symmetry

MDPI

Article

Data-Tracking in Blockchain Utilizing Hash Chain: A Study of Structured and Adaptive Process

Sungbeen Kim and Dohoon Kim *

Department of Computer Science, Kyonggi University, Suwon-si 16227, Republic of Korea; beensk@kyonggi.ac.kr
* Correspondence: karmy01@kyonggi.ac.kr

Abstract: This study presents a series of structured and adaptive processes aimed at tracking and verifying transactions recorded on the blockchain. Permissioned blockchains are employed across diverse enterprises for various purposes, including data recording, management, the utilization of blockchain services, and authentication. However, the processes of data tracking and transactions incur substantial resource and time expenditure. Furthermore, there is potential for information asymmetry within the blockchain ledger due to data breach attacks. Consequently, we propose a contract structured as a hash chain to mitigate resource and time consumption in the tracking and verification processes by organizing transaction hash values and content in a hash chain format based on cryptography. We generate a hash chain for the recorded transactions along the process line and expedite the tracking and verification process by navigating the relevant hash chain. This approach achieves faster and more accurate tracking procedures compared to conventional transaction tracking processes, simultaneously maintaining data symmetry within the blockchain ledger. We conduct a comparative analysis of a contract-based hash-chain-employing structure and two contracts related to tracking in terms of tracking time, CPU usage, and network traffic, among other metrics. The findings suggest that structuring transaction data in the form of a hash chain significantly enhances the efficiency and integrity of the data-tracking and verification processes. Consequently, in this study, we advocate for the adoption of contracts based on the hash chain format when leveraging the blockchain for tracking and verification purposes across various institutions.

Keywords: blockchain; contract; efficiency; hash chain; process line; data tracking; transaction; verification

Citation: Kim, S.; Kim, D. Data-Tracking in Blockchain Utilizing Hash Chain: A Study of Structured and Adaptive Process. *Symmetry* **2024**, *16*, 62. https://doi.org/10.3390/sym16010062

Academic Editor: Chin-Ling Chen

Received: 18 November 2023
Revised: 23 December 2023
Accepted: 25 December 2023
Published: 3 January 2024

1. Introduction

The blockchain functions as an immutable ledger within networks dedicated to handling business transactions by recording the data associated with transactions and uniformly sharing them across different nodes [1–3]. As the network scales (with an increase in the number of participating nodes), the counts of recorded transactions and block sizes increase. Consequently, a comprehensive examination of the ledger to track and verify data and transactions recorded on the blockchain can introduce time delays and consume substantial computing resources [4,5]. Moreover, the configuration of a blockchain network is influenced by the characteristics of the participating nodes [6]. For example, permissioned blockchains are predominantly adopted when the objective is to record sensitive data for secure storage [7,8]. In a permissioned blockchain, only authorized nodes can participate in the blockchain network, and the data that can be accessed are carefully managed to ensure confidentiality and controlled accessibility based on this structural environment. In this study, we propose the introduction of a hash chain [9,10] as an advantageous means of facilitating the tracking and verification of data recorded on the blockchain.

Given the challenges related to maintaining confidentiality, speed, and efficient resource utilization while tracking transactions recorded in existing blockchain networks and ensuring integrity during the process, this study proposes a blockchain-based data-tracking

management process utilizing a hash chain as a scalable solution [11,12]. In a blockchain, transactions are organized into blocks, and these generated blocks form a chain based on the hash value present in the block header [13]. Consequently, as the blocks are continuously linked, the integrity of the previous blocks can be robustly ensured [14]. This study maintains this structure, forming a blockchain and leveraging the hash chain structure within the transactions and contracts that occur internally [15,16].

This study primarily designs and validates a process based on the Hyperledger Fabric, a permissioned blockchain commonly used in business environments [17]. Additionally, we enhance confidentiality by encrypting the data recorded on the blockchain using a hash function. In this process, a structurally connected set of transactions forming a hash chain is assigned a process line number, and chained data and transaction tracking are conducted based on this line. We streamline the existing blockchain transaction-tracking process and implement the structure of hash chains through contracts to save tracking time and computing resources. Thus, we aim to provide scalability for tracking transactions and verifying the integrity of existing blockchains. The strengths and contributions of this study are enumerated as follows:

- The enhanced security of data using the hash function;
- The assessment of quantitative and qualitative efficiency through performance evaluation;
- The reduction in data-tracking time and resource consumption through the utilization of a hash chain structure;
- Ensuring data symmetry in the blockchain ledger through a hash chain structure.

In Section 2, we discuss the existing tracking processes and hash graphs [18]. In Section 3, we briefly examine the blockchain platform and encryption algorithms used in this study. In Section 4, we describe the actual tracking process for hash-chain-based data recorded on the blockchain. Section 5 describes the experiments and analyzes the results. Finally, in Sections 6 and 7, we conclude the paper with discussions and conclusions regarding our research.

2. Related Work

In Section 2, we investigate prior studies that explored tracking processes utilizing existing blockchain platforms. We specifically focus on delineating the differences and proposing solutions for the tracking process utilized in this research. Additionally, we conduct a comparative analysis between the proposed tracking process based on hash chains and the existing research on hash graphs. This section highlights the distinctions and investigates the relevant aspects of the hash-chain-based tracking process proposed in this study compared with hash-graph-based approaches and existing tracking applications.

2.1. Tracking Process

Montaser proposed the potential of a tracking process based on the blockchain [19]. In this study, a series of flows involving data processing, tracking, and security are elucidated by leveraging the characteristics of transactions and blocks recorded on the blockchain, namely, integrity. The proposed blockchain-based data-tracking process utilizing a hash chain in this study references the potential proposed by Montaser and seeks to examine its applicability in specific domains (cargo distribution tracking, personal information verification). In addition, Liu proposed a contract structure that generates information about drugs produced by each manufacturer, recording data on the process of distribution to consumers on the blockchain, thereby monitoring and tracking the total production process of the drug [20]. However, their study provided ambiguous details regarding the tracking process and lacked a clear specification of the lifecycle of transactions. Koyama proposed a tracking system for COVID-19 vaccines using blockchain technology [21]. Mendi proposed a food (grocery)-tracking process using the blockchain to track the extent of price increases caused by food crises [22]. Attia proposed a healthcare-monitoring and tracking application using blockchain technology [23]. Marbouh reviewed the potential

application of the blockchain in various fields, such as medical supply, close contacts, donations, and treatments, in the context of COVID-19 [24]. Leng conducted a comparative analysis of the process of uploading hospital medical records and medical-related data to the blockchain ledger for monitoring and management [25].

All the above-mentioned literature introduced tracking processes and applications based on the blockchain. However, the authors proposed processes that leveraged the inherent characteristics of the blockchain; hence, a thorough examination of transactions may be necessary during the tracking process. Furthermore, while discussing the potential applications of blockchain technology in various domains, some studies lacked adequate content regarding data encryption. Table 1 compares the proposed data-tracking process based on hash chains with traditional blockchain-based tracking applications. Existing applications did not typically mention data security and relied on transactions for tracking purposes. Thus, in this study, we propose the application of hash chains to facilitate transaction tracking, enhance efficiency, and improve security and trust by utilizing hash functions [26] for encrypted data transmission.

Table 1. Comparison of related works and proposed process.

Ref.	Application Domain	Blockchain Platform	Structure	Consensus Algorithm	Data Security	Performance Evaluation	Tracking Usability
Liu [20]	Drug Tracking	Hyperledger Fabric	On–Off Chain	pBFT	Not Considered	Considered	Depending on Transaction
Koyama [21]	Vaccine Tracking	Substrate	On–Off Chain	Proof of Stake	Not Considered	Not Considered	Depending on Transaction
Mendi [22]	Healthcare Monitoring	Hyperledger Fabric	On Chain	Not Specified	Not Considered	Not Considered	Depending on Transaction
Attia [23]	Food Tracking	Hyperledger Fabric	On–Off Chain	Not Specified	Not Considered	Considered	Depending on Transaction
Marbouh [24]	COVID-19 Application	Ethereum	On–Off Chain	Proof of Stake	Considered	Considered	Depending on Transaction
Leng [25]	Hospital Monitoring	Hyperledger Fabric	On–Off Chain	Not Specified	Considered	Not Considered	Depending on Transaction
Proposed Process	**Private Data Tracking**	**Hyperledger Fabric**	**On–Off Chain**	**Raft**	**Considered**	**Considered**	**Depending on Hash Chain**

2.2. Hash Graph

The hash graph, a novel approach for achieving a consensus algorithm [27] in a distributed network, utilizes a directed acyclic graph (DAG) to record transactions and establish consensus among participating nodes in the network [28,29]. A crucial element in constructing a hash graph is the gossip protocol that allows nodes to exchange information regarding transactions and events [30–32]. The nodes participating in the network determine the order of the transactions through a voting process, with each participant allocating weights to the other nodes based on the received information. Because of these characteristics, blockchains and hash graphs differ in how they connect blocks and transactions. In Figure 1, the structure of a traditional blockchain connects blocks based on sequentially agreed-upon blocks, whereas a hash graph chooses to connect verified data nodes in the graph regardless of the order of creation. Nodes A, B, C, and D, participating in the hash graph, arbitrarily validate and connect data to the graph. Consequently, the

connected data nodes are linked in the order verified by the participating nodes, and all validated data are recorded in the form of a DAG.

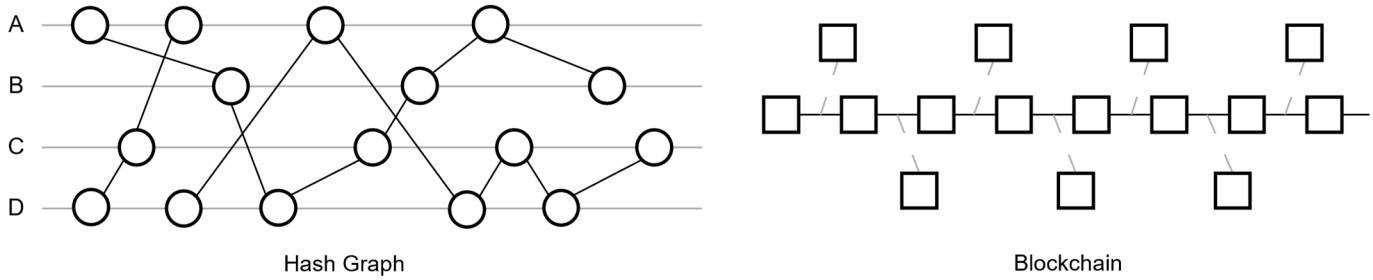

Figure 1. Comparison of hash graph and traditional blockchain structures.

In this study, we model contracts through a branching pattern similar to a hash graph, but based on process lines rather than validating nodes. Consequently, the lifecycle of transactions can be more clearly verified, and multiple other transactions related to a single transaction can be easily tracked and verified by connecting them, like branches on a tree. This connection pattern of transactions is applied to the hash chain intended for use in this study, where contracts are executed, and the process involves linking and tracking. A detailed description of this process is provided in Section 4.

3. Background

In Section 3, we analyze the blockchain platform and encryption algorithms used in this study. Our experiment focuses on a specific target: permissioned blockchains. We analyze the characteristics of permissioned blockchains by comparing them with public blockchains in terms of their advantages, disadvantages, and distinctions. Furthermore, in the subsequent section, we elucidate the encryption algorithms necessary for constructing the data-tracking process based on hash chains, with the aim of facilitating comprehension.

3.1. Blockchain Platform

In the current landscape of blockchain technology, two main categories are widely recognized: public blockchains [33] and permissioned blockchains [7,8]. A public blockchain is characterized by its openness and distributed network, allowing anyone to access the network. Consequently, it maintains transparency, where all participating nodes can read and verify the recorded transactions. In contrast, a permissioned blockchain follows a controlled-access approach, making it more suitable for industrial applications with features such as privacy preservation. In other words, only the authorized nodes can generate transactions and validate their legitimacy. The industrial sector is witnessing an increasing demand for permissioned blockchains owing to these network characteristics.

Permissioned blockchains provide some advantages in the industrial domain. First, their permissioned nature reduces malicious activities by allowing only validated participants into the network, thereby enhancing security. Second, a permissioned blockchain can encrypt sensitive data or restrict access to specific participating nodes. In addition, compared to public blockchains, permissioned blockchains offer faster transaction-processing speeds and higher scalability, making them especially applicable to industrial applications where efficiency, performance, and security are paramount.

