agriculture

MDPI

Article

Agricultural IoT Data Storage Optimization and Information Security Method Based on Blockchain

Yingding Zhao, Qiude Li, Wenlong Yi * and Huanliang Xiong

School of Software, Jiangxi Agricultural University, Nanchang 330045, China
* Correspondence: yiwenlong@jxau.edu.cn

Abstract: Given the issues of low efficiency of agricultural Internet of Things (IoT) data collection and data storage security, this study proposes a fast and reliable storage method for IoT data based on blockchain. Firstly, it performs RC5 encryption for data in the IoT sensor module. Secondly, it aggregates the same batch of collected data in the gateway into a transaction and reconstructs the Merkle ordered tree to verify the data integrity. Finally, it modifies the configuration rules of blockchain to improve the efficiency of blockchain data storage. Compared with experimental results for hash values of blockchain storage data and the stored data itself in the blockchain, the proposed method has significant advantages in data writing, and its efficiency in data reading was nearly 10 times higher than the other methods. At the same time, the method has the advantages of confidentiality, integrity, availability, controllability and non-repudiation of information security. The study can provide a solution for efficient collection and secure storage of agricultural IoT data, and it can provide technical support for realizing decentralized agricultural IoT data collection.

Keywords: agricultural IoT; blockchain; batch writing; Merkle ordered tree; encryption

Citation: Zhao, Y.; Li, Q.; Yi, W.; Xiong, H. Agricultural IoT Data Storage Optimization and Information Security Method Based on Blockchain. *Agriculture* **2023**, *13*, 274. https://doi.org/10.3390/agriculture13020274

Academic Editors: Raul Morais dos Santos and Yanbo Huang

Received: 12 December 2022
Revised: 14 January 2023
Accepted: 20 January 2023
Published: 23 January 2023

1. Introduction

The safety of agricultural products is a major livelihood issue directly related to both the health of consumers and increases in farmers' income. In recent years, with the rapid development of modern agriculture technologies, China has achieved a total balance of agricultural products. After the fundamental needs of food and clothing were satisfied, the types and behaviors of public consumption changed, and people's attention shifted from eating enough to eating well and safely. Consequently, the problem of poor agricultural product quality has attracted much attention. In order to ensure the safety of agricultural products, it is necessary to build a reliable quality control system and improve the credibility of the public service platform for information traceability. A traceability system improves the credibility of quality control of the whole chain of agricultural products, prompting producers to improve their products' quality while enhancing consumers' confidence [1]. A traceability system realizes the transformation of product quality management from the physical world to the digital world and involves the management of information from multiple supply chain links, such as production, processing, transit, sales and consumption [2].

In an agricultural product traceability system, the production link is the source of the whole supply chain, and production safety is the first line of defense for the safety of agricultural products, while the real source of growth in data on agricultural products is the premise for the system's transaction credibility. Therefore, ensuring the security of agricultural data is the primary goal of building a traceability system.

In terms of agricultural data security, there are two major issues that urgently need to be addressed: they are the collection and storage of agricultural data. Due to the diversity and real-time characteristics of agricultural data, traditional manual monitoring methods cannot be applied on a 24 h basis, and the collection of data prone to human-induced

errors such as bias and other limitations is no longer sufficient for current production models [3,4]. Agricultural Internet of Things (IoT) technology can provide an automated method of collecting production data in real time. It uses various sensors, software and hardware systems, and network communication devices to record information from the agricultural production environment and realize the interaction between machines as well as between machines and people, thus providing a new solution for the collection of agricultural data [5]. The introduction of IoT into agriculture has accelerated the formation of precision agricultural production models in field planting, which can help reduce the waste of important agricultural materials such as water and fertilizer. In addition, IoT can provide data support for intelligent decisions to reduce agricultural production costs and increase crop yields. Li et al. [6] proposed an architecture model with four layers of sensors, networks, services and applications to help deploy an intelligent agricultural system with limited energy consumption.

Existing blockchain systems perform intrinsic logic to generate blocks, which have two impacts on the performance of the blockchain. One is to consume additional on-chain computing resources to construct the Merkle tree and package the transactions; second, each generated block needs to consume storage space to record block information. Faced with the large amount of data generated by the agricultural IoT, the current blockchain system is difficult to adapt to such scenarios. To solve the limitations, this study uses the off-chain Merkle tree to transfer part of the on-chain calculation to the off-chain, saving computing resources on the chain, and changes the block generation rule. In addition, the study combines the agricultural IoT technology and blockchain technology to realize data security in the process of transferring agricultural data from collection to storage, and proposes a transaction batch writing method to optimize the data read and write speeds. The main contributions are as follows:

- The Merkle tree is reconstructed, and a Merkle ordered tree is formulated to achieve ordered arrangement of leaf nodes.
- Transaction patterns are established, and a transaction batch writing method is proposed to optimize data read and write speeds.
- The block generation rules are modified, and a "one transaction, one block" storage strategy is proposed to improve the data storage performance.

2. Related Works

Li et al. [6] proposed an intelligent IoT-based system for greenhouse management that can remotely manage temperature, humidity and irrigation in greenhouses, thus greatly reducing the labor costs and effectively improving the agricultural production efficiency of greenhouse cultivation. Navulur et al. [7] designed a mobile sensor system for agricultural automation that can help remotely manage field monitoring activities such as water pumping scheduling and the monitoring of plants as well as the presence of animals and insects through mobile sensors and mobile applications, thus changing the way farmers conduct their agricultural activities. Gao et al. [8] designed an IoT and UAV-based framework for agricultural pest monitoring that collects real-time weather parameters for crop growth through sensors and captures farm images using UAVs, thus providing information for an understanding of the specific relationship between pest damage and the farming environment.

For agricultural data storage, data are usually stored in local databases or in the cloud, but both have certain drawbacks: databases are vulnerable to human data tampering, the cloud is vulnerable to data leakage, and data integrity is not guaranteed in either case [9]. Blockchain is a distributed ledger technology that integrates consensus algorithms, cryptography, P2P protocols and other components. Since its inception, it has been aimed at replacing traditional centralized network architectures and can be regarded as a distributed database for trusted storage by virtue of its tamper-proof traceability and decentralized characteristics. Therefore, blockchain has some inherent advantages for agricultural data storage applications [10–12]. The development of blockchain technology has allowed

more and more scholars to employ it in the field of agriculture and carry out some useful explorations. For example, Yang et al. [13] designed a blockchain-based traceability system that utilizes blockchain's distributed storage, hash encryption, and programmable smart contract features to guarantee the authenticity and reliability of data in the supply chain management of agricultural products. Xie et al. [14] designed a blockchain-based solution for secure and efficient storage of agricultural traceability data, thus safeguarding food safety. Liu et al. [15] proposed a storage scheme for agricultural environmental monitoring data based on sustainable blockchain technology, thus ensuring data security and privacy. The combination of blockchain and IoT technology provides an effective solution for the design of agricultural product traceability systems, and can enhance the security of agricultural data in the process of collection and storage [16–18].

In recent years, many scholars have conducted deep studies on how to introduce blockchain into IoT to realize data storage security from two aspects, as shown in Table 1. First, after hash mapping is performed on the data collected by IoT, the obtained values are stored in the blockchain, and the original real data is stored in the Distributed Hash Table (DHT) [19], Inter-Planetary File System (IPFS) [20], or the cloud storage device. For example, Zyskind and Nathan [21] stored data in a DHT implemented by Kademlia [22], while the data hashes were stored in the blockchain; thus, data were separated from indexes, which was beneficial for data retrieval. Ali et al. [23] used IPFS to store data, and data was hashed on the chain, which relieved the pressure of blockchain storage and avoided the centralized management of data. Liang et al. [24] proposed to save the data to the cloud, while the data hash values were stored in the blockchain to ensure the integrity of cloud data. Second, data can be directly stored in the blockchain. For example, some work has put the real data on the chain, and there are some differences in specific strategies [25–28]. In this way, the data can be safely stored without using third-party tools. In addition, some studies have provided new solutions for data storage; for example, Francati et al. [29] proposed Audita, which is a blockchain-based storage audit framework, allowing data to be stored reliably by setting up storage nodes and block creators; Miyachi et al. [30] proposed hOCBS, which is a blockchain framework for privacy protection of healthcare data. Through on-chain and off-chain system design, hOCBS can adapt to different requirements of data sharing, storage, access and computing execution; Xu et al. [31] proposed Slimchain, which expands blockchain transactions through off-chain storage and parallel processing to improve the scalability of the system. However, in the face of real-time data on crop growth collected by sensors, if the data are stored directly on the blockchain, not only is data integrity verification lacking before on-chain, but also each data block needs to be hashed, which leads to more calculations by the system. Hashing algorithms are suitable for the integrity verification of a single piece of data, while the Merkle tree can verify the integrity of multiple pieces of data by using hashing algorithms and hashing merge operations many times. Compared with current works, this study proposes the off-chain Merkle tree method, which can perform data integrity verification in large batches, thus effectively establishing a balance between the performance and data integrity of agricultural IoT data transactions.

