Hindawi
Mobile Information Systems
Volume 2021, Article ID 8840399, 15 pages
https://doi.org/10.1155/2021/8840399

Research Article

Logisticschain: A Blockchain-Based Secure Storage Scheme for Logistics Data

Hongzhi Li[1], **Dezhi Han**[1] and **Mingdong Tang**[2]

[1]*College of Information Engineering, Shanghai Maritime University, Shanghai, China*
[2]*School of Information Science and Technology, Guangdong University of Foreign Studies, Guandong, China*

Correspondence should be addressed to Hongzhi Li; 1071260932@qq.com and Dezhi Han; 17855009630@163.com

Received 5 July 2020; Revised 22 November 2020; Accepted 19 December 2020; Published 3 February 2021

Academic Editor: Floriano Scioscia

Copyright © 2021 Hongzhi Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the rapid development of information technology, logistics systems are developing towards intelligence. The Internet of Things (IoT) devices throughout the logistics network could provide strong support for smart logistics. However, due to the limited computing and storage resources of IoT devices, logistics data with user sensitive information are generally stored in a centralized cloud center, which could easily cause privacy leakage. In this paper, we propose Logisticschain, a blockchain-based secure storage scheme for logistics data. In this scheme, the sensing data from IoT devices should be encrypted for fine-grained access control, and a customized blockchain structure is proposed to improve the storage efficiency of systems. Also, an efficient consensus mechanism is introduced to improve the efficiency of the consensus process in the blockchain. Specific to the logistics process, the sensing data generated from IoT devices will be encrypted and aggregated into the blockchain to ensure data security. Moreover, the stored logistics records can be securely audited by leveraging the blockchain network; both IoT data and logistics demands cannot be deleted or tampered to avoid disputes. Finally, we analyze the security and privacy properties of our Logisticschain and evaluate its performance in terms of computational costs by developing an experimental platform.

1. Introduction

Smart logistics systems have been proposed to significantly improve efficiency and accuracy, break geographical restrictions to achieve remote logistics monitoring, and ensure timely delivery of information to users. The Internet of Things (IoT) is a promising technology that provides important support for the construction of the smart logistics system. Generally, a smart logistics system is mainly composed of data collection and classification, data mining and analysis, and application layer. Specifically, we can briefly describe these components as follows. (1) The data collection procedure based on IoT is responsible for solving the real-time information collection issue, which constitutes the basis of smart logistics systems. (2) The data mining and analysis procedure is used to mine effective real-time logistics information and design a reasonable transportation scheduling scheme. (3) The application layer could provide detailed information and consulting services for logistics providers and customers [1–4].

In a smart logistics system, IoT devices (e.g., RFID, GPS sensors, temperature, and humidity sensors) are utilized to keep collecting logistics records during the logistics process. Usually, these logistics data are sent to a local gateway to perform further data processing and aggregation and then sent to the smart logistics system for analysis so that customers or users can track the real-time status. In fact, due to the limitations of IoT devices (e.g., limited battery supply and storage capabilities) and other economic factors (e.g., large-scale built-in storage for sensors is expensive and self-built data center is time-consuming and expensive), the collected logistics data are likely to be outsourced to cloud servers [4–6].

The use of cloud servers can significantly improve the efficiency of smart logistics systems compared with traditional systems [13]. Specifically, traditional logistics systems are generally based on physical sensing devices and database systems and need to rely on manual data entry. The traditional scheme is usually only suitable for small-scale logistics

systems, which is not conducive to large-scale data management because the traditional database system cannot realize the flexible management of storage resources. However, there are still several disadvantages of the cloud-assisted scheme. (1) Cloud servers can be considered as centralized third parties with a single-point bottleneck to some extent. Thus, if cloud servers are compromised or break down, all users might be affected. Furthermore, for large-scale edge computing with wide geographical distribution, the centralized cloud storage usually requires higher network bandwidth and leads to high response time. (2) Logistics data usually contain sensitive information about users, which should be well protected. However, cloud servers may sniff user privacy for commercial benefits. For instance, cloud service providers may be interested in the address information, contact information (e.g., identity, telephone, email, and location) of users, and transportation paths. (3) Logistics records stored in the cloud may still be tampered, and the credibility of the stored records cannot be guaranteed. Therefore, the cloud-assisted logistics system cannot provide reliable solutions to logistics disputes.

Blockchain technology, which is first applied in the financial field, such as Bitcoin [7] and Ethereum [8], can be used to provide a decentralized and trusted environment. With the development of blockchain technology, especially the widespread use of Ethereum and Hyperledger Fabric [9], blockchain has become one of the key technologies for IoT scenarios. The ledger of blockchain consists of a series of blocks, which are connected as a linked list, and each block contains several transactions. Since there is no centralized node in the blockchain network, it is impossible to modify the stored content in the blockchain.

Blockchain technology provides a promising solution to the security challenges of logistics data. However, the existing blockchain technology for the financial field is not suitable for IoT-based logistics scenarios. Moreover, there is no mature and feasible data protection scheme for logistics systems based on the blockchain network.

In this work, we propose a secure storage scheme for logistics data, which is named *Logisticschain*. In our scheme, users can submit logistics demands to the blockchain as transactions. Smart devices are responsible for collecting status information during the logistics process. These logistics records are usually published to the blockchain as transactions for security concerns. However, it should be noted that the complete logistics record could not be stored in the blockchain due to the limited resources of blockchain nodes. Depending on specific scenarios, a conventional DBMS (e.g., Mysql or MongoDB), a storage cloud service (e.g., S3, AWS, or Azure), or a distributed storage system (e.g., IPFS or Storj) can be used for original records' storage [10, 11]. Specifically, the main contributions of this study can be summarized as follows:

(i) We propose a blockchain-assisted secure storage scheme for logistics data, named Logisticschain. In Logisticschain, users are enabled to submit their logistics demands as transactions and logistics providers are authorized to provide logistics services. Also, logistics records are stored in the blockchain to ensure that the entire logistics process can be audited.

(ii) We propose a group-based PoW consensus mechanism, which can significantly improve the efficiency of traditional PoW consensus. It can effectively improve the throughput of blockchain-based IoT systems.

(iii) We implement the proposed Logisticschain and conduct experiments to evaluate the performance of our proposed scheme.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the related work. In Section 3, we introduce the necessary background and technologies. Also, we propose the design of the system model in Section 4. We depict the proposed scheme in Section 5. In Section 6, we present the security analysis of Logisticschain. In Section 7, we analyze the performance of Logisticschain based on experimental results. Finally, we conclude this study in Section 8.

2. Related Work

In this section, we survey the related work in three parts. Firstly, we present background information about logistics systems. Then, some literatures about blockchain applications in IoT scenarios are introduced. Finally, several representative consensus algorithms are discussed.

2.1. Smart Logistics System. With the development of e-commerce and the new retail industry, logistics has become an indispensable part of the modern supply chain. Many research studies focus on improving the performance of logistics systems. Lin et al. [12] introduced fog computing to the logistics centers. It aims to solve the problem that the centralized cloud computing system cannot bear the heavy computing load from thousands of IoT devices in factories. Furthermore, an efficient deployment model of fog computing is proposed in [12] to reduce the computational cost of IoT devices. In 2018, Zhang et al. [13] introduced the Internet of things and cloud-based storage technologies into the device layer interconnection design and data processing to solve the problem of energy waste and long waiting time in the process of integration of production and logistics in industrial workshops. A smart logistics framework is proposed in [14], and this framework adopts cyber-physical systems and industrial Internet of things (IIoT) to solve the problem of resource coordination in the process of logistics. In addition, the efforts of [15] focus on solving the integrated planning problem of smart food logistics systems, and a fuzzy logic method is used to optimize the logistics planning. Recently, blockchain technology as a promising approach is used to improve the performance of logistics systems. Perboli et al. [16] proposed a digital backbone logistics network based on the blockchain. Since the blockchain can ensure data immutability and public accessibility of data

streams, the use of blockchain can increase the efficiency, reliability, and transparency of the overall supply chain. Wang et al. [17] proposed a smart contract-based scheme for logistics systems, and an event response mechanism is utilized to trigger the defined smart contracts. Thus, all product transferring histories are perpetually recorded in a distributed ledger by using smart contracts. Aiming at the current issues of security threats and privacy leak in the smart logistics system, a scheme on applying blockchain in smart logistics systems is proposed in [18]. Smart contracts and blockchain ledgers are used to improve the traceability of logistics systems.

2.2. Blockchain for IoT. Blockchain has become one of the promising technologies for building secure Internet of things (IoT). Some research efforts are devoted to demonstrating the advantages of blockchain. Yang et al. [19] proposed a decentralized trust management system in vehicular networks based on blockchain techniques. In such a system, vehicles can validate the received messages from neighboring vehicles using the Bayesian Inference Model. To solve the difficulties of traditional centralized storage on the Internet of Vehicles, Jiang et al. [20] investigated how to extend the blockchain to the application of vehicle networks and proposed a model of the outward transmission of the blockchain data. Also, an integrated IoT blockchain platform for sensing data was proposed in [21], and the platform aims to afford the device owner a practical application that provides a comprehensive, immutable log and allows easy access to devices. Moreover, Ma *et al.* [22] proposed a blockchain-based trusted data management scheme (called BlockTDM) in edge computing. In the proposed BlockTDM, conditional access and decryption queries of the protected blockchain data have been designed, and a user-defined sensitive data encryption approach is utilized to achieve privacy protection. In [23], the authors proposed a multilayer secure IoT model based on the blockchain. This model divides the IoT system into a multilevel decentralized network and uses blockchain technology at all levels of the network. The authors of [24] put forward a blockchain-inspired IoT architecture, which is designed for creating a transparent food supply chain. RFID is used to identify food and other things, and the blockchain is employed to store the food-related sensitive information from IoT devices. In general, the blockchain technology is currently being applied to various IoT application scenarios so as to improve the security of IoT-based systems.