In this study, we chose Hyperledger Fabric, a permissioned-blockchain platform, to ensure efficiency in data recording, tracking, and verification when utilizing blockchain in the industrial sector. Fabric enables members to access the network and provides secure communication in an environment that ensures confidentiality through the introduction of private channels. Moreover, it ensures high throughput, low latency, and fault tolerance using consensus algorithms such as Raft [34,35].

3.2. Cryptographic Algorithms

3.2.1. Hash Function

In addition to blockchain technology, hash functions are widely used as fundamental encryption tools in various research, development, and application programs [36]. Hash functions output a fixed-size value regardless of the input length—a characteristic that this study aims to leverage. During the data-tracking process, exposing the actual values of the data poses a threat to security. Hence, this study applies the preimage resistance, collision resistance, and avalanche effect of hash functions. Furthermore, hash functions are advantageous for the rapid processing and validity verification of data in resource-constrained environments, preventing tampering with the original data. Therefore, in this study, the characteristics of hash functions are utilized to encrypt data and record them in the form of a hash chain.

3.2.2. Hash Chain

In this study, we structure the original transaction data as the input to a hash function and the unique hash value of the transaction itself in the form of a chain. A hash chain comprises a series of hash values that are sequentially arranged, with each hash value derived from the previous value [9,10]. In addition, a hash chain has the advantage of verifying the integrity of data items and easily detecting unauthorized modifications and tampering. In the context of a blockchain, the hash chain plays a pivotal role in constructing the block structure. Although a blockchain is constructed by applying the principles of a hash chain, the continuous linking of hash values (blocks) poses a challenge when retracking transactions, which consumes considerable time and resources. To address this issue, we construct an additional hash chain.

We configure a separate hash chain as a contract, branching out like a tree from the central blockchain. This additional hash chain facilitates transaction and data tracking and record management. It addresses the challenge of significant time and resource consumption when tracking transactions within the blockchain, owing to the continuous linking of hash values (blocks).

4. Data-Tracking Management Process

In blockchain-based applications, particularly within industries, the primary focus is often the development of systems for tracking data history, maintaining integrity, and enhancing security. In this study, we design a process to improve performance and efficiency by emphasizing the tracking of data history. The industries in which the tracking of data history is crucial typically include manufacturing, distribution, and trade [37]. The methods used to track data history generally involve backtracking transactions based on the order that they were created in the existing blockchain or recalling and searching for specific transaction data. The latter approach can be considered as a more efficient tracking process. However, with this method, transactions must be specified and searched for while tracking another linked process. In this process, data must be recorded somewhere, and data recorded in the system may compromise integrity and security [38]. Moreover, as a series of processes unfolds, resource and time consumption inevitably increase.

For example, in Figure 2, in a blockchain environment where data are continuously accumulated in blocks, the number of transactions to be traversed increases when searching for and tracking a specific transaction. In the traditional blockchain transaction-tracking process, to track the data for TX 4, the information of the specific transaction may be separately stored, followed by the search or the data, which may be tracked in reverse, starting from TX 12. However, by employing the hash-chain-based tracking process proposed in this study, a chain of related transactions may be constructed along a series of process lines. After recalling the specific initial value of the constructed chain, data tracking can be performed. This enables the tracking of only the target transaction data (TX 4, TX 5, TX 7, and TX 10) by searching Line 1, which is identified by the process line number, as opposed

to tracking all the transaction data from TX 12 to TX 1, as in the traditional blockchain transaction-tracking process.

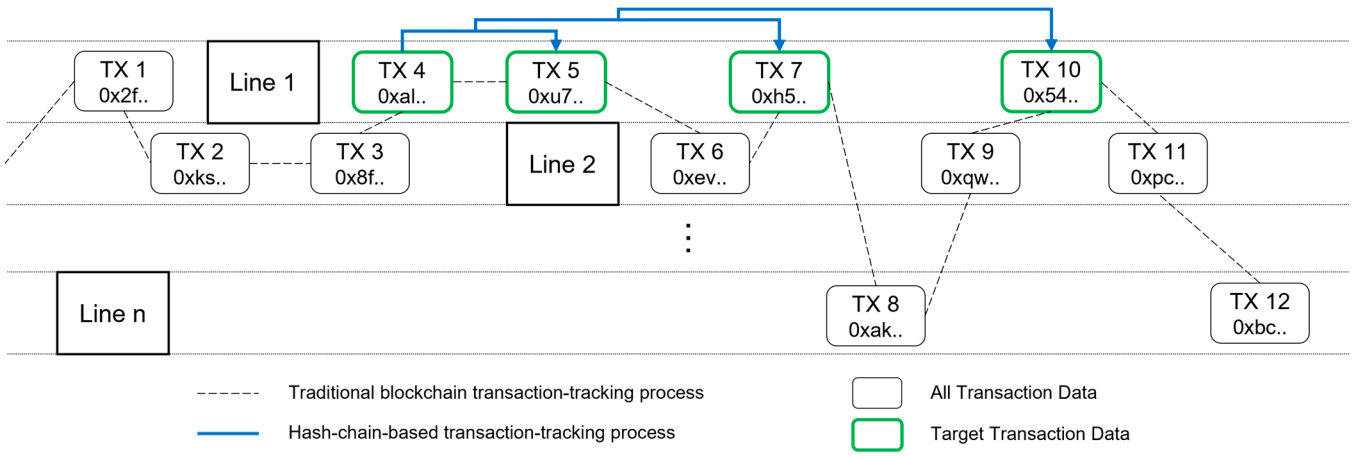

Figure 2. Traditional and proposed transaction-data-tracking processes on blockchain.

Table 2 presents a comparison between the traditional transaction-data-tracking process and the hash-chain-based transaction-data-tracking process shown in Figure 2. In the case of the hash-chain-based process, it ensures data confidentiality while reducing the time required for verification and the tracking of interconnected data through the hash chain structure. Conversely, in the traditional transaction-data-tracking process, there is a high likelihood of data confidentiality not being guaranteed. Additionally, due to the sequential recording of transactions within conventional blocks and their interconnected structure, tracing specific transaction data entails tracking a substantial number of transactions, resulting in significant resource consumption and time required for verification and tracking. Accordingly, as the blockchain lengthens, the traditional process necessitates tracking unrelated transactions. However, the hash-chain-based process proposed in this study structures related transaction data as a single process line. Consequently, only the desired data set is separately verified and tracked, streamlining the tracking process. To address the drawback of tracking all recorded transactions, this study proposes a data-tracking process based on a hash chain.

Table 2. Comparison of traditional and hash-chain-based transaction-data-tracking processes.

Category	**Traditional Transaction-Data-Tracking Process**	**Hash-Chain-Based Transaction-Data-Tracking Process**
Property	• The possibility of exposing and tracking even confidential data. • The process of tracking data recorded in transactions involves a significant number of transactions.	• Verification and tracking of data integrity through a hash-chain-based structure and hash function. • A reduced number of transactions to be traversed during the transaction-tracking process.
Data Confidentiality	Not ensured	Ensured
Linking Structure	Depending on transaction in block	Depending on hash chain structure
Linked Transactions	**TX 4** → **TX 5** → TX 6 → **TX 7**→ TX 8 → TX 9 → **TX 10**	**TX 4** → **TX 5** → **TX 7** → **TX 10**

4.1. Structure of Data-Tracking Management Process

The flow of the data-tracking process based on the hash chain is illustrated in Figure 3. In this study, data are recorded and tracked on a blockchain using three main algorithms. Algorithm 1 utilizes public key infrastructure (PKI) [39] not only to protect data transfers among blockchain nodes but also to establish a foundation for trust and integrity in the network. This is crucial in business areas where data confidentiality must be ensured, and only authorized participants should access confidential data. Algorithm 2 conducts a key role in classifying and processing data. It categorizes data into public and private segments, hashing private data for blockchain recording. This approach balances privacy and openness in blockchain applications, ensuring the integrity of data through the structure of the hash chain during transaction generation. Algorithm 3, focusing on tracking and verification, emphasizes the practicality of blockchain technology in applications like supply chain management and customs clearance, where data verification and tracking are vital. These algorithms demonstrate a comprehensive approach to data security, classification, and tracking in blockchain systems, addressing challenges like data privacy, integrity, and transparency. Section 5 details the experiments demonstrating the implementation and efficacy of these algorithms.

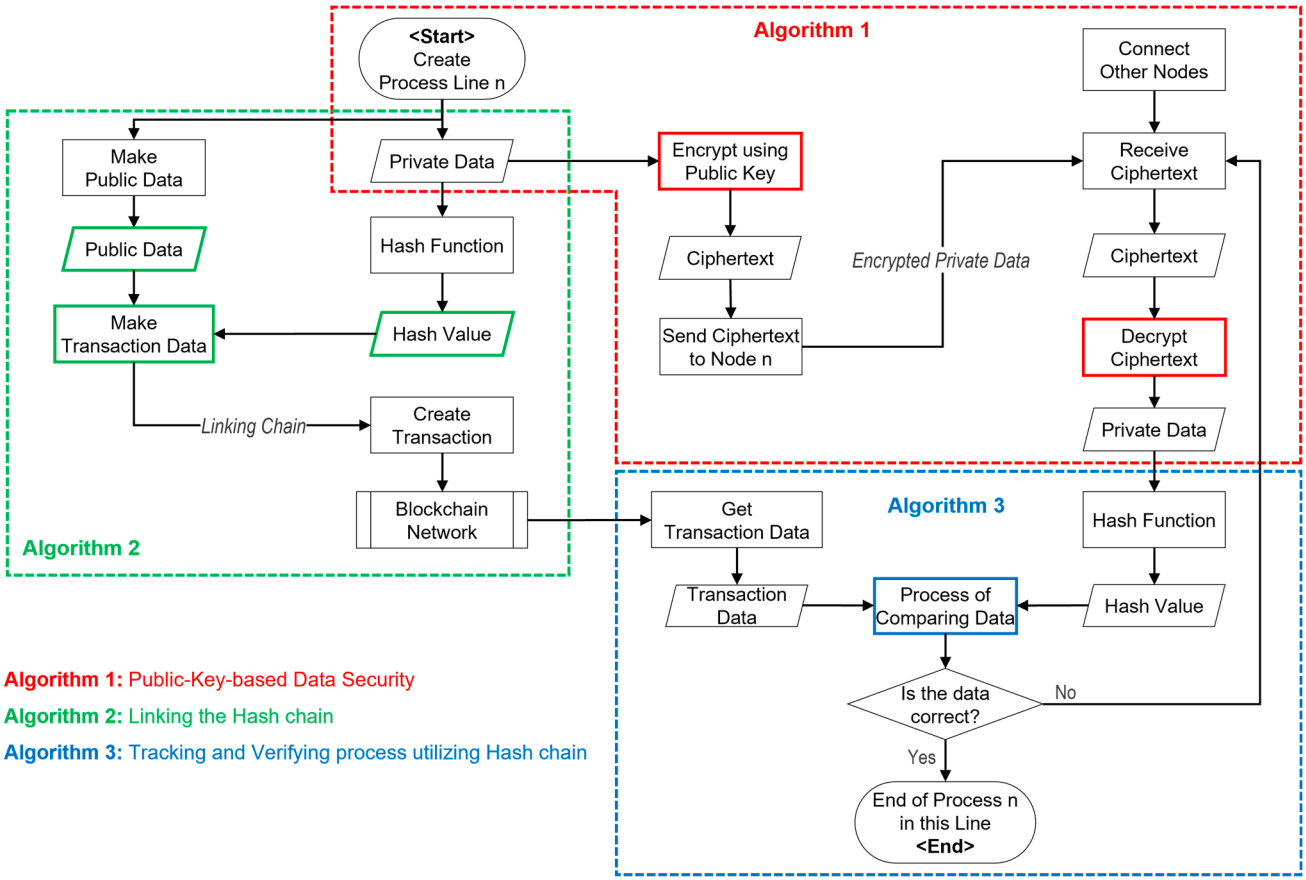

Figure 3. Flowchart of proposed process for tracking data utilizing hash chain and PKI on blockchain.

4.1.1. Algorithm 1: Public-Key-Based Data Security

A core objective of the data-tracking process based on a hash chain is data confidentiality and security. Nodes participating in the blockchain have the authority to view and verify the recorded transactions. However, in industry, there exists an environment where relationships and confidentiality between institutions must be ensured, and data that should not be exposed externally are recorded. Therefore, in this study, data transmission to other nodes is performed after the encryption process based on the PKI. The reason

for transmitting data to other nodes is that confirming whether the data recorded in the permissioned blockchain are appropriate is difficult because all the data recorded in the permissioned blockchain need to go through a permissioning process for nodes to participate in the network. The industry is vulnerable to insider attacks or social-engineering attack techniques, even for authorized nodes, due to this aspect [40]. Even for authorized nodes, internal threats may exist, and proper transaction-verification procedures may not have occurred.