Table 1. Comparison of the current solutions.

References	Storage Methods	Storage Mediums	Characteristics
[21]	Storage data hash on-chain	DHT	Good for data retrieval
[23]	Storage data hash on-chain	IPFS	Avoids data center-based management
[24]	Storage data hash on-chain	Cloud	Low storage pressure
[25–28]	Storage data on-chain	Blockchain	Data security independence
[29]	Storage nodes	Audita	Reliable data storage
[30]	On-chain and off-chain storage	hOCBS	Good data sharing
[31]	Off-chain storage nodes	SlimChain	Good system scalability

3. Materials and Methods

3.1. Framework Design

In this method, the IoT system based on blockchain is divided into three parts by function: intelligent sensors, gateways, and blockchain. As shown in Figure 1, since LoRaWAN is considered to be the most suitable communication network for low-power wide-area networks in agricultural IoT applications [32], the wireless transmission module used by the sensor uses the LoRaWAN protocol. The data collected by the sensor are encrypted and forwarded to the gateway by the wireless transmission module. In the gateways, data are packaged into a transaction (Tx) in a batch. Next, the data in the batch are generated by a Merkle ordered tree in each gateway, and each gateway verifies that its computed Merkle roots are consistent. If more than half of the Merkle roots are the same, the next operation continues, otherwise, this transaction is discarded. Multiple gateways are used to prevent a single gateway from being hijacked and uploading dummy data. In addition, data are exchanged between sensor and gateway through a wireless network [33], and, in the process, data eavesdropping and tampering can easily occur. To resolve this issue, the encryption module is first introduced into the sensor component. Then, the consistency verification of Merkle root is performed among multiple gateways. In this way, the integrity and confidentiality of the data can be protected [34,35]. Moreover, Nginx and Keeplived services are installed in the gateways, in which the former can perform load balancing to improve the system performance, and the latter can create a virtual IP (VIP) among the gateways to prevent the entire system from being unusable due to failure of a single node, thereby improving the robustness of IoT. The gateway and blockchain are connected through the internet. This system uses the batch data writing method so that only one device is needed to send transactions. In the blockchain, the Merkle tree is different from the original Merkle tree. Based on a modification of the block generation rule, one transaction can produce one block. In this system, the Merkle tree has only one leaf node. In this way, the generation speed of blocks is accelerated and the resources calculated for the Merkle tree in blocks are reduced. The data are transmitted in the form of ciphertext to ensure the security of the data. Furthermore, in the data storage stage, the tamper-proof feature of blockchain ensures data reliability, thus efficiently improving the storage security and the credit guarantee of the IoT data.

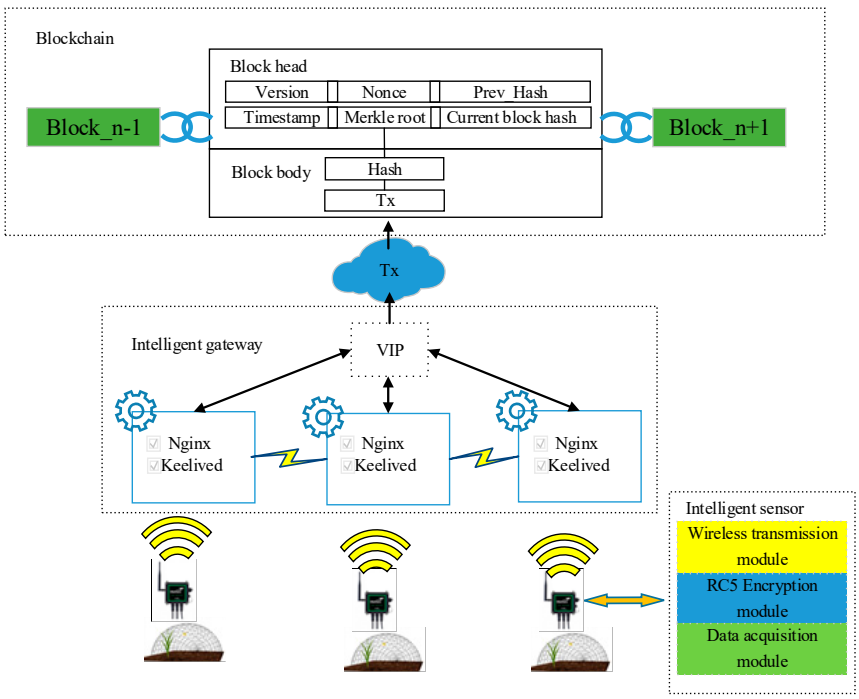

Figure 1. Overall storage framework.

3.2. Data Processing

This section describes the specific process for data in sensor-gateway-blockchain. As shown in Figure 2, in this module, the RC5 encryption algorithm is used to encrypt data. Due to the limited computing power and storage space of sensors, asymmetric encryption algorithms require more computing resources compared to symmetric encryption algorithms; therefore, symmetric encryption algorithms should be selected in sensors. In a symmetric encryption algorithm, DES and RC4 are not considered safe enough, while RC6 and AES are not suitable for sensors due to their complex calculation process. In contrast, the RC5 algorithm involves only three simple operations, such as addition, XOR and cyclic left shift; it is a fast encryption algorithm that consumes few resources and meets the needs of security [36,37]. So, it is suitable for encryption implemented in edge devices such as sensors.

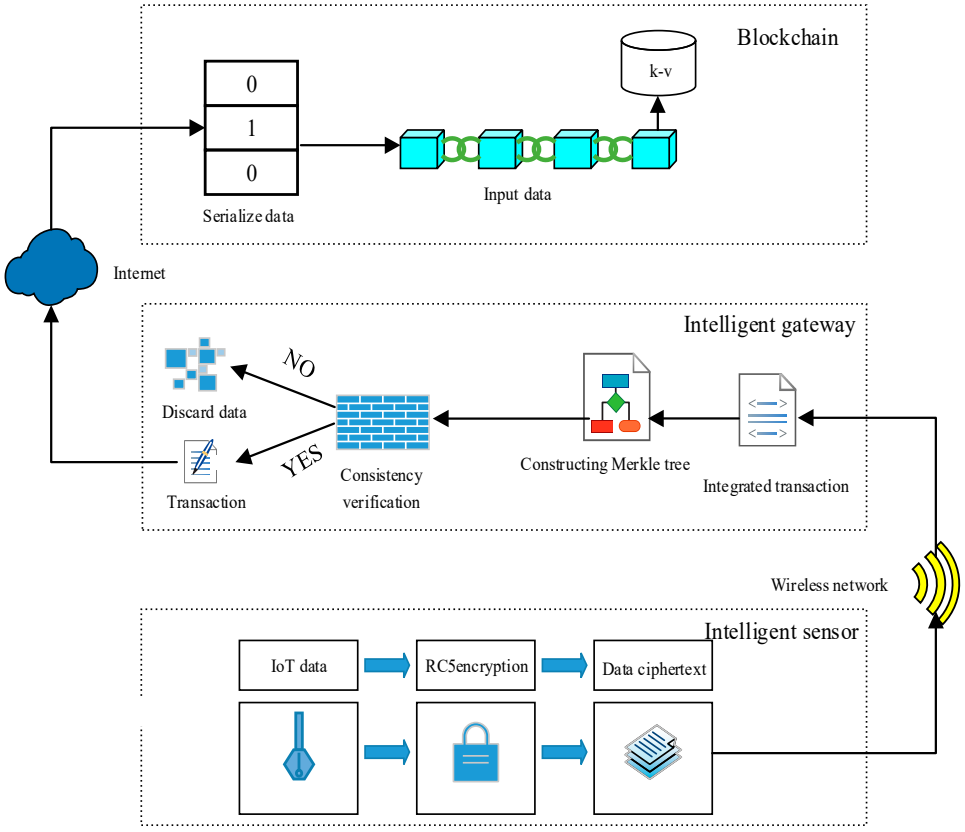

Figure 2. Flow chart of IoT data on the chain.

