electronics

MDPI

Article

A Data Sharing Model for Blockchain Trusted Sensor Leveraging Mimic Hash Mechanism

Gaoyuan Quan [1,2,3], Zhongyuan Yao [1,2,3,*], Xueming Si [1,2], Weihua Zhu [1,2,3] and Longfei Chen [1,2,3]

[1] Frontier Information Technology Research Institute, Zhongyuan University of Technology, Zhengzhou 450007, China; 2021116579@zut.edu.cn (G.Q.); 9770@zut.edu.cn (X.S.); 9773@zut.edu.cn (W.Z.); 2021016570@zut.edu.cn (L.C.)
[2] Henan International Joint Laboratory of Blockchain and Data Sharing, Zhengzhou 450007, China
[3] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450007, China
* Correspondence: yaozhongyuan@zut.edu.cn

Abstract: Blockchain, as a distributed trust database, has been widely applied in the field of trustworthy sharing of Internet of Things (IoT) sensor data. A single hash mechanism has achieved, to some extent, the trustworthy on-chain storage of blockchain sensor data, that is, the consistency of data on and off the chain. However, it still faces potential security risks such as collision attacks, short password attacks, and rainbow table attacks. To address this issue, this paper proposes a resiliently secure blockchain sensor data trustworthy sharing model based on a mimic hash mechanism. Specifically, in response to the security risks that may arise from the single hash mechanism, this study innovatively introduces a mimic hash mechanism and proposes two methods for constructing mimic hashes based on Verifiable Random Function (VRF) and Cyber Mimic Defense (CMD) in dedicated Wireless Sensor Networks (WSNs) and open public networks, respectively. Theoretical analysis and experimental results demonstrate that this model effectively solves the problem of trustworthy on-chain storage of sensor data in edge computing environments, enhancing the trustworthiness and security of the data on the chain.

Keywords: mimic hash mechanism; blockchain trustworthy sensor; data trustworthy on-chain

Citation: Quan, G.; Yao, Z.; Si, X.; Zhu, W.; Chen, L. A Data Sharing Model for Blockchain Trusted Sensor Leveraging Mimic Hash Mechanism. *Electronics* **2024**, *13*, 1495. https://doi.org/10.3390/electronics13081495

Academic Editor: Zbigniew Kotulski

Received: 25 March 2024
Revised: 10 April 2024
Accepted: 12 April 2024
Published: 14 April 2024

1. Introduction

China was the first developing country in the world to establish a mature and effective carbon market. Since its inception in 2021, China's carbon emissions trading market has covered approximately 4.5 billion tons of CO2 emissions annually, making it the largest carbon market globally in terms of greenhouse gas emissions [1]. However, the construction of China's emissions trading system faces challenges, including differences in the availability of data from companies and imperfect data quality. The most crucial step towards a long-term solution to these challenges is to clarify data sources and introduce monitoring programs [2]. In a carbon emission monitoring system, a carbon emission sensor is located at the interface between the research subject and the control system and serves as a window for sensing, acquiring, and detecting information. The production and emission data required by the carbon emission monitoring system is converted into an electrical signal that can be easily transmitted and processed through carbon emission sensors. Moreover, the data collection subsystem in the carbon emission monitoring system utilizes various instruments and devices to collect and preliminarily process signals from various carbon emission sensors [3]. Subsequently, the production and emission data in the emissions trading system are transmitted to the emissions trading market by non-data secure vendors through a series of conventional pathways. However, this approach of transmitting production and emission data through traditional pathways neglects the essential data and transmission security requirements, leading to questions about the credibility of the data sources generated by such unprotected IoT sensors.

Blockchain, as a distributed database technology, has empowered traditional Internet, Internet of Things (IoT), and Vehicle Network domains with decentralization, immutability, and traceability. It also provides reliable data storage and integrity verification services for sensor data. However, due to blockchain being a relatively closed system, it is challenging to interact directly with external data. Frequently, external data must be processed before being fed into the blockchain, making it difficult to ensure the credibility of data sources and the consistency between on-chain and off-chain data. Therefore, there is an urgent need to explore a different scheme to address the issues of data source credibility and data integrity, namely, solving the consistency problem between on-chain and off-chain sensor data.

Integrating trusted modules or trusted chip hardware into sensor devices to construct a blockchain-based trustworthy sensor can effectively address the issues of data source credibility and data integrity. However, the data consistency verification method that employs a single hash mechanism is vulnerable to security risks such as hash collision attacks, short password attacks, and rainbow table attacks. First, it may be susceptible to hash collision attacks, implying that different input data can produce identical hash values, thereby enabling attackers to craft specific inputs to deceive the system or alter data. Second, the single hash mechanism is prone to threats from short password attacks, where attackers may attempt various inputs to brute-force the hash value, thus compromising sensitive information. Additionally, rainbow table attacks present a significant risk, where attackers could utilize a precalculated hash value mapping table to locate the original data corresponding to a given hash value, further jeopardizing the integrity and security of the data.

Cyber Mimic Defense (CMD), proposed by Academician Wu Jiangxing of the Chinese Academy of Engineering as a new type of proactive defense technology designed to "change the game rules of cyberspace security", centers on dynamic heterogeneous redundancy. Its aim is to address the uncertainty threats such as unknown vulnerabilities, backdoors, or viruses and trojans in various application layers of cyberspace domains. Inspired by the concept of CMD, this paper innovatively proposes a mimic hash mechanism, utilizing Verifiable Random Function (VRF) and CMD to add dynamic and redundant elements to the original static single hash, thereby constructing a mimic hash scheme characterized by dynamic heterogeneous redundancy. Furthermore, this paper introduces a blockchain sensor data trustworthy sharing model based on the mimic hash mechanism, aimed at solving the data integrity verification issue in the process of uploading sensor data to the blockchain, that is, addressing the consistency problem between on-chain and off-chain sensor data.

The main contributions of this paper are as follows:

(1) Drawing on the concept of Cyber Mimic Defense (CMD), a new type of mimic hash mechanism is proposed. By incorporating a Verifiable Random Function (VRF) and a dynamic heterogeneous redundancy architecture, the intrinsic security and reliability of sensor data during the on-chain process are significantly enhanced.
(2) Based on the mimic hash mechanism, a blockchain sensor data trustworthy sharing model is constructed. This model not only ensures the trustworthy on-chain storage of sensor data but also improves the consistency and security of sensor data, offering a new solution for the sharing and trading of sensor data in the IoT environment.
(3) Through theoretical analysis and simulation experiments, the effectiveness of the proposed model is verified. The experimental results show that this model can effectively solve the problem of trustworthy on-chain storage of sensor data in edge computing environments while ensuring strong consistency and security of the data on the chain.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 discusses the preliminary knowledge involved in this paper. Section 4 introduces the blockchain sensor data trustworthy sharing model based on the mimic hash mechanism. Section 5 discusses two construction methods of the mimic hash mechanism.

Section 6 conducts a related security analysis. Section 7 carries out simulation experiments. Finally, Section 8 summarizes the conclusions of this paper's work and future directions.

2. Related Work

2.1. Data Consistency Verification Based on Hash Algorithms

Gao et al. [4] proposed a lightweight and robust image hash method for verifying the integrity of images in IoT devices. This technique, based on random tensors and angle features, offers efficient performance for IoT applications. D.S. Salman et al. [5] focused on the use of Hash-Based Message Authentication Codes (HMAC) to ensure data integrity; by implementing HMAC, their study enhanced the security of encrypted data in cloud environments. J. Joy et al. [6] discussed hash-based duplication techniques and their role in verifying transactions to maintain the integrity and availability of stored data, highlighting the importance of hash methods in cloud security. Chen et al. [7] explored the progress of blockchain research in terms of security, including the use of hash functions to ensure the security of data transmission and verification mechanisms.

Si et al. [8,9] integrated trusted modules or chip hardware into the sensor side, employing a single Secure Hash Algorithm to ensure the consistency of data on and off the chain. However, due to the limited functionality of generic IoT sensors and the restricted range of data types and values they collect, this method cannot defend against rainbow table attacks [10,11]. Wang et al. [12] analyzed potential attacks on hash algorithms and designed a protection mechanism against the most effective attack, the rainbow table, by introducing noise into each instruction, significantly increasing the attacker's burden and thus enhancing the security of the monitoring model and protecting instruction information. Qi et al. [13] pointed out the misconceptions of traditional MD5 hash algorithms and salted hash algorithms in certain application scenarios. They optimized the application process of salted hash algorithms using the PBKDF2 iterative algorithm, reproduced the hashing process through Python programming, and verified through testing that this slow hash algorithm's application could render dictionary or brute-force attacks impractically slow with high-performance hardware. Zhang et al. [14] took the security design of a clinical feedback system commonly used inside and outside hospitals as a case study and proposed a password storage scheme for a "salted slow hash" algorithm. They chose the bcrypt algorithm as the implementation scheme for the system and provided a detailed process for password encryption storage and verification, system design, and system integration. Zhu et al. [15] designed a salted digest model using salting techniques, cryptographic technology, and several basic digest algorithms to address the security and availability issues faced by real-world digest services. Schemes [12–15] resist rainbow table attacks by adding "salt" as interference information to the original data and using slow hash algorithms to increase the difficulty of dictionary or brute-force attacks. However, the "salt" here acts like a key in asymmetric cryptography, whose security must be ensured through complex cryptographic methods.