The data encrypted through the PKI in this manner are utilized in the tracking process of the hash chain, which is intended to verify the integrity of the data recorded in the transactions in Algorithm 3. Additionally, the original data, which are private before encryption, are used as the original data to record the hash-chain data in transactions in Algorithm 2.

4.1.2. Algorithm 2: Linking the Hash Chain

The data-tracking process is performed utilizing a hash chain to establish an environment that ensures data integrity. The blockchain fundamentally operates on a structure in which transactions are generated as individual blocks and linked together in a chain. Through this blockchain structure, the integrity of the data can be ensured. However, as previously mentioned, some nodes participating in the blockchain may pose internal threats that can compromise data integrity.

Therefore, in this study, we use a contract to create transactions to be recorded in the blockchain and structure the hash chain format, which is then recorded in the blockchain. Unlike the traditional method of recording data in a blockchain, we segregate the data into public and private categories. Public data are, as the name suggests, openly accessible and do not cause harm to industries or processes when verified by anyone. However, private data refers to data that, if exposed, may potentially harm industries and processes. This type of data is encrypted and recorded in the blockchain. Consequently, public data are recorded as they are, and private data are encrypted using a hash function to generate transaction data, which are then recorded in the blockchain. The data recorded and linked in this manner are tracked using Algorithm 3 and undergo a verification process, during which the structure of the hash chain is utilized.

4.1.3. Algorithm 3: Tracking

To track the data recorded in a hash chain, understanding the process involved and identifying the specific hash chain intended for tracking are essential. In this study, we use public data to explore a particular hash chain. Figure 3 shows two types of data, and the data necessary for tracking are public data. During the initial creation of the hash chain, a specific hash value must be obtained. Therefore, we designate the initial value as the hash value of the public data. Consequently, all the nodes aspiring to track only need to understand the publicly accessible portion of the data. In other words, private data can be entirely preserved, with public data playing a crucial role in tracking.

By hashing private and public data, a specific hash chain can be secured based on the resulting values. Subsequently, we track the data of this hash chain and perform a verification process to ensure the integrity of the data. Public data are information that can be made public without any issues. In this study, the hash values recorded in the hash chain are generated using salt values [41]. Using these generated hash values as references, we proceed with the tracking and verification processes using Algorithm 3.

4.1.4. Algorithm 3: Verification

After recording the data in the form of a hash chain on the blockchain, hash functions are employed during the verification process. The same hash function used to encrypt private data in the encryption process of Algorithm 2 is used. Public and private data are available when examining transactions recorded on the blockchain. We use public data for identification to determine which hash chain of the desired data to track. In essence,

private data are the data that need to be verified during the actual tracking process. The node receiving the data decrypts the encrypted data using its private key through the PKI and hashes it again using a hash function. The resulting hashed value is compared with the private data inside the hash chain recorded in the transaction to verify whether the correct data were transmitted in that process.

In this process, the structure of the hash chain ensures the integrity and reliability of the data, thereby verifying the ongoing process through the nodes participating in the blockchain and establishing trust in the data. This process flow allows for the tracking of data recorded on the blockchain, ensuring the integrity and reliability of the data. The detailed algorithm for this process is described below.

4.2. Algorithms for Data-Tracking Management Process

Figure 3 provides a structural explanation of the flow of each algorithm. Each algorithm manages and transmits data using PKI, a hash chain, and hash functions. In this section, we provide detailed explanations of the operational processes of each algorithm. For all algorithms, we have the initially generated private data, the public and private keys of each node participating in the blockchain network, and the process line number for recording and tracking the data. Each algorithm is executed through a contract within Hyperledger Fabric, during which it generates the structure of the hash chain and utilizes it to perform data integrity verification and the tracking of recorded data on the blockchain ledger. Each node participating in Fabric executes the contract, and the contract records the results of each algorithm's execution in the ledger. Executing contracts in Fabric incurs no separate costs, and it ensures high efficiency as it operates based on the commands of participating nodes. The time complexity for each algorithm is expressed using Big-O notation [42], as follows: Algorithm 1 has a complexity of O(1), Algorithm 2 also has a complexity of O(1), and Algorithm 3 has a complexity of O(m), where "m" represents the number of connected chains. When analyzing these time complexities, it is evident that Algorithm 3 has the most significant impact on performance and resource consumption. Therefore, this study conducts focused comparative experiments on this particular algorithm. The time complexity analysis for each algorithm is conducted in detail in Section 4.3, and, in Section 4.4, we analyze the scalability of future algorithms through an explanation of abstract data types (ADTs). This section presents the pseudocode for each algorithm followed by an analysis. Various abbreviations are used in this process, and detailed explanations of these abbreviations are provided in the Appendix A, specifically in Table A1.

4.2.1. Algorithm 1

The first algorithm encrypts data using the PKI structure and transmits the data to other nodes. During this process, the public and private keys utilize the PKI provided by Fabric CA [43]. Each node performs the process with a PKI in place, and private data encryption occurs during this process. Private data are encrypted with the public key of the receiving node and then transmitted to that node. Upon receiving data from the sending node, the receiving node retrieves the hashed data of the existing blockchain transaction to verify the data integrity. The previously hashed data are encrypted by the node that generated the initial data and process using a hash function before being recorded. The receiving node decrypts the received data with its private key, hashes the data, and compares them with the H_Data retrieved from the transaction to verify the integrity of the data.

Algorithm 1 outlines the procedures for securely transmitting data between nodes, and we use the PKI for this purpose. However, depending on the domain and service area, alternative encryption algorithms, such as secure sockets layer [44], may be used, or this step may not be necessary. The decision to encrypt data lies within the domain of the operating services and platforms of the administrators. Therefore, the execution of Algorithm 1 falls within the realm of recommendations and is not mandatory. In this

study, we exclude Algorithm 1 while verifying the efficiency and applicability of the hash-chain-based data-tracking process.

Algorithm 1: Transfer and Verify Private Data to Another Node using PKI

```
Given:
- The private data is transmitted to the First Sender after the data is created.
- Each node's public and private keys are issued through Fabric CA.
- Line No. is in the blockchain that records the data.
Step 0. FabricCA
    while do
        X_Pub, X_Priv = fabricCA(X)
    END
Step 1. Node n (Sender n)
    if the data to be recorded on the blockchain is prepared then
        n_Priv_Data = privateDataInProcess
        E = m_Pub(n_Priv_Data)
        transmit (E, Receiver m)
    END
END
Step 2. Node m (Receiver m)
    if receiving data from Sender n then
        H_Data = getTransaction(Line No., Key="H_Data")
        E = m_Pub(n_Priv_Data) = receive()
        m_Priv_Data = m_Priv(E) = m_Priv(m_Pub(n_Priv_Data))
        m_Data = SHA256(m_Priv_Data)
        if H_Data == m_Data then
            return Approval
        else
            return Refuse
        END
    END
END
```

4.2.2. Algorithm 2

The second algorithm verifies private data utilizing the structure of a hash chain. In step 1 of Algorithm 2, the node initially performs step 1 of Algorithm 1 to receive the data. In step 2, the node compares the received data with the data recorded in the existing blockchain to perform verification. Initially, the node searches for the hash values of the public and private data of the desired process line in the existing transaction. During this process, the node needs to receive the values of the two previously recorded data points before the m-th node, because the current node needs to confirm the accuracy of the previously recorded data before recording. After saving the values of the n-th and n-1-th data points, the node uses the data from the current process to generate a hash value, which is then compared with the previous hash value. If the comparison results are correct, confirming the integrity of the current process, the node creates a transaction and links it on the blockchain to proceed to the next stage of the process.

In this process, within Fabric's contract, data are converted into hash values to link them to the corresponding process line with a hash chain, and these data are recorded in the ledger. Specifically, the contract searches for the existing process line recorded in the ledger using the process line no. and links the next chain based on the last hash value configured in that process line's hash chain. If a new process line needs to be formed, the hash value of the first relevant data is used as the seed value for constructing the hash chain of that process line. Therefore, data associated with a specific process line are linked to their own hash chain, and the initial values of each hash chain branch are based on the hash value of the initially created process, which are recorded in the blockchain ledger.

Algorithm 2 serves as a fundamental part of the hash-chain-based data-tracking process proposed in this study. It is used to construct a hash chain for tracking the lifecycle of data and validating whether a node with private data verifies its correctness, thereby forming an essential process for ensuring data integrity during the proposed hash-chain-based data-tracking process.

Algorithm 2: Record the Data on Blockchain Utilizing Hash Chain Structure

Given:
- The private data is transmitted to the First Sender after the data is created.
- Each node's public and private keys are issued through Fabric CA.
- Line No. is in the blockchain that records the data.
Step 1. *Node n (Receiver n || Sender n)*
    Step 1 *of* ***algorithm 1***
END
Step 2. *Node m (Receiver m)*
    n_Pub_Data, n_Hash Value = getTransaction(Line No., "n")
    n-1_Hash Value = getTransaction(Line No., "n-1")
    m_Pub_Data = JSON(Line_No, Departures, Arrivals, . . .)
    m_Hash_Value = Next_Hash = SHA256(n_Hash_Value + m_Pub_Data)
    prev_Hash_Value = SHA256(n-1_Hash_Value + n_Pub_Data)
    if *n_Hash_Value == prev_Hash_Value* ***then***
        return Approval
    else
        return Refuse
    END
    execTransaction(H_Data, m_Hash_Value, m_Pub_Data, Line No.)
END

4.2.3. Algorithm 3

The third algorithm involves the tracking and verification of recorded data and serves as a tool in the industrial sector to address errors and anomalies encountered during the course of a process. It primarily examines tampering with the private and public data. First, the transaction data and hash chain values related to the corresponding process line are stored in an array. The stored data are then sequentially examined using a for-loop. For the first transaction associated with the process, the algorithm checks the data by computing the hash value of the private data along with the public data because the initial value is the hash value of the private data. Subsequently, starting from the second transaction, the algorithm performs the verification process by computing the hash value based on the previous hash value and public data for the number of transactions recorded in the process line. After completing the verification, the algorithm compares the number of transactions in the process and the number of computations performed after the operations to verify whether any interruptions in the for-loop occurred owing to errors.

In this process, within Fabric's contract, the verification of data recorded in the corresponding process line is performed through the structure of the hash chain. While Algorithm 2 continuously hashes the data to link the hash chain, Algorithm 3 rapidly re-verifies the values recorded in this connected hash chain through a hash function within the contract. This allows for the progression of data-tracking and verification processes.

Algorithm 3: Data-Tracking Process using Hash Chain

```
Given:
- The private data is transmitted to the First Sender after the data is created.
- Each node's public and private keys are issued through Fabric CA.
- Line No. is in the blockchain that records the data.
Step 1. Node n
    if Node m tracks and verifies the values of the hash chain then
        n_Pub_Data[], n_Hash Value[] = getTransaction(Line No.)
        hash = ""
        for n = 0 to length(Line No.) do
            if n == 0 then
                hash = H(SHA256(privateData) + n_Pub_Data[n])
                if hash == n_Hash Value[n] then
                    continue;
                else
                    unMatch()
                    break;
            else
                hash = H(hash + n_Pub_Data[n])
                if hash == n_Hash Value[n] then
                    continue;
                else
                    unMatch()
                    break;
                END
            END
            if n == length(Line No.) then
                Match()
            END
        END
    END
```

In this study, a hash chain is constructed for each process line comprising transactions to execute the integrity-verification and tracking processes. Based on the results of these processes, we can ascertain whether data breaches or attacks have occurred in the ongoing process line. Furthermore, this approach streamlines the tracking procedure in terms of time and resources, as it eliminates the need for a comprehensive examination of all transactions. Thus, this approach enables efficient data-history management.

4.3. Complexity Analysis of Algorithms

A crucial aspect of algorithmic assessment is understanding the time complexity, which provides insights into the efficiency and scalability of an algorithm under different conditions. In this section, we present the complexity analysis for Algorithms 1 through 3, which are integral to our proposed process.

- **Algorithm 1:** Algorithm 1 is designed to perform a specific task that requires a constant amount of time, irrespective of the input size. This is achieved by ensuring that the operations within the algorithm do not depend on the number of transactions and chains in the blockchain. As such, the time complexity of Algorithm 1 is O(1), denoting constant time complexity, meaning that its execution time does not change regardless of the size of the data set.
- **Algorithm 2:** Similar to Algorithm 1, Algorithm 2 also exhibits constant time complexity, which is O(1). The operations within this algorithm are designed to be completed in a fixed number of steps, which do not vary with the size of the input data. In Algorithm 2, the use of hash functions and the processing of transaction generation are crucial, which are processes identically performed across all blockchain platforms and

contracts. This constant time performance is crucial for ensuring predictable execution times, particularly in scenarios where quick response times are essential.

