

Article

Enhancing Data Preservation and Security in Industrial Control Systems through Integrated IOTA Implementation

Iuon-Chang Lin ^{1,*} , Pai-Ching Tseng ^{2,3}, Pin-Hsiang Chen ¹ and Shean-Juinn Chiou ⁴

¹ Department of Management Information Systems, National Chung Hsing University, Taichung 40277, Taiwan; benson0714@gmail.com

² Ph.D. Program of Business, Feng Chia University, Taichung 40724, Taiwan; tpcp630@gmail.com

³ Natures Bank Exchange Co., Ltd., Taichung 40724, Taiwan

⁴ Department of Mechanical Engineering, National Chung Hsing University, Taichung 40227, Taiwan; sjchiou@dragon.nchu.edu.tw

* Correspondence: iclin@nchu.edu.tw

Abstract: Within the domain of industrial control systems, safeguarding data integrity stands as a pivotal endeavor, especially in light of the burgeoning menace posed by malicious tampering and potential data loss. Traditional data storage paradigms, tethered to physical hard disks, are fraught with inherent susceptibilities, underscoring the pressing need for the deployment of resilient preservation frameworks. This study delves into the transformative potential offered by distributed ledger technology (DLT), with a specific focus on IOTA, within the expansive landscape of the Internet of Things (IoT). Through a meticulous examination of the intricacies inherent to data transmission protocols, we present a novel paradigm aimed at fortifying data security. Our approach advocates for the strategic placement of IOTA nodes on lower-level devices, thereby streamlining the transmission pathway and curtailing vulnerabilities. This concerted effort ensures the seamless preservation of data confidentiality and integrity from inception to storage, bolstering trust in the convergence of IoT and DLT technologies. By embracing proactive measures, organizations can navigate the labyrinthine terrain of data management, effectively mitigate risks, and cultivate an environment conducive to innovation and progress.

Keywords: DLT; IoT; data security; Docker; container technology; IOTA; Tangle



Citation: Lin, I.-C.; Tseng, P.-C.; Chen, P.-H.; Chiou, S.-J. Enhancing Data Preservation and Security in Industrial Control Systems through Integrated IOTA Implementation.

Processes **2024**, *12*, 921. <https://doi.org/10.3390/pr12050921>

Academic Editor: Sergey Y. Yurish

Received: 29 March 2024

Revised: 27 April 2024

Accepted: 28 April 2024

Published: 30 April 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the emergence of Industry 4.0, various innovative industrial technologies, including MES or AI technology, have gained attention. Traditional factories have gradually transformed into intelligent factories, and the IoT also plays an important role in it. As IoT devices generate a large amount of data, security and integrity that satisfy data preservation requirements are increasingly valued. Data preservation technology has become indispensable. Currently, most of the models or technologies are implemented using encryption technology. Generally, a trusted third party is required to store the data, which may lead to data leakage or attack. In recent years, the DLT has developed vigorously. It can ensure the security and integrity of data to satisfy data preservation requirements on the chain and solve the problem of a single point of failure (SPOF) in the system. Currently, some methods have emerged to combine with DLT to save data. However, most of the methods focus on replacing the original third-party data storage and only uploading the data to DLT, ignoring the risks that may occur during the uploading process. In the existing method [1,2], the author uses IOTA as the DLT for storing data in the IoT. IOTA uses a tangle network based on Directed Acyclic Graph (DAG) technology, which is different from traditional DLTs. It does not require transaction fees and has faster transaction speeds. It is often used in industrial environments. In the IoT network, data are received through sensors, computers, and servers, and finally uploaded to the node of the DLT. Complicated paths may lead

malicious attackers to use devices in the transmission path to attack, thereby destroying the integrity of the data.

- In an Industry 4.0 environment, a manufacturing execution system (MES) is a comprehensive dynamic software system that ensures production quality. It can help enterprises monitor, track, record, and control the data generated in the manufacturing process, from receiving orders, production, and process control to products.
- According to the MESA model [3] proposed by MESA, “data collection and acquisition”, and “product tracking and historical records” are important components of MES. On the traditional MES, the historical record of the product will upload the data to the database in the system. However, the data stored in the database may be hijacked by hackers, and the tampered data may greatly affect the judgment of decision-makers or the accuracy of AI training models. Therefore, maintaining data integrity is a major challenge for enterprises in terms of information security.

In a blockchain network, the author Satoshi Nakamoto [4] divided the network into two roles: miners and users. Miners consume a significant amount of computing power to provide proof of work (POW) to connect blocks. This reward structure poses a significant obstacle in the machine-to-machine economy because small payments between machines may be less than the cost of payment required. In IOTA, there is no difference between miners and users, and all nodes can participate in consensus. The person who initiates the transaction performs lightweight proof of work, and the transaction must be verified by other people before it can be uploaded. Therefore, the more users there are, the faster the verification speed, and the better the efficiency. In contrast, the performance of DLT deteriorates as the number of transactions increases. In industrial environments with a large number of IoT devices and a focus on efficiency, IOTA is more suitable than traditional blockchain. However, IOTA has some weak points, which are described below:

1. The size of each data element in IOTA cannot exceed 32 kb. Therefore, uploading pictures or videos may not be possible due to the size of the data.
2. IOTA does not provide an access control system. That is, anyone can access all data on the IOTA Tangle network, which may lead to a loss of confidentiality, one of the three elements of information security.

DLT can ensure the integrity of data on the ledger, but it still needs to store an index pointing to the location of the data. For example, after uploading data to IOTA, a message ID is generated that corresponds to the location of the data stored on IOTA. If this message ID is stored in a local database, there is a risk of substitution. If the message ID is replaced with a malicious one, the integrity of the data cannot be verified.

Man-in-the-middle attacks, a common attack method, such as ARP spoofing, DNS spoofing, IP spoofing, etc., are common attack methods. The traditional information flow involves data passing through sensors, edge computers, and servers, and finally being uploaded to a distributed ledger. Although this is easier to manage, it makes the transmission path very complex and increases the possibility of data being tampered with during transmission. The objective of this paper is as follows:

- Ensure data preservation for data stored on the server.
- Ensure integrity and confidentiality of data transmitted from sensors to the server and IOTA.
- Successfully upload data exceeding IOTA storage capacity.
- Ensure the integrity of data stored on the server and detect any tampered data.

The main contributions of our architecture are as follows:

- The data will be uploaded to IOTA, and the immutable nature of DLT will ensure the integrity of the data after they are uploaded to IOTA.

- Use containerization technology [5] to set up IOTA nodes, which can reduce the difficulty of setting up nodes, because there is a significant number of different hardware devices in the IoT environment.
- Set up the IOTA node on the Raspberry Pi, upload the data to Tangle after the sensor receives the data, and successfully reduce the transmission path before uploading to the DLT.
- By utilizing the method we proposed, the data will be preprocessed to ensure its integrity during transmission and storage on the server, thus satisfying data preservation requirements in the IoT environment.

2. Related Works

2.1. Distributed Ledger Technology

IoT covers a significant number of hardware devices, and there are many different communication paths between IoT devices. To protect the communication security between IoT devices and Tangle, many studies have proposed methods to detect and avoid data attacks [6]. However, the risk of data being attacked on the transmission path is still unavoidable, so how to reduce the transmission path is the key issue to address.