2.3. Consensus for IoT-Based Blockchain. Recent studies have utilized different consensus algorithms to establish blockchain-enabled networks for data storage. Proof of Work (PoW) [25] is first used in public blockchains. However, the rapid growth of transactions generated by IoT devices are different from public blockchains, and the traditional PoW algorithm is both resources and time consumption, which may not be suitable for IoT. Puthal et al. [26] presented a novel consensus algorithm called Proof-of-Authentication (PoAh), which introduces a cryptographic authentication mechanism to replace PoW for resource-constrained devices. Furthermore, in [27], the authors proposed a novel lightweight Proof of Block & Trade (PoBT) consensus algorithm for blockchain-based IoT systems, which validates not only the transactions but also the created blocks. Besides, in [27], a complete working solution for the integration of PoBT and Hyperledger Fabric is presented. The consumption of resources and time of PoBT is analyzed through a set of experiments. To improve the security of the Industrial Internet of Things (IIoT), Huang et al. [28] investigated how to extend the blockchain technology to the resource-constrained IoT environment. In detail, a credit-based Proof-of-Work (PoW) mechanism for IoT devices is proposed in [28], which guarantees the system security and the transaction efficiency simultaneously. Liu et al. [29] proposed an anonymous reputation system for IIoT-enabled retail marketing. In detail, the reputation management in the consumer-retailer system is the focus of this work and a blockchain-based reputation management scheme is proposed to provide high-level privacy protection. These accumulated reputation scores are treated as stakes of the corresponding blockchain account. Finally, a PoS-based consensus mechanism is proposed, which utilizes reputation scores to improve the efficiency of systems.

3. Preliminaries

In this part, we briefly introduce the necessary background and technologies used in our scheme.

3.1. Basic Security Functions. There are several basic algorithms used in our scheme, which are listed as follows:

(i) Setup (1^k): it takes as input a security parameter 1^k and returns the system parameter params, which includes the system keys v and u and a hash function H.

(ii) KeyGen (params, id): it takes as input a public system parameter params and a user identity id and then returns the public and private keys of user $\mathrm{AK}_i = \{\mathrm{Prik}, \mathrm{Pubk}\}$.

(iii) $H(m)$: this function is used to generate a fixed-length hash code T of message m.

(iv) Sign (Prik, m): this function is used to generate a digital signature Sig of message m using user private key Prik.

(v) VerifySig (Pubk, SigText): this function is used to verify the validity of the signature SigText. Here, the SigText is signed by the corresponding private key of Pubk. Outputs are *true* if the verifications hold or *false* otherwise.

(vi) Verify (params, Pubk, H (Cert)): this function takes the system parameter params and a specific public key Pubk as input parameters to verify the validity of the input certificate Cert.

(vii) Encrypt (Key, $\mathrm{PT}_{(m)}$): this function is used to encrypt a plain text $\mathrm{PT}_{(m)}$ by a key Key and returns

the corresponding cipher text $\mathrm{CT}_{(m)}$. This function is suitable for both symmetric and asymmetric encryption.

(viii) Decrypt(Key, $\mathrm{CT}_{(m)}$): this function is used to decrypt a cipher text $\mathrm{CT}_{(m)}$ by a key Key and returns the corresponding plain text $\mathrm{PT}_{(m)}$. Also, this function is suitable for symmetric and asymmetric decryptions.

3.2. Related Data Structure. Ublock: Ublock (the block of *Userchain*) contains transactions about logistics requests. Similar to the block structure of Ethereum, Ublock consists of a block header and a block body. As the core part of blockchain, the block header contains some important parameters about the blockchain, which can be defined as

$$U\text{header} = \{\text{ParentHash}, \text{Ind}, Tx\text{Hash}, \text{Root}, \text{Num}, Ts\}, \tag{1}$$

where ParentHash is the hash value of parent block and blocks are linked to each other through this parameter, TxHash is defined as Merkle Patricia Tries root of transactions, Ind is the hash value of Uheader, Root is the MPT (Merkle Patricia Tries) root formed by accounts, and Num and Ts are the number and the generated time of Ublock, respectively.

The block body only contains logistics transactions, which are usually published by *User nodes*, and can be defined as

$$Tx_{\text{logistics}} = \left\{\text{ID}_{U_i}, S_i, htxI_i, \text{Content}, Ts_i\right\}, \tag{2}$$

As shown in equation (2), the transaction $Tx_{\text{logistics}}$ contains the user identity ID_{U_i}, timestamp TS_i, etc. To specific, the signature S_i is signed with the private key Prik_i as $S_i = \text{Sign}(\text{Prik}_i, H(\text{ID}_{U_i} \| \text{Content} \| Ts_i))$. Content in equation (2) is the specific information about logistics demands, and this parameter can be a JSON (JavaScript Object Notation) string. Besides, $htxI_i$ is the identity of $Tx_{\text{logistics}}$, which can be calculated as $htxI_i = H(\text{ID}_{U_i} \| S_i \| \text{Content} \| Ts_i)$.

$Tx_{\text{logistics}}$ is used to publish user logistics demands, which could be obtained by only authorized logistics providers. To authorize logistics providers, the authorization transaction $Tx_{\text{authorized}}$ should be utilized:

$$Tx_{\text{authorized}} = \left\{\text{ID}_{U_i}, \text{Env}_i, \text{Sig}_i, htxA_i, Ts_i\right\}, \tag{3}$$

$$\text{Env}_i = \left\{\text{ID}_{P_{1,2,\ldots,j}}, htxI_i, \text{transkey}_i\right\}. \tag{4}$$

As shown in equation (3), the user identity ID_{U_i} is contained in $Tx_{\text{authorized}}$, and the Sig_i is signed with the private key Prik_i as $\text{Sig}_i = \text{Sign}(\text{Prik}_i, H(\text{ID}_{U_i} \| \text{Env}_i \| Ts_i))$. $htxA_i$ is the identity of $Tx_{\text{authorized}}$, which can be calculated as $htxA_i = H(\text{ID}_{U_i} \| \text{Env}_i \| \text{Sig}_i \| Ts_i)$. Also, Env_i contains the basic information of providers to be authorized, which can be defined as equation (4). Here, $\text{ID}_{P_{1,2,\ldots,j}}$ denotes the identities of providers that need to be authorized and $htxI_i$ is the identity of $Tx_{\text{logistics}}$ that can be obtained by $\text{ID}_{P_{1,2,\ldots,j}}$. Also, transkey_i is used to encrypt and decrypt the Content field of $Tx_{\text{logistics}}$.

Dblock. Similar to Ublock, Dblock can be divided into block header and block body. Here, the design of Dblock is the same as the Ublock, which can be defined as Dheader = {ParentHash, Ind, TxHash, Root, Num, Ts}.

In detail, only one type transaction (i.e., Tx_{Data}) should be aggregated into Dblock. Tx_{Data} contains the logistics data from IoT devices, which can be defined as

$$Tx_{\text{data}} = \left\{\text{ID}_j, htxI_j, \text{HEIoT}, \text{Sig}_j, Ts_j\right\}. \tag{5}$$

As shown in equation (5), ID_j is the identity of logistics provider who provides logistics services using IoT devices. To reduce the storage overhead, only the hash (i.e., HEIoT) of the encrypted IoT data should be contained in Tx_{Data}. Also, Sig_j is the signature of the private key Prik_j as $\text{Sig}_j = \text{Sign}(\text{Prik}_j, H(\text{ID}_j \| \text{HEIoT} \| Ts_j))$. $htxI_j$ is the hash of all the other parts in Tx_{Data}, which can be calculated as $htxI_j = H(\text{ID}_j \| \text{HEIoT} \| \text{Sig}_j \| Ts_j)$. Note that $htxI_j$ can be used as the identity of Tx_{Data}.

4. System Model and Design Goals

In this section, we introduce the system model and design goals of our proposed *Logisticschain*.

4.1. System Model. As illustrated in Figure 1, the proposed *Logisticschain* mainly includes different components, which can be described as follows:

(i) Users: users should first become legitimate members of the Userchain and submit their logistics requests through terminal devices (e.g., smartphones or desktop computers). In fact, these logistics requests would be collected and encrypted to the Userchain. It is important to note that users do not participate in the construction of blockchain and just connect to blockchain nodes through terminal devices.

(ii) Logistics providers: usually, logistics providers should choose logistics requests based on their abilities to provide logistics services.

(iii) IoT devices: these devices may be RFID, GPS sensors, and scanner devices. In fact, these devices are deployed in warehouses and logistics centers; each of these is connected to one and only one data node as its management node. Besides, these devices are responsible for collecting various logistics-related information to the data node periodically.

(iv) Logistics centers: logistics centers are responsible for receiving and processing logistics items. Enterprises usually deploy computing and storage devices in these centers to process logistics data.