- **Algorithm 3:** In contrast to Algorithms 1 and 2, Algorithm 3 has a linear time complexity of O(m), where "m" represents the number of chains linked to the hash chain. The time required to execute this algorithm linearly increases with the number of chains, as it involves operations that must be individually performed on each chain. This linear relationship highlights the importance of optimizing the number of chains to maintain efficient performance, especially in large-scale blockchain applications. Furthermore, since Algorithm 3 is not affected by the number of transactions recorded in the blockchain ledger, it is confirmed to be more efficient in ensuring scalability than the conventional transaction-tracking methods.

The analysis of the time complexity of each algorithm demonstrates the scalability of the hash-chain-based data-tracking process proposed in this study. Algorithms 1 and 2 are not influenced by the number of transaction data recorded in the blockchain ledger, while Algorithm 3 is affected by the number of chains linked to the hash chain, rather than the total number of transactions. This approach can be applied to various blockchain platforms and contracts and is largely independent of the specific abstract data types (ADTs) currently employed.

4.4. Abstract Data Types (ADTs) of Algorithms and Advanced ADTs

In this section, we provide a structural explanation of the abstract data types used in the algorithms proposed in our study, along with an analysis of advanced ADTs. In the case of the ADTs of algorithms, there is an array of other structures within the structure itself, which leads to an increase in the number of arrays as the hash chain becomes linked. A detailed description of these structures is provided in Section 4.4.1, while Section 4.4.2 explains the algorithm complexity in relation to advanced ADTs.

4.4.1. ADTs of Each Algorithm

1. **Asset ADTs:** In our study's proposed hash-chain-based data-tracking process, these represent the overall ADTs constituting a single process line, including the process line number, the hash value of private data, a pointer value to link the hash chain, a collection of public data, and a continuously linked chain:
   - Process_Line_No: String, a unique identifier for the asset within the process;
   - H_Data: String, hashed data relevant to the private data;
   - N_Hash_Value: String, the next hash value for maintaining chain integrity;
   - Public: Public, a subset of the asset's data available for public viewing;
   - Chain: Array of "Chain", a sequence of chains to which the asset belongs.
2. **Chain ADTs:** These are arrays of structures included in the asset, representing each piece of data linked to the hash chain. They contain the previous chain's hash value for using the hash chain structure, the hash value of private data, a pointer value for linking to the next chain, and a public data set:
   - Previous_Hash: String, the hash value of the previous data set that is N_Hash_Value in the asset and chain;
   - H_Data: String, data relevant to the current block's hash;
   - N_Hash_Value: String, the anticipated next hash value for forward integrity;
   - Public: Public, a subset of the asset's data available for public viewing.
3. **Public ADTs**: Public ADTs are not fixed and vary depending on the business domain and the data set intended to be recorded. In essence, the key ADTs for linking the hash chain are the asset and chain, while Public ADTs are composed as a variable element.

4.4.2. Algorithm Complexity in Relation to Advanced ADTs

In this study, the proposed algorithms utilize conventional ADTs in the form of arrays of structures. However, the use of advanced ADTs, such as balanced trees, hash tables, or graph structures, can allow for more efficient data organization and retrieval. These structures can reduce the operational time and resource consumption as the size of the blockchain network increases.

- **Potential impact on Algorithms 1 and 2:** Algorithms 1 and 2, which currently operate with a constant time complexity (O(1)), may not experience significant changes in time complexity with the introduction of advanced ADTs due to their inherent constant time characteristics. However, these advanced ADTs can provide other benefits in blockchain applications' research and development, such as reduced space complexity and improved data integrity.
- **Potential impact on Algorithm 3:** Algorithm 3, which utilizes linear time complexity (O(m)), can be optimized through advanced ADTs. For example, instead of using an array of structures to manage the chains corresponding to a process line, a graph structure can be considered. By recording edges between the data linked by chains in the blockchain transactions, the structure in which the data is connected can be altered. Additionally, combining hash tables with tree structures can allow for recording the main process lines in hash tables, while the data corresponding to those process lines can be recorded using tree structures. By more efficiently organizing the data recorded in the blockchain, the potential to reduce linear complexity is realized. This can lead to faster processing times and more effectively address scalability issues, especially as the number of chains increases.

5. Methods and Experiments

5.1. Case Study

In this study, experiments are conducted with the objective of enhancing the efficiency of resource utilization and reducing the time required to record and track data within a service platform utilizing blockchain technology. Various methods are available for recording data on a blockchain. However, we utilize the structure of a hash chain to enhance efficiency and integrity while tracking the recorded data. To experiment with this, we design the hash-chain-based data-tracking process proposed in this study using several cases as a basis. Figure 4 shows a case process related to data movement, encryption, and node-process execution in the proposed process.

Node 1 generates a process line associated with the data to be recorded on the blockchain. During this process, the private data are fed into a hash function to derive a hash value. The derived hash value is designated as the first seed value of the hash chain for the process line, denoted as H_Data. H_Data serves as the seed value for the process and is recorded at all process stages to ensure integrity and reliability. Next, Node 1 combines the public data relevant to Node 1 with H_Data using the + operation (concatenation), resulting in N1_Hashed_Data when fed into the hash function.

In essence, to ensure integrity, H_Data, n_Hash_Value linked to the hash chain, and n_Pub_Data corresponding to the process performed at each node are recorded in the transaction. After the completion of the process at the first node, Node 1, the transaction is recorded on the blockchain, and the next node continues the process. For example, Node 3 verifies and records data based on the transactions recorded by Node 1. During this process, N1_Hashed_Data and 1_Pub_Data are retrieved from the transaction. Node 3 then generates 1_Hash_Value and compares it with the 1_Hash_Value recorded in the transaction, verifying its integrity. If the integrity of the data is ensured, Node 3 feeds the private data into the hash function to generate N3_Hashed_Data. Subsequently, H_Data, 3_Hash_Value, and 3_Pub_Data are recorded in the transaction, and the process is concluded.

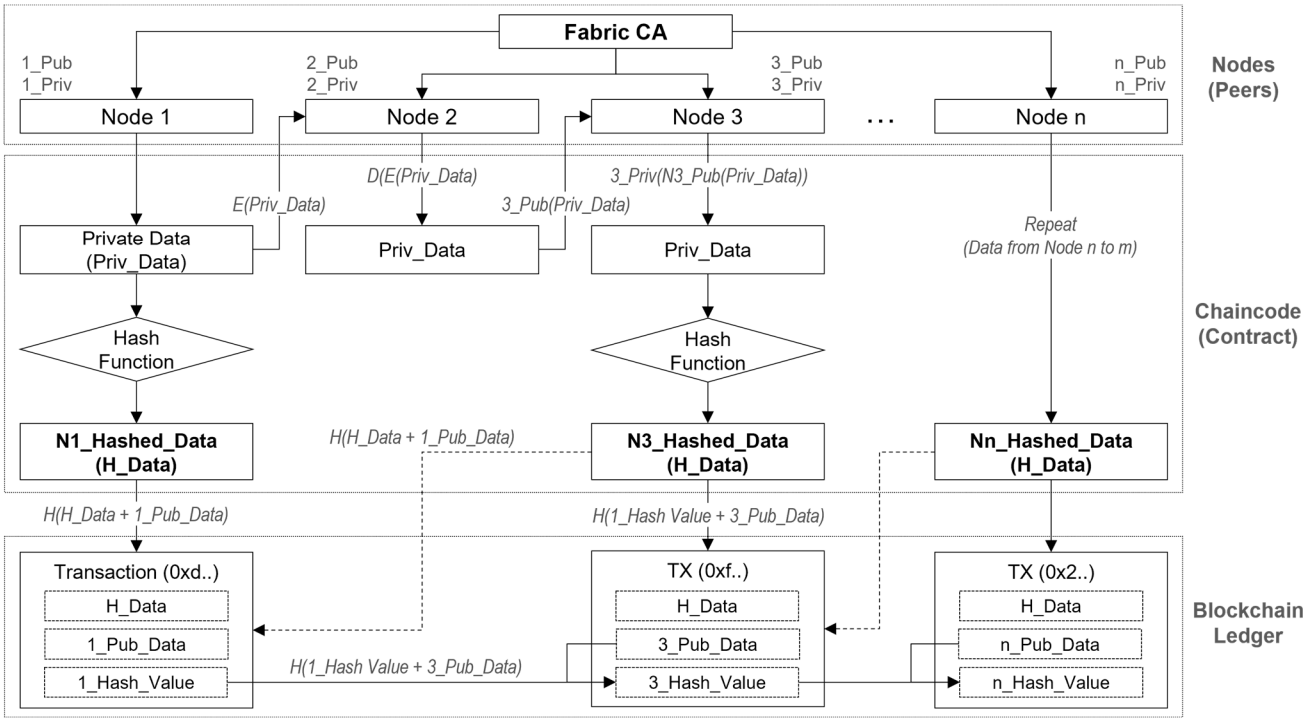

Figure 4. Proposed process for ensuring private data utilizing hash chain on blockchain.

Thus, the nodes associated with this process perform consecutive operations to generate transactions and record data on the blockchain by linking hash chains. Finally, Node n retrieves H_Data, the hash value of private data, from the blockchain to track and verify the data for this process, following the verification procedure in Algorithm 3.

5.2. Materials and Methodologies

5.2.1. Experimental Environment

The experimental conditions used in this study are listed in Table 3. The host operating system runs in a Windows 11 environment, and a virtual blockchain network is established by setting up a Hyperledger Fabric network using a CPU, GPU, and Docker [45]. The guest operating system, configured within Docker, is based on Ubuntu 20.04 [46]. The versions of Fabric and Golang [47] are consistent across all blockchain nodes. Data collection and monitoring tools such as Cadvisor [48,49], Prometheus [50], and Grafana [51,52] are utilized for the monitoring and data analysis of all nodes constituting the blockchain network. These tools enable real-time collection of data on the CPU, network traffic, and other relevant metrics that are crucial for subsequent result analysis.

Table 3. Experimental environment.

Category	Description
OS	Windows 11
CPU	Intel i5-11400
GPU	RTX 3070 Ti
RAM	16 GB
Docker OS	Ubuntu 20.04
Blockchain Environment	Hyperledger Fabric v2.53
Cadvisor Version	v0.47.2
Prometheus Version	v2.32.1
Grafana Version	V8.3.4
Golang Version	v1.20.4
Docker Version	v24.0.2

Figure 5 shows the structural environment of the experiments conducted in this study. Each node constituting the Hyperledger Fabric blockchain network operates in the capacity of a node, executes the chaincode to initiate the creation of the process line, and executes the chain configuration and tracking processes. In this process, data are relayed to Grafana through Prometheus, where the final data to be analyzed (CPU usage and network traffic) are extracted in the CSV format for further analysis.

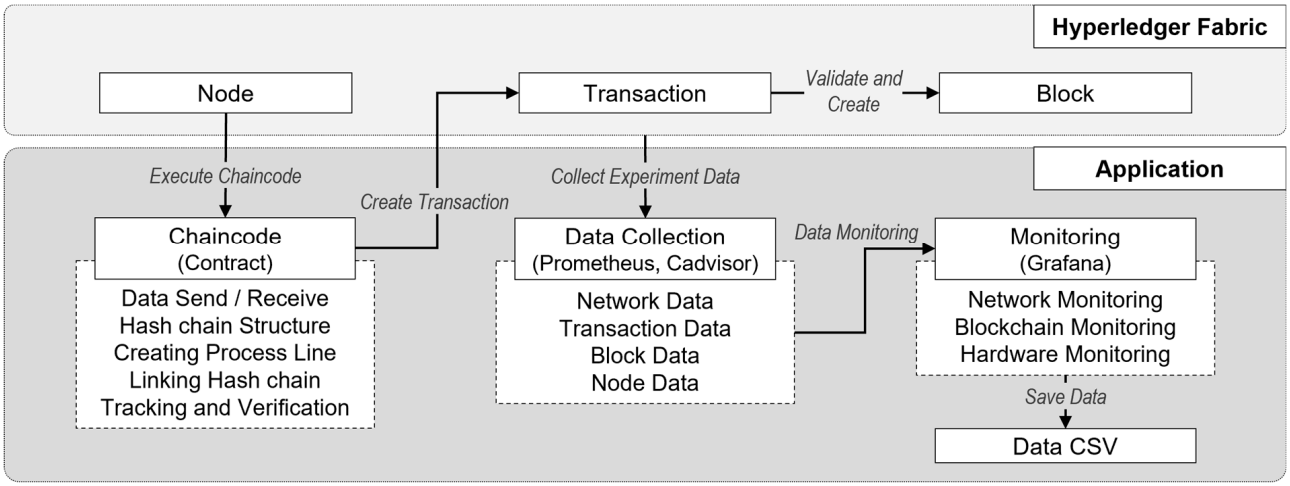

Figure 5. Structure of experiment on hash-chain-based data-tracking process.