2.2. Transmission Path Selection from Sensors to DLT

There are three architectures for sending data from the sensor to Tangle in the industrial environment with IoT devices. This is inspired by W. F. Silvano and R. Marcelino [7]. The transmission paths are divided into three types:

Architecture (i): Set up the IOTA node on the server: After the data are read by the sensors, send them to the computer, use the computer to process the data and then transfer them to the server or database, collect the data centrally, and then upload the data to Tangle.

Architecture (ii): Set up the IOTA node on the computer. After the data read by the sensor are sent to the computer, the computer will upload the processed data to the tangle through the IOTA node.

Architecture (iii): Set up the IOTA node on Raspberry Pi. After the sensor sends a signal to Raspberry Pi, it directly uploads data to Tangle.

Architecture (i) will be used in fields that require a significant amount of preprocessing, which can greatly reduce transmission time. Our architecture usually exists in today's smart field, where data can be centrally managed and uploaded. However, data may be lost or hacked during the transmission process, or a single point of failure may occur during the transmission process. Architecture (ii) is more secure than the previous architecture, as the computer receives the data and processes them before uploading. Architecture (iii) is the most ideal architecture in the intelligent factory, which can directly upload data to Tangle, and can also avoid the loss of transmission data caused by SPOF.

2.3. IOTA

When the traditional DLT is applied to the IoT, the increase in the number of transactions may lead to the consumption of handling gas fees and scalability issues. The author Popov [8] proposed that Tangle solves the problem of gas fee and scalability. The traditional DLT needs to use miners to verify each transaction to propose a block and connect each block to form a DLT structure like a linked list. Tangle, as proposed by the author, belongs to the mesh DLT structure. It can add new transaction blocks from any direction, improving transaction speed and ensuring its scalability. The consensus protocol adopted by Tangle is proof of work (PoW). When a transaction occurs, the node we set up will need to verify two transactions on Tangle before requesting other nodes to verify the transaction we sent. Its characteristic is that more users can make the transaction faster.

IOTA is a distributed ledger technology specifically created for the Internet of Things (IoT) ecosystem. In contrast to traditional blockchain-based systems, IOTA utilizes a directed acyclic graph (DAG) structure, known as Tangle, to handle transactions. Tangle enables the concurrent processing of transactions and does not require miners to vali-

date transactions, resulting in a more scalable and energy-efficient solution compared to blockchain-based systems.

IOTA, like other blockchains or distributed ledgers, requires nodes to communicate. IOTA nodes can communicate with Tangle and act as validators to verify other people's transactions. When uploading data to Tangle, the data are first verified by IOTA nodes with two transactions on Tangle and then uploaded to Tangle via IOTA nodes.

Each node in the IOTA Tangle maintains a local database, or ledger, which records all transactions and balances. The ledger is distributed among nodes in the network, meaning that nodes share their copy of the ledger with other nodes. This distributed sharing of the ledger makes the ledger a "distributed ledger".

2.4. InterPlanetary File System

The InterPlanetary File System (IPFS) is a decentralized peer-to-peer (P2P) system designed for storing and sharing files. IPFS operates by utilizing a content-addressed storage system that identifies content using its unique hash. With its ability to offer a decentralized and secure solution for file storage, IPFS has become increasingly popular in recent years as an alternative to traditional centralized storage systems.

Some researchers have explored the use of IPFS in various applications. For example, V. Mani, P. Manickam, Y. Alotaibi, S. Alghamdi, and O. I. Khalaf [9] proposed an IPFS-based solution for storing and sharing electronic health records (EHRs). They demonstrated that IPFS can provide a secure and efficient way to store and share sensitive health data while preserving patients' privacy.

In the IoT environment, real-time data reception and transmission with data integrity assurance are essential. Therefore, the authors Alsbou et al. [10] proposed the mobile-agent distributed intelligence Tangle-based approach (MADIT) to address the challenges of massive data transmission and efficiency in IoT environments. They utilized the IOTA masked authenticated messaging (MAM) protocol to ensure data privacy on Tangle. While data on the ledger are publicly transparent, there may be a need to upload sensitive data to the ledger. Hence, the authors Zhang et al. [11] proposed LDP, which uploads data to the distributed ledger technology (DLT) while preserving data confidentiality. Depending on different contexts, there may be a need to upload varying sizes of data that could exceed the single transaction limit of the ledger. Therefore, the authors J. Jayabalan and N. Jeyanthi [12] proposed a model that encrypts medical data and stores them on IPFS, while storing the index generated by IPFS on the DLT, ensuring both data integrity and confidentiality.

2.5. Cipher Feedback

Cipher Feedback (CFB) is a mode of operation for block ciphers that allows for the encryption of plaintext data of any length. CFB is a widely used mode of operation due to its security and ease of implementation.

Figure 1 shows that CFB is a block cipher mode that operates similarly to CBC (Cipher Block Chaining), in that the previous ciphertext block is used in the encryption of the current block. Like CBC, CFB uses an initialization vector. However, the key difference is that, in the CFB mode, the previous ciphertext block is encrypted first, and then XOR-ed with the plaintext block to produce the ciphertext block for the current iteration.

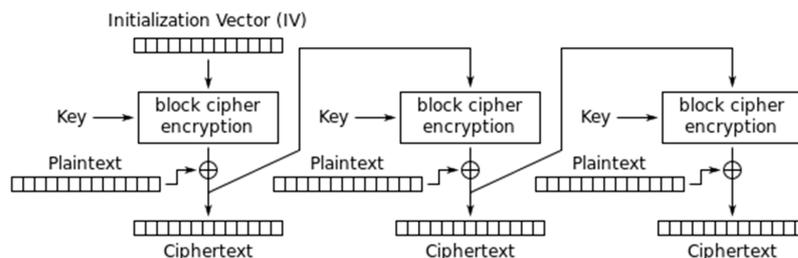


Figure 1. Cipher Feedback (CFB) mode encryption.

2.6. Summary

We were inspired to propose this architecture based on the related works mentioned above. The entire process is depicted in Figure 2. Firstly, I conducted research on distributed ledger technology (DLT) and found it to be highly suitable for use in the context of intelligence factories, as it can ensure the integrity of the stored data. Therefore, I chose IOTA to implement this technology because it can upload data to the blockchain in real time. However, I encountered a limitation with IOTA, as it cannot store data exceeding 3 MB. To address this challenge, I explored the InterPlanetary File System (IPFS) to store large datasets and then stored the hash value of the data in IOTA, effectively resolving the data storage issue while ensuring data integrity.