In the gateway, first, the encrypted ciphertext data transmitted through the wireless network are obtained. Second, the same batch of data from different sensors is packaged into a single transaction. Then, this batch of data is used to construct the Merkle tree to verify whether the Merkle root calculated in the gateway is consistent. If the number of identical Merkle roots is less than half, this means that the data consistency verification has failed and the transaction will be discarded; otherwise, verification is successful. Only gateways that pass the data consistency verification have the authority to upload a transaction. After that, the Merkle root verification success rate for each smart gateway is updated. When the Merkle root verification success rate for any smart gateway is lower than a preset threshold, the gateway is deactivated and inspected manually for tampering. The Merkle root verification success rate for a smart gateway is calculated using the equation:

$$A = \begin{cases} \frac{r+1}{N+1}, & s = true \\ \frac{r}{N+1}, & s = false \end{cases} \quad (1)$$

where A is the Merkle root verification success rate for the smart gateway, r is the number of verification successes counted last time, N is the total number of calculations counted last time. s is the status of the current verification; if the verification is successful, s is marked as true, otherwise, it is marked as false.

In the blockchain, the data are first serialized, then the index in the transaction is taken as the k value, and this transaction itself is taken as the v value. They are written into the blockchain world in the key-value form of k-v. In the meantime, new blocks are generated and linked after the previous block [38,39].

In the above process, compared with the traditional method, there are two links to help improve the data reading and writing rate; one is the use of batch writing in the gateway, which reduces the computational overhead of the blockchain on the chain and facilitates the batch reading of the same batch of data, which greatly improves the data reading rate. Second, the blockchain adopts the "one transaction, one block" block generation strategy, which accelerates the block generation rate.

3.3. Data Interaction Process

This section introduces the interaction of IoT data in the space-time dimension from the perspective of users. As shown in Figure 3, the user is the sender of instructions and the monitor of data. The sensor is the receiver of instructions and the provider of data. The gateway is the processor and uploader of data. The blockchain is the storage of data. First, the user issues a working order. After receiving the order, the sensor begins to collect data. The collected data are sent to the gateway after encryption. The gateway first integrates the data and then generates the Merkle tree to verify the integrity of the data. If the verification fails, then an error message will be directly sent to the user. If the verification succeeds, then the packaged transaction will be sent to the blockchain. After storing the transaction, the blockchain sends a success message to the user. At this point, the whole IoT data interaction process is finished. By generating the Merkle ordered tree in the gateway, the user can quickly master the integrity of data and obtain the source of tampered data. The tampered data cannot be uploaded, and the maliciously damaged sensors can be repaired, thus ensuring the authenticity and reliability of IoT data. There is little energy cost for users under normal system operation. The data collection, processing, and storage are all automated.

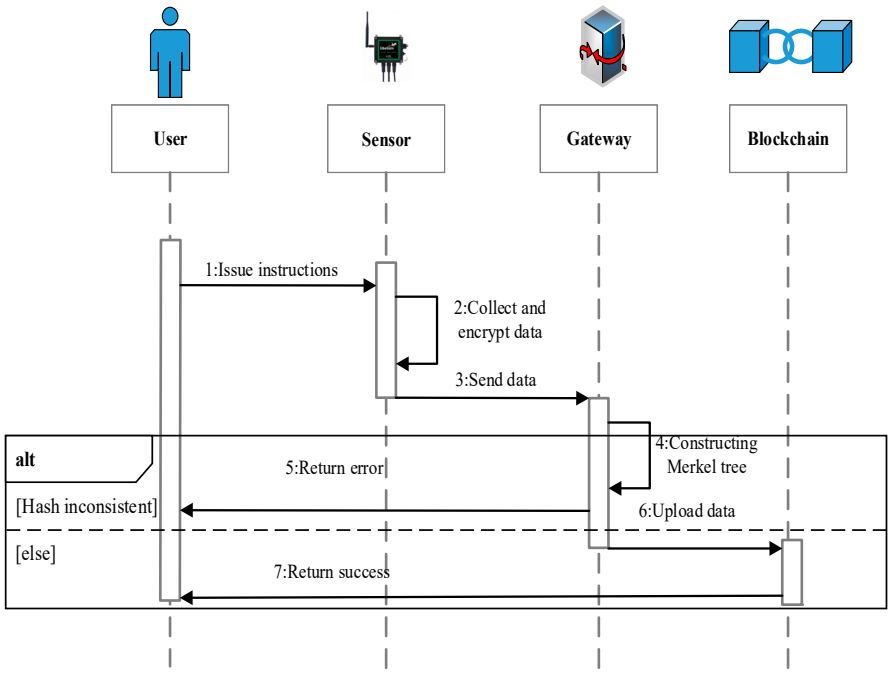

Figure 3. Time sequence of IoT data processing.

3.4. Detailed System Design

3.4.1. Intelligent Sensor Module

An intelligent sensor is a sensor with an information processing function [40]. In this study, a cryptographic module is loaded into a conventional intelligent sensor. It is decomposed into three modules according to functions: an acquisition module, a cryptographic module, and a transmission module. The working principle of this intelligent sensor module is shown in Figure 4. The left box is the data acquisition module, which is responsible for converting the external signals perceived by sensors, such as physical parameters, chemical parameters, and biological parameters, into processable electrical signals uniformly according to the electrical signal conversion protocol [41]. The middle box is the data cryptographic module, which is responsible for encrypting the data collected by the acquisition module through the RC5 encryption protocol to ensure data security. Each sensor corresponds to a product ID (PID), and the RC5 key is generated randomly by running the key generation algorithm. The right box is the wireless transmission module, which is responsible for sending ciphertext to the nearby gateway according to the wireless transmission protocol [42]. After the external information is collected, encrypted, and transmitted by the sensors, useful information can be obtained and sent to the gateway in a secure form. The applications of intelligent sensors ensure information security at the perception layer in IoT.

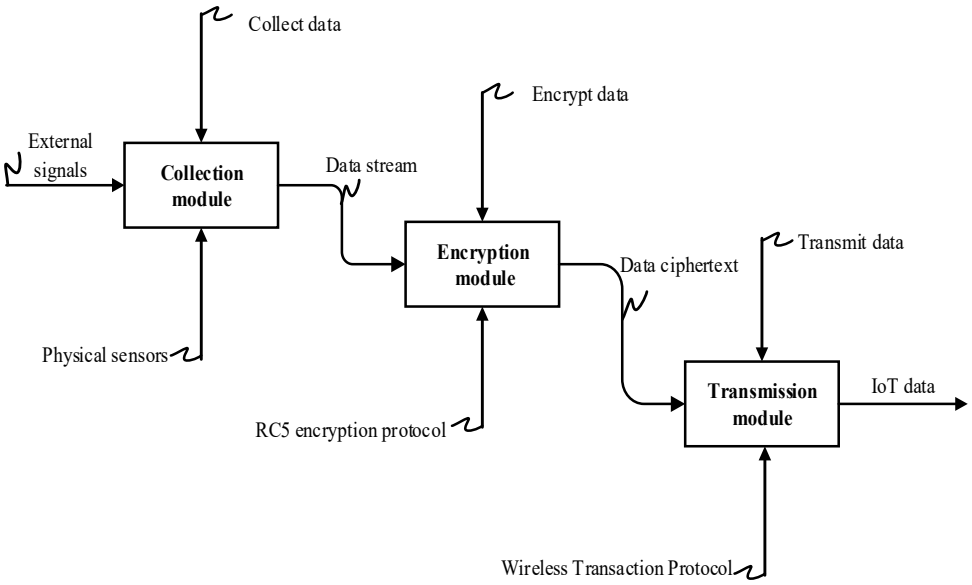

Figure 4. Data processing of the proposed smart sensor.