2.2. Blockchain-Based Trustworthy Sharing of Sensor Data

In the domain of blockchain-trustworthy sensors, solutions [9,10,16] have added trusted modules or trusted chip hardware at the sensor end to create a type of blockchain-trusted sensor. Reference [9] generates random key seeds based on key hash chains and produces disposable key hash chains through a recursive relationship. It broadcasts the first node containing the node device ID and the key hash chain. During each data transmission, it sends an authentication to neighboring nodes and eventually chains the verification result to ensure the authenticity and trustworthiness of the data source. Additionally, a single secure hash algorithm is used to ensure the consistency of data on and off the chain.

Reference [10] proposes a blockchain data transmission method based on keyless signatures, where the trustworthy sensor module operates in predefined cycles, receiving plaintext data transmitted by sensors. It computes digests for the plaintext data within the cycle to obtain digest hash values. The trustworthy sensor module aggregates these

digest hash values and uploads the current aggregated global hash tree information to an aggregation server for layered aggregation. The aggregation of the top-level node in the layered aggregation is performed by blockchain nodes, achieving and storing the root node information of the hash calendar. Through the aggregation of digest hash values by both the trustworthy sensor module and the aggregation server, and the completion of the top-level aggregation by blockchain nodes, a hash calendar is obtained. Based on the top-level root node information stored in the blockchain and the characteristics of the hash calendar, recursive verification of each layer of nodes in the hash calendar can be conducted, ensuring the authenticity and validity of the data, and effectively reducing the data storage requirements of the blockchain system.

Reference [16] proposed a blockchain data storage method and terminal oriented to sensor networks, which packages the collected sensor data into the sensor network blockchain. After the sensor network blockchain terminal data meets the storage or time conditions, the blockchain data are offloaded to the blockchain server, which then forwards it for storage in the data backup blockchain. After successful storage, a verification confirmation message is sent to the sensor network blockchain. Therefore, using a dual-chain data storage method can effectively avoid data overflow problems caused by insufficient storage resources, ensuring the timeliness and traceability of the original data without affecting the operation of the original blockchain network, thus improving the communication quality of the sensor network.

2.3. Endogenous Security Mimic Defense

Wu [17] analyzed the unbalanced state of network security and its fundamental issues, focusing on the principles and methods of using untrustworthy software and hardware components to compose a highly reliable and secure information system through a dynamic heterogeneous redundancy architecture. He also briefly introduced the basic concepts of mimetic defense. Finally, he described the testing and preliminary conclusions of a mimetic defense principle verification system. Tong et al. [18] proposed a mimetic defense model based on the "dynamic heterogeneous redundancy" structure and described the defense principles and characteristics of the mimetic defense model. They built a mimetic defense web server based on this model, introduced its architecture, and analyzed the implementation of mimetic principles on web servers.

Ma et al. [19] introduced heterogeneous redundant functional entities into router architecture based on mimetic defense technology. Through a dynamic scheduling mechanism, multiple heterogeneous functional entities are randomly selected to work under the same external stimuli. By comparing the output results of various heterogeneous functional entities, abnormal detection is carried out on the functional entities to achieve proactive defense of the router system. Wang et al. [20] proposed a simple and easy-to-deploy proactive defense architecture with intrusion tolerance—Mimetic DNS (M-DNS)—to ensure DNS security. This architecture consists of a selector and a server pool containing multiple heterogeneous DNS servers. First, the selector dynamically selects several servers to process requests in parallel, and then a voting mechanism is used to determine the final valid response. Xu et al. [21], addressing the potential security issues in blockchain, proposed a blockchain security solution called mimetic blockchain, inspired by dynamic heterogeneous redundancy architecture and cryptographic lottery drawing. This solution is based on dynamic heterogeneous consensus mechanisms and dynamic heterogeneous redundancy signature algorithms, combining security definitions and parameter selection rules.

3. Preliminary Knowledge

3.1. Blockchain

Blockchain technology originated from a foundational paper published in a cryptography mailing list in 2008 by an individual or group under the pseudonym “Satoshi Nakamoto” titled “Bitcoin: A Peer-to-Peer Electronic Cash System”. Blockchain is a decentralized, immutable, traceable, and collaboratively maintained distributed database. It can

integrate multiple isolated databases, which are traditionally maintained by single parties involving their own business, and distribute them across multiple nodes maintained by various parties. No single party can completely control these data; updates can only be made according to strict rules and consensus. This achieves trustworthy information sharing and supervision among multiple parties, avoids cumbersome manual reconciliation, improves business processing efficiency, and reduces transaction costs. Blockchain solves the issue of data trustworthiness through the integration of various technologies such as P2P protocol, asymmetric encryption, consensus mechanisms, and the blockchain structure itself. By applying blockchain technology, trustworthy, peer-to-peer value transfer can be achieved without the need for any third-party trusted institution among parties that do not know or trust each other [22].

To ensure the immutability of data, blockchain introduces a chain structure with blocks as units. While different blockchain platforms may vary in the specifics of their data structures, the overall concept remains primarily the same. Taking Bitcoin as an example, each block consists of a block header and a block body. The block body contains multiple transactions that have occurred since the previous block; the block header contains the previous block's hash (PreBlockHash), a nonce (Nonce), and a Merkle root (Merkle Root), among others.

Blockchain ensures data immutability through two hash structures: the Merkle tree and the linked list, as illustrated in Figure 1 [23]. Bitcoin employs the simplest form of a binary Merkle tree. Each node in the tree is a hash value, with each leaf node representing the SHA-256 hash of a transaction within the block. By concatenating the values of two child nodes and then performing a hash operation, the value of the parent node is obtained. This process of pairwise hashing is repeated until the root hash value, or the Merkle root of transactions, is generated. The Merkle root allows for the detection of any tampering with transaction data within the block, thus ensuring the integrity of the transaction data. Without needing the participation of additional nodes in the tree, the existence of a transaction in the block can be confirmed using Simplified Payment Verification (SPV), based on the direct branch from the transaction node to the Merkle root.

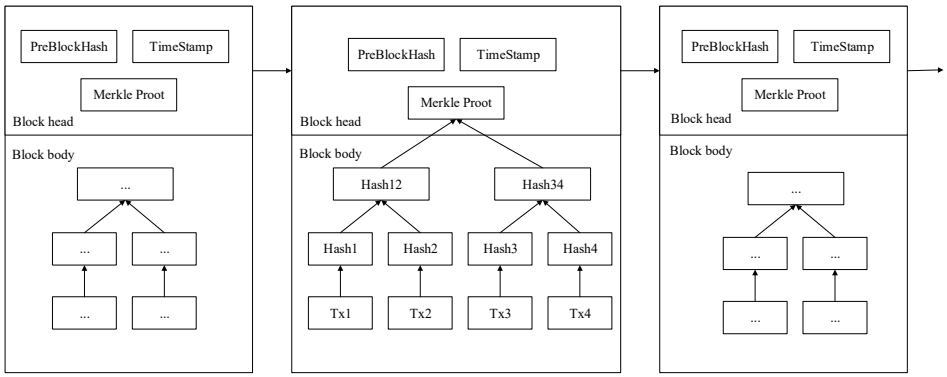

Figure 1. The Two Hash Structures of Blockchain.

The PreBlockHash stores the block hash of the previous block, and all blocks are linked together in the order they were generated using the PreBlockHash as a hash pointer, forming a blockchain-linked list. The block header includes the transaction Merkle root; therefore, the block hash can be used to verify whether the block header and the transaction data within the block have been tampered with. The block header also contains the previous block hash (PreBlockHash), so the block hash can further verify whether all blocks up to the genesis block preceding it have been altered. Relying on the hash pointer of the previous block (PreBlockHash), all blocks are interlocked, and any tampering with a block would trigger a chain reaction altering the hash pointers of all subsequent blocks. When downloading a certain block and all preceding blocks from an untrusted node, the block hash can be used to verify whether any of the blocks have been modified.

3.2. Cyber Mimic Defense (CMD)

Cyber Mimic Defense (CMD) [17] is a type of proactive defense behavior. Inspired by biological phenomena, in 2008, Academician Wu Jiangxing and his team proposed the concept of Dynamical, Heterogeneous, and Redundant (DHR) construction based on the axiom of relative correctness, drawing from reliability and automatic control theories. By employing a mimicry mechanism, this approach renders the system externally unpredictable, thereby endowing it with intrinsic security features. Combining mimic defense with existing security technologies can exponentially increase defensive gains. The use of mimic defense technology allows for the normalization of traditional and non-traditional security issues and enables the system to acquire generalized robust control properties.

A mimic defense system typically consists of components such as the information system, mimic transformations, heterogeneous executors, and a voter. The system achieves transitions between different states of the information system by actively changing the state of its constituent elements, a process known as mimic transformation. Mimic transformations effectively alter the static and deterministic nature of systems upon which network attacks rely, and these transformations will be described in detail later. The introduction of a voter aims to further confuse attackers, reduce the risk of system attacks, and increase system reliability. It introduces an uncertainty mapping relationship between the target object's meta-function and its equivalent multifarious implementation structures or algorithms, thereby concealing the real relationship between system inputs and outputs. The Moving Target Defense (MTD) system can be considered a specific case of a mimic defense system, which endows the system with dynamic characteristics through various mimic transformations but does not apply a heterogeneous redundant architecture [24].