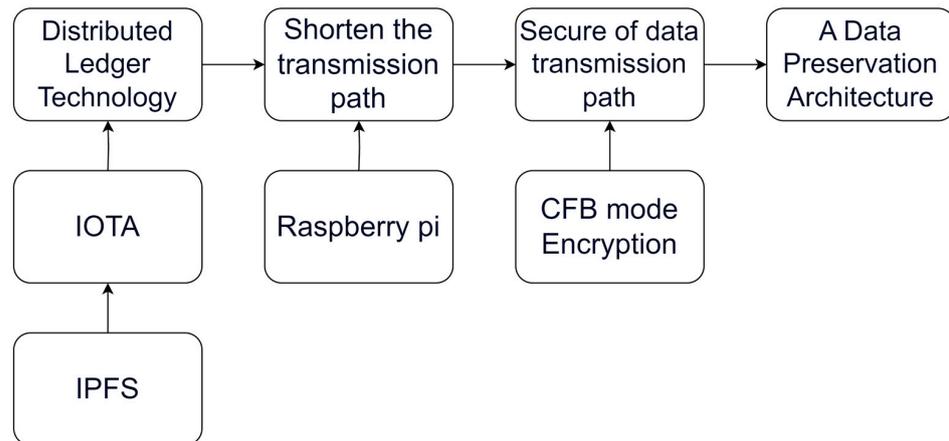


Figure 2. The technique relationships of our proposed architecture.

Furthermore, I studied the data transmission path method proposed by W.F. Silvano and R. Marcelino [7]. I learned that minimizing the data transmission path is crucial for ensuring data security. Therefore, I deployed IOTA nodes on Raspberry Pi to enable sensors to transmit data to IOTA via the shortest path.

In terms of data transmission security, I utilized CFB Encryption to encrypt the data. This encryption method not only ensures data integrity but also guarantees data confidentiality.

By integrating all these methods, I developed a data preservation architecture. It ensures both the confidentiality of data during transmission and the integrity of data storage.

3. Proposed Method

3.1. System Architecture

We categorize data into confidential and non-confidential data. Confidential data include information that should not be accessible to others, such as official documents, authorization contracts, technical cooperation agreements, etc. Non-confidential data, on the other hand, include data opposite to confidential data, such as raw data generated by machines, publicly available company data, etc. Based on these two types of data, we provide different data preservation architectures.

The proposed system architecture and method extended the previous work [13]. It ensures the integrity of the data before and after uploading to Tangle. It allows for selective confidentiality based on the sensitivity of the data. Before uploading, we have developed an algorithm based on CFB encryption for data preprocessing, which securely stores the original data locally and verifies data integrity using DLT. We utilize containerization technologies to establish nodes locally, minimizing the transmission path to Tangle, which can reduce the risk of attacks. Furthermore, our proposed architecture does not require significant computational capabilities from IoT devices in resource-constrained IoT environments. We also proposed a method to address the issue of large data that cannot be stored directly in IOTA. By combining IPFS and IOTA, data can be uploaded without being limited by the single transaction capacity of IOTA while ensuring data integrity. We have

chosen IOTA as our DLT of choice, as it offers efficient transaction verification compared to Ethereum and can meet the real-time data exchange requirements in IoT environments. The key factors to ensure data integrity are as follows:

1. Using DLT: The DLT and various consensus mechanisms to maintain the operation of the entire ledger. By utilizing the tamper-evident nature of DLT, it can successfully guarantee the integrity of data after they are uploaded to the chain.
2. Reducing the transmission path before uploading to the DLT: Taking the industrial control field as an example, after data are uploaded to the DLT, the DLT can ensure the integrity of the data. However, before uploading to the DLT, the data will first go through sensors, edge computers, and servers. The more transmission paths that the data goes through, the higher the possibility of intrusion. Therefore, reducing the transmission path before uploading to the DLT is a key factor to ensure data integrity.

3.2. Shorten the Transmission Path

To shorten the transmission path, we need to set up an IOTA node on Raspberry Pi, which can be more challenging compared to installing it on a regular desktop computer because Raspberry Pi uses an arm64 CPU architecture (Sony's UK manufacturing plant, Pencoed, South Wales), which cannot directly install the official packages designed for the amd64 architecture. Therefore, this article utilizes containerization technology to address this issue.

Figure 3 demonstrates how to install the IOTA node on Raspberry Pi. The main process is as follows:

Step 1: Create container: install docker on Raspberry Pi and create a container by docker. This allows us to use containerization technology on our devices.

Step 2: Setup IOTA node: set up the IOTA node by using packages on the IOTA website and modifying the configuration file. The main purpose of modifying the configuration file is to increase the surrounding neighbor nodes, which makes it synchronize with Tangle. Many other configuration files can be modified.

Step 3: Connect the sensor to Raspberry Pi: connect sensors (CO₂ sensor, water sensor, temperature sensor, etc.) to Raspberry Pi, and design a program to collect data from the sensor automatically.

Step 4: Design a program: Design a program for uploading data to the tangle.

Step 5: Activate the IOTA node: enable programs that collect data and upload data simultaneously.

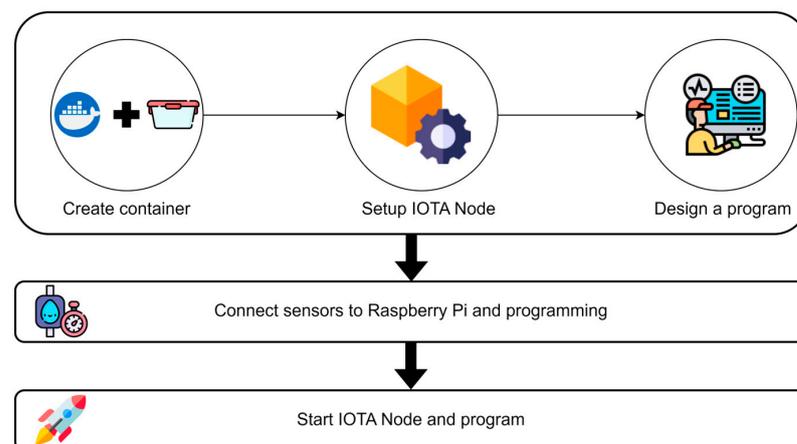


Figure 3. A flowchart of creating and setting up an IOTA node in a container.

3.3. Proposed System Model

This section presents a confidential data transmission architecture for uploading data from IoT sensors to IOTA Tangle, involving four participants: Raspberry Pi with sensors,

DLT, server, and data user. As shown in Figure 4, sensors will transmit the raw data to Raspberry Pi in real time, and the hash value of the data will be sent to IOTA through containerization technology for subsequent verification. Meanwhile, the data will be encrypted or preprocessed based on their type using the algorithm proposed in this study. Finally, before the data are used by the data user, they will undergo integrity verification through IOTA and the algorithm proposed in this study to ensure that the data have not been tampered with.

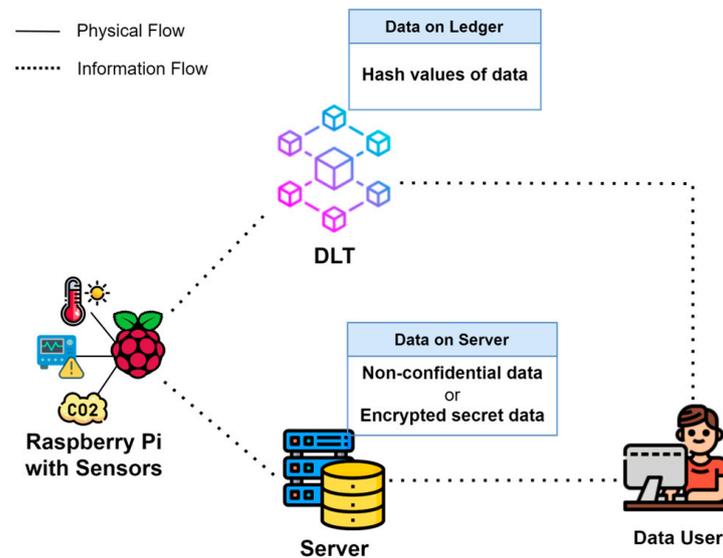


Figure 4. Proposed system architecture to ensure data integrity.