3.4.2. Merkle Ordered Tree

In this study, each data batch will not be uploaded individually; instead, the same batch of data is uploaded as a whole in a transaction. This improves the speed of data uploading and facilitates the centralized processing of data and extraction of key features. As shown in Figure 5, the structure of transaction data uploading is divided into two parts. One part is the index of the transaction, which is used as the k value when on-chain verification is performed; the other part is the data array. In the data array, the same batch of data information, including the sensor PID and data ciphertext, is stored in JSON form, and the data in this array also serve as the root node data for the following Merkle ordered tree.

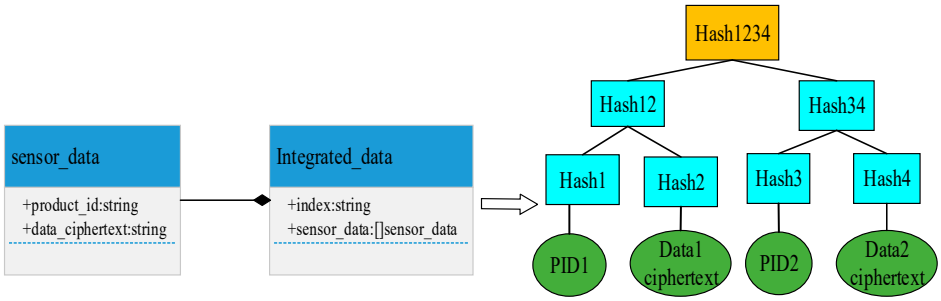

Figure 5. Construction of the Merkle ordered tree.

The transactions in the leaf nodes of the Merkle tree generated in the blockchain are disordered [43], which is not beneficial for the association among transactions. In this study, based on generation of the Merkle ordered tree, during verification of the data consistency, the types of transactions can be quickly located and the correlations among transactions can be analyzed. At the same time, the integrity of the data can be verified off-chain by introducing a Merkle ordered tree to avoid dirty data on-chain. Compared with the original Merkle tree, the Merkle ordered tree proposed in this study increases the position ordering between the data recorded by the leaf nodes. As shown in the right part of Figure 5, the same batch of data is stored in leaf nodes in the sequence. In the leaf nodes, the odd-numbered nodes store the PID of the sensors, while the even-numbered nodes store the data ciphertext transmitted by the sensors. The PIDs of the adjacent odd-numbered nodes are the only numbers of the sensors used in data collection in even-numbered nodes. The PID can be used for finding the RC5 encryption key in the sensor to facilitate the subsequent data decryption. At the same time, if the sensor collecting data is determined to be unreliable, the PID can be used to quickly locate the malfunctioning device and facilitate its timely maintenance. The generated Merkle ordered tree is a full binary tree. The Merkle ordered tree used in this study is a binary tree. When an error occurs in the data of any of its leaf nodes, the time complexity of the positioning is $T(n) = O(\log n)$, in which n is the number of leaf nodes, which is proportional to the amount of data loaded in the Merkle ordered tree.

3.4.3. Finite State Machine Model of the Off-Chain Markel Tree

The off-chain Merkle tree involves the creation of the tree, the consistency verification of the Merkle root, and the final destruction process. This study adopts a finite state machine to describe the state transition. As shown in Figure 6, a deterministic finite state machine (DFSM) $M = (S, \sum, f, S_0, Z)$ is defined, where:

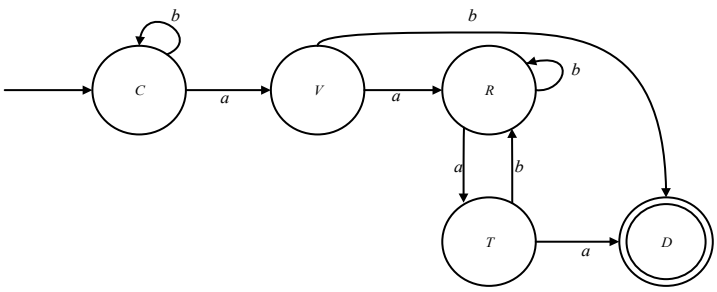

Figure 6. The DFSM model of the off-chain Merkle tree state transition.

$S = \{C, V, R, T, D\}$ is a finite state set,
$\sum = \{a, b\}$ is an input alphabet,
f is the transition function, and the function is from $S * \sum$ to S,
$S_0 = C$ is the only initial state, and
$Z = \{D\}$ is the final state set.

The DFSM describes the state changes of generating Merkle trees in edge devices, where C represents Create, V represents Verify, R represents Run, T represents Test, D represents End, a represents operation success, b represents operation failure, $f(S, a) = S$ represents that this state operation fails to enter the next state, and $f(S, b) = S$ represents that this state operation succeeds to enter the next state. The specific process is as follows: first, each smart gateway creates a transaction from the batch of data obtained and generates a Merkle ordered tree. After that, the Merkle roots are compared with those of other smart gateways; if the number of identical Merkle roots compared does not exceed half, this means that this data consistency verification fails and this transaction is discarded. Otherwise, the data consistency verification was successful and the gateway is authorized to upload this transaction. At the same time, the operation status of the system is constantly monitored during this process, and the generated Merkle tree is destroyed when the transaction has been successfully uploaded. This completes the life cycle of the Merkle ordered tree in the gateway. The DFSM can detect if the last input string is a or b; a means that the data consistency of this transaction has been verified successfully, while b means that the verification has failed.

3.5. Algorithm Description

There are three main algorithms involved in the proposed IoT algorithm, namely the RC5 encryption algorithm, the Merkle tree generation algorithm and the hash merging algorithm. Algorithm 1 describes the RC5 encryption process. Before encryption, the plaintext should be divided into two identical parts, A and B, which are stored in two w-bit registers, respectively. The S array stores w extended keys. The specific encryption process is as follows:

1. Input plaintext A and B, the number of encryption rounds r, and w-bit circular key array S; perform the addition operation for A with the first key in S; perform the addition operation for B with the second key in S; and obtain the new A and B.
2. Perform the following operations r times: perform bitwise xor for A and B; perform ring shift left by B bits for the obtained values; perform the addition operation for the processed results with the key whose subscript is twice that of the current cycle index in array S and eventually reassign it to A; process B the same as A, only change the ring shift left by A bits; the key of the addition operation is the value in the array S whose index is two times the number of cycles i plus 1.
3. When the cycle ends, merge the obtained A and B, which are the two parts of the ciphertext, and obtain the final, complete ciphertext.

Algorithm 1: RC5 Encryption

Input: *A, B, r, S*
Output: *a, b*
1: $A \leftarrow A$+S [0];
2: $B \leftarrow B$+S[1];
3: **for** *i=1* **to** *r* **do**
4: $A \leftarrow((A \oplus B)<<<B)$ +S[2i];
5: $B \leftarrow((B \oplus A)<<<A)$ +S[2i+1];
6: $i \leftarrow i+1$;
7: **end for**
8: **return** $a \leftarrow A$, $b \leftarrow B$;

The main process of Merkle tree generation is described in Algorithm 2; the specific process is as follows:

1. Hash the data in the leaf nodes to generate a hash value with a fixed length.
2. Execute the hash merging algorithm operation circularly until the current hash node is unique.

3. After the cycle ends, obtain the hash value, which is the Merkle root, and thus complete the Merkle tree generation.

Algorithm 2: Generate Merkle tree

Input: *IoTData*
Output: *rootHash*
dataHash ← *Hash(IoTData);*
while *dataHash.length* ≠ *1* **do**
    dataHash ← *getNewDataHash(dataHash);*
end while
return *rootHash* ← *dataHash*[0];

The hash merging algorithm is the process of merging the current hash set in pairs from left to right in order. As shown in Algorithm 3, the specific process is as follows:

1. Set the array index, where the initial value is 0.
2. Perform the following operations in the cycle continuously: take the hash value of the array corresponding to the current index as the left node, increase the index value by 1, assign the value of the right node to an empty string, and determine whether the current index is equal to the length of the current array; if not equal, take the hash value corresponding to the current index as the right node; afterwards, the hash values corresponding to the left and right nodes are connected in series, and the hash algorithm is adopted to generate a new hash and store it into the new hash array; then, increase the current index value by 1 and end this cycle.
3. When the index value is not less than the length of the hash array, end the cycle, and the obtained new array hash is the result of the combination of the input array hashes.