3.3. The Blockchain Trusted Sensors

Blockchain trusted sensors consist of sensors and trusted modules with certain data processing capabilities, forming a reliable hardware system. These sensors leverage cryptography, blockchain, fault diagnosis, and privacy protection technologies to ensure the trustworthiness of data collected at the source. This enables the data to remain unaltered and reliably transmitted from the point of collection to the server, addressing the consistency issues between data on the blockchain and off the blockchain. The forms of blockchain-trusted sensors include trusted blockchain sensor components and trusted blockchain sensor chips, which are used to ensure the reliability of the data sources being added to the blockchain.

The trusted blockchain sensor component includes sensors and trusted sensor modules, with its structural diagram shown in Figure 2. The trusted sensor chip comprises modules for security protection, fault detection, plaintext critical data encoding, and critical data privacy protection, with the structural diagram of the trusted sensor chip illustrated in Figure 3.

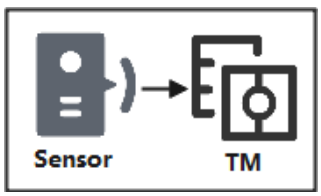

Figure 2. Trusted Blockchain Sensor Component. Herein, "Sensor" refers to the sensor, and "TM" stands for Trusted Module.

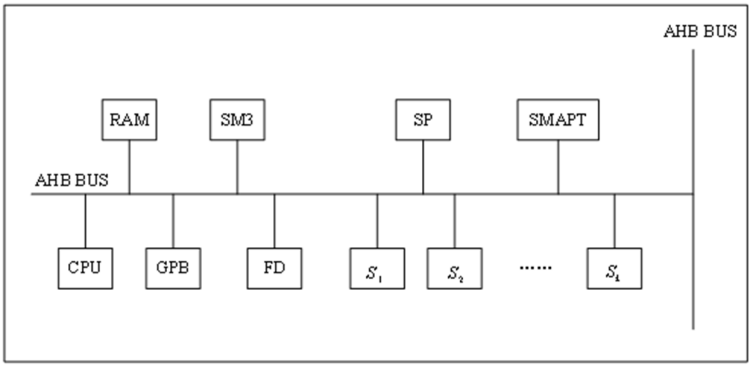

Figure 3. Structure of the Trusted Sensor Chip. Within this context, $S_1, S_2, \ldots\ldots, S_k$ represent the k connected sensors. RAM denotes Random Access Memory. SP stands for the Safety Protection module, FD refers to the Fault Diagnosis module, GPB is the Google Protocol Buffer encoding module, and SMART signifies the data privacy protection module.

4. Blockchain Sensor Data Trustworthy Sharing Model Based on Mimic Hash Mechanism

4.1. Model Architecture

Solutions [4,5] have developed a blockchain trusted sensor that utilizes cryptography, blockchain, fault diagnosis, and privacy protection technologies to ensure the trustworthiness of data collected at the source. This ensures the data from the collection point to the server is immutable and reliably transmitted, addressing the issue of consistency between data on the blockchain and off the blockchain. However, considering the hash chain mode of existing single hash algorithms, it is susceptible to specific attacks targeting the hash algorithm. This paper posits that randomness and dynamism are fundamental to the security of hash mechanisms. Building upon the original blockchain sensor scheme, this paper constructs a blockchain sensor data trustworthy sharing model based on a mimic hash mechanism, as illustrated in Figure 4.

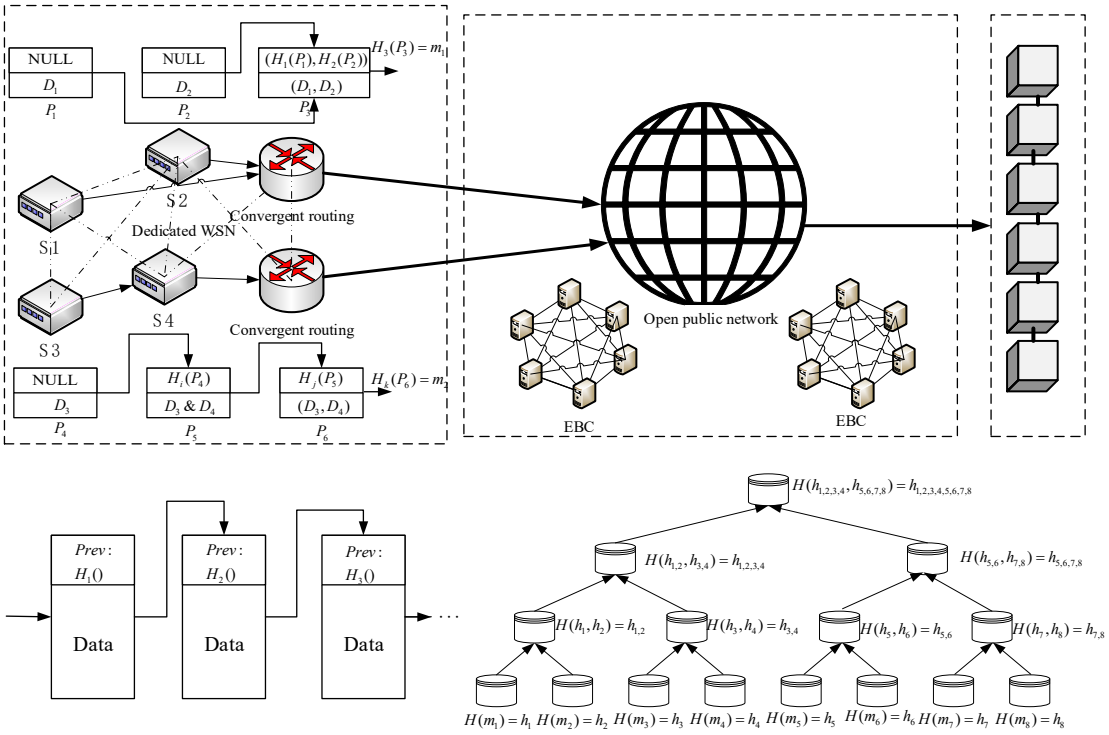

Figure 4. Blockchain Trusted Sensor Data Sharing Model Based on the Mimic Hash Mechanism.

The blockchain sensor data trustworthy sharing model based on the mimic hash mechanism is divided into cloud, edge, and end layers. The cloud layer is composed of data blockchains, responsible for storing the final sensor data and its verification information. The edge layer consists of multiple parallel edge blockchains deployed in open public networks, formed by mimic hash servers. The edge blockchains primarily handle the transmission and circulation security of sensor data within the edge network and may temporarily store part of the data based on system status. This not only disperses the network burden of massive data uploads to the data blockchain but also provides end-layer users with more real-time data services. The end layer is made up of blockchain-trusted sensors, responsible for collecting and transmitting business data, with the blockchain-trusted sensors at the end layer together forming a dedicated wireless sensor network.

Addressing the issue of limited resources in terminal sensor devices and the security risks posed by unreliable edge nodes that may suffer from physical attacks and threats of malicious software intrusion, this paper proposes distinct mimic hash schemes for dedicated wireless sensor networks and public open networks, respectively, to solve the potential security challenges they face.

The mimic hash mechanism, based on mimic defense, is a current network security defense technology whose core principle is dynamic heterogeneous redundancy. It can actively change the state of information system components, facilitating the migration of information across different state spaces. Current integrity verification systems based on hash functions can be considered as simple isomorphic static structures, where nodes use the same hash function for every transmission and verification. The proof algorithms for integrity verification of transmitted data in networks such as sensor networks, vehicular networks, and the Internet of Things represent such isomorphic static structures. This structure offers convenience to attackers and is a significant reason why current hash function-based integrity verification systems face security threats.

The blockchain sensor data trustworthy sharing model based on the mimic hash mechanism will utilize the core of mimic defense, dynamic heterogeneous redundancy (DHR), to add dynamic and heterogeneous elements to the original static homogenous data integrity verification foundation. This approach constructs a dynamic and heterogeneous blockchain sensor network data security transmission and integrity verification scheme, enhancing the security of data integrity verification and thereby ensuring the authenticity of blockchain on-chain data and the credibility of its source.

4.2. Network Deployment

The blockchain trusted sensor data sharing model based on the mimic hash mechanism adopts a master-slave collaborative chain architecture design, referencing the deployment of cloud-edge-device architecture. It includes three layers: the device side, edge side, and cloud side. The device side deploys a dedicated WSN (Wireless Sensor Network), while the edge side and cloud side are deployed in an open public network, as shown in Figure 5.

The cloud side includes the user data main chain and third-party cloud computing services purchased by users, with the user data main chain used for storing all data hashes in the business scenario.

The edge side consists of service sub-chains built by untrusted edge nodes located near the source data collected by sensors. Multiple parallel sub-chains can be constructed according to actual business needs. The edge blockchain can provide computing and data caching services for device-side sensors and aggregation routers. The caching function of the edge blockchain is similar to that of the database Redis, allowing cached data on the edge blockchain to provide query services for users within the business cycle. When the business cycle completes or the storage limit of the edge blockchain is reached, historical data will be overwritten by new data.