(v) Userchain and Datachain: the proposed Userchain and Datachain are both consortium blockchains (i.e., permissioned blockchains). In this work,

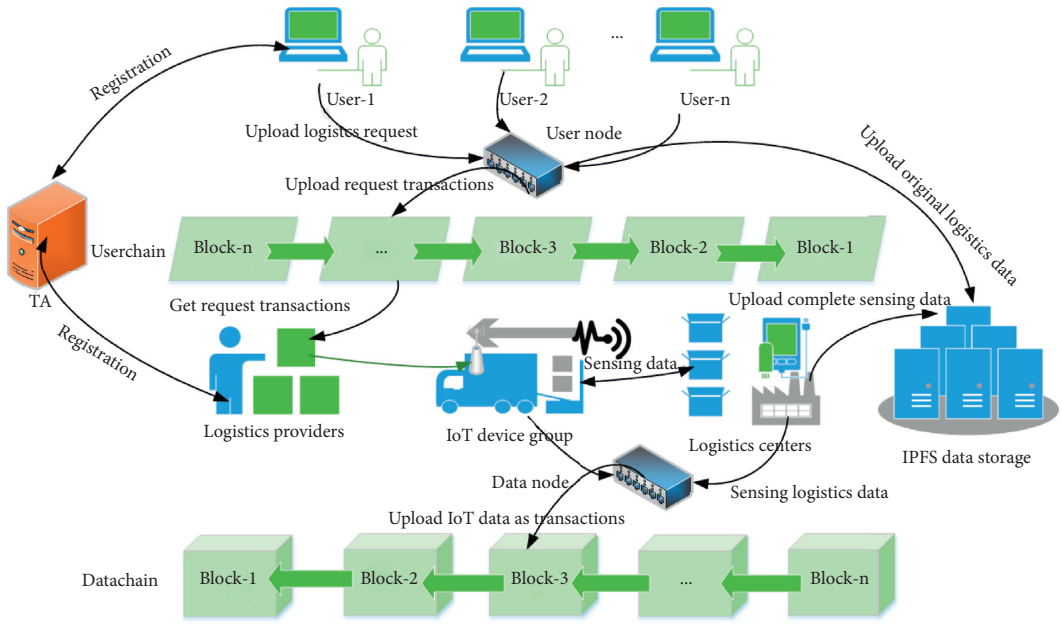

Figure 1: System model.

blockchains are usually deployed in logistics centers with enough computing and storage resources.

(vi) IPFS storage: the complete logistics records, which include logistics requests from users and sensing data from IoT devices, should be encrypted and stored in the off-blockchain storage system. Here, IPFS (InterPlanetary File System) is used to store complete records due to its decentralized design and storage performance.

(vii) Trusted Authority (TA): as the initializer of our system, any entity which attempts to join the blockchain should register its identity information and obtain the public and private keys.

Overview. *Users* (customers) and *logistics providers* should complete the identity registration in the *TA*. During logistics, users could submit logistics demands through terminal devices, and these logistics demands will be packaged into transactions and submitted to the *Userchain*. Generally, logistics providers could obtain the logistics request from the *Userchain*, and then the logistics provider is responsible for managing the specific transportation process. *IoT devices* deployed in transportation vehicles and *logistics centers* are responsible for collecting logistics status information (e.g., temperature, humidity, and geographical location information). To be specific, the generated logistics records will be aggregated and packaged as transactions into the *Datachain*.

4.2. Design Goals. In this work, we aim to achieve the secure storage of logistics data and the following design goals should be met:

(i) Supporting IoT devices: for the smart logistics system, more and more IoT devices will be connected to generate logistics data. With these devices, users are enabled to monitor logistics information in time, and the system should be able to connect massive IoT devices.

(ii) Data security: logistics data can be treated as the digital assets of users; thus, the sensitive information related to personal privacy should be protected.

(iii) Efficiency: logistics records need to be analyzed and stored in time. Hence, the system should satisfy the practical access requirements on effectiveness and scalability.

(iv) Auditable: it is used to prevent disputes between users and logistics providers. Then, logistics providers should be responsible for the uploaded logistics data. All logistics data should be auditable to ensure disputes' resolution.

5. The Proposed Scheme

In this section, we describe the proposed scheme in detail. The key notations used in this paper are listed in Table 1.

5.1. System Initialization. *TA* (Trusted Authority) as the system administrator is responsible for system initialization. *TA* publishes the system parameters with a given security parameter 1^k, which can be described as the following steps:

(i) Step 1: big enough prime number q is chosen; then, TA generates three cyclic groups G_1, G_2, and G_T of the same order q with generator g and bilinear map e: $G_1 \times G_2 \longrightarrow G_T$.

(ii) Step 2: TA chooses hash function H: $\{0,1\}^* \longrightarrow Z_q^*$.

(iii) Step 3: TA chooses a symmetric encryption function, such as AES, E: $\{0,1\}^* \longrightarrow \{0,1\}^*$, and an

Table 1: Key notations.

Notation	Description
k, p, G	Security parameter, prime number, and cyclic group
$H(.)$	Keyed hash function
ID_i, U_i	Identity of the user i
REQ_i, RES_i	Logistics request and response
Cert_i	Certificate of the node i
Ts	Timestamp
Pubk_i, Prik_i	Public key and private key of the node i
sAddr, dAddr	Destination and source addresses of items
$D\mathrm{node}_i$, $U\mathrm{node}_i$	Member node of the blockchain
Tx_{data}, $Tx_{\mathrm{logistics}}$	Blockchain transactions used in this scheme
CT_x	Encrypted content for x
P_{au_j}	The logistics service provider j
$\mathrm{transkey}_i$	Key to encrypt logistics data in transactions.
$\mathrm{Encrypt}(.)$, $E(.)$	Encryption functions
N_a	Number of nodes' group

asymmetric encryption function $\mathrm{Encrypt}(.)$, such as ECC and RSA algorithms.

(iv) Step 4: finally, TA publishes the system parameters as $\{q, G_1, G_2, G_T, e, H, E, \mathrm{Encrypt}\}$.

5.2. Entity Registration. Users and logistics providers should first register their identities and obtain certificates from the system. The detail process can be described as the following steps:

(i) Step 1: a user or logistics provider U_i sends the identity information Info_i to TA through a secure channel as $U_i \longrightarrow \mathrm{TA}$: Info_i.

(ii) Step 2: TA should encrypt the received Info_i as $E(\mathrm{Key}_{\mathrm{TA}}, \mathrm{Info}_i \| Ts_i) \longrightarrow \mathrm{CT}_{(\mathrm{Info}_i)}$. Then, TA uploads the encrypted information $\mathrm{CT}_{(\mathrm{Info}_i)}$ to IPFS and obtains a storage credential HEInfo_i, which can be used to retrieve the stored information in IPFS.

(iii) Step 3: U_i should generate the public and private keys as $\mathrm{KeyGen}(\mathrm{params}, \mathrm{Info}_i) \longrightarrow (\mathrm{Pubk}_i, \mathrm{Prik}_i)$ and register his/her blockchain account in Userchain or Datachain to obtain the corresponding blockaddress $\mathrm{BlockAddr}_i$ with the assistance of TA.

(iv) Step 4: TA publishes a certificate to U_i as $\mathrm{TA} \longrightarrow U_i$:$\mathrm{Cert}_i = \{\mathrm{HEInfo}_i, \mathrm{Pubk}_i, \mathrm{BlockAddr}_i, \mathrm{Sig}_{\mathrm{TA}}, Ts_i\}$.

Once U_i has completed the registration in *Logisticschain*, the authorized Cert_i should be stored properly.

5.3. Logistics Requesting. A user U_i with the identity ID_{U_i} (i.e., HEInfo_i), starting and ending timestamp t_1 and t_2, and a current location loc_i could publish his/her logistics request transactions to *Userchain* so that logistics providers could obtain the real-time logistics demands. This process can be described as follows:

(i) U_i collects a batch of items Transitems and extracts the type of Transitems as $\tau = \varphi(\mathrm{Transitems})$. The estimated weight of Transitems is ω and the required latest delivery time is Ts_l. Besides, U_i should clearly define the source address sAddr and the destination address dAddr.

(ii) U_i defines the logistics request as $\mathrm{REQ}_{U_i} = \{\mathrm{ID}_{U_i}, \mathrm{Env}(\mathrm{Transitems}, \tau, \omega, \mathrm{sAddr}, \mathrm{dAddr}, Ts_l), \mathrm{Sig}_{U_i}\}$, where $\mathrm{Sig}_{U_i} = \mathrm{Sign}(\mathrm{Prik}_i, H(\mathrm{REQ}_{U_i}))$.

(iii) REQ_{U_i} is encrypted as $\mathrm{Encrypt}(\mathrm{transkey}_i, \mathrm{REQ}_{U_i}) \longrightarrow \mathrm{CT}_{(\mathrm{REQ}_{U_i})}$ using the $\mathrm{transkey}_i$ with the assistance of terminal devices. Then, U_i should construct a logistics transaction as equation (2) and pack $\mathrm{CT}_{(\mathrm{REQ}_{U_i})}$ as the content field.

(iv) Finally, U_i should publish an authorization transaction $Tx_{\mathrm{Authorized}}$ as equation (3), which includes identities of the logistics providers to be authorized. Note that one authorization transaction can be used to decrypt only one $Tx_{\mathrm{logistics}}$.

In fact, $Tx_{\mathrm{Authorized}}$ should be broadcast throughout the *Userchain*, but only several authorized providers could obtain the $\mathrm{transkey}_i$ to decrypt the logistics transaction $Tx_{\mathrm{logistics}}$. Overall, the simplified process of publishing logistics transactions can be illustrated in Figure 2.

5.4. Logistics Responding. After receiving a $Tx_{\mathrm{logistics}}$ and the authorization transaction $Tx_{\mathrm{authorized}}$, a specific node $U\mathrm{node}_i$ in the Userchain should check the validity of $Tx_{\mathrm{logistics}}$ by invoking $\mathrm{VerifySig}(\mathrm{Pubk}_i, Tx_{\mathrm{logistics}}.(S_i))$. If it holds, $U\mathrm{node}_i$ would extract the identities of authorized providers as $\mathrm{IdSet} = \{\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_j\}$ from the received $Tx_{\mathrm{authorized}}$. After that, $U\mathrm{node}_i$ would check its connected providers whether they are in IdSet. If any authorized provider is connected to $U\mathrm{node}_i$, the received $Tx_{\mathrm{authorized}}$ and $Tx_{\mathrm{logistics}}$ would be sent to the provider through a secure channel, and the generation process of response can be described as follows:

(i) Step 1: an authorized provider P_{au_j} who obtains the transaction (i.e., $Tx_{\mathrm{logistics}}$ or $Tx_{\mathrm{authorized}}$) would check if the transaction has expired by verifying $(Ts_{\mathrm{now}} - Tx_{\mathrm{logistics}}.(Ts_i)) < \Delta t$. If $Tx_{\mathrm{logistics}}$ has expired, the transaction would be discarded.