Algorithm 3: Get a new DataHash

Input: *dataHash*
Output: *newDataHash*
index ← *0;*
while *index* < *dataHash.length* **do**
    left ← *dataHash*[index];
    index ← *index* + *1;*
    right ← *0;*
    if *index* ≠ *dataHash*[index] **then**
        right ← *dataHash.length;*
    else
        continue;
    end if
    sha2HexValue ← *getSHA2HexValue(right* + *left);*
    newDataHash.add(sha2HexValue);
    index ← *index* + *1;*
end while
return *newDataHash;*

4. Results and Discussion

4.1. Experimental Environment

In the experiments, the Hyperledger Fabric [44] platform was used for testing, and the experiments were divided into a stand-alone environment and a multi-machine environment, as shown in Table 2. The single experiment was as follows: Docker technology [45] was used to deploy a 2-originization and 4-node network in one cloud server. The Solo [46] mode in the test environment was adopted for the consensus algorithm. The environment configuration was the 64-bit Ubuntu 16.04 LTS operating system with dual-core CPU, 2 GB memory, 1 Mbps broadband, Golang programming language, and Fabric v1.2.0. The multi-machine experiment was as follows: two virtual machines were used to deploy a

1-originization and 2-node network on each machine through Docker, which also formed a 2-originization and 4-node network. The Raft algorithm was adopted for the consensus algorithm. The environment configuration for each virtual machine was the 64-bit Ubuntu 16.04 LTS operating system with quad-core CPU, 8 GB memory, 1 Mbps broadband, Golang programming language, and Fabric v1.2.0.

Table 2. Description of experimental environment.

Experimental Environment	Experimental Platform	Consensus Algorithm	Number of Servers
Stand-alone environment	Hyperledger Fabric	Solo	1
Multi-machine environment	Hyperledger Fabric	Raft	2

4.2. Data Sources

The test data used were obtained through probability distribution functions. It is not the data content itself that affects the system's input or output, but the size of the data. Therefore, it is sufficient to generate random numbers that conform to a certain probability distribution to obtain test data with a length suitable to simulate the data sources expected in real environments. In this study, four probability distribution functions were selected for comparative analysis: namely, the uniform, Poisson, normal, and exponential distributions. The corresponding probability distribution functions are shown in Equations (2)–(5):

$$f(x) = \begin{Bmatrix} \frac{1}{b-a} & a \le x \le b \\ 0 & else \end{Bmatrix} \tag{2}$$

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad k = 0, 1, 2, \cdots \tag{3}$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}} \tag{4}$$

$$f(x) = \begin{Bmatrix} \lambda e^{-\lambda x} & x \ge 0 \\ 0 & x < 0 \end{Bmatrix} \tag{5}$$

To find the best probability distribution, in this study the random numbers generated conformed to the above four probability distributions. Each probability distribution was used to generate a set of data values, where each set consisted of 10 values with a mean value of 50. The relevant specific input parameters are shown in Table 3.

Table 3. Parameters of the probability distribution functions.

Distribution Function	Expectation	Variance	Upper Limit	Lower Limit
Uniform	-	-	55	45
Poisson	50	-	-	-
Normal	50	5	-	-
Exponential	50	-	-	-

The data generation results are shown in Figure 7. The exponential distribution fluctuated sharply around the mean value, while the other three distributions exhibited relatively gentle changes. In field applications, the on-chain data will be collected automatically and obtained using IoT devices, and the size of each data sample will not change much. The data will only be affected by variables such as temperature, humidity, and carbon dioxide concentration, and these variables all change over time. The Poisson distribution is suitable for describing the number of random events that occur per unit time, and it is frequently used in practical applications. The experimental data generated using the Poisson distribution also had a mean of 50 and a variance of 5.

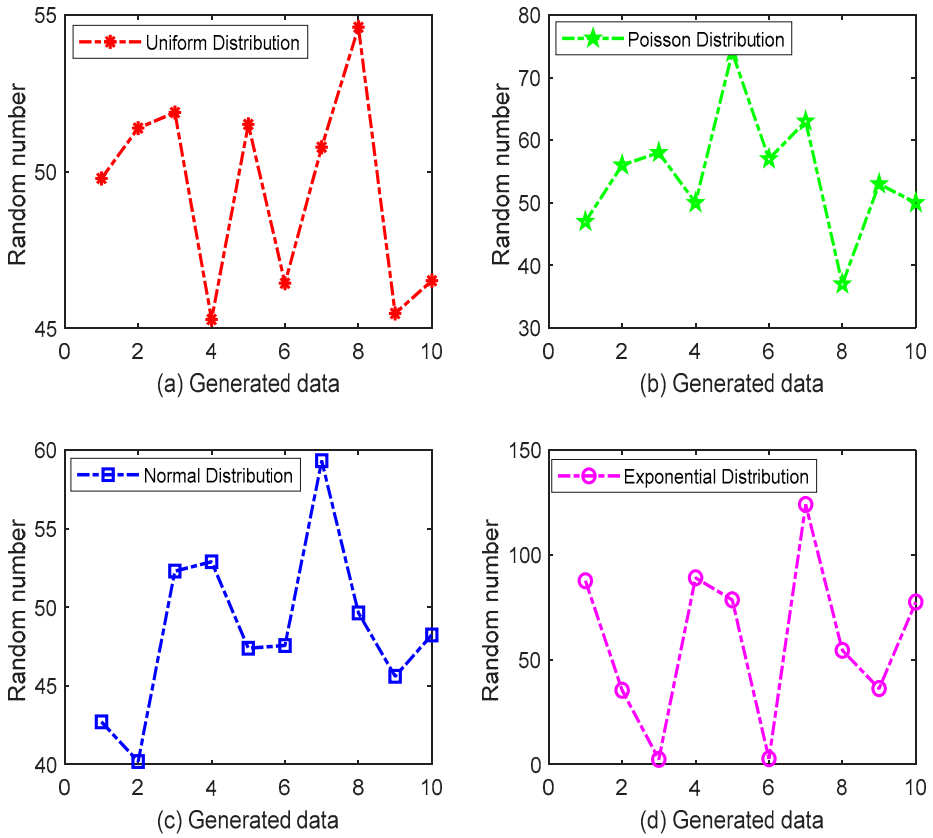

Figure 7. Random numbers generated using the four probability distributions.

4.3. Experimental Scheme

This method was compared to two other schemes, namely data storage and hash storage, as shown in Table 4. To clarify, the scheme in this paper was to directly pack the same batch of data into one transaction on the chain; the stored data scheme was to use each data as one transaction on the chain, and the stored data hash scheme was to use the hash transformed by the SHA-256 algorithm for each data batch as one transaction on the chain. The size of the simulated data conformed to the Poisson distribution. The test operation included the data input and query on the blockchain, and the test transaction volume increased from 1000 to 10,000. The default Fabric block generation rules were adopted for the compared schemes, i.e., Rule 1: there are 10 transactions; Rule 2: time reached up to 2 s; and Rule 3: the size of transactions may reach up to 99 MB. A new block was generated when any one of the three rules was satisfied. To facilitate the experimental comparison, the pressure testing tool Apache ab was used to simulate 10 clients that conduct concurrent transactions and allow the first block generation rule to be satisfied. In the scheme of this study, the first block generation rule was changed to one transaction, and only one client was used for the transaction in the experiment. In general, the blockchain will set the maximum number of transactions in a block to a larger value in order to allow each block to store as many transactions as possible, which will cause a waste of storage space. At the same time, if the number of concurrent transactions is not a multiple of the maximum number of transactions set in a block, there will be some blocks generated by the longest time, which will cause additional upload overhead. The constraint of one transaction per block imposed in this study optimizes both the storage space and the upload time of the block chain. The specific process is as follows: first, in the initial system deployment of the Hyperledger Fabric, the rules for a block corresponding to a transaction are realized by modifying the parameters of the block generation part of the configuration file configtx.yaml; then, in the gateway of the agricultural IoT, the preset sensor data type

and the corresponding collected data are obtained and packaged into a transaction and uploaded to the blockchain.

Table 4. Test details of three schemes.