Figure 5 illustrates the detailed process of uploading non-confidential data to the server and IOTA. To prevent data from being stolen in the public Tangle network, the original data are hashed before being uploaded to IOTA. Figure 6 is the method based on Algorithm 1, Figure 7 shows the process of data retrieval by the data user. Using the message ID returned by IOTA, the data are preprocessed using the algorithm proposed in this study (Algorithm 1) and then uploaded to the server. When the data user wants to access the data, the stored message ID in the server undergoes integrity verification using the verification algorithm (Algorithm 2) proposed in this study. Upon successful integrity verification of the message ID, the hash value of the data is retrieved from IOTA using the message ID for further verification, ensuring that the accessed data have not been tampered with.

Algorithm 1 A method to hash data based on CFB

```

1: Input:  $IV T, HDM$ 
2: Output:  $HDML$ 
3: /* The output will return a list consisting of the hashed  $D$  and the hashed  $msgID$ .*/
4:  $PD \leftarrow$  Search previous cipher data on raspberry pi
5: if  $PD \neq null$  or  $timeout \leq t$  then
6:    $CT \leftarrow H(PD) \oplus HDM$ 
7:    $HDML.append(CT)$ 
8:   return  $HDML$ 
9: else
10:  Generate  $IV T$  with timestamp
11:   $CT \leftarrow H(PD) \oplus IV T$ 
12:   $HDML.append(CT)$ 
13:  return  $HDML$ 
14: end if

```

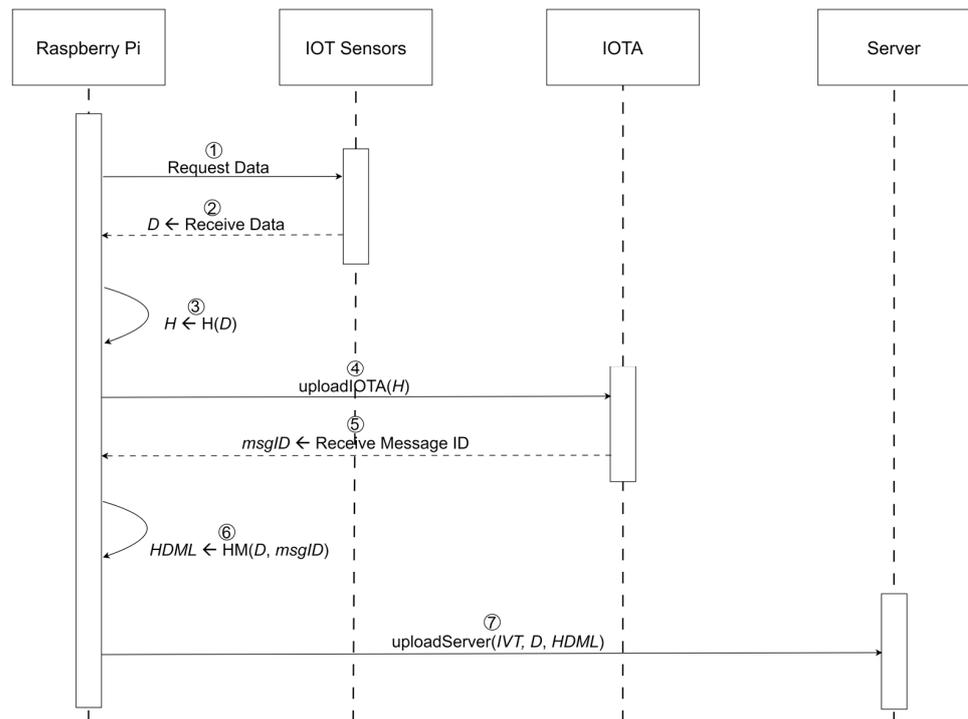


Figure 5. Sequence diagram for non-confidential data upload.

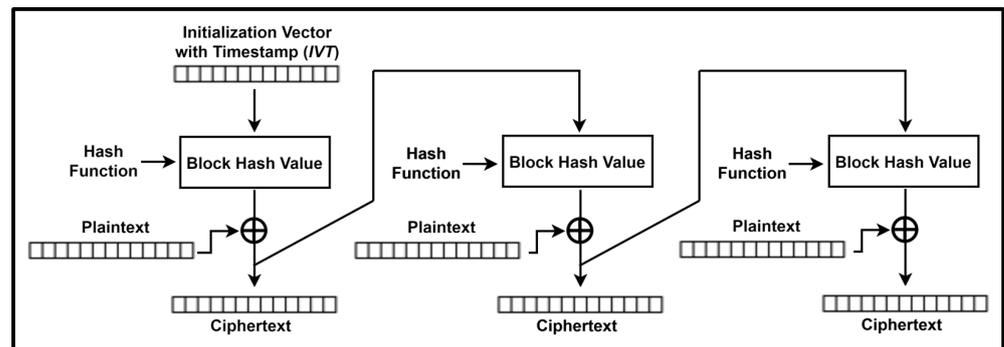


Figure 6. Proposed method based on CFB encryption.

Algorithm 2 Verify integrity of msgID

```

1: Input:  $BL$ ,  $HDML$ 
2: Output: true or false
3: /*The result of return true indicates that the data are correct and has not been tampered with.
false indicates that the data has been maliciously modified.*/
4:  $FD \leftarrow H(BL[0]_{IVT} \oplus BL[0]_{msgID} \oplus BL[0]_D)$  //First Data
5: if  $FD \neq HDML[0]$  then
6:   Data has been tampered with someone
7:   return false
8: end if
9:  $n \leftarrow$  number of data in  $BL$ 
10: for  $i = 1$  to  $n - 1$  do
11:   if  $H(BL[i - 1]_D \oplus BL[i]_{msgID} \oplus BL[i]_D) \neq HDML[i]$  then
12:     Data has been tampered with someone
13:     return false
14:   end if
15: end for
16: return true