(ii) Step 2: P_{au_j} checks the validity of $Tx_{\mathrm{authorized}}$ by verifying $Tx_{\mathrm{authorized}}.(\mathrm{Env}_i.htxI_i) \stackrel{?}{=} Tx_{\mathrm{logistics}}.htxI_i$. If it holds, P_{au_j} could extract transaction keys from $Tx_{\mathrm{authorized}}$ as $\mathrm{ObtainTranskeys}(Tx_{\mathrm{authorized}}) \longrightarrow \mathrm{transkey}_i$.

(iii) Step 3: P_{au_j} could obtain the original logistics request REQ_i as $\mathrm{Decrypt}(\mathrm{transkey}_i, Tx_{\mathrm{logistics}}.(\mathrm{Content})) \longrightarrow \mathrm{REQ}_i$. Moreover, P_{au_j} verifies the validity of the received P_{au_j} by computing $\mathrm{VerifySig}(\mathrm{Pubk}_i, \mathrm{REQ}_i.(\mathrm{Sig}_{U_i}))$.

(iv) Step 4: P_{au_j} should generate a response RES_j to the received $Tx_{\mathrm{logistics}}$. In detail, we can define RES_j as $\{P_{\mathrm{au}_j}, \mathrm{Feedback}_{j \longrightarrow i}$ (accept/decline), $htxI_i, Ts_j, \mathrm{Sig}_{\mathrm{au}_j}\}$, where $\mathrm{Sig}_{\mathrm{au}_j}$ in REQ_j can be calculated as $\mathrm{Sig}_{\mathrm{au}_j} = \mathrm{Sign}(\mathrm{Prik}_{\mathrm{au}_j}, H(P_{\mathrm{au}_j} \| \mathrm{Feedback}_{j \longrightarrow i} \| htxI_i \| Ts_j))$.

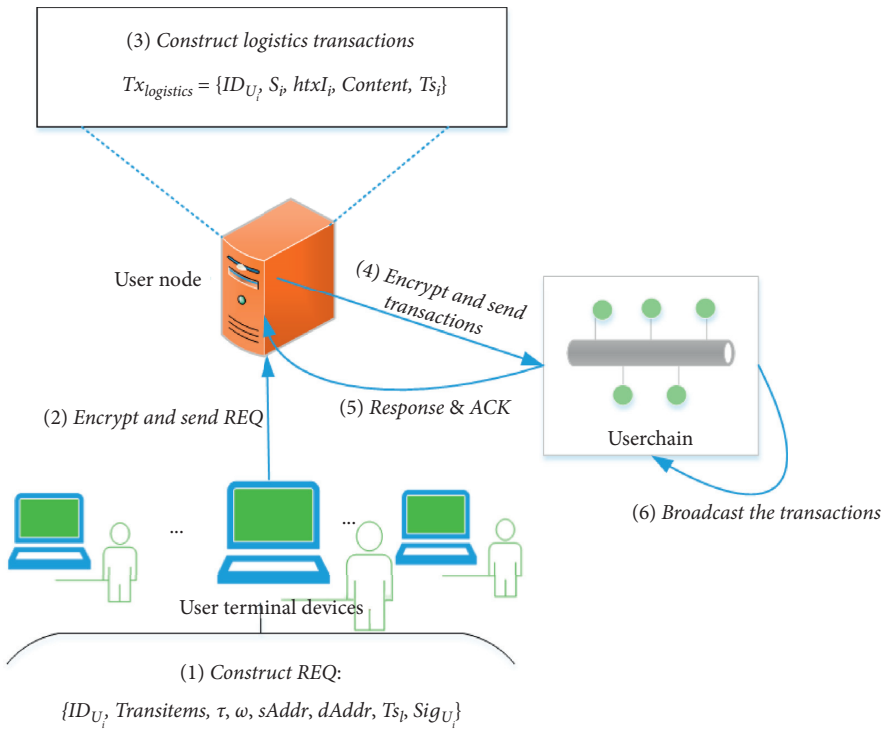

Figure 2: Flowchart of publishing logistics transactions.

(v) Step 5: RES_j is encrypted as $E(\mathrm{transkey}_i, \mathrm{RES}_j) \longrightarrow \mathrm{CT}_{\mathrm{RES}_j}$. Thus, we can define the response package as $\mathfrak{R} = \{\mathrm{CT}_{\mathrm{RES}_j}, H(\mathrm{transkey} \| \mathrm{CT}_{\mathrm{RES}_j})\}$, which should be sent to $U\mathrm{node}_i$ through terminal devices. In fact, $\mathfrak{R}$ would be packed into a transaction and broadcast throughout the blockchain (i.e., Userchain).

Figure 3 shows the process of publishing response from logistics providers.

5.5. Data Uploading from IoT Devices. When a logistics provider P_{au_j} accepts a logistics transaction Tx_{logistcs}, the corresponding member nodes (i.e., *Datachain node*) should authorize the connected IoT devices (i.e., issuing certificates to these devices). Specifically, IoT devices $\mathrm{IotSet} = \{\mathrm{iot}_1, \mathrm{iot}_2, \ldots, \mathrm{iot}_n\}$, which may be temperature and humidity sensors, and member nodes are responsible for filtering and summarizing information from IotSet in distributed warehouse centers. In *Logisticschain*, the data storage function consists of the original records storage based on IPFS and the hash value storage based on Datachain. Thus, the storage process can be described as follows:

(i) We define $s\mathrm{Data}_i = \{s_1', s_2', \ldots, s_n'\}$ to represent the sensing data from a specific IoT device iot_i, and iot_i is connected to the $D\mathrm{node}_i$ (i.e., a *Datachain node*). Besides, iot_i should store the identity of $Tx_{\mathrm{logistics}}$ (i.e., $htxI_i$) so that the sensing data could be associated with the specific logistics request $Tx_{\mathrm{logistics}}$.

(ii) The device iot_i sends a data package as $d\mathrm{pack}_i = \{s\mathrm{Data}_i, htxI_i, Ts_i\}$ to $D\mathrm{node}_i$ through a secure channel.

(iii) $D\mathrm{node}_i$ first collects $d\mathrm{pack}_i$ from iot_i and aggregates the sensing data based on the contained $htxI_i$. Then, $D\mathrm{node}_i$ verifies the obtained $s\mathrm{Data}_i$ and encrypts $s\mathrm{Data}_i$ with $\mathrm{transkey}_i$ as $E(\mathrm{transkey}_i, s\mathrm{Data}_i) \longrightarrow \mathrm{CT}_{(s\mathrm{Data}_i)}$.

(iv) $D\mathrm{node}_i$ generates $\{htxI_i, \mathrm{CT}_{(s\mathrm{Data}_i)}, Ts, \varsigma_i\}$, where ς_i in this package is signed by P_{au_j} with the private key $\mathrm{Prik}_{\mathrm{au}_j}$.

(v) $\varsigma_i = \mathrm{Sign}(\mathrm{Prik}_{\mathrm{au}_j}, H(htxI_i \| \mathrm{CT}_{(s\mathrm{Data}_i)} \| Ts))$.

(vi) The generated package should be sent to the IPFS system through a secure channel; then, the storage hash HEIoT_i would be returned from the IPFS system. After that, $D\mathrm{node}_i$ generates a logistics data transaction as $Tx_{\mathrm{data}} = \{\mathrm{Pubk}_{\mathrm{au}_j}, htxI_j, \mathrm{HEIoT}_i, \mathrm{Sig}_j, Ts_j\}$, where $\mathrm{Sig}_j = \mathrm{Sign}(\mathrm{Prik}_{\mathrm{au}_j}, H(\mathrm{Pubk}_{\mathrm{au}_j} \| \mathrm{HEIoT}_i \| Ts_j))$. Finally, the generated transaction Tx_{data} would be published to the Datachain.

After completing the storage process of logistics data, another important thing is to update and maintain the mapping between the published logistics request and the stored logistics sensing data. A smart contract should be defined and deployed in *Userchain* to maintain the relationship between logistics requests and sensing data. To be specific, such a smart contract can be executed by authorized blockchain nodes and a mapping type variable $\mathrm{LogisticsRecordsMap} = \mathrm{Map}\langle htxI_i \Rightarrow \mathrm{LogisticsRd}[\,]\rangle$ is used to store the relationship. Here, $htxI_i$ is the identity of

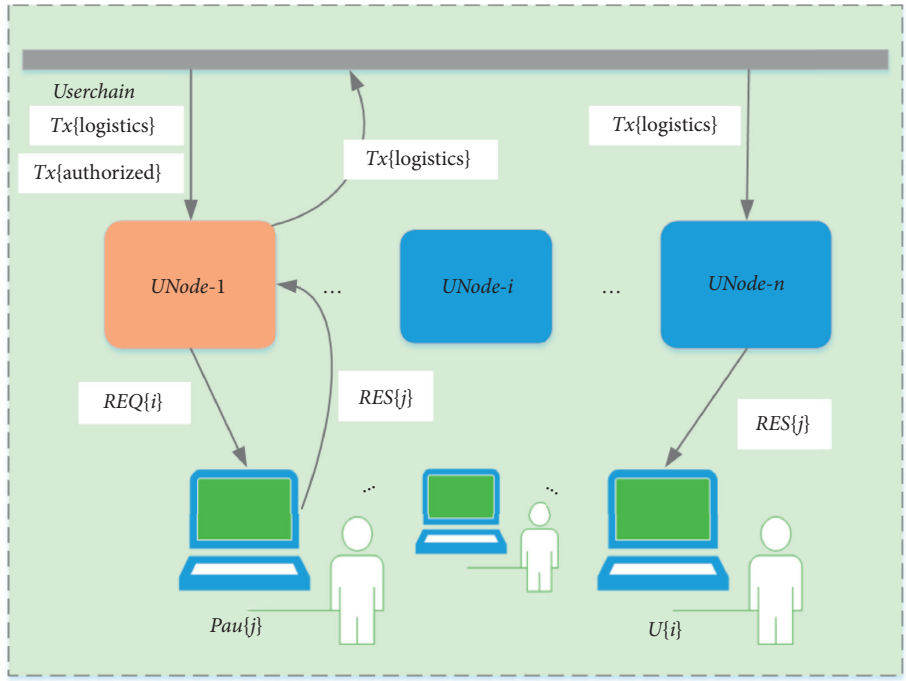

Figure 3: Flowchart of publishing logistics response.

logistics request transaction, and we can define the *LogisticsRd* as follows:

Typedefine Struct LogisticsRd {TransId$_{sensing}$: String, Key$_{iot}$: String, Pubk$_{P_{au_j}}$: String, IpAddr: String, BlockAddr: Address, Timestamp: Long }.