Experimental Scheme	Comparison	Number of Block Scheduled Transactions	Number of Transaction Concurrency	Number of Transaction Integration
Kokoris-Kogias et al. [28]	Stored data	10	10	1
Xu et al. [47]	Stored data hash	10	10	1
Authors' method	Stored block data	1	1	10

4.4. Evaluation Method

In this section, the read and write performance of the three compared schemes is evaluated and analyzed. The relevant variables are described as follows. Let T be the total time, m be the total number of transactions, n be the transactions per second (TPS) for data writing, a and b be the temporary variables, and q be the TPS for data query. Since the stored data scheme and the stored data hash scheme have the same performance in data writing and query, the total time of the two for writing m transactions is calculated by (6).

$$T_1 = \frac{m}{n} \tag{6}$$

The proposed method packages 10 transactions together but uses a serial manner for writing data. Thus, the TPS will be increased, but it will not exceed 10 times. Therefore, $0 < a < 10$, and the total time to write m transactions is given by (7).

$$T_2 = \frac{m}{an} \tag{7}$$

The total time to query m transactions for the stored data scheme and the stored data hash scheme is calculated by (8).

$$T_3 = \frac{m}{q} \tag{8}$$

The proposed method has the same TPS as the comparison schemes when querying the data, but the amount of data for each transaction is 10 times that of the comparison schemes. At this time, $b = 10$, and the evaluation time of the query transaction is given by (9).

$$T_4 = \frac{m}{bq} \tag{9}$$

The performance comparison is shown in Table 5. The author's method achieved a partial improvement in the efficiency of data writing and a nearly 10-fold improvement in the efficiency of data query.

Table 5. Performance analysis of comparison schemes.

Experimental Scheme	Number of Transactions	TPS for Data Writing	TPS for Data Query	Time of Data Writing	Time of Data Query
Traditional scheme	m	n	q	$\frac{m}{n}$	$\frac{m}{q}$
Authors' method	m	n	q	$\frac{m}{an}$	$\frac{m}{bq}$

4.5. Experiment Result

To analyze the relationship between the writing time and the data size, the writing speed of the data values generated under the four probability distributions described above was tested when taking the maximum and minimum values. Figure 8a is a comparison of the on-chain speed in a stand-alone environment. For the uniform distribution, the change in the amount of data is less than that of the exponential distribution, but the difference

in the on-chain speed is significantly greater than the exponential distribution. It can be inferred that the on-chain speed is not directly related to the data size. Figure 8b shows a comparison of the on-chain speeds in a multi-machine environment. Due to the need for additional communications between nodes, the on-chain time is slightly greater than that with the same data size in a stand-alone environment, while the on-chain time fluctuates within a certain range when the data size changes. Through comprehensive analysis, it can be considered that within a certain range, the on-chain speed is not affected by the size of the data.

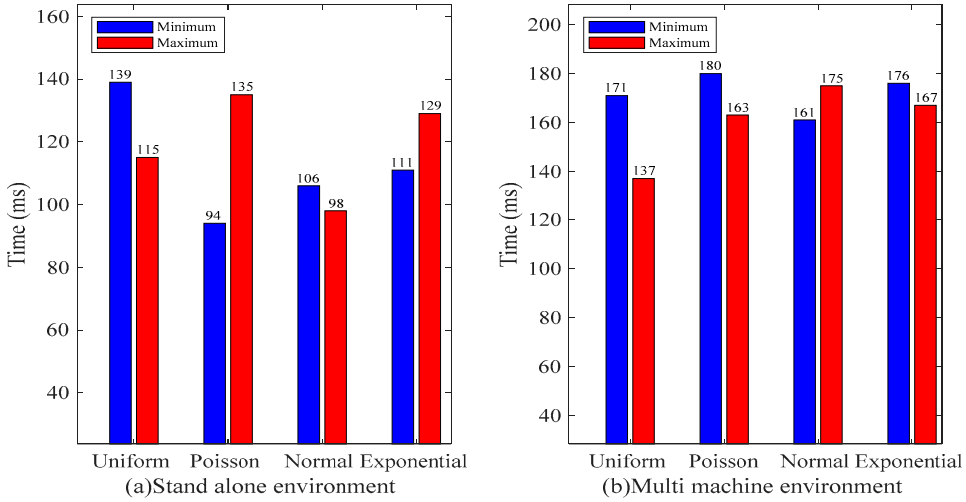

Figure 8. Comparison of data write times for different data sizes.

The experiment tested the data writing times of three schemes in the blockchain. As shown in Figure 9a, the times consumed by the three schemes increased linearly with the increase in test transactions, and the times consumed for data storage and the data storage hash were basically the same when the same volume of transactions was written. The proposed method had advantages under the same volume of tests, and the advantages became more obvious with the increase in the volume of transactions. Based on a comparison of the data storage scheme with the data storage hash scheme, it was found that there was no direct relationship between data writing time and data volume within a certain range. The experiment simulated the optimal environment under the default block configuration rule, which means that, under the condition without network fluctuation and transmission delay, 10 concurrent transactions resulted in the fastest block generation speed. However, in an actual environment, there are inevitably network fluctuations among devices. In the worst case, it will cost 2 s to generate blocks. Therefore, the proposed scheme has an obvious advantage in storage performance.

In the experiment, the data reading times of the three schemes in the blockchain were tested. As shown in Figure 9b, the curves of the three schemes were positively correlated with the volume of transactions. The curve slopes of the data storage scheme and data hash storage scheme were the same, which were 10 times that of the proposed scheme. Different from the data writing experiment, in the data reading experiment, all of the experimental schemes adopted one device for serial reading. In this scheme, the batch writing mode was used, and a transaction integrated the same batch of IoT data. In the experiment, 10 pieces of data were defined as a batch. Therefore, reading one transaction in the scheme proposed in this study was equivalent to reading 10 transactions in the other two schemes. Figure 9b shows that the reading performance of the scheme in this study was exactly 10 times faster than that of the other two schemes.

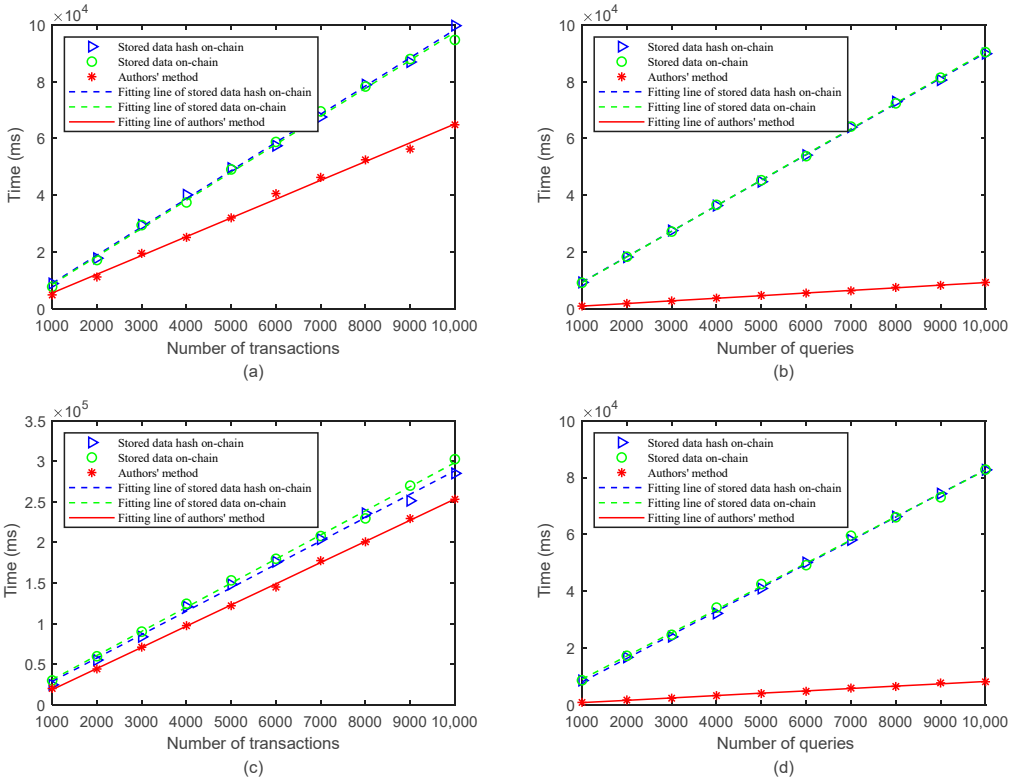

Figure 9. Data read–write performance of three schemes under different environments. (**a**) Comparison of data write time under the single-machine environment; (**b**) comparison of data read time under the single-machine environment; (**c**) comparison of data write time under the multi-machine environment; (**d**) comparison of data read time under the multi-machine environment.