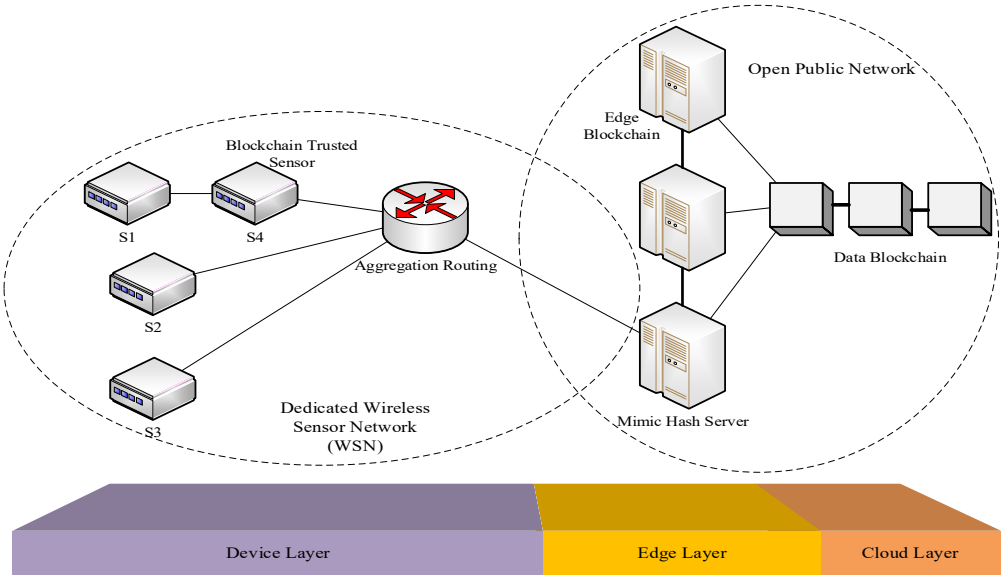

Figure 5. Network Deployment of the Blockchain-Trusted Sensor Data Sharing Model Based on the Mimic Hash Mechanism.

The device side includes blockchain-trusted sensors and aggregation routers; blockchain-trusted sensors are responsible for data collection, initial hashing of source data, and data forwarding. Aggregation routers are tasked with the aggregation and forwarding of data.

The data flow process of the blockchain trusted sensor data sharing model based on the mimic hash mechanism is as shown in Figure 6:

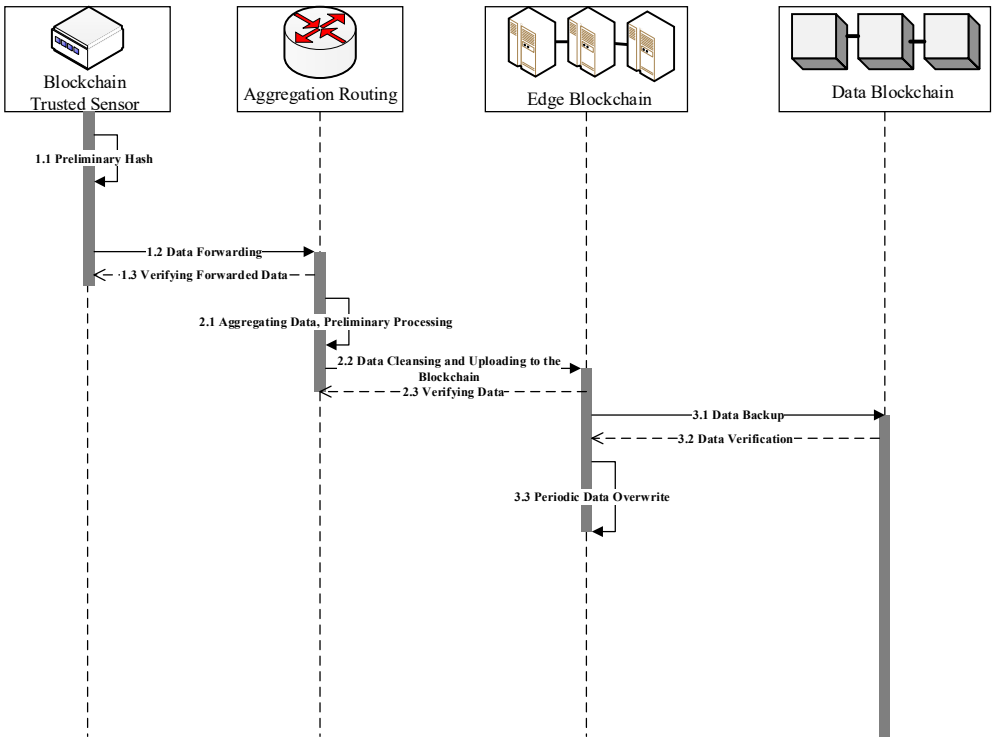

Figure 6. Data On-Chain Process of the Blockchain-Trusted Sensor Data Sharing Model Based on the Mimic Hash Mechanism.

Step 1: The blockchain trusted sensor collects source data, initially hashes the data using the hash algorithm pre-burned into the trusted blockchain chip or component, and forwards it to the aggregation node through a dedicated network.

Step 2: The aggregation node first verifies and preliminarily processes the aggregated data set, then uploads it to the edge blockchain.

Step 3: The edge blockchain cleanses the data by removing dirty data according to business needs and uploads the final data to the data blockchain for backup; it also periodically overwrites historical data based on the actual storage limits of the edge blockchain.

5. Mimic Hash Mechanism

Considering the diversity of security defense methods and data transmission modes across different networks, this paper proposes distinct approaches to construct mimic hashes in dedicated WSN networks and open public networks, aiming to achieve consistency between on-chain and off-chain sensor data.

5.1. Mimic Hash Scheme Based on Verifiable Random Function

Wireless Sensor Networks (WSN) are composed of numerous sensor nodes tasked with gathering data and transmitting it to a central node, commonly known as the base station. The transmission of data packets typically encompasses the following steps:

(1) Data Collection:

Sensor nodes harvest data from their surroundings, such as temperature, humidity, and pressure.

(2) Data Packaging:

The gathered data are encapsulated into data packets. This stage may involve formatting the data and appending essential header details, like the sender and receiver's addresses and the packet's sequence number.

(3) Routing Decision:

Sensor nodes determine the optimal route for sending the data packet to the base station, either through direct communication with the base station or via multi-hop transmission through other network nodes.

(4) Data Transmission:

Data packets are conveyed over wireless channels, a process that may include signal modulation and encoding, alongside potential error detection and correction techniques.

(5) Relay Transmission (for Multi-hop Transmission):

In scenarios employing multi-hop transmission, data packets traverse multiple intermediate nodes. Each node receives, possibly processes, and subsequently forwards the packet to the next node.

(6) Arrival at the Base Station:

Data packets ultimately arrive at the base station, where the collected data undergo processing, storage, or further transmission.

(7) Possible Acknowledgment and Feedback:

Certain WSN protocols may enable the base station or intermediary nodes to dispatch acknowledgment messages back to sensor nodes, confirming the successful reception of the data packets.

Therefore, drawing inspiration from the concept of mimic defense, this paper introduces a mimic hash mechanism that incorporates three different hash algorithms. Moreover, the mimic hash mechanism utilizes Verifiable Random Function (VRF) technology. For every data packet forwarding moment, each subsequent hop data packet aggregator must create a VRF certificate based on the hash values of the proximate data packets received in

the sensor network and the current data packet sequence number's hash value signature. The function of the VRF certificate is to determine the hash algorithm that the data packet aggregator should use in this round according to the VRF certificate. The following outlines the specific steps of the mimic hash mechanism:

(1) System Model and Assumptions
   a. System Model: Consider a Wireless Sensor Network (WSN) composed of multiple sensor nodes, each responsible for collecting environmental data and transmitting it via wireless communication.
   b. Security Assumptions: It is assumed that there are at least three different secure hash algorithms available within the network, and adversaries cannot predict in advance which hash algorithm each node will use.

(2) Core Design of the Mimic Hash Mechanism
   a. Hash Algorithm Selection: Define three different hash algorithms H_1, H_2, H_3.
   b. Round Definition: Define the round r as the sequence of data packet transmission in the WSN, initialized to $r = 0$.

(3) Data Collection and Processing
   a. Data Collection: In each round r, sensor nodes collect environmental data D_r.
   b. Hash Certificate Generation: Nodes generate a hash certificate Ce_r for their data D_r and the state from the previous round $State_{r-1}$.

(4) Dynamic Selection of Hash Algorithm
   a. Certificate Calculation: The ith sensor calculates the Verifiable Random Function certificate $Ce_i^r = H(sig_i(D_r||State_{r-1}||r))$.
   b. Algorithm Determination: Dynamically select the hash algorithm H_i^r based on Ce_i^r, where:

$$H_i^r = \begin{cases} H_1, Ce_i^r \equiv 0 \text{mod} 3 \\ H_2, Ce_i^r \equiv 1 \text{mod} 3 \\ H_3, Ce_i^r \equiv 2 \text{mod} 3 \end{cases}.$$

(5) Hash Processing and State Update
   a. Hash Processing: Hash the data D_r using the selected hash algorithm to generate the hash value H_{val}^r.
   b. State Update: Update the node state to $State_r = H(State_{r-1}||H_{val}^r)$.

