our scheme has the lowest overall time consumption for one audit.

In addition, we test the time consumption of the *Response* algorithm running locally and the time consumption of the *VerifyProof* algorithm running in the encapsulated smart contract. Our blockchain platform uses Hyperledger Fabric 2.2.0, and we build a test network on a virtual machine (Ubuntu 20.04). Let P_x indicates a probability and t means the number of corrupted blocks, we get

$$P_x \geq 1 - \left(\frac{n-t}{n}\right)^c, \tag{13}$$

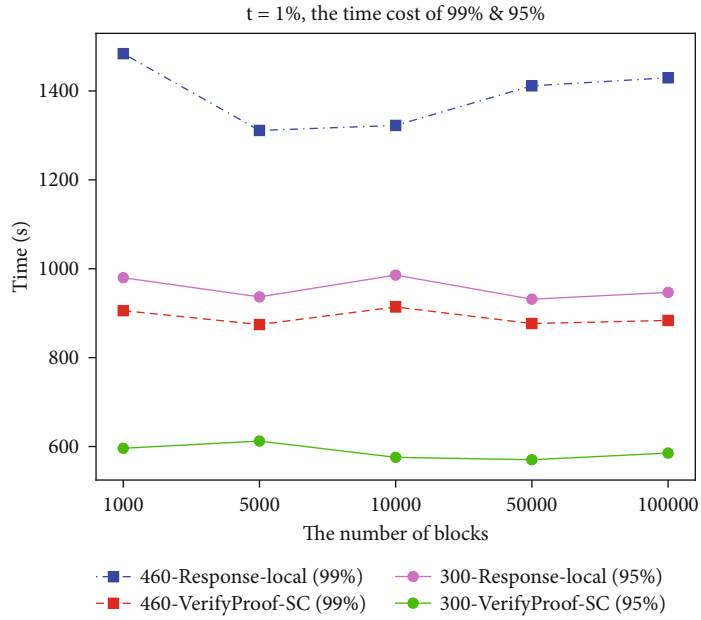

Figure 7: The time cost of Response and VerifyProof algorithms (99% and 95%).

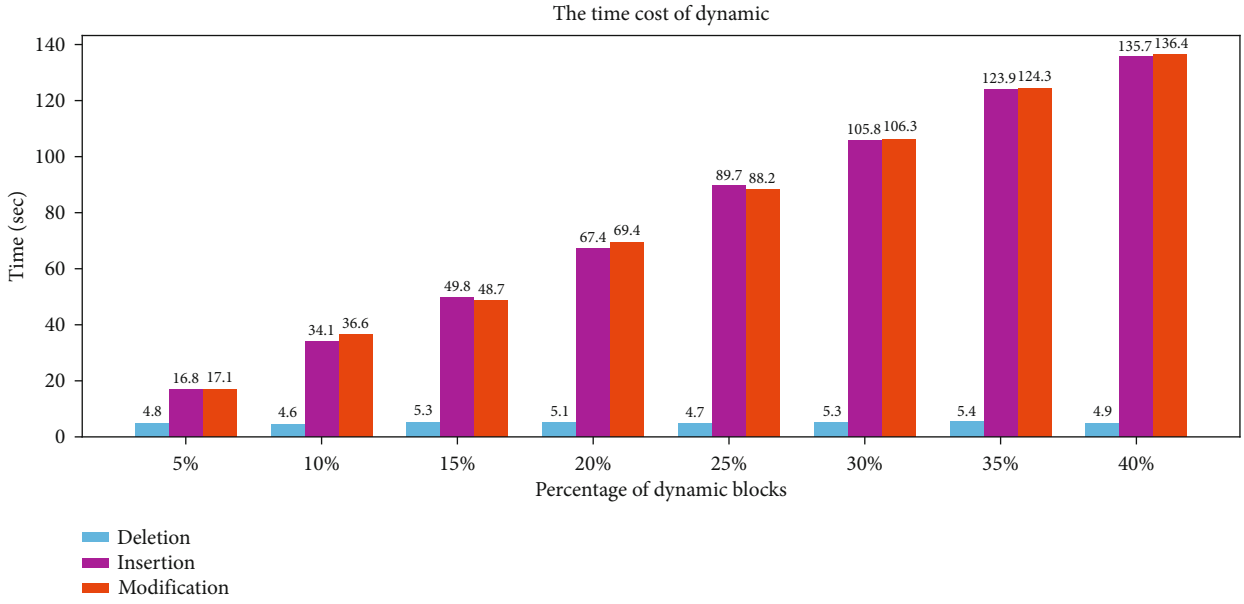

Figure 8: Time cost of dynamicoOperations.

Equation (13) shows that when t blocks are corrupted, different values of c will produce different confidence levels. Therefore, we assume that when 1% blocks corrupted, we set $c = 300$ and $c = 460$ to get 95% and 99% confidence, respectively, As presented in Figure 7, the time cost of *Response* and *VerifyProof* algorithm will remain even with the increase of the number of blocks.