The experiment tested and compared the data writing performance of the three schemes under a multi-machine environment. As shown in Figure 9c, compared with the single-machine experiment, the writing times consumed by the three schemes increased sharply, and the data storage scheme and data hash storage scheme were still close to each other. By contrast, the time consumed by the proposed scheme was lower than that of the other two schemes, but its advantage was lower than under the single-machine environment. This is because, under the multi-machine environment, the communication delay among nodes was larger. Compared with the transaction writing data, the weight of the time of various nodes reaching consensus increased. This scheme still had an advantage in data writing under the multi-machine environment and had lower requirements on the volume of concurrent transactions. To achieve the experimental effect in Figure 9c, this scheme only needed one client for the transaction. However, the compared schemes had to meet the requirements of 10 or more clients for concurrent transactions.

The experiment tested and compared the data reading performance of the three schemes under a multi-machine environment. As shown in Figure 9d, the curve changes and reading performance of the three schemes were basically consistent with those under the single-machine environment. Under the multi-machine environment, the data reading was not affected by the consensus mechanism, and it was irrelevant to the number of nodes and node distribution. Transactions could be obtained directly from any node in the blockchain. Therefore, the data reading performance in the multi-machine environment and that in the single-machine environment showed no major difference.

Through experimental comparison, it can be seen that the performance of the scheme in this study was improved in different environments compared with the traditional scheme; especially in the data reading stage, the proposed scheme had the obvious advantage of being able to read the same batch of data at one time, which improved the data reading

efficiency while reducing the extra workload. Moreover, the best experimental parameters were chosen for the comparison scheme in the experiment; in fact, the amount of data on the chain at the same time in life is mostly not an integer multiple of 10, so the real data on the chain time will be much larger than the experimental time and inevitably cause a waste of block storage space. The scheme has flexibility in parameter configuration and can select the amount of data for integrated transactions according to the actual situation, so as to maximize transaction chain speed and use block storage space. In the agricultural IoT, since the collected data are not continuous and the data uploaded by various sensors are delayed, the data obtained by traditional methods may be incomplete or fragmented, which is not conducive to subsequent data analysis. The method in this study can be stored in the blockchain after obtaining complete data, which can improve the utilization rate of data. Therefore, the method has better advantages than the traditional methods in the face of discrete data storage.

4.6. Security Analysis

The security of the method proposed in this paper is mainly supported by three links: smart sensors, smart gateways and blockchain. An RC5 encryption algorithm is used in the smart sensor, a Merkle tree is constructed in the smart gateway, and Hyperledger Fabric is used as a platform in the blockchain. Therefore, the security is influenced by the RC5 encryption algorithm, Merkle tree algorithm and Hyperledger Fabric platform.

RC5 algorithm: at the beginning of RC5 design, the RSA lab spent considerable time analyzing the RC5 algorithm for 64-bit grouping, and the results showed that the statistical properties looked very good after five cycles. After eight cycles, each plaintext bit affected at least one cycle shift. In fact, differential analysis is safe after six rounds, and Rivest recommends at least 12 rounds, possibly even 16.

Merkle tree: it has two major characteristics: first, a slight change in any leaf node will lead to a huge change in the root node of the Merkle tree. The second is to quickly locate and modify. If a node is modified, it will affect adjacent nodes upward to the root of the tree, and the tampered node can be quickly located along this path.

Hyperledger Fabric: it operates in a mutually isolated channel, which ensures the data security and privacy of Consortium Blockchain. Users register their identities with the CA through applications, log into the Fabric network using identity certificates, and initiate transaction proposals. After the transaction is simulated, endorsed, signed, and sorted through the Fabric network, it is packaged and generated into a block. The block accounting is completed in the network nodes, and the status ledger is updated at the same time.

In summary, this study analyzed the five characteristics of information security, namely confidentiality, integrity, availability, controllability and non-repudiation, as follows.

- **Confidentiality:** the RC5 cryptographic module is loaded in the intelligent sensor to realize the confidentiality of the whole data process, from collection, transmission, and storage to reading.
- **Integrity:** the Merkle root of the collected data is calculated on multiple edge devices to ensure the integrity of data in the process of transmission, and the tamper-proof feature of the blockchain ensures the integrity of data in the process of data storage.
- **Availability:** using Hyperledger Fabric consortium blockchain architecture, any organization or node can only access the network if authorized by a CA, ensuring data availability.
- **Controllability:** data are not secure only when 51% or more of nodes in a blockchain are controlled. The more nodes in the network, the stronger the attack resistance. Generally, the system is under a legitimate user's effective control.
- **Non-repudiation:** the traceable feature of blockchain ensures that every transaction is undeniable.

Through the above analysis, it is clear that the method proposed in this study satisfies the five requirements of information security.

5. Conclusions

Agricultural product traceability is a complex system involving many fields such as technology, law and society. Data security cannot be fully guaranteed at the purely technical level, and legal measures need to be established from the government's perspective to cope with frequent hacker attacks. At the same time, those employed in agriculture should also cultivate the appropriate awareness of security to avoid inappropriate actions that could cause system vulnerabilities. At the technical level, we propose the reconstruction of the off-chain Merkle tree, which improves certain aspects of data security and fault tolerance compared to traditional methods, and optimizes the block generation rules in the block chain, making it more suitable for the characteristics of agricultural IoT data to improve the system performance. However, our method requires the use of additional hardware overhead to support the data security calculations. At the same time, in the designed storage framework, the data collected by the sensor are not processed for abnormal data detection before uploading on the blockchain. In the future, we need to study the above two limitations and apply this technology to a decentralized traceability system for the quality of agricultural products.

Author Contributions: Conceptualization, W.Y., Q.L. and Y.Z.; methodology, W.Y. and Y.Z.; software, Q.L. and H.X.; validation, Y.Z.; formal analysis, W.Y.; investigation, W.Y.; resources, Y.Z.; data curation, Y.Z. and Q.L.; writing—original draft preparation, W.Y., Q.L. and Y.Z.; writing—review and editing, W.Y.; visualization, H.X.; supervision, W.Y.; project administration, W.Y.; funding acquisition, W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financially supported by the National Key Research and Development Program of China (Grant No. 2020YFD1100600) and the Natural Science Foundation of Jiangxi Province (Grant No. 20212BAB202015).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liao, Y.; Xu, K. Traceability system of agricultural product based on block-chain and application in tea quality safety management. *J. Phys. Conf. Ser.* **2019**, *1288*, 012062. [CrossRef]
2. Salah, K.; Nizamuddin, N.; Jayaraman, R.; Omar, M. Blockchain-based soybean traceability in agricultural supply chain. *IEEE Access* **2019**, *7*, 73295–73305. [CrossRef]
3. Neethirajan, S.; Kemp, B. Digital livestock farming. *Sens. Bio-Sens. Res.* **2021**, *32*, 100408. [CrossRef]
4. Yang, X.; Shu, L.; Chen, J.; Ferrag, M.A.; Wu, J.; Nurellari, E.; Huang, K. A survey on smart agriculture: Development modes, technologies, and security and privacy challenges. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 273–302. [CrossRef]
5. Farooq, M.S.; Riaz, S.; Abid, A.; Abid, K.; Naeem, M.A. A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming. *IEEE Access* **2019**, *7*, 156237–156271. [CrossRef]
6. Li, Z.; Wang, J.; Higgs, R.; Zhou, L.; Yuan, W. Design of an Intelligent Management System for Agricultural Greenhouses Based on the Internet of Things. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 154–160.
7. Navulur, S.; Prasad, M.G. Agricultural management through wireless sensors and internet of things. *Int. J. Electr. Comput. Eng.* **2017**, *7*, 3492. [CrossRef]
8. Gao, D.; Sun, Q.; Hu, B.; Zhang, S. A framework for agricultural pest and disease monitoring based on internet-of-things and unmanned aerial vehicles. *Sensors* **2020**, *20*, 1487. [CrossRef]
9. Gupta, M.; Abdelsalam, M.; Khorsandroo, S.; Mittal, S. Security and privacy in smart farming: Challenges and opportunities. *IEEE Access* **2020**, *8*, 34564–34584. [CrossRef]
10. Yi, W.; Huang, X.; Yin, H.; Dai, S. Blockchain-based approach to achieve credible traceability of agricultural product transactions. *J. Phys. Conf. Ser.* **2021**, *1864*, 012115. [CrossRef]