5.2.2. Experimental Procedure

To conduct experiments on the data-tracking process based on the hash chain, the environment outlined in Table 3 must be set up, and the contract code should be executed within the Fabric network environment. In this study, we utilize three different contracts: (1) hash-chain-based contract, (2) hashed-data-based contract, and (3) hashed-data and hash-chain-based contract to conduct performance-comparison experiments.

In the proposed algorithm for hash-chain-based contracts, a chain is constructed using a structure array to link the data. During this process, the hash value of the original data, H_Data, is recorded. Additionally, to facilitate easy integrity verification within the connected chain and expedite the tracking process, N_hash_Value, computed by performing an operation on H_Data and public data, is recorded within the connected chain.

Unlike a hash-chain-based contract, a hashed-data-based contract does not rely on a connected chain structure. Instead, it uses a traditional transaction-linking structure. In this process, H_Data is recorded during the transaction to track specific processes. Because this contract does not have a connected chain structure, a thorough examination of the transactions recorded on the blockchain is required to extract transactions bearing the same H_Data for subsequent integrity checks.

Finally, a contract that utilizes the structure of the hashed data and chain does not employ structure arrays for the chain. Instead, it inserts a transaction number that is linked to the transaction itself. The key difference from a hash-chain-based contract is the absence of structural arrays, and the structure of the chain is directly established within the transactions. However, it cannot simultaneously fetch transactions related to the process line. Table 4 provides a detailed analysis of this contract.

Table 4. Comparison of contracts' properties.

Category	Hash Chain on Structure (1)	Hashed Data (2)	Hash Chain on Transaction (3)
Connection	Hash Chain	Transaction	Transaction
Structure	Struct Array	Struct	Struct
Chained	Chained	Not Chained	Chained
Time Complexity	O(m)	O(nm)	O(m)
CPU Usage	Low	Middle	High
Network Traffic	Low	High	High
Hash Function	SHA-256	SHA-256	SHA-256
Tracking Time	Fast	Middle	Slow
Scalability	High	Middle	Low

Connection refers to the basic method of tracking data in each contract. Structure refers to the form in which data are recorded. Chained indicates whether data are chained in each process line. Time complexity is expressed in a simple Big-O notation based on the code of each contract. M = number of chains, n = number of transactions, O = Big-O notation.

5.3. Results

In this section, we present a qualitative performance comparison based on a quantitative comparison of the experimental results of the hash-chain-based data-tracking process proposed in this study. Using the three contracts described earlier, we measure and analyze the tracking time, hardware resources, network traffic, and other relevant factors to evaluate the performance of the proposed algorithm considering external factors. The experiments are categorized into three main types for result analysis. For each experiment type, we performed repeated experiments based on the total number of transactions, number of connected chains, and type of contract. The average value of each metric from the experiments are used for analysis. Tables A2–A6 present the average values of the data used in the graphs and are included in Appendix B.

5.3.1. Qualitative Analysis

Before performing a quantitative comparative analysis of each contract, we first conduct a qualitative analysis of each contract. The time complexity of each contract can be expressed using Big-O notation as follows: for contract (1), O(m): m = number of chains; for contract (2), O(nm): n = number of transactions, and m = number of chains; for contract (3), O(m): m = number of chains. These time complexities are derived from a simple analysis of the code within the chain. Table 4 provides a qualitative comparison of the experimental results for each contract, and the specific experimental results for each contract are graphically represented.

In Table 4, the connection type used for tracking in the contract is divided into hash chains and transactions. This division is determined by the fundamental elements on which the tracking process is built during the implementation. For transactions, the tracking process resembles a traditional transaction-tracking process. However, the key distinction lies in the utilization of a hash chain in which the structure for recording data is organized into a separate chain. "Structure" and "Chained" indicate whether each contract is structured and composed in a chain-like form. For a hash chain on structure, the connection is established through an array of structures, whereas the hash chain on transaction links the desired chain data key value within the structure using variables. However, in the case of hash data, data are recorded through structures but are not linked in a chain-like structure. Moreover, consistent hashing using the SHA-256 [53] hash function is maintained across all contracts.

When analyzing the CPU usage and network traffic, we observed that the transaction-based hash chain contract consumed more resources and traffic. Consequently, in terms of tracking time, we noted that the CPU and network traffic significantly influenced tracking time, apart from time complexity. While the time complexity is the same for the hash chain

on structure and hash chain on transaction, the tracking time is faster for the hash chain on structure owing to lower resource consumption and traffic. A more detailed explanation is provided below through quantitative analysis and specific descriptions using graphs.

5.3.2. Quantitative Analysis

Figure 6 and Tables A2–A6 present the experimental results for the three contracts, considering different factors such as the number of chains and transactions, and tracking time, CPU usage, and network traffic are used as evaluation metrics. Analysis of the experimental results for each contract is explained below based on the respective graphs. The units for each experimental metric are as follows: the tracking time is measured in milliseconds (ms), CPU usage is expressed as a percentage (%), and network traffic is denoted in kilobytes (KB).

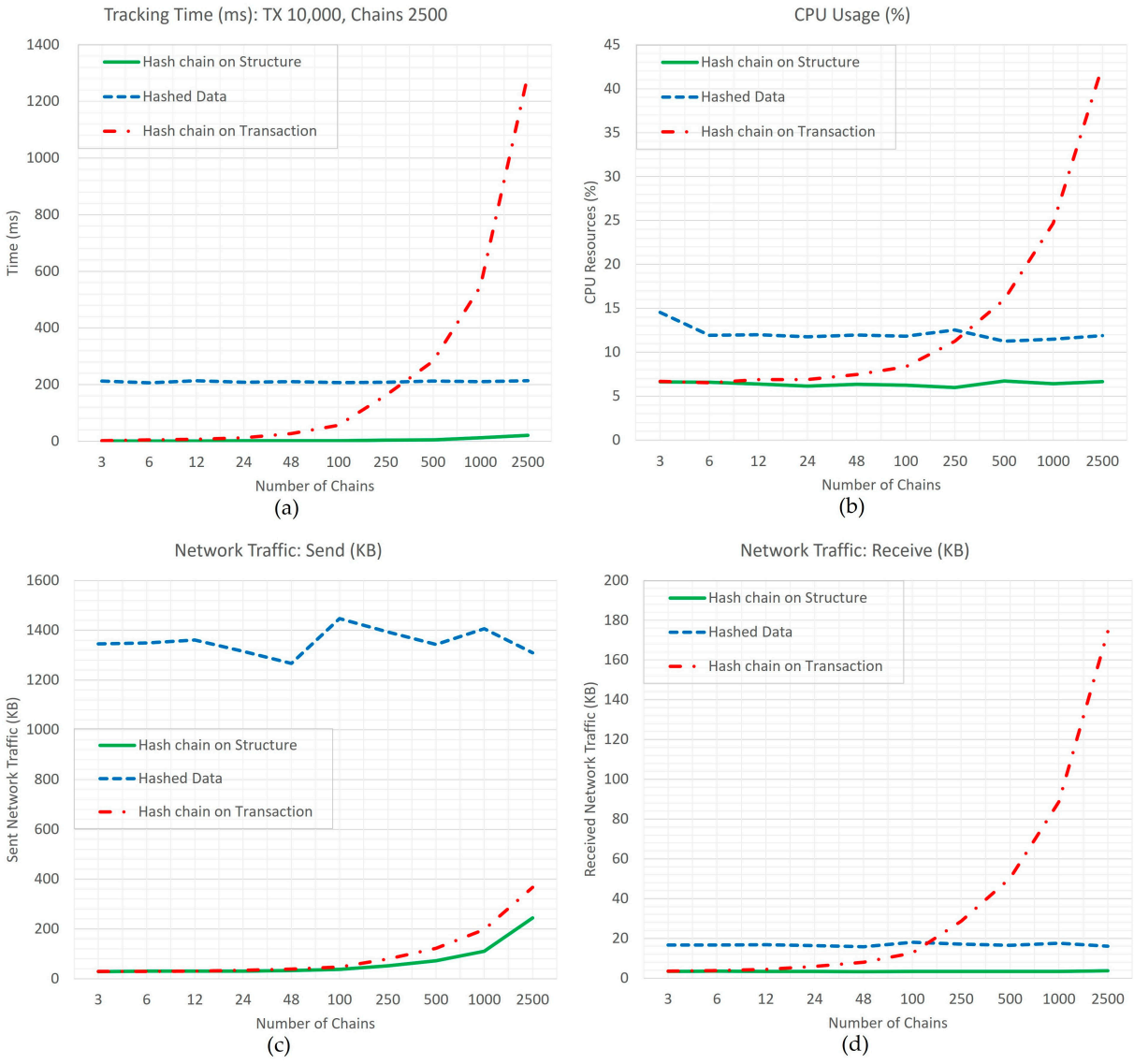

Figure 6. *Cont.*

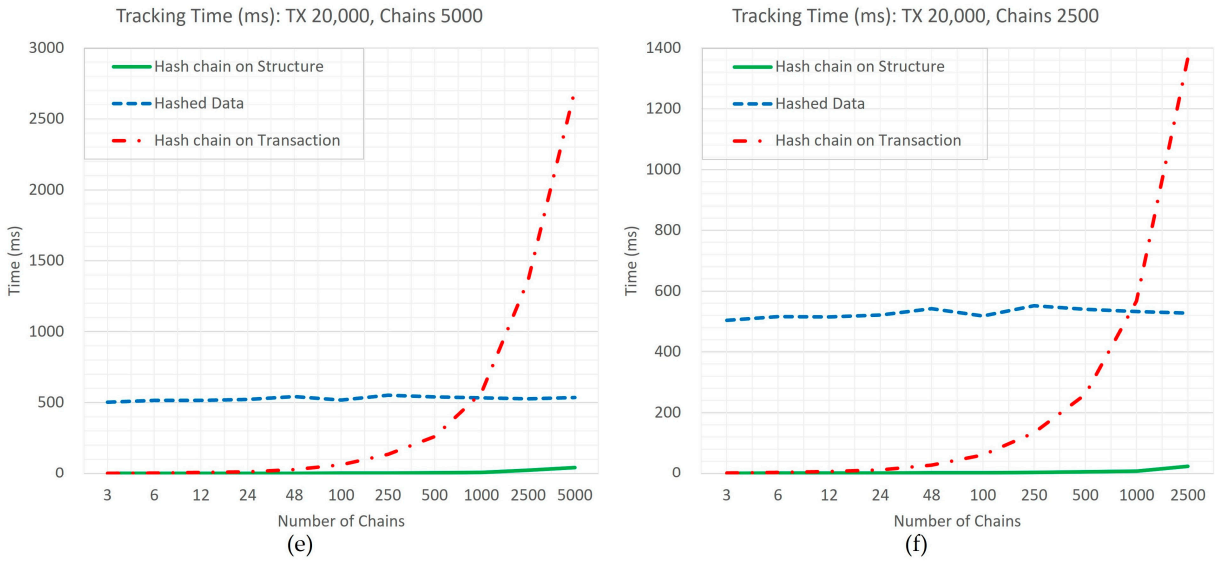

Figure 6. Comparative analysis of tracking time, CPU usage, and network traffic across three contracts. (**a**) Tracking Time (ms): TX 10,000, Chains 2500, (**b**) CPU Usage (%), (**c**) Network Traffic: Send (KB), (**d**) Network Traffic: Receive (KB), (**e**) Tracking Time (ms): TX 20,000, Chains 5000, (**f**) Tracking Time (ms): TX 20,000, Chains 2500.

(a) Chain Counts—Tracking Time: Referring to Table 4, the hashed data are considered to have the longest execution time owing to their time complexity. However, the actual experimental results depicted in Figure 6a show that the hash chain on transaction exhibits the longest execution time. This highlights the fact that the tracking time is not simply equivalent to the time complexity of the contract.

(1) The hash chain on structure, the tracking process proposed in this study, constructs a chain based on a structure. It achieves a faster tracking time than the other two contracts. During the initial process line formation, the target asset (process) to be tracked is recorded. Subsequently, when consecutively executing the same process as the identified process line, the structure is connected in array form to the existing asset, thus forming a chain. Based on the analysis of the experimental results, the primary reason for the fastest tracking time in this contract is the utilization of a structure array.