(6) Blockchain Integration
   a. Blockchain Records: Store $State_r$ along with related information (such as timestamps, node ID, etc.) in the blockchain to ensure data integrity and traceability.

(7) Communication and Verification
   a. Data Transmission: Transmit the updated state $State_r$ and related verification certificates through the WSN.
   b. Verification: The receiving node verifies the incoming $State_r$ and its associated verification certificates to ensure the data's integrity and authenticity.

(8) Update of Transmission Round
   a. Increment Round: After completing a round of data transmission and verification, increment the round r to prepare for the next cycle of data collection and processing.

The WSN Mimic Hash Mechanism is shown in Algorithm 1 below:

Algorithm 1: WSN Mimic Hash Mechanism

Input: Sensor data set D, previous round state $State_{r-1}$, round r
Output: Updated state $State_r$
1: function $MIMETIC_{HASH}(D,\ State_{r-1}, r)$
2: Initialize hash algorithms H_1, H_2, H_3
3: Initialize round $r = 0$
4: Generate initial state $State_0$ for each node
5: for each round r do
6: // Data collection
7: $D_r \leftarrow COLLECT_{DATA}()$
8: // Generate hash certificate
9: $Ce_r \leftarrow H(sig(D_r\ ||State_{r-1}||\ r))$
10: // Select hash algorithm based on certificate
11: switch $(Ce_r \bmod 3)$
12: case0 : $H_i^r \leftarrow H_1$
13: case1 : $H_i^r \leftarrow H_2$
14: case2 : $H_i^r \leftarrow H_3$
15: end switch
16: // Calculate hash value
17: $H_{val_r} \leftarrow H_i^r\ (D_r)$
18: // Update state
19: $State_r \leftarrow \mathrm{H}(State_{r-1}\ ||H_{val_r})$
20: // Record to blockchain
21: $RECORDTOBLOCKCHAIN\ (State_r,\ r,\ node_{info})$
22: // Transmission and verification
23: $TRANSMITANDVERIFY(State_r,\ node_{next})$
24: // Update round
25: $r \leftarrow r + 1$
26: end for
27: return $State_r$
28: end function

5.2. Mimic Hash Scheme Based on Mimic Defense

5.2.1. Model Architecture

The mimic hash based on mimic defense is a new hash mechanism for data integrity verification inspired by the concept of mimic defense. Mimic hash will utilize a dynamic heterogeneous redundancy architecture (DHR) to add dynamic and heterogeneous elements on top of the traditional static hash algorithm for data integrity verification. This constructs a dynamic and heterogeneous scheme for secure data transmission and integrity verification, as shown in Figure 7.

The blockchain trusted sensor is composed of target sensors, trusted sensor components/chips, aggregation nodes (routers or blockchain trusted sensors), blockchain BaaS (Blockchain as a Service) platforms, identity authentication systems, and the mimic hash system.

5.2.2. Operating Mechanism

The mimic hash system based on mimic defense primarily consists of a request distribution module, heterogeneous executors, a central scheduler, a response redundancy voter, and a monitoring module.

The request distribution module acts as the genuine entry point for network requests. It replicates user requests and dynamically forwards them to the heterogeneous executor modules according to a scheduling strategy, laying the groundwork for the dynamic variation of the heterogeneous executors. The response redundancy voter, following a predetermined algorithm, votes on multiple responses to the same request from users, produces a unified output result and feeds this result back to the central scheduler. The central scheduler, based on the scheduling algorithm and feedback information, generates

and sends a scheduling strategy to the request distribution module. The pool of heterogeneous executors, implementing the same functionality through different constructions, significantly increases the system's dynamism and reduces the likelihood of common-mode vulnerabilities. The monitoring module monitors the status of sensors or sensor networks and generates scheduling parameters based on the monitoring results. The structure is illustrated in Figure 8.

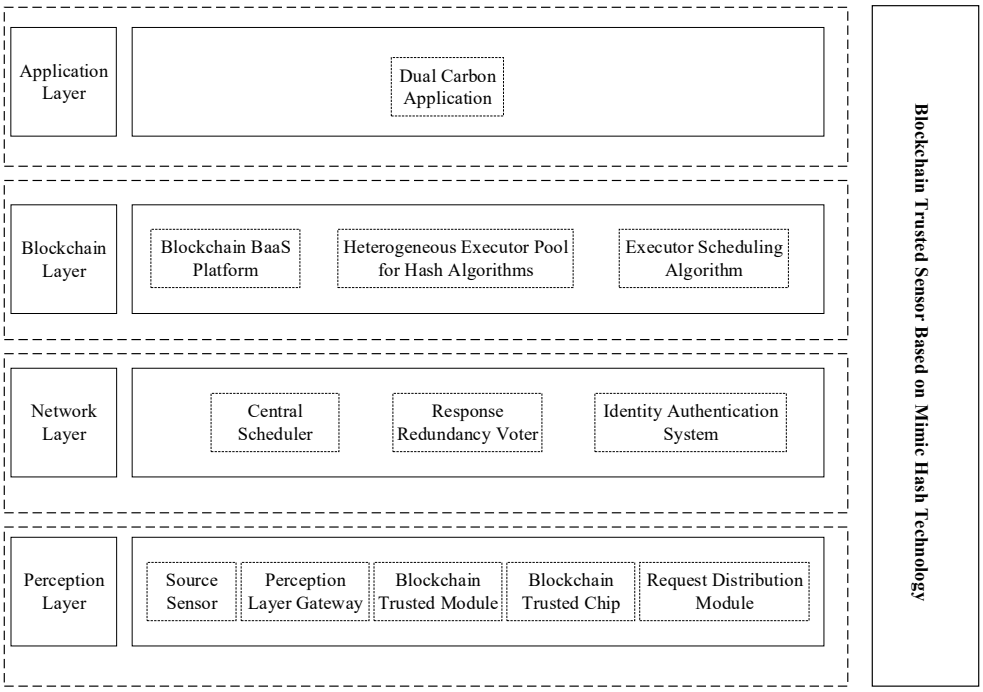

Figure 7. Architecture of the Mimic Hash Model Based on Mimic Defense.

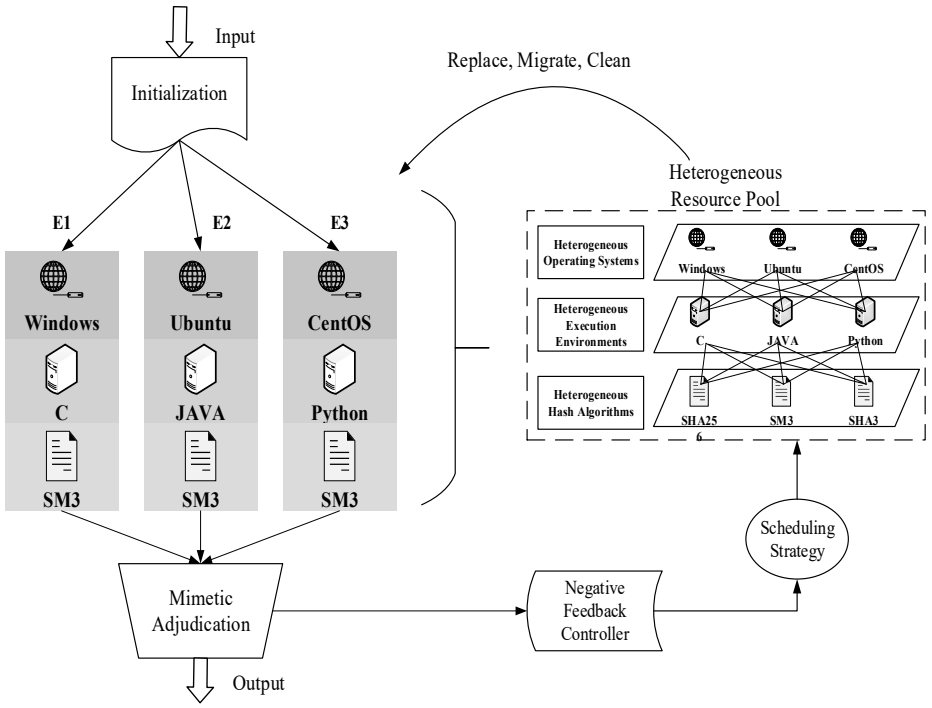

Figure 8. Describing the Operating Mechanism of the Mimic Hash System Based on Mimic Defense from the Perspective of Functionally Equivalent Heterogeneous Executors.

In the mimic hash system based on mimic defense, all modules, including the input proxy, central scheduler, heterogeneous executors, and redundancy voter, are designed at the software level. Their final implementation is distributed across various levels of the blockchain trusted sensor system, such as trusted components, trusted chips, or edge servers, within the blockchain.