In the defined *LogisticsRd*, TransId$_{sensing}$ is the identity of Tx_{data}, which is generated by $D\text{node}_i$. IpAddr and BlockAddr denote the IP and the blockchain address of $D\text{node}_i$, respectively. In addition, Key$_{iot}$ is the public key of the specific device that generates the sensing data. Algorithm 1 presents the functionality when a member node tends to set up mapping records.

5.6. Data Auditing. Once there are logistics disputes between users and logistics providers, a reliable audit mechanism could provide a basis for resolving these disputes. Therefore, we propose a blockchain-based audit mechanism for logistics data in this study. For instance, a user U_i can audit his/her logistics transaction $Tx_{logistics}$ as follows:

(i) Step 1: in this process, we should first obtain the transaction id $htxI_i$ of the audited $Tx_{logistics}$; then, utilize $htxI_i$ to retrieve the transaction from the defined mapping variable LogisticsRecordsMap as LogisticsRecordsMap$[H(htxI_i)] \longrightarrow$ LogisticsRds.

(ii) Step 2: we can use LogisticsRds.$\{\text{TransId}_{sensing_i}\}$ (i.e., the identity of Tx_{data}) to obtain the data transaction Tx_{data}, and then acquire the identity (i.e., $Tx_{data}.\{\text{HEIoT}_i\}$) of original data from Tx_{data}.

(iii) Step 3: because the IPFS system is based on content addressing (i.e., the access address of a file is generated by hashing the content of the file), if we could obtain the sensing data from the IPFS system using the obtained $Tx_{data}.\{\text{HEIoT}_i\}$, then the stored original logistics data are still reliable and have not been modified. Otherwise, the data are untrustworthy.

Based on the decentralized environment provided by our scheme, we can strictly audit all logistics data stored in the IPFS system, as shown in Algorithm 2.

5.7. Group-Based PoW Mechanism. In our proposed scheme, each member node can participate in block packaging. A consensus protocol is the core of blockchain, which can determine the ownership of block packaging rights. As we know, the famous PoW and PoS consensuses are widely used in blockchain applications (e.g., Bitcoin, Ethereum, and EoS). However, the PoW consensus usually leads to unnecessary cost of computational resources, and the PoS consensus would reduce the decentralization of the blockchain network. These issues may affect the performance and security of the blockchain-based systems. Specific to our scheme, the efficiency and security of the employed blockchain should be guaranteed because the logistics data are relevant to customer benefits and privacy. Therefore, under the premise of ensuring the decentralization of blockchain, we propose a novel group-based PoW consensus, which allows nodes to be grouped freely to avoid large-scale hash computing and reduce the number of participating nodes. To make our proposed consensus better understood, we introduce the consensus process as follows:

(1) *Basic Setting.* We can assume that Nodes $= \{nd_1, nd_2, \ldots, nd_n\}$ represents all nodes that compete for the packaging rights at time t. Here,

Input: Rd_i denotes one of mapping records, LogisticsRecordsMap is a mapping array variable, $htxI_i$ is a transaction id of $Tx_{\text{logistics}}$.
Output: Boolean.
(1) **if** msg.sender does not exist **then**
(2) **return** false;
(3) **end if**
(4) **if** Rd_i is NULL or $htxI_i$ is NULL **then**
(5) **return** false;
(6) **end if**
(7) $H(htxI_i) \longrightarrow$ index;
(8) **if** empty(LogisticsRecordsMap[index]) **then**
(9) LogisticsRecordsMap[*index*] = new LogisticsRd [];
(10) **end if**
(11) $H(Rd_i) \longrightarrow R$ index;
(12) **if** R index**in**LogisticsRecordsMap[index]**then**
(13) **return** false;
(14) **end if**
(15) LogisticsRecordsMap[index][Rindex] = Rd_i;
(16) **return** true;

Algorithm 1: LogisticsRd storage process.

Input: $htxI_i$ is a transaction id of the logistics request, LogisticsRecordsMap is a mapping array variable.
Output: Boolean.
(1) **if** $htxI_i$ is NULL or empty(LogisticsRecordsMap)**then**
(2) **return** false;
(3) **end if**
(4) $H(htxI_i) \longrightarrow$ index;
(5) LogisticsRecordsMap[index] $\longrightarrow$ LogisticsRds;
(6) **for** $\text{obj}_{\text{logistics}}$ in LogisticsRds**do**
(7) $\text{obj}_{\text{logistics}}.\{\text{TransId}_{\text{sensing}}\} \longrightarrow \text{Id}Tx_{\text{data}}$;
(8) getTransById(IdTx_{data}, Datachain) $\longrightarrow Tx_{\text{data}}$;
(9) $Tx_{\text{data}}.\{\text{HEIoT}_i\} \longrightarrow \text{storageId}_{(\text{IPFS})}$;
(10) getFileofIPFS(storageId$_{(\text{IPFS})}$) $\longrightarrow$ retfile;
(11) **if** retfile == (NULL or FALSE) **then**
(12) **return** false;
(13) **end if**
(14) **end for**
(15) **return** true;

Algorithm 2: Data auditing process.

we attempt to divide these nodes into $N_a\,(1 \le N_a \le N)$ groups.

(2) *Group Partition.* Each node $nd_i \in$ Nodes generates a random number randNum_i and calculates its group id as

$g\text{id}_i = (\text{randNum}_i \oplus \text{Pubk}_i \oplus Ts_i) Mod N_a$. After that, nd_i will exchange its group id in the P2P (Peer-to-Peer) network by broadcasting a synchronization message
$\text{SYN}_i = \{g\text{id}_i, H(\text{Cert}_i), \text{BlockAddr}_i, \text{Sig}_i, Ts_i\}_{\text{ID}_i}$.
When a node $nd_j \in$ Nodes with the group id $g\text{id}_j$ receives SYN_i, it should verify the validity of SYN_i and check $g\text{id}_i \stackrel{?}{=} g\text{id}_j$. If it holds, nd_j will generate a group item $gt_{(j \longrightarrow i)} = \{\text{num}_i, \text{IPAddr}_i, \text{BlockAddr}_i, \text{Sig}_{(\text{node}_j)}, Ts_j\}$ and add $gt_{(j \longrightarrow i)}$ into its *group table*, as shown in Table 2. Otherwise, the SYN_i would be discarded by nd_j. After the group synchronization of Nodes has been completed within ΔT duration, all of these nodes will eventually obtain its entire *group table*.

(3) *Leader Selection.* For the autonomous group $a\text{group}_i$, $nd_i \in a\text{group}_i$ could participate in the group leader selection. Specifically, the selection process can be briefly divided into the selection proposal and selection declaration stages, and we can describe the selection process as the following steps.

Table 2: Sample content of the group table.

NodeNum	IP	BlockchainAddress	Timestamp
0	192.168.112.23	0x46c0c3795914 fB...	1582968761
1	192.168.112.26	0x6c0A4A4F08730...	1582968895
2	192.168.112.56	0x2dAbo2820A11 A...	1582968453
...	...	...	...
n	192.168.112.95	0xEE89c5b213e07e...	1582968542

(i) *Step 1.* nd_i could participate in the election of leader after the network is stable. Here, we set a network waiting time Δtw (e.g., 5s or 15s) for nd_i. If the number of member nodes in $a\text{group}_i$ does not change during Δtw, then $agroup_i$ is considered to be in a stable state. Furthermore, if no node in $a\text{group}_i$ is declared as a leader during Δtw, then nd_i should perform *Step 2.*

(ii) *Step 2.* nd_i generates a selection proposal $\text{Sel}_{\text{proposal}} = \{\text{Sid} = H(\text{Pubk}_i \| Ts_i), \text{isLeader}(\text{false}), \text{Sig}_i, Ts_i\}$ and sets a propagation time Δtp. $\text{Sel}_{\text{proposal}}$ will be broadcast to other nodes in $a\text{group}_i$ according to the generated *group table*. In fact, other nodes in $a\text{group}_i$ should store the received $\text{Sel}_{\text{proposal}}$ temporarily. Thus, if any other node nd_j has been declared as a leader during Δtp, then nd_i will drop its election proposal and marks nd_j as the leader of $a\text{group}_i$. Otherwise, the election procedure will execute *Step 3.*

(iii) *Step 3.* After Δtp, all of nodes (including the node nd_i) in $a\text{group}_i$ should aggregate the received selection proposals and choose the node with the largest $\text{Sid} \in \text{Sel}_{\text{proposal}}$ as the leader of $a\text{group}_i$. It is clearly that if only nd_i puts forward a selection proposal or there are multiple proposals and $\text{Sid} \in \text{Sel}_{\text{proposal}}$ of nd_i is the largest, then nd_i will be marked as the leader.