```

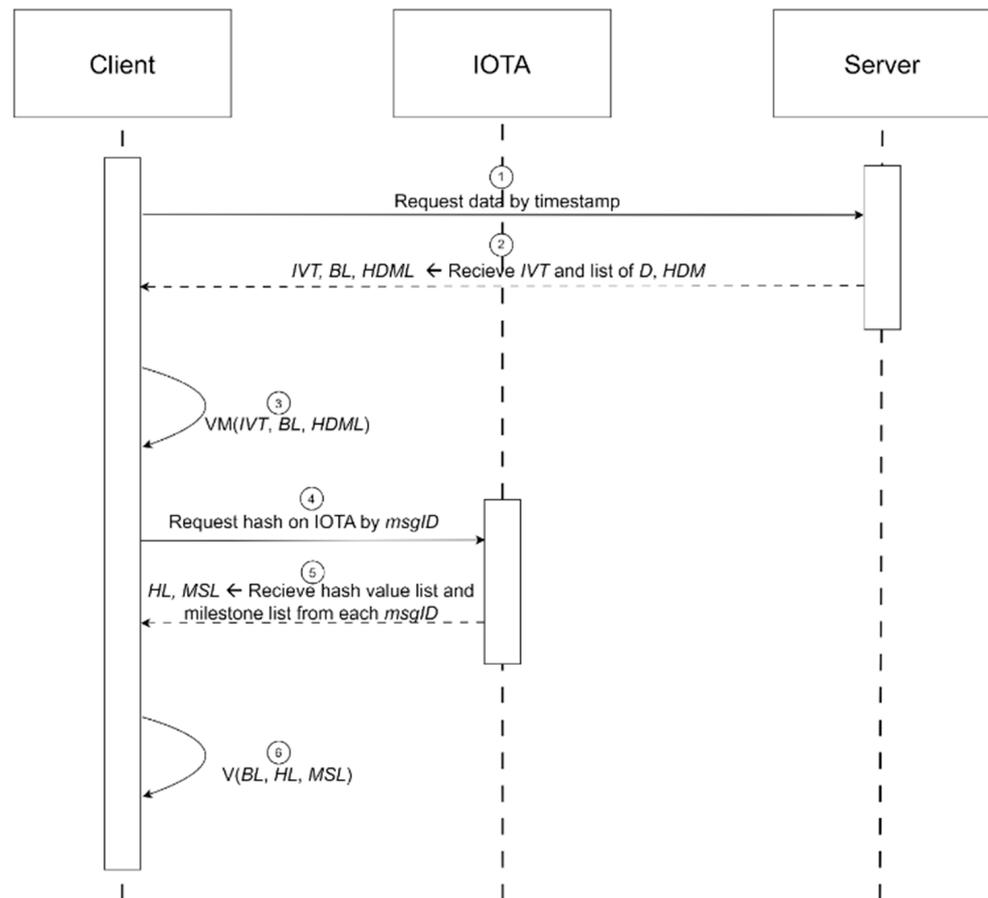


Figure 7. Sequence diagram for non-confidential data retrieval.

Table 1 shows the list of symbols used in the proposed architecture. This paper proposes an architecture that includes two different data processing methods, which are designed for non-confidential data and confidential data, respectively.

Algorithm 3 Verify data integrity

```

1: Input:  $BL, HL, MSL$ 
2: Output: true or false
3: /*The result of return true indicates that the data is correct and has not been tampered with.
false indicates that the data has been maliciously modified.*/
4:  $n \leftarrow$  number of data in  $BL$ 
5: for  $i = 0$  to  $n - 1$  do
6:   if  $H(BL[i]_D) \neq HL[i]$  then
7:     Data have been tampered with someone
8:   return false
9: else if  $MSL[i - 1] \geq MSL[i]$  then
10:  Data have been tampered with someone
11:  return false
12:  end if
13: end for
14: return true
  
```

Table 1. Definition of notations.

Notation	Definition
D	Data receive from IOT sensors
H	Hash value of D
$msgID$	The location of data that save on IOTA
IVT	The initialization vector with timestamp
$H(\cdot)$	A secure one-way hash function
HM	The method we proposed based on CFB (Algorithm 1)
HDM	A hash value of $D \oplus msgID$
BL	The list with a bundle of $D, msgID, IVT$
HDML	The hash list of HDM
$VM(\cdot)$	The algorithm to verify integrity of $msgID$ in DL (Algorithm 2)
HL	The hash list corresponds to each D in DL
MSL	The milestone list on IOTA tangle
$V(\cdot)$	The algorithm to verify integrity of D (Algorithm 3)
$E(\cdot)$	A encrypt algorithm for secret data (Algorithm 4)
key	Secret key generated by AES
CT	Ciphertext return by Algorithm 4
P	Plaintext after CFB Decryption, which is $D \oplus msgID$
CL	The encrypted cipher list with $D, msgID, IVT$

Algorithm 4 Encrypt data based on CFB

```

1: Input:  $D, msgID, key$ 
2: Output:  $CT$ 
3: /* Return the ciphertext( $CT$ ) generated by using CFB encryption and the initialization vector
with timestamp( $IVT$ ).*/
4:  $PD \leftarrow$  Search previous encrypted data on raspberry pi
5: if  $PD \neq null$  or  $timeout \leq t$  then
6:    $CT \leftarrow E_{key}(PD) \oplus (D \oplus msgID)$ 
7:   return  $CT$ 
8: else
9:   Generate  $IVT$  with timestamp
10:   $CT \leftarrow E_{key}(IVT) \oplus (D \oplus msgID)$ 
11: return  $CT, IVT$ 
12: end if

```

3.4. Non-Confidential Data Upload Method

This section provides a detailed explanation of the general data transmission path and the preprocessing steps that data need to undergo. Figure 5 illustrates the data transmission path when data are transmitted from IoT sensors to the server. The processing of data on Raspberry Pi only requires the use of hash and exclusive or (XOR) operations and does not require a significant amount of computing resources, making it suitable for IoT environments with resource-constrained. After preprocessing, data preservation is ensured on the server.

The entire process of uploading non-confidential data involves four actors: Raspberry Pi, IoT sensors, IOTA, and server. Raspberry Pi is responsible for receiving and transmitting information, the IoT sensors receive various signals, IOTA verifies the integrity of the data, and the server is used to store data and $msgID$. The entire process is divided into seven steps:

Step 1–2: Raspberry Pi collects raw data from sensors using a program.

Step 3: The original data are hashed to obtain the hash value of the data, to prevent data theft when uploading to IOTA.

Step 4–5: The hash value of the raw data is uploaded to IOTA and the returned *msgID* is obtained.

Step 6–7: Algorithm 1 is used to preprocess the *msgID*, and the preprocessed data are uploaded to the server, ensuring the integrity of the data and *msgID* stored on the server.

Before uploading data to the server, Algorithm 1 is applied for pre-processing to ensure data integrity on the server. Algorithm 1 is based on the method shown in Figure 6, which first randomly generates an Initialization Vector with Timestamp (*IVT*), hashes the *IVT* using a hash function, and then uses XOR to create a hash value with the raw data. The previous ciphertext is repeatedly hashed using a hash function and then using XOR with the plaintext. These steps can allow for all data to have relatedness. If any data is tampered with, the modified data can be easily identified during verification. To avoid spending a significant amount of time verifying data, a timeout is set in advance on the third line of Algorithm 1, with new *IVT* is generated periodically and preventing data from forming excessively long chains, which can make data verification more efficient. In lines 4 and 8 of Algorithm 1, the hash function and XOR operation are used to make the hash values of each data item correlate. This is to identify any tampering of the data. Hash and XOR operations do not require significant computational power. Therefore, this method can be used for preprocessing data when dealing with large amounts of data that require timely processing, and it is suitable for execution in resource-constrained IoT environments after preprocessing all the data before the timeout. They will be concatenated into a message chain, and all the cipher texts will be stored in a list, which is the *HDML*.