(2) The hashed data do not show a noticeable increase as the number of chains increases because they involve an exhaustive inspection of transactions. Surprisingly, even when the number of chains increases, the tracking time tends to decrease. This finding indicates that external factors can influence the results. However, we observe that the tracking time for the hash data is significantly longer than that for the hash chain on structure. This experimental result suggests that exhaustive inspection of transactions affects the tracking time. Additionally, the decrease in tracking time suggests that even with all conditions set to be the same during the experiment, external factors such as the CPU and network affect the experimental results. This is further detailed in Figure 6b–d.

(3) Hash chain on transaction exhibits a steep increase in tracking time. As expected, the time complexity of this contract is O(m), which increases with the number of chains. This increase is attributed to the execution time of the function retrieving transactions with the recorded connected chains from the blockchain ledger. Hyperledger Fabric provides application programming interface (API) functions for recording and retrieving transactions in the ledger. The execution time of this function influences the increase in tracking time.

(b) Chain Counts—CPU Usage: The CPU usage was used to observe trends with respect to the number of chains. The Hyperledger Fabric chain code is executed by nodes

participating in the blockchain network by utilizing the CPU when executed on each node. During this process, as the CPU usage increases, there is potential for interference with the ability to perform processes outside the Fabric. Additionally, a higher CPU usage consumes more resources and, consequently, prolongs the tracking time. Therefore, in this study, a process is proposed to ensure the integrity and security of data recorded on the blockchain, while reducing resource consumption. The experimental results for this are illustrated in Figure 6b.

(1) An examination of the CPU usage of the contract utilizing a hash chain on structure shows that it utilizes approximately 5% of the total usage on one node. In this study, the proposed contract forms a chain in the form of a structure. Consequently, in the process of tracking data related to the process line, only one transaction is fetched, resulting in significantly fewer accesses to the blockchain ledger compared with the other two contracts. Therefore, a notably low level of CPU usage is observed. Even with a manifold increase in the number of chains, the CPU usage remains almost constant.
(2) For hashed data, a nearly consistent time is required, regardless of the number of chains, as evident in the tracking time graph. Moreover, in Figure 6b, CPU usage remains relatively stable. This finding suggests that contracts are minimally influenced by the number of chains. Consequently, it is inferred that the tracking time and CPU usage for this contract are determined by factors other than the number of chains, which is further elaborated in Figure 6e,f.
(3) The hash chain on transaction exhibits an upward trend in both tracking time, as seen in Figure 6a,b, and CPU usage. As previously explained, this can be attributed to an increase in CPU usage during the process of fetching transactions from the blockchain ledger. Initially, when the number of chains is three, the CPU usage is similar to that of the hash chain on structure. However, as the number of chains increases, a similar upward trend in the CPU usage is observed. Although this contract enhances integrity and security through the structure of chains, it has a lower code efficiency, resulting in increased tracking time.

(c) Chain Counts—Network Traffic: The presented experimental results in Figure 6c,d represent the measurement of network traffic at the node directly involved in tracking the data. In the data-tracking process, network traffic occurs because of the process of requesting and receiving the results of transactions recorded in the blockchain ledger. Therefore, in the proposed data-tracking process, network traffic is an essential factor that must be considered in addition to tracking time. As explained previously, two cases are considered: sending and receiving.

(1) In the hash chain on structure, both sending and receiving show remarkably low results. One of the factors that generate network traffic is the number of data. In this contract, the process of requesting and receiving data related to a specific process line is concluded through network traffic, resulting in a minimal amount of traffic to complete the tracking process. In other words, the contract algorithm itself significantly reduces network traffic.
(2) As previously explained, a contract based on hashed data performs a complete examination of all transactions recorded in the blockchain. In this contract, network traffic occurs during an exhaustive examination of all transactions. As observed in the graph, the network traffic remains constant regardless of the number of chains, instead of varying with the total number of transactions. This contract examines all transactions, thereby generating network traffic. As shown in the graph, the network traffic remained consistent regardless of the number of chains.
(3) A contract utilizing the structure of the hash chain on transaction constructs chains based on transactions. It is necessary to request and receive all relevant transactions to track the data on the chains associated with a process line. Consequently, the graph for this contract shows that the network traffic increases with the number of chains.

However, compared to the hash-data-based contract, it exhibits lower traffic. This is because this contract requests transactions based on the number of chains but does not conduct a complete examination of transactions. Thus, while not exhaustively examining transactions, the network traffic increases with the number of chains.

(d) Transaction Counts—Tracking Time: In this experiment, we analyze graphs to understand how tracking time varies with the number of transactions. In experiment (a), we analyze the results based on 10,000 transactions and 2500 chain counts. However, Figure 6e,f present the results of the experiments conducted with 20,000 transactions and 2500 chains and 20,000 transactions and 5000 chains as the baseline, respectively. In other words, we analyze the tracking time based on the number of transactions and chain counts for the three contracts and conduct an analysis of the consistency of the results.

(1) As observed in Figure 6e, the contract based on the hash chain on structure yields similar experimental results when the number of chains is fixed at 2500, irrespective of the total transaction count. This can be attributed to the fact that, similar to the CPU usage and network traffic discussed earlier, it only requires the retrieval of data associated with a specific process line and related chains that are recorded. Figure 6e,f indicate that with a fixed transaction count of 20,000, an increase in the number of chains leads to an increase in the tracking time. Thus, the performance of this contract varies based on the number of connected chains, with no impact from the total transaction count.

(2) In the case of the hashed data, the performance varies with the total number of transactions. As shown in Figure 6a, the tracking time is approximately 200 ms for 10,000 transactions, and, in Figure 6e, it is approximately 500 ms for 20,000 transactions. This suggests that the tracking time in this contract is directly proportional to the number of transactions and is unaffected by the number of connected chains. Hence, as the blockchain network grows and more transactions are recorded, the tracking time increases.

(3) When comparing Figure 6a,f with the same settings but with a change in transaction count from 10,000 to 20,000, the hash chain on transaction shows that the total transaction count, much like in (1), does not significantly influence tracking time. Figure 6e shows that the tracking time increases based on the number of connected chains. However, with an increase in the number of chains, the slope of the graph sharply steepens, indicating that, as the number of connected chains increases, the tracking time significantly increases.

5.3.3. Summary

The experiments conducted in this study compare the performance of three contracts, including the hash-chain-based data-tracking process, proposed as a means to track the history of data recorded on the blockchain. To track the recorded data, tracking functions are implemented within the contracts to measure tracking time, CPU usage, and network traffic. As previously described, the contract based on the hash chain on Structure demonstrates the highest performance. To explain this, we refer to each comparative experimental result as follows:

- **CPU Usage:** Both contracts based on the hash chain on structure and hash chain on transaction involve verification processes for the hash values of private data and verification of chain values using public data. However, in the latter case, when 2500 chains are connected, 2500 transactions must be fetched from the blockchain ledger, resulting in higher CPU usage. Therefore, analysis of the CPU usage measured in this experiment reveals a tendency for increased CPU usage during the process of fetching transactions recorded in the blockchain.
- **Network Traffic:** Blockchain, a distributed ledger in which multiple nodes form a network, incurs network traffic during the process of fetching and verifying transactions. In this study, three contracts are tested. The hashed-data-based contract, which

requires comprehensive verification of transactions, exhibits an increase in network traffic based on the number of transactions. Furthermore, in the case of the hash chain on transaction, the increase in the number of chains leads to an augmentation of network traffic owing to the process of fetching and verifying the connected transactions within the blockchain network. In conclusion, the network traffic is significantly influenced by the process of fetching transactions from the blockchain network. Notably, the hash chain on structure proposed in this study, which verifies connected chains through a single transaction, exhibits the least network traffic.

- **Number of Transactions:** One of the metrics used to evaluate the performance of a blockchain network is transactions per second [54], which highlights the importance of transaction processing. The experiments conducted in this study aim to compare contracts to ensure efficiency in tracking the data recorded in transactions. Therefore, considering metrics such as CPU usage and network traffic, the hash chain on structure constructs a separate chain within each transaction to minimize the computational load per transaction and mitigate the impact on the total number of transactions during the verification process.
- **Number of Chains:** The structure of the chain represents one of the technical elements proposed in this study to facilitate and effectively execute the process of tracking data history. Chains connected to a process line contain data related to the actions, results, and procedures associated with a specific process. As the number of connected chains increases, the tracking time logically increases. However, the hash chain on structure proposed in this study utilizes a structure of arrays to minimize the impact of the number of connected chains, resulting in the highest efficiency and performance.
- **Tracking Time:** Connecting the above analysis to the tracking time shows that the hash chain on structure utilizes the fewest resources during the processes involving CPU usage, network traffic, and others. It demonstrates the fastest tracking performance by minimizing the impact of both transaction and chain quantities. In addition, when considering the tracking time within the blockchain, the contract's time complexity and external resource consumption must be accounted for.

5.3.4. Prediction and Comparison of Tracking Time Based on Simple Linear Regression

Additionally, the simple linear regression [55] functions for each contract according to the number of transactions and chains are represented by the following equations. $T_{t,n}$ signifies the tracking time according to the number of transactions and contracts as represented in Equations (1)–(3), while the simple linear regression function to predict the changes in the final tracking time based on the number of Chains C in Equation (4) is represented in Equations (6)–(8). R^2, the coefficient of determination, is used as a measure to indicate the explanatory power of the regression model for Equations (6)–(11). In other words, a value of R^2 close to 1 implies that the regression model explains the variability of the dependent variable well.

$$T_{t,n} = Tracking\ Time\ (ms) \tag{1}$$

$$t = Number\ of\ Transactions \tag{2}$$

$$n = Contract\ Number\ (1, 2, 3) \tag{3}$$

$$C = Number\ of\ Chains \tag{4}$$

$$R^2 = r^2 = \frac{SSR}{SST} = \frac{\sum\left(\hat{Y} - \overline{Y}\right)^2}{\sum\left(Y - \overline{Y}\right)^2} : Coefficient\ of\ Determination \tag{5}$$

Equations (6)–(8) are the simple linear regression functions for contracts (1), (2), and (3) with 10,000 transactions. The predicted tracking time (ms) according to the number of

chains for these simple linear regression functions (6)–(8) is detailed in Table A7, which is included in Appendix B.

$$T_{10000,1} = 0.008350941C + 1.35252795,\ R^2 = 0.9825 \quad (6)$$

$$T_{10000,2} = 0.001543091C + 209.4076746,\ R^2 = 0.2169 \quad (7)$$

$$T_{10000,3} = 0.518595789C + 9.087888095,\ R^2 = 0.9989 \quad (8)$$

The following Equations (9)–(11) are the simple linear regression functions for contracts (1), (2), and (3) with 20,000 transactions. The predicted tracking time (ms) according to the number of chains for these functions is presented in Table A8, as included in Appendix B.

$$T_{20000,1} = 0.00837555C + 1.05303041,\ R^2 = 0.9958 \quad (9)$$

$$T_{20000,2} = 0.00218473C + 525.828953,\ R^2 = 0.0569 \quad (10)$$

$$T_{20000,3} = 0.54191816C + 2.57352809,\ R^2 = 0.9999 \quad (11)$$

When comparing Tables A7 and A8 with Tables A2 and A3, it can be seen that, for contracts (1) and (3), the actual experimental results and the predictions begin to align when the number of chains exceeds 24. This confirms that the simple linear regression functions derived from the results of this study produce similar outcomes according to the number of chains. Equations (7) and (10) show a small slope and a large Y-intercept, indicating that the tracking time is almost unaffected by the number of chains, whereas the relatively larger slopes of Equations (6), (8), (9), and (11) indicate that the tracking time is influenced by the number of chains. Moreover, it is proven using the above equations that contract (1) is less affected by the number of chains compared to contract (3), meaning that, even as the process line lengthens, the increase in tracking time is minimal.

Graphs comparing the actual experimental results with the tracking time predictions based on the values from Tables A7 and A8 are observed in Figure 7a,b. The data connected by lines represent the actual experimental results, while the data points represent the predictions generated through the simple linear regression functions. Comparing these, it can be seen that the actual experimental results and the predictions are similar. Therefore, this study demonstrates that such simple linear regression functions are used to predict the tracking time as the number of chains increases, and it explains the scalability and analytical potential of applying a hash-chain-based data-tracking process in business processes.