(iv) *Step 4.* We assume that nd_i is the selected leader of $a\text{group}_i$; then, nd_i should generate a selection declaration $\text{Sel}_{\text{declaration}} = \{\text{Pubk}_i, \text{is Leader}(\text{true}), \text{Sig}_i, Ts_i\}$ and broadcast the generated $\text{Sel}_{\text{declaration}}$ to other nodes in $a\text{group}_i$. After receiving and verifying $\text{Sel}_{\text{declaration}}$ from nd_i, all nodes in $a\text{group}_i$ will remove the received selection proposals and mark nd_i as their leader locally.

It is important to note that only one round of selection is needed to determine a leader node due to the unique $\text{Sid} = H(\text{Pubk}_i \| Ts_i)$ in $\text{Sel}_{\text{proposal}}$.

(4) *Block Generation.* $\text{Leaders} = \{ln_1, ln_2, \ldots, ln_n\}$ denotes the selected group leaders of the network, and each $ln_i \in \text{Leaders}$ competes for block generation through the PoW protocol [25] (i.e., compare the power of computing by large-scale mathematical calculations). To specific, $ln_i \in \text{Leaders}$ that has obtained the block packaging rights should aggregate transactions into a new block and notify member nodes to synchronize the generated block. Once the blockchain has been updated, all the autonomous groups would be cancelled. In the next round of block generation, autonomous groups need to be rebuilt to avoid the nodes with strong computing power controlling the whole network. Here, we can summarize the above selection process, as shown in Algorithm 3.

In practical terms, we assume that there are N nodes in the system and would be divided into N_a groups. The total computational time T_t is composed of the groups partition time T_g, the leaders selection time T_e, and the block generation and consensus formation time T_v; then, $T_t = \sum T_i$, $i \in [g, e, v]$. To be specific, T_g is generally composed of gid_i generation and time and SYN_i generation and broadcast time; T_e is composed of $\text{Sel}_{\text{proposal}}$ and $\text{Sel}_{\text{declaration}}$ generation and broadcast time. As for T_v, we just consider the computational time of the target hash value. For simplicity, let us denote the entity (e.g., $g\text{id}_i$, SYN_i, and $\text{Sel}_{\text{proposal}}$) generation time as $t_{g_{\text{obj}}}$, the broadcast time as $t_{b_{\text{obj}}}$, and the hash computing time in PoW as th; then, the block generation time T_t can be computed as

$$T_t = \sum_{i=1}^{N} \left(t^i_{g_{(gid\&SYN)}} + t^i_{b_{(gid\&SYN)}} \right) + \sum_{i=1}^{\overline{k}} \left(t^i_{g_{(\text{Sel})}} + t^i_{b_{(\text{Sel})}} \right) + \sum_{i=1}^{N_a} \text{th}^i, \tag{6}$$

where $\overline{k}$ is the number of nodes participating in leader election of the whole network, and the leaders' election time is $T_e = \sum_{i=1}^{\overline{k}} (t^i_{g_{(\text{Sel}_{(\text{pro\&dec})})}} + t^i_{b(\text{Sel}_{(\text{pro\&dec})})})$. Also, the consensus process is participated by leaders of all groups and the consensus formation time can be calculated as $T_v = \sum_{i=1}^{N_a} \text{th}^i$. Furthermore, we can see that the time complexity of Algorithm 3 is related to the execution frequency of lines 10–13. Then, we assume that the number of nodes in groups is $\{\text{num}_1, \text{num}_2, \ldots, \text{num}_{N_a}\}$ and $\sum_{i=1}^{N_a} \text{num}_i = N$. Hence, the execution frequency of lines 10–13 is $F(n) = 4 * \sum_{i=1}^{N_a} \text{num}_i = 4 * N$. Thus, the time complexity of Algorithm 3 is $O(N)$. Note that if $N_a = N$, our group-based PoW consensus is equivalent to the traditional PoW method.

6. Security Analysis

In this section, we analyze the security requirements of Logisticschain based on the design goals in Section 3.

6.1. Data Security. Logistics requests and sensing data from IoT devices usually contain sensitive information that need to be inaccessible to illegal adversaries. As for Userchain,

Input: Nodes = $\{nd_1, nd_2, \ldots, nd_n\}$ represents all nodes, N_a is the number of groups.

(1) $\{nd_1, nd_2, \ldots, nd_n\} \longrightarrow$ Nodes, $groups$, N_a;
(2) **for** $nd_i \in$ Nodes**do**
(3) nd_i.generateRandom $\longrightarrow$ randomNum;
(4) $g\mathrm{id}_i = (\mathrm{randomNum} \oplus \mathrm{Pubk}_i \oplus Ts_i)\%N_a$;
(5) nd_i.broadcastNetwork($g\mathrm{id}_i$, Nodes);
(6) nd_i.obtainGroup $\longrightarrow \{\mathrm{groups}, \mathrm{groupTable}_{(g_1,\ldots,g_n)}\}$;
(7) **end for**
(8) **for** $g_i \in$ groups **do**
(9) **for** $nd_j \in g_i$**do**Δ*Leader Selection*
(10) waittoStable(Δtw);
(11) $\mathrm{Sel}_{\mathrm{proposal}} = nd_j$.genProposal;
(12) nd_j.broadcast($\mathrm{Sel}_{\mathrm{proposal}}$, $\mathrm{groupTable}_{(g_i)}$);
(13) waitPropagation(Δtp);
(14) **end for**
(15) l_i = getLeader([$\mathrm{Sel}_{\mathrm{proposal}_1}, \ldots, \mathrm{Sel}_{\mathrm{proposal}_n}$]);
(16) $\mathrm{Sel}_{\mathrm{declaration}} = l_i$.genDeclaration;
(17) l_i.broadcastTogroups($\mathrm{Sel}_{\mathrm{declaration}}$, g_i);
(18) leaders.add(l_i);
(19) free([$\mathrm{Sel}_{\mathrm{proposal}_1}, \ldots, \mathrm{Sel}_{\mathrm{proposal}_n}$]);
(20) **end for**
(21) **for** $l_i \in$ leaders **do**
(22) execute(PoWprotocol) $\longrightarrow$ miner;
(23) **end for**
(24) miner.generateBlock;
(25) miner.notify;
(26) free(groups);

Algorithm 3: Group-based PoW consensus.

only authorized users could access the Userchain. The published logistics requests are encrypted with the transaction key $\mathrm{transkey}_i$, which is published by the request owner. Only the authorized logistics providers in $Tx_{\mathrm{authorized}}$ could get $\mathrm{transkey}_i$. Therefore, when $\mathrm{transkey}_i$ is not compromised, adversaries cannot get encrypted $Tx_{\mathrm{authorized}}$.

Similarly, *Da tachain* only contains the hash HEIoT_i of encrypted logistics data, and adversaries can only get $\mathrm{CT}_{(s\mathrm{Data}_i)} = \mathrm{Enc}(\mathrm{transkey}_i, s\mathrm{Data}_i)$ from the IPFS storage system. The sensing data should first be signed with $\mathrm{Key}_{\mathrm{iot}_i}$ by IoT devices and then encrypted with $\mathrm{transkey}_i$ in *Data chain nodes*. Therefore, if $\mathrm{transkey}_i$ is not compromised, adversaries cannot obtain any detail information about the logistics data. Overall, our scheme achieves to provide conditional security of logistics data.

6.2. Entity Authentication. We assume that an external adversary U_a attempts to impersonate a legitimate entity U_i. In general, each entity in our scheme should register its information, and the system will authenticate any operation of U_i with its valid certificate. However, U_a does not register basic information in IPFS without a unique certificate issued by *TA*. Therefore, if the certificate of U_i is not compromised, the adversary U_a cannot get any valid information by impersonating a legitimate U_i. Since each legitimate entity U_i has its unique certificate Cert_i issued by *TA*, it is almost impossible for malicious nodes to pretend multiple identities illegitimately by forging certificates.

6.3. Security Analysis of Group-Based PoW. In the *Group Partition* and *Leader Selection* stages, all the messages exchanged by legal nodes needs to be signed with the hash value of its unique certificate Cert_i. Then, each node performs signature verification on the received messages and invokes the method Verify(.) to verify the hash value of certificate. Therefore, we can effectively prevent the spread of fake messages from damaging the consensus process in this way.

Considering such a scenario that a malicious node in the blockchain modifies the data stored in a block on its node, the malicious node links the modified block to form a chain by competing for new blocks. Hence, there are two versions of blockchain, namely, the honest chain and the modified chain. Therefore, if the malicious node attempts to tamper successfully, it must make the modified chain become the longest chain. Assuming that the length between the modified chain and the honest chain differs by z blocks, the modified chain becomes the longer chain and succeeded in making up for the distance gap. The possibility of this process can be approximately considered as Gambler's Ruin problem [30] and can be calculated as equation (7).

z = block distance between the honest chain and the modified chain.

p = probability the honest chain gets the next new block.

q = probability the modified chain gets the next new block.

p and q represent the probability of mutually exclusive events and $p + q = 1$.

q_z = probability the modified chain catches up from z blocks behind.

$$q_z = \begin{cases} 1, & p <= q, \\ \left(\frac{q}{p}\right)^z, & p > q. \end{cases} \tag{7}$$

Assuming that the blockchain generates a new block per the average expected time, the extended length of the modified chain will be a Poisson distribution. Then, the probability can be calculated as

$$p_a = \sum_{k=0}^{\infty} \left(\frac{\lambda^k e^{-\lambda}}{k!}\right)^* \begin{cases} \left(\frac{q}{p}\right)^{(z-k)}, & k <= z, \\ 1, & k > z. \end{cases} \tag{8}$$

We use matlab to simulate the above process based on equation (8), and the results are shown in Figure 4. We can see that when $q = 0.5$ or more, it is possible to control all the data of the entire blockchain and break through the consensus algorithm.