3.5. Non-Confidential Data Retrieval Method

This section proposes the method that data users need to use when accessing data so that data users will verify the integrity of the data before accessing them to ensure that the data have not been tampered with. Figure 7 illustrates the entire process of accessing data, which will use two different verification algorithms to verify the integrity of the data. The entire process is divided into six steps:

Steps 1–2: Use the timestamp on the *IVT* to select which data to access, and then request the entire data from the server with a specific timestamp. The obtained data include *IVT*, *BL*, and *HDML*. *IVT* is a random number that contains a timestamp; *BL* is a list that contains *D*, *msgID*, and *IVT*; and *HDML* is a list that contains *HDM*. *HDM* is the hash value of $D \oplus msgID$.

Step 3: Verify the integrity of *msgID* using Algorithm 2 to prevent hackers from replacing the *msgID* and leading data users to the wrong location to search for the hash value on the IOTA and return true or false after the verification is completed, wherein true indicates successful verification and false indicates verification failure.

Steps 4–5: Obtain the hash value of the entire data and its corresponding milestone on the IOTA using the *msgID* that has integrity. Verify whether the milestone indicates whether the data have been uploaded in order, or whether the *msgID* has been replaced by a malicious user.

Step 6: Perform integrity verification on the data using the verification method proposed in Algorithm 3 and return true or false after the verification is completed, wherein true indicates successful verification and false indicates verification failure.

In step 3 of Figure 7, Algorithm 2 is used to verify the integrity of the *msgID* stored on the server. In the second line, the first data's hash value is calculated using the first *IVT*, *msgID*, and *D* in *BL*, and then compared with the first data's hash value in *HDML* to determine the integrity of the first data. The loop in line 8 will compare all remaining *msgID* or data in *BL* to verify whether they are correct. Finally, true or false will be returned,

wherein true indicates that the *msgID* has not been tampered with, while false indicates that the data have been tampered with.

In Step 6 of Figure 7, Algorithm 3 is used to verify the integrity of the data by comparing them with the hash value on the IOTA Tangle. In the conditional statement in line 4, the algorithm first compares the hash value on the IOTA Tangle with the local hash value of the data to verify that the two data items are consistent. In the conditional statement in line 7, leveraging the characteristics of DLT, the milestone generated by the transaction uploaded to the IOTA Tangle earlier will always be smaller than that of the later ones. Therefore, this algorithm verifies whether the milestones on the IOTA Tangle are sorted in order. If there is an issue of milestones being out of order, this means that the *msgID* has been replaced, and the data do not have integrity. Finally, this algorithm will return whether the data have been tampered with. If the data have been tampered with, it will return false; otherwise, it will return true.

3.6. Confidential Data Upload Method

This section provides a detailed explanation of the encryption method required for uploading confidential data, which incorporates the concept of CFB encryption. By concatenating all the data, the method ensures the integrity of the data stored on the server. Although this method requires more computational power than the previously proposed method for uploading non-confidential data, it ensures that the data can only be accessed by those with the key, ensuring the confidentiality of the data. This study has written the encryption method in Algorithm 4 to provide a data preservation method on the server side.

Figure 8 shows the entire process of uploading confidential data, involving four actors: Raspberry Pi, IoT sensors, IOTA, and server. Raspberry Pi is used for receiving and transmitting information, IoT sensors are used for receiving various signals, IOTA is used for verifying the integrity of the data, and the server is used for storing the data and *msgID*. The entire process includes seven steps:

Step 1–2: Request data from sensors and return the received signal from the sensors to Raspberry Pi.

Step 3: Convert the data into a hash value, which is *H*, by using a hash function.

Step 4–5: Upload *H* to IOTA to ensure that the data cannot be stolen from the public ledger. After uploading *H*, receive the *msgID* returned by IOTA.

Step 6: Use Algorithm 4 proposed in this study to encrypt the data using CFB encryption. This results in a ciphertext generated within the timeout, which is *CT*.

Step 7: Transmit the *D* and *CT* which are generated by Algorithm 4 to the server and store them in the corresponding message chain using the timestamp of the *IVT*.

In step 6 of Figure 7, Algorithm 4 is used to encrypt the original data using CFB encryption. The algorithm requires an AES key to encrypt all of the data. Due to the nature of CFB, each ciphertext is related to the previous data, making it difficult for hackers to tamper with the data. In line 4, a timeout is used to prevent the message chain formed by CFB encryption from becoming too long. If the time does not exceed the timeout, the previous data are encrypted, and XOR operation is used with the current data. In line 7, enough message chains have been generated, so a new *IVT* is produced. If this timeout is not set, verification will take a significant amount of time, so the *IVT* is regularly updated at intervals to generate a new message chain.

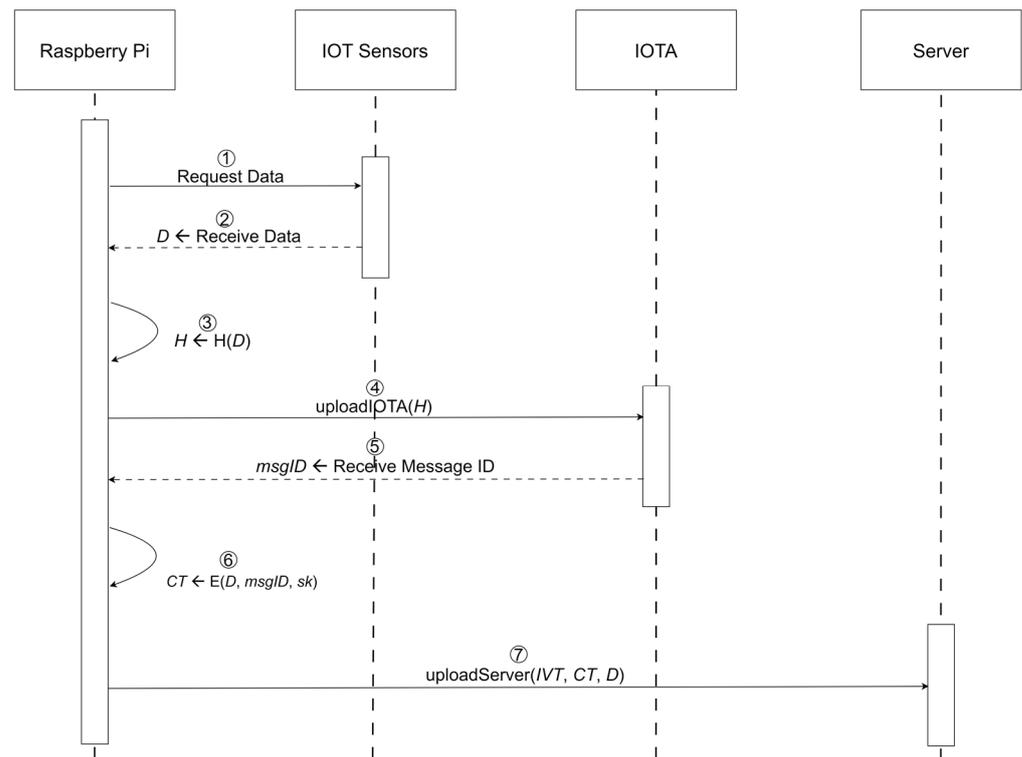


Figure 8. Sequence diagram for uploading confidential data.