Describing the operating mechanism of the mimic hash system based on mimic defense from the perspective of functionally equivalent heterogeneous executors, as illustrated in Figure 8:

(1) Initialization: Select an initial set of three heterogeneous executors (including random and different *S* and *R*, and the same *H*) from the heterogeneous resource pool as the working set.
(2) Input Proxy: According to the instructions of the negative feedback controller, the input proxy distributes the input sequence (original data) to the corresponding (3) heterogeneous executors.
(3) Reconfigurable Executor Set: The three heterogeneous executors, stimulated by the input, should under high-probability conditions be able to work normally and independently produce output vectors (hash results) that meet given semantics and syntax.
(4) Multimode Adjudicator: Based on the adjudication parameters or algorithm-generated adjudication strategy, assesses the consistency of the multimode output vector content and forms an output response sequence. If an undesired state is detected, it activates the negative feedback controller.
(5) Negative Feedback Controller: Upon activation, decides whether to send a command to the output proxy to replace (migrate) the "output anomaly" executor based on the control algorithm generated by control parameters, or instruct the suspected problematic executor to implement online/offline cleaning and recovery operations (including triggering additional background processing functions), or perform a combination of reassembly, reconstruction, and reconfiguration operations on the anomalous executor under functionally equivalent conditions based on software and hardware components. This activation process continues until the inconsistency in the output vector disappears in the multimode adjudication phase or the frequency of such occurrences falls below a given threshold, at which point it pauses.

5.2.3. Blockchain-Based Mimic Hash Executor Scheduling and Consensus Adjudication Method

In the edge computing environment, we have defined a blockchain-based mimic hash executor scheduling and consensus adjudication method. The heterogeneous components *S* and *R* are distributed and deployed on edge servers, while the heterogeneous component *H* is stored in the edge blockchain composed of edge servers.

Assuming there are w heterogeneous operating systems (abbreviated as *S*) such as Windows, Ubuntu, CentOS, m heterogeneous execution environments (abbreviated as *R*) such as C, JAVA, Python, C++, and n heterogeneous hash algorithms (abbreviated as *H*) such as MD5, SHA256, SHA3, SM3. We consider heterogeneous operating systems, execution environments, and hash algorithms collectively as heterogeneous components, each assigned a unique identifier i.

To quantify the performance and reliability of heterogeneous components, we define a heterogeneous component weight function as follows:

$$W_i = \alpha P_i + \beta Q_i + \gamma C_i$$

Here, P_i represents the performance of the heterogeneous component, Q_i represents the historical success rate of the component's executions, and C_i represents the credibility value of each component. The performance metric P might be calculated based on specific performance indicators, such as computation speed, storage space, network bandwidth, etc., with a value range between [0 and 1]. The historical execution success rate Q is derived from the proportion of tasks successfully executed by the component in the past.

The credibility value C represents the credibility of the input data. α, β, and γ are weight coefficients, adjusted according to the actual requirements of the system, and satisfying $\alpha + \beta + \gamma = 1$.

Next, we need to construct weight lists for heterogeneous components, namely $List < \mathrm{S} >$, $List < R >$, $List < H >$, and construct the weight of the heterogeneous executor (abbreviated as E), where the weight of E is the weighted average of the weights of the components $W_E = (W_S + W_R + W_H)/3$.

During the executor scheduling phase, we employ the PBFT (Practical Byzantine Fault Tolerance) consensus algorithm, with the following specific steps:

a. Select the top j weights from $List < S >$, $List < R >$, and one H to from j executors $E(1 \leq j \leq n$, where n is the number of all possible E combinations in the system). Launch jEs online and form an endorsing node list $List < E >$ sorted by weight.
b. When a new task arises, only these endorsing nodes are required to compute and produce results.
c. Once an endorsing node completes its computation, it sends its result to other nodes. When more than $2/3$ of the nodes receive the same result, the task is considered complete. The nodes that complete the computation are then awarded certain weight rewards, which might be achieved by increasing the node's Q value (execution success rate) or directly increasing the node's weight.

During the consensus adjudication phase, a threshold T is set, which determines whether an executor can remain in the list of endorsing nodes.

In this phase, each executor can participate in voting. However, if an executor's number of votes exceeds the threshold T within a certain period (indicating frequent disagreement with the computed results), its reliability or correctness may be questioned. In such cases, the executor will be removed from the list of endorsing nodes, and the negative feedback controller will be triggered for further action. This action may include reducing its weight or even removing it from the system altogether. In essence, the threshold T is a safety mechanism we set; if an executor's voting frequency exceeds T, it suggests potential issues with the executor, necessitating further processing. The initial value of T can be dynamically adjusted based on system requirements and node behavior.

a. If a node finds that its computation result is inconsistent with the results of the majority (over $2/3$) of nodes, it initiates a vote, requesting all nodes to verify this result.
b. If a node's number of votes exceeds the threshold T within a certain period, it will be removed from the list of endorsing nodes, triggering the negative feedback controller for processing.

During the heterogeneous executor scheduling phase, machine learning algorithms (for example, decision tree algorithms based on historical data) are introduced to predict which executors might produce erroneous voting results and to process them accordingly.

a. If a node's weight is reduced or it is removed from the list of endorsing nodes, its relevant historical data will be collected, including changes in its weight, number of votes, and accuracy of computation results.
b. Use this historical data to train a decision tree model to predict which nodes may produce erroneous voting results. Divide the data into a training set and a test set, use the training set to train the model, and finally use the test set to evaluate the model's performance.
c. Based on the results predicted by the model, take preemptive action on nodes that may pose problems, such as reducing their weight or removing them from the list of endorsing nodes.

6. Security Analysis

6.1. Security Objective Definitions

Definition 1 (Data Integrity $\mathcal{I}$). *Data integrity is defined as a function $\mathcal{I}(D, D') = true$, if and only if data D and D'remain consistent throughout any operation process.*

Definition 2 (Data Privacy $\mathcal{S}$). *Data privacy is defined as an assertion $\mathcal{S}(D, U) = true$, if and only if user U cannot access unauthorized data D.*

Definition 3 (System Reliability $\mathcal{R}$). *System reliability is defined as a probability function $\mathcal{R}(S) > \theta$, where S represents the system state and θ is the acceptable minimum threshold of reliability.*

6.2. Security Proof

Theorem 1 (Data Integrity). *For any data D and D' , if D after being processed by the mimic hash mechanism $\mathcal{M}_{mim}(D)$ produces the same hash value as $\mathcal{M}_{mim}(D')$, then D must be equal to D'.*

Proof. According to Definition 1, data integrity $\mathcal{I}(D, D') = true$ holds if and only if data D and D' remain consistent throughout any operation process. Considering the mimic hash mechanism $\mathcal{M}_{mim}$, which processes data by dynamically selecting a hash algorithm, if $\mathcal{M}_{mim}(D) = \mathcal{M}_{mim}(D')$ produces the same hash value, then under the assumption of collision resistance, it is considered that D must equal D', thereby ensuring data integrity. □.

The mimic hash mechanism $\mathcal{M}_{mim}$ dynamically selects a hash algorithm at each data transmission. Assuming the system selects three hash algorithms H_1, H_2, H_3, and each time data is transmitted through the dedicated wireless sensor network at the terminal layer, the mimic hash based on Verifiable Random Function (VRF) technology randomly chooses one of these three algorithms. Assuming D and D' produce the same hash value after being processed by $\mathcal{M}_{mim}$, it can be considered that they were processed in the same round and selected the same hash algorithm H_i. Since each data packet's processing is independent and the selection of the hash algorithm is random, the probability of two different data packets D and D' using the same hash algorithm H_i is $1/3$.

By using a proof by contradiction, it is easy to prove that assuming $D \neq D'$ but $\mathcal{M}_{mim}(D) = \mathcal{M}_{mim}(D')$, the collision resistance of the hash function H_i makes the probability of different inputs producing the same output extremely low. Therefore, this assumption is unlikely to be held under a high probability.

Theorem 2 (Collision Resistance). *Under the mimic hash mechanism $\mathcal{M}_{mim}$, the probability $P(\mathcal{M}_{mim}(D) = \mathcal{M}_{mim}(D'))$ of finding two different data D and D' that produce the same hash value d is extremely low.*

Proof. According to Definition 2, data privacy $\mathcal{S}(D, U) = true$ holds if and only if an unauthorized user U cannot access data D. The mimic hash mechanism $\mathcal{M}_{mim}$ provides an additional layer of security for data transmission by dynamically changing the hash algorithm used. This unpredictability increases the difficulty for unauthorized accessors to obtain valid data, thus enhancing data privacy $\mathcal{S}$. Even if attackers intercept the data, the dynamic selection of hash algorithms significantly reduces their ability to decode the original data, thereby protecting data privacy. □

As discussed in Section 5, the design of the mimic hash includes randomness and unpredictability, making it difficult for attackers to predict the selection of the next hash algorithm. Given the randomness of each hash algorithm selection, finding two different inputs D and D' that produce the same hash value requires coincidentally choosing a hash function capable of producing a collision in every round. This probability is extremely low,

especially considering the collision resistance of each hash function itself, reducing the probability to $1/3^n$ (where n is the number of times data are transmitted). The dynamism and heterogeneity of the mimic hash mechanism further reduces the likelihood of collisions, because even if attackers find a collision for one hash function, the constantly changing system quickly renders that hash function obsolete.