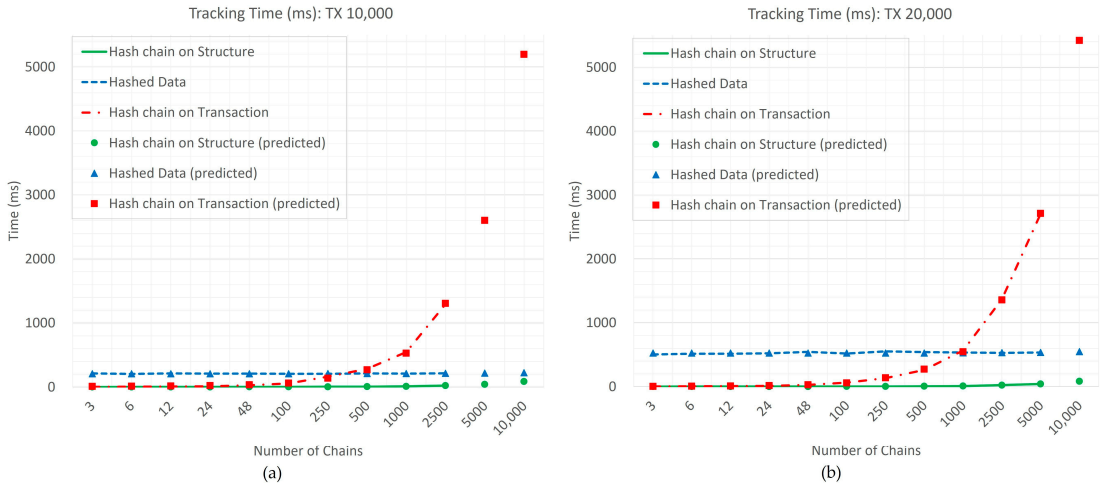

Figure 7. Comparing data-tracking times between experimental results and predictions. (**a**) Experimental and Predicted Tracking Time (ms): TX 10,000, (**b**) Experimental and Predicted Tracking Time (ms): TX 20,000.

6. Discussion

In this study, we propose a process for executing contracts in the form of a hash chain to reduce resource consumption and shorten the time required to track the transaction data recorded in the blockchain, which is utilized by services using existing blockchain platforms. This process constructs a hash chain in a branching pattern similar to a hash graph, but with the distinction that the data are sequentially recorded rather than randomly. A hash-chain-based data-tracking process is proposed to enhance efficiency, integrity, and traceability when recording data on permissioned blockchains in the industry. Because the blockchain length increases with the continuous influx of transactions, the process of retroactively tracking transactions consumes more time and resources. This study aims to address this issue. The experiments conducted in this study involve adding a tracking process to the contract on a Hyperledger Fabric basis and evaluating its performance.

6.1. Application in Public Blockchains

It is important to discuss the applicability of the hash-chain-based data-tracking process proposed in this study in different blockchain platforms. We believe that platforms that support smart contracts, such as Ethereum, can adopt our approach with minimal modifications. The proposed method is designed to operate independently of the underlying consensus algorithm, and, therefore, it is not significantly affected by the choice of consensus mechanism. However, blockchain platforms with slower transaction processing speeds, such as those utilizing Proof of Work (PoW) [13], may experience delays due to the time required to reach consensus on transactions.

We acknowledge that the efficiency benefits demonstrated in this research are most prominent within permissioned blockchain environments, where transaction speeds are generally faster, and governance is more centralized. In contrast, public blockchains present distinct challenges, including higher transaction volumes and the need for more robust consensus mechanisms, which can impact the overall effectiveness of the hash chain structure. In future studies, we aim to expand the experimental limits and apply various blockchain platforms and consensus algorithms for experimentation.

6.2. Constrained Experimental Environment and Scalability

The experiments conducted in this study are limited to a maximum of 20,000 total transactions recorded in the blockchain ledger and 5000 chains that compose the hash chain. To address this limitation, the experimental results are used to predict the tracking time through simple linear regression functions, and comparative analyses are conducted. It is observed that while the total number of transactions influences the existing blockchain transaction-data-tracking process, it hardly affects the hash-chain-based transaction-data-tracking process proposed in this study. That is, with the traditional tracking process, as the size of the blockchain increases, both the tracking time and resource consumption increase. However, the hash-chain-based tracking process is affected only by the number of linked chains, resulting in a relatively smaller increase in time delay and resource consumption, even as the blockchain size grows.

Indeed, continuously increasing the number of transactions and chains for further experimentation is constrained by the current experimental environment. Nevertheless, by utilizing the simple linear regression functions to analyze scalability, we can confirm that the hash-chain-based tracking process proposed in this study offers clear advantages. The actual resource consumption and tracking time, when the process is applied In practice, could differ from the experimental results. This discrepancy may be due to external factors such as the usage of machines operating as blockchain nodes and network conditions. Therefore, future research should involve selecting specific business processes and developing actual pilot applications to collect data. This will allow for a comparative analysis of the experimental results, predictive values, and real-world application outcomes.

6.3. Application Domains and Case Studies in Data Tracking

While the hash chain structure proposed in this study is optimized to reduce tracking time and resource consumption for transaction data recorded on blockchain, it is crucial to acknowledge services where this process may be less beneficial or not applicable. For example, platforms that handle non-sensitive, transient data, where the emphasis is on throughput rather than traceability, may not benefit from the added complexity and overhead of a hash-chain-based tracking system. Consider fast-moving consumer goods (FMCG) industries, where the primary concern is inventory turnover rather than detailed data tracking. Implementing a hash chain structure in such a context could lead to unnecessary computational overhead, negating the inherent high-speed nature of these businesses. Similarly, in service platforms where data are frequently updated or replaced, the permanence offered by the blockchain may serve more as a hindrance than a benefit. Moreover, systems that operate with a high level of trust among participants and do not require the immutability guarantee of the blockchain, such as internal data management systems of a single organization, might find the hash chain structure redundant. In these cases, traditional databases might be more cost-effective and simpler to maintain. Furthermore, in service domains where the efficiency of data tracking is unnecessary, research can be conducted on access control for data using time-based one-time passwords [56] instead of employing hash chains.

Optimizing the implementation of the proposed hash chain structure for diverse use cases involves a cost–benefit analysis to determine if the benefits of increased traceability and integrity outweigh the potential trade-offs in system complexity and performance. For sectors where tracking is crucial, such as finance, healthcare, and supply chain management, the optimization might include the development of hybrid models that combine hash chains with selective encryption using the hash function to protect confidential data, while still providing the speed and efficiency required for high-volume transactions. In light of these considerations, future work could focus on developing adaptive hash chain mechanisms that are capable of identifying the specific needs of a service platform and scaling their complexity accordingly.

7. Conclusions

We proposed a hash-chain-based data-tracking process to streamline and enhance the efficiency of the tracking process within blockchain-based service platforms. The existing blockchain transaction-tracking process requires a thorough investigation of data to retroactively track transactions recorded in the network, relying on the provided APIs, which presents a challenge. However, we connected the hash values of the data recorded in each transaction in the form of a hash chain. Thus, by knowing the desired process line number, we could track transactions related to that line, thereby reducing the exposure risk to the original data. In addition, with the hash values linked in a chain structure, verifying the integrity of the data handled in each transaction and process line became easy. We conducted performance comparison experiments between the hash-chain-based data-tracking process and the traditional transaction-tracking process, facilitating both quantitative and qualitative assessments of the proposed process.

In this study, through the proposed contract-based process, we streamlined the transaction-tracking process in traditional blockchains, reducing time and resource consumption. Utilizing hash functions in a permissioned blockchain ensured both data integrity and confidentiality, while the implementation of a hash chain structure secured symmetry in the data recorded on the blockchain ledger. Furthermore, we evaluated the applicability of data tracking in permissioned blockchains. By simplifying this process, we proposed a method to effectively utilize blockchains across various industries that handle sensitive data. We hope that insights gained from future research will enable the proactive utilization of diverse blockchain platforms. We anticipate that the blockchain will be actively applied and utilized, especially in specific domains such as logistics, freight transportation, customs clearance, and document verification.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00251241) and also supported by Kyonggi University's Graduate Research Assistantship 2023.

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

Appendix A. Abbreviations

Table A1. Abbreviation list.

Term	Full Abbreviation
H_Data	Hashed origin (private) data
n	Node n [n = 1, 2, . . ., n]
n_Pub	Public Key (of Node n)
n_Priv	Private Key (of Node n)
n_Priv_Data	Private Data (of Node n)
n_Pub_Data	Public Data (of Node n)
n_Pub(T)	Encrypt the T [T: Data]
n_Priv(n_Pub(T))	Decrypt the n_Pub(T)
n_Hash_Value	Hash value from X for "hash chain"
H(T)	Encrypt the T using Hash Function [T: Data]
T1 + T2	Concatenate T1 and T2 [T: Data]
TX	Transaction

Appendix B

These tables represent the average values of the experimental results corresponding to the graphs shown in Figures 6 and 7. In addition, each contract in the table is denoted as follows: (1) hash-chain-based contract, (2) hashed-data-based contract, and (3) hashed-data and hash-chain-based contract.

Table A2. Experiment results: tracking time (TX 10,000).

Contract	Number of Chains				
	3	**6**	**12**	**24**	**48**
(1)	0.981	1.104	1.169	1.347	1.348
(2)	243.207	205.907	213.126	208.254	210.144
(3)	2.235	4.555	7.111	12.444	27.122
	100	**250**	**500**	**1000**	**2500**
(1)	1.936	4.086	5.387	11.956	21.315
(2)	207.261	207.997	212.142	210.641	228.255
(3)	56.693	162.163	282.986	546.410	1293.281

The tracking time corresponding to each contract and number of chains is expressed in milliseconds (ms).

Table A3. Experiment results: tracking time (TX 20,000).

Contract	Number of Chains					
	3	**6**	**12**	**24**	**48**	**100**
(1)	0.986	1.058	0.987	1.276	1.789	2.109
(2)	503.674	516.222	514.679	521.332	542.653	518.605
(3)	1.761	3.671	6.602	11.986	27.223	61.244
	250	**500**	**1000**	**2500**	**5000**	.
(1)	3.049	5.345	7.839	24.010	42.227	.
(2)	551.930	540.123	533.014	527.309	535.207	.
(3)	134.259	260.673	567.403	1366.25	2704.56	.

The tracking time corresponding to each contract and number of chains is expressed in milliseconds (ms).

Table A4. Experiment results: CPU usage.

Contract	Number of Chains				
	3	**6**	**12**	**24**	**48**
(1)	6.620	6.595	6.377	6.157	6.350
(2)	14.550	11.931	12.006	11.757	11.956
(3)	6.702	6.535	6.886	6.902	7.473
	100	**250**	**500**	**1000**	**2500**
(1)	6.264	5.993	6.722	6.423	6.642
(2)	11.817	12.547	11.261	11.490	11.892
(3)	8.336	11.258	15.989	24.660	42.585

The CPU usage corresponding to each contract and number of chains is expressed as a percentage (%).

Table A5. Experiment results: network traffic (send).

Contract	Number of Chains				
	3	**6**	**12**	**24**	**48**
(1)	27.920	29.745	30.051	29.823	32.307
(2)	1345.737	1348.769	1360.853	1315.287	1267.190
(3)	29.056	29.457	30.635	33.407	38.305
	100	**250**	**500**	**1000**	**2500**
(1)	37.404	51.199	72.032	109.715	243.566
(2)	1447.415	1393.477	1343.042	1406.506	1310.048
(3)	47.421	78.749	122.234	197.927	367.543

The network traffic corresponding to each contract and number of chains is expressed in kilobytes (KB).

Table A6. Experiment results: network traffic (receive).

Contract	Number of Chains				
	3	**6**	**12**	**24**	**48**
(1)	3.350	3.510	3.356	3.330	3.234
(2)	16.666	16.659	16.764	16.356	15.813
(3)	3.562	3.867	4.412	5.866	8.027
	100	**250**	**500**	**1000**	**2500**
(1)	3.362	3.407	3.422	3.365	3.712
(2)	17.993	17.087	16.484	17.528	16.043
(3)	12.619	28.600	50.344	88.687	174.221

The network traffic corresponding to each contract and number of chains is expressed in kilobytes (KB).

Table A7. Predicted results: tracking time (TX 10,000).

Contract	Number of Chains					
	3	**6**	**12**	**24**	**48**	**100**
(1)	1.378	1.403	1.453	1.553	1.753	2.188
(2)	209.412	209.417	209.426	209.445	209.482	209.562
(3)	10.644	12.199	15.311	21.534	33.980	60.947
	250	**500**	**1000**	**2500**	**5000**	**10,000**
(1)	3.440	5.528	9.703	22.230	43.107	84.861
(2)	209.793	210.179	210.951	213.265	217.123	224.839
(3)	138.737	268.386	527.684	1305.577	2602.067	5195.05

The tracking time corresponding to each contract and number of chains is expressed in milliseconds (ms).