7.2.3. Dynamic. For dynamic simulation tests, we use $n = 50000$. The time it takes to edit a block in our situation is determined by the time it takes for the blockchain to write the record and the time it takes to produce HVTs. We configured an endorsement node in our test network to write operation records to the Hyperledger (https://hyperledger-fabric.readthedocs.io/en/latest/index.html). The time consumption of insertion and modification is linear as the number of dynamic blocks rises, but the time consumption of deletion remains constant, as shown in Figure 8. Because deletion does not involve the creation of new HVTs, insertion and modification take longer than deletion. The time spent on deletion is primarily due to the time spent writing records by the endorsing node. Because the method for both operations is the same, insertion and modification take almost the same amount of time.

8. Conclusion

This paper mainly solves three problems, including the problem of TPA's privacy leakage and collusion attack, the problem of poor blockchain scalability, and the security problem of blockchain-based integrity verification schemes. To address the problems above, we propose a scalable blockchain-based integrity verification scheme that implements fully dynamic operations and blockless verification. The scheme builds scalable homomorphic verification tags based on ZSS short signatures. We exploit smart contract technology to replace TPA for integrity verification tasks, which not only eliminates the risk of privacy leakage but also resists collusion attacks. Furthermore, we formally define a blockchain-based security model that captures the semantic security of adaptive chosen message attacks. We show that our scheme is secure under the security assumption of cryptographic primitives. Finally, the mathematical analysis of our scheme shows that both the communication complexity and the communication complexity of an audit are $O(c)$, in which c is the number of challenge blocks. We compare our scheme with other schemes, and the results show that our scheme has the lowest time consumption to complete an audit.

Data Availability

The data that support the findings of this study are not openly available due to reasons of sensitivity and are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Key R&D Program of Zhejiang Province (Grant No. 2022C01055), the Key R&D Program of Zhejiang Province (Grant No. 2020C05005), the Hangzhou Innovation Institute, Beihang University, under Grant 2020-Y5-A-022, and the Beijing Natural Science Foundation (No.4202036). An earlier version of this paper has been presented at conference in 2021 IEEE SmartWorld Ubiquitous Intelligence and Computing Advanced and Trusted Computing Scalable Computing and Communications Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI) .

References

[1] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492–6499, 2019.

[2] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," *2009 17th International Workshop on Quality of Service*, pp. 1–9, 2009.

[3] Y. Deswarte, J. J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Working conference on integrity and internal control in information systems*, pp. 1–11, Boston, MA, 2004.

[4] G. Ateniese, R. Burns, R. Curtmola et al., "Remote data checking using provable data possession," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 1–34, 2011.

[5] X. Luo, Z. Zhou, L. Zhong, J. Mao, and C. Chen, "An effective integrity verification scheme of cloud data based on bls signature," *Security and Communication Networks*, vol. 2018, 11 pages, 2018.

[6] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," *HotOS*, 2007.

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

[8] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[9] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1703–1713, 2014.

[10] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for iot data," in *2017 IEEE International Conference on Web Services (ICWS)*, pp. 468–475, Honolulu, HI, USA, 2017.

[11] H. Wang and J. Zhang, "Blockchain based data integrity verification for large-scale iot data," *IEEE Access*, vol. 7, pp. 164996–165006, 2019.

[12] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 598–609, 2007.

[13] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 584–597, 2007.

[14] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International conference on the theory and application of cryptology and information security*, pp. 90–107, Berlin, Heidelberg, 2008.

[15] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 4, pp. 1–29, 2015.

[16] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective datasanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[17] M. A. Shah, R. Swaminathan, and M. Baker, "Privacypreserving audit and extraction of digital contents," *pasos revista de turismo y patrimonio cultural*, 2008.

[18] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial Networks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–38, 2021.

[19] Z. Cai and Z. He, "Trading private range counting over big iot data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 144–153, Dallas, TX, USA, 2019.

[20] S. Mitsunari, R. Sakai, and M. Kasahara, "A new traitor tracing," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 85, no. 2, pp. 481–484, 2002.

[21] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[22] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in p2p cloud storage," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 561–568, Singapore, 2018.

[23] 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 *2017 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID)*, pp. 468–477, Madrid, Spain, 2017.

[24] X. Zheng and Z. Cai, "Privacy-Preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[25] C. Wang, S. Chen, Z. Feng, Y. Jiang, and X. Xue, "Block chain-based data audit and access control mechanism in service collaboration," in *2019 IEEE International Conference on Web Services (ICWS)*, pp. 214–218, Milan, Italy, 2019.

[26] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, 2008.

[27] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," 2016.

[28] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of iot devices using blockchain-enabled fog nodes," in *2018 IEEE/ACS 15th international conference on computer systems and applications (AICCSA)*, pp. 1–8, Aqaba, Jordan, 2018.

[29] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *International Workshop on Public Key Cryptography*, pp. 277–290, Springer, 2004.

[30] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International conference on the theory and application of cryptology and information security*, pp. 514–532, Berlin, Heidelberg, 2001.