Theorem 3 (Effectiveness of Dynamic Defense). *In a mimic hash system based on mimic defense, the probability $P_{def}(a)$ that the system can dynamically adjust executors to resist any given attack a is higher than in static systems.*

Proof. According to Definition 3, system reliability $\mathcal{R}(S) > \theta$, where θ is the acceptable minimum reliability threshold. In the mimic hash system based on mimic defense, the system dynamically adjusts executors in response to detected threats. This capability allows the system to adapt quickly in the face of attacks, by swapping executors to evade known attack vectors, thereby maintaining the system state S at a level of reliability higher than θ. The dynamic defense mechanism ensures that the system maintains the predetermined reliability standards under various attacks, aligning with the definition of system reliability $\mathcal{R}$. □

As discussed in Section 5, the system dynamically adjusts executors based on monitored threats. Each time an attack occurs, the system evaluates the current executors and makes adjustments as necessary. Due to the diversity of executors, any specific attack strategy is only effective under a specific configuration of executors. By continually changing the configuration of executors, the system reduces the time window of being exposed to a specific attack.

For a specific attack a, its success probability depends on the system's configuration against that attack vector. The diversity and dynamic adjustment of system executors decrease the probability of attack a succeeding compared to static systems.

7. Experimental Analysis

7.1. Experimental Environment

The experimental environment for this study comprises a simulated Wireless Sensor Network (WSN) and a private Ethereum blockchain platform. The WSN was established using the Cooja Simulator within the Contiki OS environment, including 50 distributed sensor nodes and 2 aggregation nodes for data collection and relay, simulating the data collection and transmission processes of Internet of Things (IoT) devices in the real world. The private Ethereum blockchain platform was rapidly deployed using Ganache, aimed at simulating the on-chain process of sensor data, providing immutability and traceability for the data.

Experimental Environment: The Alibaba Cloud server was used to run the private Ethereum blockchain. Hardware configuration: Ubuntu 20.04 LTS operating system, Intel Xeon Processor (8 cores, 2.20 GHz), 16 GB RAM memory, and 500 GB SSD storage; Software environment: Ganache 2.5.4, Geth 1.9.25, Node.js 12.22.12, Solidity 0.5.16. A laptop was used for running WSN simulation and experimental control. Detailed configuration: Windows 10 operating system, Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz 1.80 GHz, 8 GB memory; Software environment: Contiki OS 3.0, Python 3.8; Libraries: hashlib, pysha3 1.0.2. Table 1 shows the Network Attack Simulation Experiment Environment.

Experimental Setup: In the experiments, two mimic hash schemes based on Verifiable Random Function (VRF) and Cyber Mimic Defense (CMD), along with a traditional MD5 hash scheme, were implemented to assess their security and performance. For constructing the two mimic hash schemes and to ease the simulation process, both methods utilized the same three hashes: MD5, SHA-1, and SHA-256 in rotation. The WSN environment was subjected to simulated network attacks, such as collision attacks, rainbow table attacks, and short password attacks, to evaluate the security of the various hash schemes [25,26].

Table 1. Network Attack Simulation Experiment Environment.

Scheme	MD5	SHA-1	SHA-256
Hash collision attacks	md5collgen	SHA1collision	-
Rainbow table attacks	RainbowCrack		
Short password attacks	https://github.com/FrankGrimmer/dictionary-password-cracker, accessed on 8 February 2024.		

Dataset Construction: Thermocouples are temperature sensors widely used in various industrial, scientific, and engineering applications due to their broad temperature range, robustness, and direct conversion of thermoelectric difference into voltage. Thermocouples are available in different types, such as K, J, T, etc., each made from different metal combinations suitable for various temperature ranges and environments. Among them, K-type thermocouples can measure temperatures approximately ranging from −200 °C to 1350 °C. The accuracy of thermocouples is usually defined within a specific temperature range and can be expressed as a percentage of the reading or as a fixed value in Celsius or Fahrenheit. In this paper, the simulation experiment data is based on K-type thermocouple temperature sensors, with accuracy set to 0.1 °C to construct the experimental dataset, which contains a total of 15,550 temperature data samples.

Hash Collision Attack: Currently, the academic community does not have publicly available example data, tools, or source code for SHA-256 collision attacks. Therefore, this experiment only simulates the mimic hash collision attack process for MD5 and SHA-1. A set of data, A, from the K-type thermocouple sensor data sample set, is used. In the SEED experimental environment, md5collgen is utilized to generate control data groups B and C with the same MD5 hash values, and SHA1 collision is used to generate control groups D and E with the same SHA-1 hash values. Then, the data groups A, B, C, D, and E are shuffled to obtain the hash collision test dataset U. Different collision datasets are obtained by changing the initial conditions or inputs of the md5collgen and SHA1 collision generation algorithms. During the simulation of hash collision attack experiments, two pieces of data are randomly selected from the hash collision test dataset U for hashing and matching the hash results. If the match is successful, the collision is considered successful; otherwise, the result is a collision failure. The distribution of states in the mimic hash collision attack test is shown in Table 2.

Table 2. Distribution of States in Mimic Hash Collision Attack Testing.

Hash Algorithm	Dataset	State Distribution
MD5	B or C	Successful
	Except B & C	Failed
SHA-1	D or E	Successful
	Except D & E	Failed

Rainbow Table Attack: RainbowCrack is used to construct rainbow tables for MD5, SHA-1, and SHA-256 based on temperature sensor data. By specifying the three hash algorithms and the set of all possible sensor readings as the input space characteristics, RainbowCrack generates the corresponding rainbow tables for MD5, SHA-1, and SHA-256. During a rainbow table attack, one can attempt to crack a given hash value by looking up the corresponding plaintext (temperature readings) in the table.

Short Password Attack: The short password attack exploits the tendency of users to utilize simple, common words or phrases as passwords. Attackers prepare a list containing a large number of common passwords (known as a "dictionary") and attempt these passwords to find matching hash values, also known as a dictionary attack. Given the fixed and singular target data range of sensors, they are highly susceptible to short password attacks.

7.2. Experimental Analysis

Figure 9 shows the number of successful collisions in traditional hash and mimic hash during 100 simulated collision attacks. The traditional hash collision test had a success rate of 18%, while the success probability for the mimic hash collision test based on cryptographic lottery decreased to 12%, and the success probability for the mimic hash collision test based on mimic defense further decreased to 7%. Based on the results of this experiment, it can be analyzed that mimic hash can reduce the probability of hash collision attacks to a certain extent.

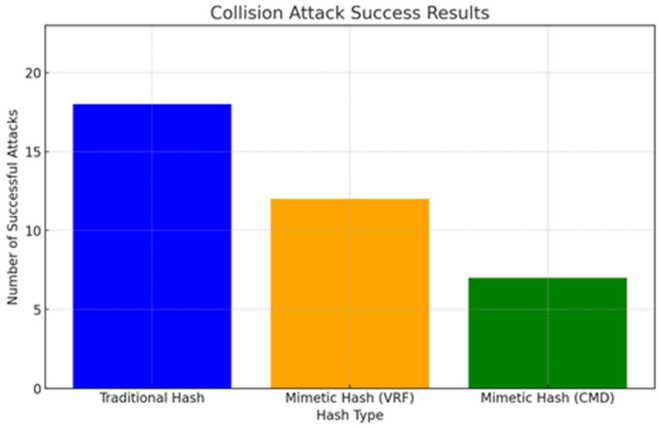

Figure 9. Comparison of Success Rates in Mimic Hash Collision Attacks.

In the rainbow table attack simulation experiment, for the same 15,500 data points from a thermocouple sensor, a single hash requires only one rainbow table to be constructed, whereas mimic hash necessitates the construction of corresponding rainbow tables for all hash algorithms in the mimic hash heterogeneous executor pool. As can be seen from Table 3, the generation time for rainbow tables in mimic hash increased by 77.6% compared to the traditional hash, and the storage size of rainbow tables in mimic hash increased by 71.9% compared to the traditional hash. This indicates that mimic hash significantly increases the time and space costs of constructing rainbow tables. However, the computational complexity of rainbow table attacks remains O(n) for both single hash and mimic hash. Reference [13] utilizes the PBKDF2 algorithm to perform iterative calculations for salted slow hashing to resist rainbow table attacks. Therefore, the next step in this work could draw on this algorithm to address the aforementioned shortcomings.

Table 3. Mimic Hash Rainbow Table Attack Simulation Analysis.

Comparison Item	Single Hash Mechanism	Mimic Hash
Data Volume (unit: items)	15,500	15,500
Number of Rainbow Tables (unit: count)	1	3
Rainbow Table Generation Time (unit: seconds)	3503	15,609
Rainbow Table Storage Size (unit: KB)	545	1937
Rainbow Table Attack Time Complexity	O(n)	O(3n)

Figure 10 shows the hashing time for short password attacks on 15,500 data items from thermocouple sensors encrypted with traditional hash and mimic hash. For the 15,500 data items encrypted with traditional hash, the total time for short password attacks was 1.1 h; for the data encrypted with the VRF-based mimic hash, the total time for short password attacks was 2.6 h, which is a 57.7% increase compared to the total time for short password attacks on traditional hash; for the data encrypted with CMD-based mimic hash, the total time for short password attacks was 24.1 h, marking a 95.4% increase compared to

traditional hash. These experimental results are specific to a dataset of 15,500 items, while the actual data sample space considered during hacker attacks is much larger than that of this experiment. Therefore, mimic hash can significantly increase the time cost for hackers to use brute force or short password attacks.