As for our proposed group-based PoW, a node n_i needs to become the leader of its group to participate in the competition for block packaging. We assume that the current network has N nodes, which are divided into N_a groups. Since the group leader is elected by random proposals, the probability of node n_i being selected is (N_a/N). Then, the probability of the modified chain gets the next new block can be defined as $q_g = (N_a/N) * q$, where q is used in equation (8) to indicate the ability of a node to acquire blocks in a pure PoW competitive environment. According to the above analysis, to achieve data tampering, q_g of the malicious node should be bigger than 0.5. According to the definition of q_g, it is impossible to realize data tampering by improving the computing power, especially in a large-scale network.

7. Performance Evaluation

To validate the effectiveness and feasibility of Logisticschain, we have carried out a series of experiments. In this section, we first introduce the experimental environment and the capacity of Ublock and Dblock. Then, we analyze the generation time of transactions and compare the efficiency of our scheme with that of other schemes. Finally, we evaluate the performance of the group-based PoW and compare it with well-known consensus protocols.

7.1. Experimental Environment and Block Capacity. A trial system of Logisticschain has been implemented to evaluate its efficiency and effectiveness. To implement the trial system, we simulate users and logistics providers with smart phones, and a desktop computer (Lenovo ThinkCentre M720) is used to run the *TA* procedure. Besides, we use ten desktop computers to run our proposed blockchain and the IPFS (v0.4.14) system.

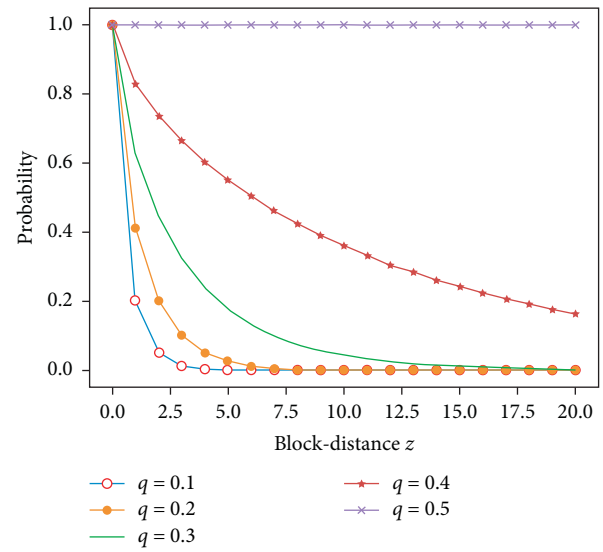

Figure 4: Data tampering success probability.

We use the defined structures of *Ublock* and *Dblock* as mentioned in Section 3. The length of ParentHash and Ind are both set as 64 bytes; TxHash and Root are both with the length of 40 bytes; Num and Ts are set as 4 bytes. The length settings of block header are shown in Table 3. We choose 1024 bits RSA and 160 bits ECC for asymmetric encryption and signature, 256 bit AES for symmetric encryption, and SHA-256 for hash computing.

Based on the above cryptographic settings, we can calculate that the size of $Tx_{\text{logistics}}$ in equation (2) is about 528 Bytes and the size of $Tx_{\text{authorized}}$ and Tx_{data} are 328 Bytes and 292 Bytes, respectively. Hence, we can conclude that Ublock of 1M Bytes could contain about 1985 $Tx_{\text{logistics}}$ or 3196 $Tx_{\text{authorized}}$ and Dblock of 1M Bytes contains 3590 Tx_{data}. Assuming the generation time of Ublock and Dblock is set to 1 minute, the throughput can reach about 33 $Tx_{\text{logistics}}$ per second or 53 $Tx_{\text{authorized}}$ per second, and as for the Dblock, the throughput can reach about 59 Tx_{data}.

7.2. Performance of Logisticschain. For the evaluation of computational overhead, we implement a trial system based on the proposed scheme, which includes Userchain, Datachain, and IPFS system. Our evaluation test is based on the Apache JMeter 5.2. We simulated 500, 1000, and 1500 logistics requests, respectively, to *Userchain* and the same number of transaction requests to *Datachain*. Then, we set the number of test threads as 50, 100, and 150, and for each thread, set the loop count as 10. Furthermore, we compare the computational costs of the proposed scheme with that of [23, 28]. The results are shown in Figure 5, from which we can see that our scheme performs better than the other two schemes. To deploy the blockchain platform, we use ten

Table 3: Setting of the block header.

Parameters	ParentHash	Ind	TxHash	Root	Num	Ts
Length (bytes)	64	64	40	40	4	4

desktop computers where the system configuration is Ubuntu 16.04 (64 bits) with an Intel(R) Core(TM) i7-6700 CPU 3.40 GHZ and 3 GB RAM to construct a network with 20 virtual nodes. As for *Userchain*, the computational cost of our scheme is at a low level and remains stable. For instance, when the number of requests reaches 800, the average run time of each transaction request is 13.56 ms, while the methods in [23, 28] are about 17.4 ms and 26.1 ms, respectively. The throughput of Datachain is slightly higher than that of Userchain, since Datachain only contains the hash value of sensing data and the storage overhead of sensing data in the IPFS is not included. Compared with the schemes used in this experiment, our *Logisticschain* reduces the processing time of logistics data due to the customized transaction structure.

Furthermore, we evaluate the off-chain performance including users' registration, logistics request/response generation, and IPFS storage. In Table 4, experimental results show that the computation incurs a few milliseconds.

To evaluate the efficiency of the audit mechanism, which is used to avoid logistics disputes, we conduct a performance test that simulated audit requests from 100 to 1000 (including valid transactions and some tampered transactions), and the results are shown in Figure 6.

As illuminated in Figure 6, with the same network size, the computational cost increases as audit requests increase. The highest TPS is over 200, and the auditing consumption remains stable. Besides, the throughput of audit requests is affected by the network size. We can observe that the throughput of the network of 20 node is higher than that of the network of 40 node about 21.2%. The throughput decreases with the network size increase, but does not follow a linear relationship.

7.3. Performance of Group-Based PoW Consensus. In our scheme, the proposed group-based PoW consensus takes into consideration of the limited computing resources. To evaluate the performance of our group-based PoW, we have designed and implemented a comparison test with traditional PoW and PoS consensus protocols. In this test, we use multithreading technology to simulate member nodes. In fact, the number of node groups N_a has a certain influence on the efficiency of group-based PoW consensus. Here, we simulate user nodes with 100 threads and set the difficulty of PoW to 4 (i.e., the computational target is Hash (header) ≤ (MAX/difficulty)) so as to evaluate the impact of N_a on consensus efficiency.

As shown in Table 5, N_a has a significant impact on the efficiency of the group-based PoW. When $N_a = 7$, the runtime reaches a lower level of 59 ms. Obviously, if the value of N_a is too small or too large, the consensus efficiency decreases significantly (e.g., when N_a is 2 and 5, the runtime is 99 ms and 121 ms, respectively). Therefore, we should adopt an appropriate value of N_a according to the size of network.

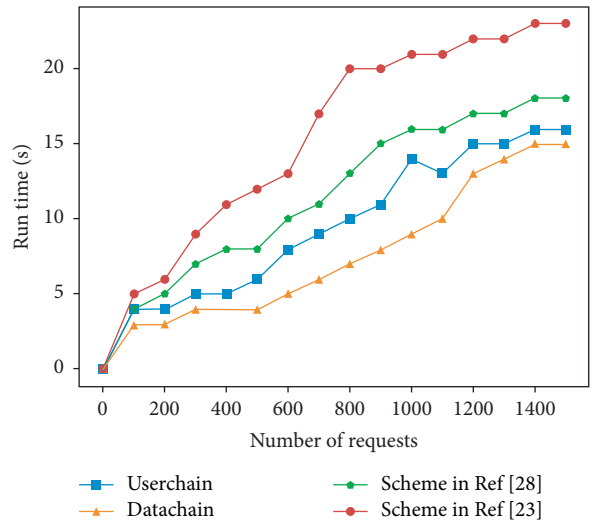

Figure 5: Comparision of computational overhead.

Table 4: Off-chain overhead.

Operations	Involved entities	Time (ms)
Users registration	User/Providers	521
RES/REQ generation	Userchain	678
IPFS storage	Userchain/Datachain	439
SHA-256 signing	Userchain/Datachain	125
AES encryption	Userchain/Datachain	207

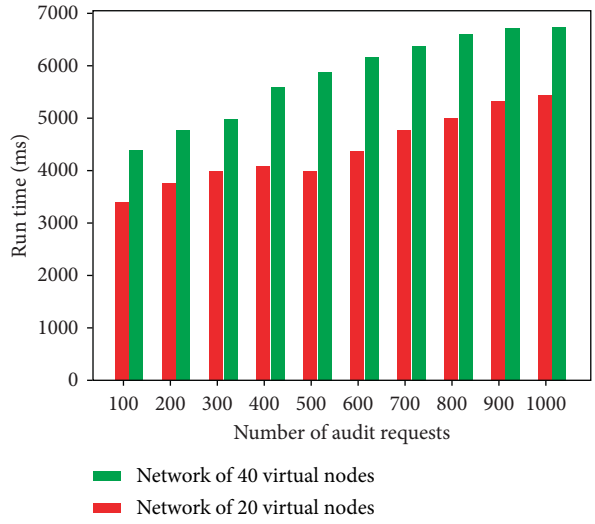

Figure 6: Runtime of audit requests.