3.7. Confidential Data Retrieval Method

After preprocessing and uploading data using our proposed method, data users with the key can use this method to verify the integrity of the data and retrieve it. Figure 9 shows the detailed process of data retrieval, which involves three actors: the client, IOTA, and the server. The client is the data user who must have the key to decrypt the encrypted CT. IOTA stores the hash value corresponding to the data and can verify the integrity of the data stored on the server. The server is used to store CL, which contains all ciphertexts within a certain period and can be decrypted using the key to obtain P. The complete process consists of seven steps:

Steps 1–2: Using the timestamp, the client retrieves the required data from the server, which returns CL and BL. CL is a list of ciphertexts, while BL only includes IVT and D because the integrity of the msgID has not yet been verified.

Step 3: The retrieved CL is decrypted using CFB decryption and the key to generate P, which is $D \oplus msgID$. Then, the BL $[0, 1, \dots, n]D$ obtained from the server is used to XOR each P to obtain the msgID with integrity.

Step 4: After obtaining the verified msgID, it can be added to BL. Therefore, BL includes D, IVT, and msgID.

Step 5–6: Using msgID, the IOTA is queried for the hash value of D. All hash values corresponding to msgID are placed in a list, which is HL. The milestone generated by the transaction is queried, and all queried milestones are placed in a list to form MSL.

Step 7: Using Algorithm 3, the integrity of D is verified. BL, HL, and MSL are verified to ensure that the hash value of the data are consistent. The order of the milestones is also verified to detect any abnormalities in the milestones, which may indicate that the msgID has been replaced.

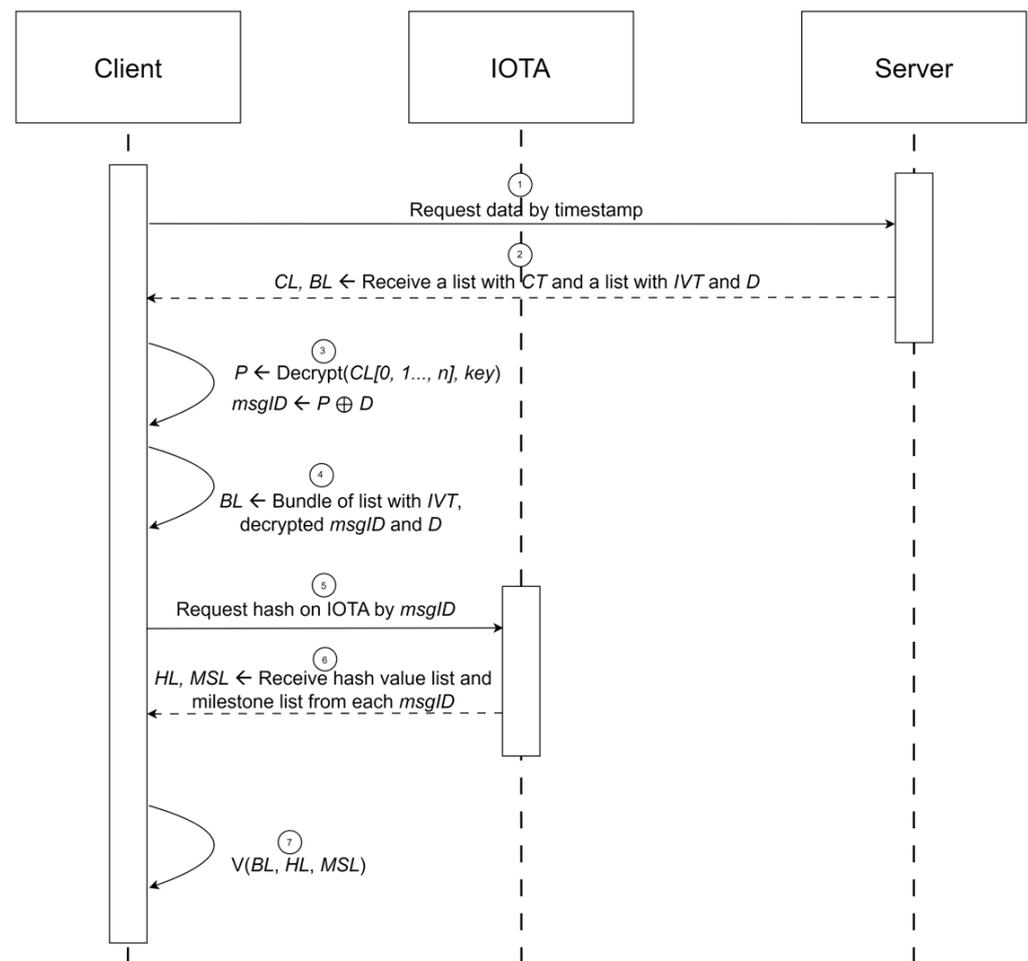


Figure 9. Sequence diagram for confidential data retrieval.

3.8. Large Data Upload Method

This section uses IPFS to solve the problem of limited capacity for storing data on IOTA. IPFS can upload any type of data without size limitations. In IoT scenarios wherein larger data such as photos or videos need to be uploaded, the 3 MB limit per transaction in IOTA can result in data being unable to be uploaded successfully. The method proposed in this section does not consider the integrity verification of *msgID* when it is stored on the server. If *msgID* integrity needs to be ensured, the algorithm used in the previous section can be applied to preprocess *msgID*. This section only addresses the capacity limitation of storing data on IOTA. Figure 10 shows the detailed process of large data uploads. The process of uploading data involves seven steps:

Steps 1–2: Retrieve data from IoT devices and return them to Raspberry Pi.

Steps 3–4: Upload the data to IPFS and retrieve the IPFS *CID*, which can be used to locate the file on IPFS.

Steps 5–6: Upload the IPFS *CID* to IOTA and retrieve the corresponding *msgID*.

Step 7: Upload the obtained *msgID* to the server for storage.