Table A8. Predicted results: tracking time (TX 20,000).

Contract	Number of Chains					
	3	**6**	**12**	**24**	**48**	**100**
(1)	1.078	1.103	1.154	1.254	1.455	1.891
(2)	525.836	525.842	525.855	525.881	525.934	526.047
(3)	4.199	5.825	9.077	15.580	28.586	56.765
	250	**500**	**1000**	**2500**	**5000**	**10,000**
(1)	3.147	5.241	9.429	21.992	42.931	84.809
(2)	526.375	526.921	528.014	531.291	536.753	547.676
(3)	138.053	273.533	544.492	1357.369	2712.164	5421.755

The tracking time corresponding to each contract and number of chains is expressed in milliseconds (ms).

References

1. Nofer, M.; Gomber, P.; Hinz, O.; Schiereck, D. Blockchain. *Bus. Inf. Syst. Eng.* **2017**, *59*, 183–187. [CrossRef]
2. Wang, H.; Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352. [CrossRef]
3. Pierro, M.D. What is the blockchain? *Comput. Sci. Eng.* **2017**, *19*, 92–95. [CrossRef]
4. Hsiao, S.-J.; Sung, W.-T. Blockchain-Based Supply Chain Information Sharing Mechanism. *IEEE Access* **2022**, *10*, 78875–78886. [CrossRef]
5. Wu, Z.; Liu, J.; Wu, J.; Zheng, Z.; Chen, T. TRacer: Scalable graph-based transaction tracing for account-based blockchain trading systems. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2609–2621. [CrossRef]
6. Ismail, L.; Materwala, H. A Review of Blockchain Architecture and Consensus Protocols: Use Cases, Challenges, and Solutions. *Symmetry* **2019**, *11*, 1198. [CrossRef]
7. Dabbagh, M.; Choo, K.-K.-R.; Beheshti, A.; Tahir, M.; Safa, N.S. A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Comput. Secur.* **2021**, *100*, 102078. [CrossRef]
8. Yang, R.; Wakefield, R.; Lyu, S.; Jayasuriya, S.; Han, F.; Yi, X.; Yang, X.; Amarasinghe, G.; Chen, S. Public and private blockchain in construction business process and information integration. *Autom. Construct.* **2020**, *118*, 103276. [CrossRef]
9. Hu, Y.-C.; Jakobsson, M.; Perrig, A. Efficient constructions for one-way hash chains. In Proceedings of the International Conference on Applied Cryptography and Network Security, New York, NY, USA, 7–10 June 2005; pp. 423–441. [CrossRef]
10. Lee, D. Hash function vulnerability index and hash chain attacks. In Proceedings of the 3rd IEEE Workshop on Secure Network Protocols, Beijing, China, 16 October 2007; pp. 1–6. [CrossRef]
11. Kim, S.; Kwon, Y.; Cho, S. A survey of scalability solutions on blockchain. In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 17–19 October 2018; pp. 1204–1207. [CrossRef]
12. Zhou, Q.; Huang, H.; Zheng, Z.; Bian, J. Solutions to scalability of blockchain: A survey. *IEEE Access* **2020**, *8*, 16440–16455. [CrossRef]
13. Nakamoto, S. Bitcoin: A Peer-To-Peer Electronic Cash System Bitcoin. 2009. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 19 November 2023).
14. Zikratov, I.; Kuzmin, A.; Akimenko, V.; Niculichev, V.; Yalansky, L. Ensuring data integrity using blockchain technology. In Proceedings of the 20th Conference of Open Innovations Association (FRUCT), St. Petersburg, Russia, 3–7 April 2017; pp. 534–539. [CrossRef]

15. Zou, W.; Lo, D.; Kochhar, P.S.; Le, X.-B.D.; Xia, X.; Feng, Y.; Chen, Z.; Xu, B. Smart contract development: Challenges and opportunities. *IEEE Trans. Softw. Eng.* **2021**, *47*, 2084–2106. [CrossRef]
16. Aleksieva, V.; Valchanov, H.; Huliyan, A. Implementation of Smart-Contract, Based on Hyperledger Fabric Blockchain. In Proceedings of the 21st International Symposium on Electrical Apparatus & Technologies (SIELA), Bourgas, Bulgaria, 3–6 June 2020; pp. 1–4. [CrossRef]
17. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the 13th EuroSys Conference, Lisbon, Portugal, 23–26 April 2018. [CrossRef]
18. Baird, L.; Luykx, A. The hashgraph protocol: Efficient asynchronous BFT for high-throughput distributed ledgers. In Proceedings of the International Conference on Omni-Layer Intelligent Systems (COINS), Barcelona, Spain, 31 August–2 September 2020; pp. 1–7. [CrossRef]
19. Montaser, M.M.A.; Othman, S.H.; Radzi, R.Z.R.M. Secured Tracking and Tracing System Based on Blockchain Technology. In Proceedings of the International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021; pp. 1–6. [CrossRef]
20. Liu, X.; Barenji, A.V.; Li, Z.; Montreuil, B.; Huang, G.Q. Blockchain-based smart tracking and tracing platform for drug supply chain. *Comput. Ind. Eng.* **2021**, *161*, 107669. [CrossRef]
21. Koyama, A.; Tran, V.C.; Fujimoto, M.; Bao, V.N.Q.; Tran, T.H. A Decentralized COVID-19 Vaccine Tracking System Using Blockchain Technology. *Cryptography* **2023**, *7*, 13. [CrossRef]
22. Mendi, A.F. Blockchain for Food Tracking. *Electronics* **2022**, *11*, 2491. [CrossRef]
23. Attia, O.; Khoufi, I.; Laouiti, A.; Adjih, C. An IoT-blockchain architecture based on hyperledger framework for healthcare monitoring application. In Proceedings of the 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Canary Islands, Spain, 24–26 June 2019; pp. 1–5. [CrossRef]
24. Marbouh, D.; Abbasi, T.; Maasmi, F.; Omar, I.A.; Debe, M.S.; Salah, K.; Jayaraman, R.; Ellahham, S. Blockchain for COVID-19: Review, opportunities, and a trusted tracking system. *Arabian J. Sci. Eng.* **2020**, *45*, 9895–9911. [CrossRef] [PubMed]
25. Leng, Z.; Tan, Z.; Wang, K. Application of Hyperledger in the Hospital Information Systems: A Survey. *IEEE Access* **2021**, *9*, 128965–128987. [CrossRef]
26. Damgård, I.B. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*; Springer: New York, NY, USA, 1989.
27. Oyinloye, D.P.; Teh, J.S.; Jamil, N.; Alawida, M. Blockchain Consensus: An Overview of Alternative Protocols. *Symmetry* **2021**, *13*, 1363. [CrossRef]
28. Pervez, H.; Muneeb, M.; Irfan, M.U.; Haq, I.U. A comparative analysis of DAG-based blockchain architectures. In Proceedings of the 12th International Conference on Open Source Systems and Technologies (ICOSST), Lahore, Pakistan, 19–21 December 2018; pp. 27–34. [CrossRef]
29. Müller, S.; Penzkofer, A.; Polyanskii, N.; Theis, J.; Sanders, W.; Moog, H. Tangle 2.0 Leaderless Nakamoto Consensus on the Heaviest DAG. *IEEE Access* **2022**, *10*, 105807–105842. [CrossRef]
30. Verma, A.; Bhattacharya, P.; Madhani, N.; Trivedi, C.; Bhushan, B.; Tanwar, S.; Sharma, G.; Bokoro, P.N.; Sharma, R. Blockchain for industry 5.0: Vision, opportunities, key enablers, and future directions. *IEEE Access* **2022**, *10*, 69160–69199. [CrossRef]
31. Birman, K. The promise, and limitations, of gossip protocols. *SIGOPS Oper. Syst. Rev.* **2007**, *41*, 8–13. [CrossRef]
32. Santiago, C.; Ren, S.; Lee, C.; Ryu, M. Concordia: A Streamlined Consensus Protocol for Blockchain Networks. *IEEE Access* **2021**, *9*, 13173–13185. [CrossRef]
33. Guegan, D. *Public Blockchain versus Private Blockchain*; Working Document Center Economics Pantheon-Sorbonne University: Paris, France, 2017.
34. Huang, D.; Ma, X.; Zhang, S. Performance analysis of the raft consensus algorithm for private blockchains. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 172–181. [CrossRef]
35. Fan, C.; Ghaemi, S.; Khazaei, H.; Musilek, P. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access* **2020**, *8*, 126927–126950. [CrossRef]
36. Wang, M.; Duan, M.; Zhu, J. Research on the security criteria of hash functions in the blockchain. In Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts (BCC), Incheon, Republic of Korea, 4 June 2018; pp. 47–55.
37. Liu, X.L.; Wang, W.M.; Guo, H.; Barenji, A.V.; Li, Z.; Huang, G.Q. Industrial blockchain based framework for product lifecycle management in industry 4.0. *Robot. Comput. Integr. Manuf.* **2020**, *63*, 101897. [CrossRef]
38. Bertino, E.; Sandhu, R. Database security—Concepts approaches and challenges. *IEEE Trans. Depend. Sec. Comput.* **2005**, *2*, 2–19. [CrossRef]
39. Maurer, U. Modelling a public-key infrastructure. In Proceedings of the Computer Security—ESORICS 96: 4th European Symposium on Research in Computer Security, Rome, Italy, 25–27 September 1996; pp. 325–350. [CrossRef]
40. Salahdine, F.; Kaabouch, N. Social engineering attacks: A survey. *Future Internet* **2019**, *11*, 89. [CrossRef]
41. Gauravaram, P. Security analysis of salt||password hashes. In Proceedings of the 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuala Lumpur, Malaysia, 26–28 November 2012; pp. 25–30. [CrossRef]

42. Chivers, I.; Sleightholme, J. An introduction to algorithms and the big O notation. In *Introduction to Programming with Fortran*; Springer: Cham, Switzerland, 2015; pp. 359–364. [CrossRef]
43. Gayathri Santhosh, M.; Reshmi, T. Enhancing PKI Security in Hyperledger Fabric with an Indigenous Certificate Authority. In Proceedings of the IEEE International Conference on Public Key Infrastructure and its Applications (PKIA), Bangalore, India, 8–9 September 2023; pp. 1–5. [CrossRef]
44. Wagner, D.; Schneier, B. Analysis of the SSL 3.0 protocol. In Proceedings of the 2nd USENIX Workshop on Electronic Commerce Proceedings, Oakland, CA, USA, 18–20 November 1996; Volume 1, pp. 29–40. [CrossRef]
45. Anderson, C. Docker [Software Engineering]. *IEEE Softw.* **2015**, *32*, 102-c3. [CrossRef]
46. Ubuntu 20.04. Available online: https://releases.ubuntu.com/focal/ (accessed on 19 November 2023).
47. The Go Programming Language. Available online: https://go.dev/ (accessed on 19 November 2023).
48. Cadvisor. Available online: https://github.com/google/cadvisor (accessed on 19 November 2023).
49. Tolaram, N. Cadvisor. In *Software Development with Go: Cloud-Native Programming Using Golang with Linux and Docker*; Tolaram, N., Ed.; Apress: Berkeley, CA, USA, 2022; pp. 347–376. [CrossRef]
50. Prometheus. Available online: https://prometheus.io/ (accessed on 19 November 2023).
51. Chakraborty, M.; Kundan, A.P. Grafana. In *Monitoring Cloud-Native Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 187–240.
52. Grafana. Available online: https://grafana.com/ (accessed on 19 November 2023).
53. Gilbert, H.; Handschuh, H. Security analysis of SHA-256 and sisters. In Proceedings of the International Workshop on Selected Areas in Cryptography, Ottawa, ON, Canada, 14–15 August 2003; pp. 175–193.
54. Chauhan, A.; Malviya, O.P.; Verma, M.; Mor, T.S. Blockchain and scalability. In Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, Portugal, 16–20 July 2018; pp. 122–128. [CrossRef]
55. Zou, K.H.; Tuncali, K.; Silverman, S.G. Correlation and simple linear regression. *Radiology* **2003**, *227*, 617–628. [CrossRef]
56. M'Raihi, D.; Machani, S.; Pei, M.; Rydell, J. TOTP: Time-Based One-Time Password Algorithm. Wilmington, DE, USA. 2011. Available online: https://tools.ietf.org/html/rfc6238 (accessed on 17 November 2023).