To further evaluate the performance of our group-based PoW, under the same hardware and software (go1.13.4) experimental environment. We compare the computational

Table 5: Runtime with different N_a.

N_a	2	3	4	5	6	7	8
Runtime (ms)	99	96	76	121	110	59	81

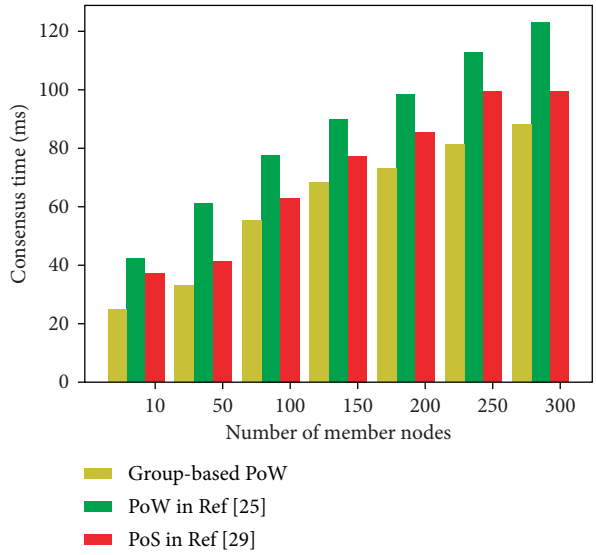

Figure 7: Comparison with other consensuses.

overhead of our group-based PoW with that of the traditional PoW and PoS, respectively. From the results in Figure 7, it can be clearly seen that our group-based PoW performs better than traditional PoW consensus [25] and PoS consensus [29]. Specifically, we have simulated a network of nodes from 10 to 300, the average consensus time of our group-based PoW is still less than 67 ms. Meanwhile, the average consensus time of traditional PoW is about 84 ms, which is about 19% higher than that of the group-based PoW. In fact, under the same difficulty setting, our group-based PoW has better performance than the traditional PoW because our proposed consensus can greatly reduce the hash calculations while maintaining decentralization. As for the PoS consensus in this test, we use random number in a given range (e.g., 100–1000) to represent the stake of virtual nodes, and then the nodes compete for block packaging through its own stakes. As shown in Figure 7, the performance of our proposed consensus is similar to that of PoS algorithm, both of which have high computing efficiency. It should be noted that we need to adjust N_a according to the size of network to achieve better results.

Furthermore, to evaluate the resource consumption of the group-based PoW, we have designed and implemented a comparison experiment. This experiment is built on a desktop computer (Lenovo ThinkCentre M720) with Intel Core i7-7500 2.7 GHz processor and 32 GB. We have simulated virtual nodes from 100 to 300 using multithreading technology, and Go language is utilized for implementing algorithms in the experiment. Specifically, we evaluate the resource consumption of algorithms in terms of memory consumption and CPU usage and use VisualVM 1.4.4 as a measurement tool to monitor the memory and CPU usage.

As shown in Table 6, we can see that our proposed consensus performs better than the other two methods

Table 6: Comparison of resources' consumption.

Consensus	Virtual nodes	MEM (MB)	CPU (%)
Group-based PoW	100	0.82	1.4
	200	1.13	2.1
	300	1.32	4.4
PoW in [25]	100	5.53	5.7
	200	6.96	13.4
	300	16.2	19.5
PoS in [29]	100	0.50	1.7
	200	1.14	2.9
	300	1.27	4.2

and the memory usage of group-based PoW is less than 2 MB, while for the traditional PoW, the maximum memory usage is more than 15 MB. In terms of the CPU usage, the group-based PoW is significantly lower than the PoW. For our consensus, the highest CPU usage is 4.4% (the number of virtual nodes is 300), which is lower than the lowest value of PoW (when the number of virtual nodes is 100, the CPU usage is 5.7%). Since the group-based PoW only contains a small amount of hash calculations, such as PoS, it is less dependent on CPU computing resources. So, these two methods are relatively close in CPU consumption. Besides, the memory consumption of our proposed consensus is slightly higher than that of traditional PoS; this is because our scheme needs to maintain the group table and other data structures in memory.

8. Conclusion

In this paper, we have proposed and implemented a blockchain-based logistics scheme *Logisticschain*. To be specific, logistics requests from users are aggregated into *Userchain*; then, logistics providers could choose logistics orders according to their demands. Furthermore, the logistics data from IoT devices are collected and aggregated into *Datachain* to ensure that all the stored logistics data cannot be tampered. Also, the group-based PoW is proposed, which can significantly reduce the computational overhead while ensuring the decentralization of the blockchain. Several simulations are carried out to evaluate the performance of our system. Analysis and evaluation show that our proposed scheme is effective and feasible for the storage of logistics data. Further studies are still needed in the future. For example, how to evaluate the reputation of logistics providers participating in *Logisticschain* is an open issue to be further studied.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] S. Winkelhaus and E. H. Grosse, "Logistics 4.0: a systematic review towards a new logistics system," *International Journal of Production Research*, vol. 58, no. 1, pp. 18–43, 2020.

[2] L. Yuan, "Intelligent logistics management application relying on the internet of things," in *Proceedings of the 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, pp. 451–457, IEEE, Jishou, China, 2019.

[3] J. Wen, L. He, and F. Zhu, "Swarm robotics control and communications: imminent challenges for next generation smart logistics," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 102–107, 2018.

[4] M. Li, P. Lin, G. Xu, and G. Q. Huang, "Cloud-based ubiquitous object sharing platform for heterogeneous logistics system integration," *Advanced Engineering Informatics*, vol. 38, pp. 343–356, 2018.

[5] T. Démare, C. Bertelle, A. Dutot, and D. Fournier, "Adaptive behavior modeling in logistic systems with agents and dynamic graphs," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 13, no. 3, pp. 1–25, 2019.

[6] Q. Liu and Y. Wu, "Analysis on cloud logistics service mode," in *Proceedings of the 2017 Chinese Automation Congress (CAC)*, pp. 7546–7548, IEEE, Jinan, China, 2017.

[7] S. Underwood, "Blockchain beyond bitcoin. Communications of the ACM," vol. 59, no. 11, pp. 15–17, 2016.

[8] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: identification, analysis, and impact," *Future Generation Computer Systems*, vol. 102, pp. 259–277, 2020.

[9] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, Porto, Portugal, 2018.

[10] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: architecture, consensus, and future trends,"" ", IEEE, in *Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, IEEE, Honolulu, HI, USA, 2017.

[11] N. Nizamuddin, K. Salah, M. Ajmal Azad, J. Arshad, and M. H. Rehman, "Decentralized document version control using ethereum blockchain and ipfs," *Computers & Electrical Engineering*, vol. 76, pp. 183–197, 2019.

[12] C.-C. Lin and J.-W. Yang, "Cost-efficient deployment of fog computing systems at logistics centers in industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4603–4611, 2018.

[13] N. Zhang, "Smart logistics path for cyber-physical systems with internet of things," *IEEE Access*, vol. 6, pp. 70 808–870 819, 2018.

[14] Y. Zhang, Z. Guo, J. Lv, and Y. Liu, "A framework for smart production-logistics systems based on cps and industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4019–4032, 2018.

[15] Y. Li, F. Chu, C. Feng, C. Chu, and M. Zhou, "Integrated production inventory routing planning for intelligent food logistics systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 867–878, 2018.

[16] G. Perboli, S. Musso, and M. Rosano, "Blockchain in logistics and supply chain: a lean approach for designing real-world use cases," *IEEE Access*, vol. 6, pp. 62 018–062 028, 2018.

[17] S. Wang, D. Li, Y. Zhang, and J. Chen, "Smart contract-based product traceability system in the supply chain scenario," *IEEE Access*, vol. 7, pp. 115 122–115 133, 2019.

[18] Y. Fu and J. Zhu, "Operation mechanisms for intelligent logistics system: a blockchain perspective," *IEEE Access*, vol. 7, pp. 144 202–144 213, 2019.

[19] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2018.

[20] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: distributed network architecture and performance analysis," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4640–4649, 2018.

[21] L. Hang and D.-H. Kim, "Design and implementation of an integrated iot blockchain platform for sensing data integrity," *Sensors*, vol. 19, no. 10, p. 2228, 2019.

[22] M. Zhaofeng, W. Xiaochang, D. K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[23] C. Li and L.-J. Zhang, "A blockchain based new secure multi-layer network model for internet of things," in *Proceedings of the 2017 IEEE International Congress on Internet of Things (ICIOT)*, pp. 33–41, IEEE, Honolulu, HI, USA, 2017.

[24] S. Mondal, K. P. Wijewardena, S. Karuppuswami, N. Kriti, D. Kumar, and P. Chahal, "Blockchain inspired rfid-based information architecture for food supply chain," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5803–5813, 2019.

[25] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," Manubot," Cryptography Mailing list at https://metzdowd.com, 2009.

[26] D. Puthal, S. P. Mohanty, V. P. Yanambaka, and E. Kougianos, "Poah: a novel consensus algorithm for fast scalable private blockchain for large-scale iot frameworks," 2020, http://arxiv.org/abs/2001.07297.

[27] S. Biswas, K. Sharif, F. Li et al., "A light weight consensus algorithm for scalable iot business blockchain," *IEEE Internet of Things Journal*, 2019.

[28] J. Huang, L. Kong, G. Chen, M.-Y. Wu, X. Liu, and P. Zeng, "Towards secure industrial iot: blockchain system with credit-based consensus mechanism," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, 2019.

[29] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3527–3537, 2019.

[30] Research summary of gambler's ruin problem. https://www.jianshu.com/p/7df33ae5fb56s, 2018.