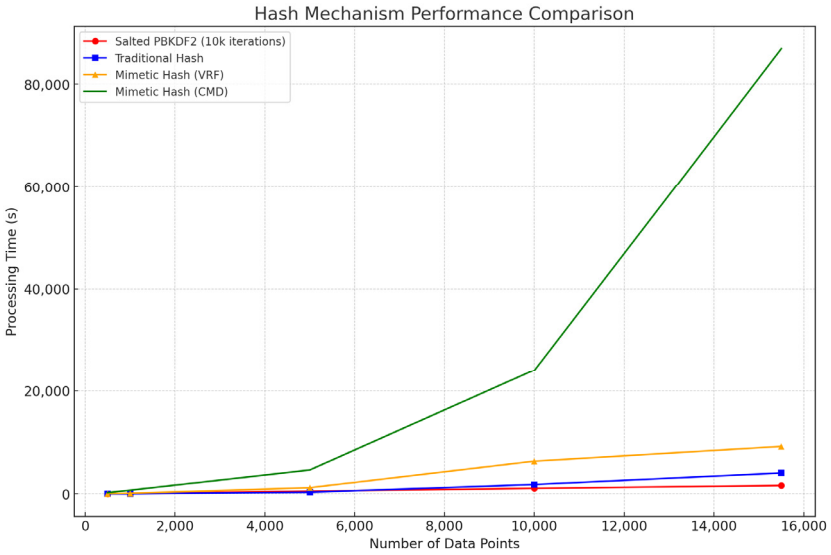

Figure 10. Comparison of Short Password Attack Time for Mimic Hash.

In summary, the mimic hash scheme exhibits significant advantages in enhancing network security, particularly against diverse network attacks. However, the improvement in security comes at the cost of reduced processing speed and increased resource consumption. Therefore, implementing the mimic hash scheme requires a comprehensive balance of actual scenario needs.

8. Conclusions

Building upon the blockchain trusted sensor scheme proposed by Si et al. [4,5], this paper addresses the potential security threats of collision attacks, rainbow table attacks, and short password attacks inherent to the single static hash mechanism. Inspired by the concept of mimic defense, we innovatively propose a mimic hash mechanism. This mechanism overcomes the inability of blockchain-trusted sensor data consistency verification schemes based on traditional single hash mechanisms to withstand uncertain network security threats, achieving trustworthy sharing of sensor data and enhancing the system's resistance to external malicious attacks and untrusted server attacks. Both theoretical analysis and experimental evidence have demonstrated the security and performance of our model, providing a novel and inherently secure solution for sensor data transmission and verification in edge computing environments.

For future work, we plan to further investigate the scalability issues of the mimic hash mechanism in real-world large-scale sensor network environments and explore the integration of machine learning techniques for anomaly detection and threat mitigation in blockchain sensor data sharing. Additionally, we will consider the impact of emerging technologies like quantum computing on the security of the proposed model.

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

Funding: This research was funded by Key Research Project for Higher Education of Henan Province, grant number 24A520059; Pre-research Project of SongShan Laboratory, grant number YYJC032022021; Open Fund of Henan Key Laboratory of Network Cryptography Technology, grant number LNCT2022- A12; Postgraduate Research and Innovation Project of Zhongyuan University of Technology, grant number YKY2023ZK52.

Data Availability Statement: The source code in this study is available on request from the corresponding author.

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

References

1. Pardeshi, M.S.; Sheu, R.-K.; Yuan, S.-M. Hash-Chain Fog/Edge: A Mode-Based Hash-Chain for Secured Mutual Authentication Protocol Using Zero-Knowledge Proofs in Fog/Edge. *Sensors* **2022**, *22*, 607. [CrossRef] [PubMed]
2. Heiss, J.; Busse, A.; Tai, S. Trustworthy pre-processing of sensor data in data on-chaining workflows for blockchain-based IoT applications. In Proceedings of the Service-Oriented Computing: 19th International Conference, ICSOC 2021, Virtual Event, 22–25 November 2021; pp. 133–149.
3. Zhang, R.; Xue, R.; Liu, L. Security and privacy on blockchain. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–34. [CrossRef]
4. Gao, H.; Gao, T.G. A lightweight robust image hash based on random tensors and angle features for IoT devices. *Signal Image Video Process.* **2024**, *18*, 1747–1761. [CrossRef]
5. Salman, D.S.; Naif, J.R. Smart cloud security using hybrid encryption algorithms with 4-d chaotic key. *J. Res. Adm.* **2023**, *5*, 1816–1837.
6. Joy, J.; Devaraju, S. Avoidance of duplicacy and compelling cloud security indifferent cloud situations. *Int. J. Creat. Res. Thoughts* **2023**, *11*, 543–555.
7. Chen, L.; Xiang, F.; Sun, Z. A survey of blockchain security technologies based on attribute-based cryptography. *Acta Electron. Sin.* **2021**, *49*, 192–200.
8. Si, X.M. A Security Verification Method, Terminal, and System Based on Keyed Hash Chains. Patent CN115883101A, 31 March 2023.
9. Si, X.M.; Guo, S.K. A Blockchain Data Transmission Method and System Based on Keyless Signatures. Patent CN115694811A, 3 February 2023.
10. Zhang, D.F.; Guan, L.; Dai, X.M. Design and implementation of the rainbow table lookup system based on password cracking. *Cyberspace Secur.* **2020**, *11*, 1.
11. Yu, H.B.; He, L.; Cheng, Z.J. Improved Cryptanalysis on Checkpoints in Perfect Rainbow Table. *J. Cryptologic Res.* **2021**, *8*, 76–86. [CrossRef]
12. Wang, X.; He, Z.H.; Xu, Y.; Pang, S.S.; Wang, X.C.; Zhou, C.; Du, P. Hash Attacks Prevention for Instruction Security in Embedded Monitoring System. In Proceedings of the 2016 Joint International Conference on Service Science, Management and Engineering (SSME 2016) and International Conference on Information Science and Technology (IST 2016), Wuhan, China, 20–21 August 2016; ISBN 978-1-60595-379-3.
13. Qi, X.; Wei, M.R.; Jiang, W.B. Security Analysis and Comparison of Password Encryption Algorithms. *Cyberspace Secur.* **2016**, *7*, 34–38.
14. Zhang, W.C.; Li, H.; Cheng, G. Research and Implementation of Secure Password Storage Based on the One-way Salted Hashing Algorithm. *China Digit. Med.* **2018**, *5*, 8–11. [CrossRef]
15. Zhu, Y.B.; Wang, C.L. A Message–digest Model with Salt and Hidden Feature of Hash. *Comput. Technol. Dev.* **2013**, *23*, 134–138.
16. Si, X.M. A Blockchain Data Storage Method and terminal for sensor Networks. Patent CN115842834A, 24 March 2023.
17. Wu, J.X. Research on Cyber Mimic Defense. *J. Cyber Secur.* **2016**, *1*, 1–10. [CrossRef]
18. Tong, Q.; Zhang, Z.; Zhang, W.H.; Wu, J.X. Design and Implementation of Mimic Defense Web Server. *J. Softw.* **2017**, *28*, 883–897. [CrossRef]
19. Ma, H.L.; Yi, P.; Jiang, Y.M.; He, L. Dynamic Heterogeneous Redundancy based Router Architecture with Mimic Defenses. *J. Cyber Secur.* **2017**, *2*, 29–42. [CrossRef]
20. Wang, Z.P.; Hu, H.C.; Cheng, G.Z. A DNS Architecture Based on Mimic Security Defense. *Acta Electron. Sin.* **2017**, *45*, 2705–2714. [CrossRef]
21. Xu, M.X.; Yuan, C.; Wang, Y.J.; Fu, J.H.; Li, B. Mimic Blockchain—Solution to the Security of Blockchain. *J. Softw.* **2019**, *30*, 1681–1691. [CrossRef]
22. Gui, Z.; Sun, Z.; Huang, Z. Spatial Query Optimization Methods by Integrating Blockchain and Database. *Acta Sci. Nat. Univ. Pekin.* **2023**, *59*, 261–270. [CrossRef]
23. Shao, Q.F.; Jin, Q.Z.; Zhang, Z.; Qian, W.N.; Zhou, A.Y. Blockchain: Architecture and Research Progress. *Chinese Journal Computers.* **2018**, *41*, 969–988.
24. Si, X.M.; Wang, W.; Zeng, J.J.; Yang, B.C.; Li, G.S.; Yuan,, C.; Zhang, F. A Review of the Basic Theory of Mimic Defense. *Strateg. Study CAE* **2016**, *18*, 62–68. [CrossRef]

25. Li, Z.M. Design and Analysis of the Hash Functions. Ph.D. Thesis, Beijing University of Posts, Beijing, China, 2009.
26. Qiu, Y.F. Implementation and Verification of Hash Algorithm. Master's Thesis, Guangdong University of Technology, Guangzhou, China, 2021.