11. Li, Q.; Yi, W.; Zhao, X.; Zhao, Y.; Yin, H.; Xu, Y. Design and Evaluation of a High-Performance Support System for Credibility Tracing of Agricultural Products. In Proceedings of the 2021 IV International Conference on Control in Technical Systems (CTS), Saint Petersburg, Russia, 21–23 September 2021; pp. 15–18.
12. Weber, I.; Lu, Q.; Tran, A.B.; Deshmukh, A.; Gorski, M.; Strazds, M. A Platform Architecture for Multi-Tenant Blockchain-Based Systems. In Proceedings of the 2019 IEEE International Conference on Software Architecture (ICSA), Hamburg, Germany, 25–29 March 2019; pp. 101–110.
13. Yang, X.; Li, M.; Yu, H.; Wang, M.; Xu, D.; Sun, C. A trusted blockchain-based traceability system for fruit and vegetable agricultural products. *IEEE Access* **2021**, *9*, 36282–36293. [CrossRef]
14. Xie, C.; Sun, Y.; Luo, H. Secured Data Storage Scheme Based on Block Chain for Agricultural Products Tracking. In Proceedings of the 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Chengdu, China, 10–11 August 2017; pp. 45–50.
15. Liu, Z.; Huang, W.; Wang, D. Functional agricultural monitoring data storage based on sustainable block chain technology. *J. Cleaner Prod.* **2021**, *281*, 124078.
16. Fan, L.; Gil-Garcia, J.R.; Werthmuller, D.; Burke, G.B.; Hong, X. Investigating Blockchain as a Data Management Tool for IoT Devices in Smart City Initiatives. In Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age, Delft, Netherlands, 30 May–1 June 2018; pp. 1–2.
17. Yánez, W.; Mahmud, R.; Bahsoon, R.; Zhang, Y.; Buyya, R. Data allocation mechanism for Internet-of-Things systems with blockchain. *IEEE Internet Things J.* **2020**, *7*, 3509–3522. [CrossRef]
18. Li, R.; Song, T.; Mei, B.; Li, H.; Cheng, X.; Sun, L. Blockchain for large-scale internet of things data storage and protection. *IEEE Trans. Serv. Comput.* **2018**, *12*, 762–771. [CrossRef]
19. Androutsellis-Theotokis, S.; Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv. (CSUR)* **2004**, *36*, 335–371. [CrossRef]
20. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
21. Zyskind, G.; Nathan, O. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 21–22 May 2015; pp. 180–184.
22. Maymounkov, P.; Mazieres, D. Kademlia: A Peer-to-Peer Information System Based on the Xor metric. In *International Workshop on Peer-to-Peer Systems*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 53–65.
23. Ali, M. Trust-to-Trust Design of a New Internet. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 2017.
24. Liang, X.; Zhao, J.; Shetty, S.; Li, D. Towards Data Assurance and Resilience in IoT Using Blockchain. In Proceedings of the MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 23–25 October 2017; pp. 261–266.
25. Kaneko, Y.; Asaka, T. DHT Clustering for Load Balancing Considering Blockchain Data Size. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), Takayama, Japan, 27–30 November 2018; pp. 71–74.
26. Yoo, H.; Yim, J.; Kim, S. The Blockchain for Domain Based Static Sharding. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1689–1692.
27. Chen, H.; Wang, Y. Sschain: A full sharding protocol for public blockchain without data migration overhead. *Pervasive Mob. Comput.* **2019**, *59*, 101055. [CrossRef]
28. Kokoris-Kogias, E.; Jovanovic, P.; Gasser, L.; Gailly, N.; Syta, E.; Ford, B. Omniledger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 583–598.
29. Francati, D.; Ateniese, G.; Faye, A.; Milazzo, A.M.; Perillo, A.M.; Schiatti, L.; Giordano, G. Audita: A Blockchain-Based Auditing Framework for Off-Chain Storage. In Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing, Hong Kong, China, 7 June 2021; pp. 5–10.
30. Miyachi, K.; Mackey, T.K. hOCBS: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design. *Inf. Process. Manag.* **2021**, *58*, 102535. [CrossRef]
31. Xu, C.; Zhang, C.; Xu, J.; Pei, J. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing. *Proc. VLDB Endow.* **2021**, *14*, 2314–2326. [CrossRef]
32. Miles, B.; Bourennane, E.-B.; Boucherkha, S.; Chikhi, S. A study of LoRaWAN protocol performance for IoT applications in smart agriculture. *Comput. Commun.* **2020**, *164*, 148–157. [CrossRef]
33. Vellidis, G.; Tucker, M.; Perry, C.; Kvien, C.; Bednarz, C. A real-time wireless smart sensor array for scheduling irrigation. *Comput. Electron. Agric.* **2008**, *61*, 44–50. [CrossRef]
34. Saini, H.; Bhushan, B.; Arora, A.; Kaur, A. Security vulnerabilities in Information communication technology: Blockchain to the rescue (A survey on Blockchain Technology). In Proceedings of the 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kannur, India, 5–6 July 2019; pp. 1680–1684.
35. Shrestha, R.; Bajracharya, R.; Shrestha, A.P.; Nam, S.Y. A new type of blockchain for secure message exchange in VANET. *Digit. Commun. Netw.* **2020**, *6*, 177–186. [CrossRef]

36. Rivest, R.L. The RC5 Encryption Algorithm. In *International Workshop on Fast Software Encryption*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 86–96.
37. Shahzadi, R.; Anwar, S.M.; Qamar, F.; Ali, M.; Rodrigues, J.J. Chaos based enhanced RC5 algorithm for security and integrity of clinical images in remote health monitoring. *IEEE Access* **2019**, *7*, 52858–52870. [CrossRef]
38. Baliga, A.; Solanki, N.; Verekar, S.; Pednekar, A.; Kamat, P.; Chatterjee, S. Performance Characterization of Hyperledger Fabric. In Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, Switzerland, 20–22 June 2018; pp. 65–74.
39. Sukhwani, H.; Wang, N.; Trivedi, K.S.; Rindos, A. Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network). In Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 1–3 November 2018; pp. 1–8.
40. Zhang, H.; Han, W.; Xu, K.; Zhang, Y.; Lu, Y.; Nie, Z.; Du, Y.; Zhu, J.; Huang, W. Metallic sandwiched-aerogel hybrids enabling flexible and stretchable intelligent sensor. *Nano Lett.* **2020**, *20*, 3449–3458. [CrossRef]
41. Guerrero-Ibáñez, J.; Zeadally, S.; Contreras-Castillo, J. Sensor technologies for intelligent transportation systems. *Sensors* **2018**, *18*, 1212. [CrossRef] [PubMed]
42. Khalaf, O.I.; Abdulsahib, G.M. Energy efficient routing and reliable data transmission protocol in WSN. *Int. J. Advance Soft Compu. Appl.* **2020**, *12*, 45–53.
43. Xu, J.; Wei, L.; Zhang, Y.; Wang, A.; Zhou, F.; Gao, C.-Z. Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures. *J. Netw. Comput. Appl.* **2018**, *107*, 113–124. [CrossRef]
44. Cachin, C. Architecture of the Hyperledger Blockchain Fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*; held in conjunction with ACM Symposium on Principles of Distributed Computing, PODC 2016; Chicago, IL, USA, 25-29 July 2016.
45. Potdar, A.M.; Narayan, D.; Kengond, S.; Mulla, M.M. Performance evaluation of docker container and virtual machine. *Procedia Comput. Sci.* **2020**, *171*, 1419–1428. [CrossRef]
46. Xu, X.; Sun, G.; Luo, L.; Cao, H.; Yu, H.; Vasilakos, A.V. Latency performance modeling and analysis for hyperledger fabric blockchain network. *Inf. Process. Manag.* **2021**, *58*, 102436. [CrossRef]
47. Xu, Z.; Han, S.; Chen, L. CUB, a Consensus Unit-Based Storage Scheme for Blockchain System. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 16–19 April 2018; pp. 173–184.