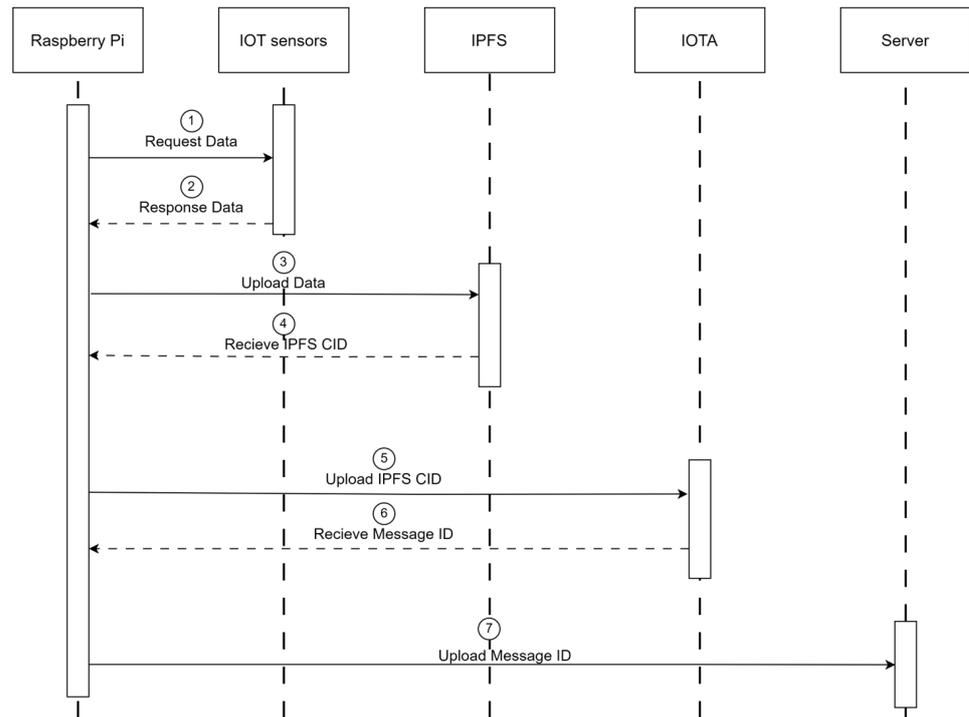


Figure 10. Sequence diagram for large data upload.

3.9. Large Data Retrieval Method

When receiving data, IPFS may be used to retrieve the data as they are stored on different nodes. However, this may not be more efficient than traditional data retrieval due to the scattered storage of the data. Figure 11 illustrates the entire receiving process, which is divided into six steps:

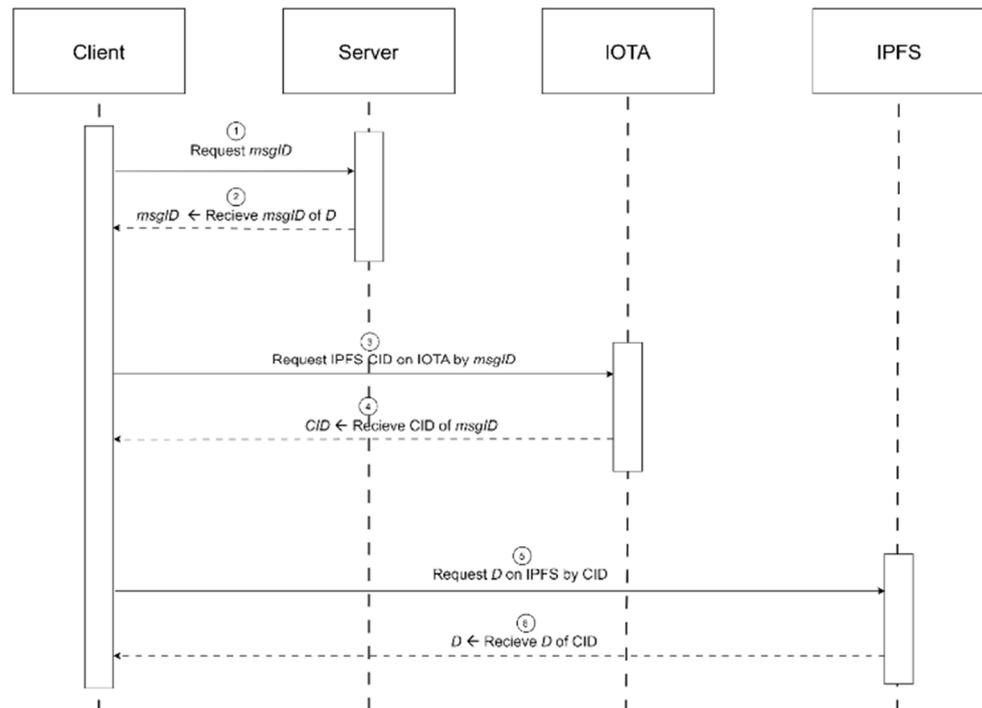


Figure 11. Sequence diagram for large data retrieval.

Steps 1–2: Request a specific *msgID* of *D* from the server and receive the server's response containing the *msgID*.

Step 3–4: Use the *msgID* to request the IPFS CID from IOTA and receive the *CID*.

Step 5–6: Use the *CID* to request *D* from IPFS and wait for IPFS to retrieve the data from the nodes and return *D* to the client.

4. Experimental Results

We implemented that the data received by the sensor are immediately uploaded to the IOTA. Our proposed architecture can be applied in the current industrial environment, and it can thus guarantee the integrity of the data. This article will implement the use of Docker to set up an IOTA node to upload data to the server and IOTA and evaluate the integrity of the data. Finally, different methods of uploading data will be evaluated.

- Raspberry Pi: The Raspberry Pi specifications are Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC, 8 GB RAM, and OS with Red Hat Enterprise Linux9.
- DHT11 sensor: The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin.

4.1. Implementation

The implementation of using Docker with containerized technology to set up nodes is shown in Figure 12 below. It shows that all functions have been set up, and also successfully executed. IOTA nodes are an essential part of the IOTA network, and they can participate in transaction verification and network security. Adding more nodes can enhance the overall security of the network because the more nodes there are, the more difficult it is for attackers to target the network. IOTA nodes participate in verifying and confirming transactions. Adding more nodes can speed up the transaction verification process and reduce the time it takes to confirm transactions. Therefore, deploying multiple IOTA nodes on various industrial devices using Docker can increase the security of the entire network and speed up transaction processing.

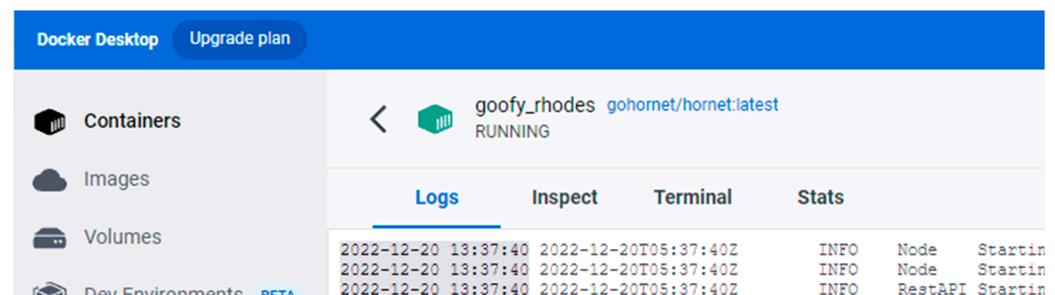


Figure 12. IOTA node built with Docker.

After setting up the node, we need to upload the data to IOTA Tangle. The entire process of this implementation can be divided into three steps:

Step 1. Receive temperature and humidity signals through the DHT11 sensor.

Step 2. Preprocess the data using the algorithm and send the received data to IOTA and MongoDB in real time through IOTA node on Raspberry Pi.

Step 3. Before accessing the data, we can verify whether the data have been tampered with through IOTA.

After uploading data, the integrity of the data are verified through our proposed algorithm. If the data are correct, they can pass the algorithm smoothly. If the data have been tampered with, an error will be reported.

Figure 13 shows the uploaded data from the dht11 sensor query using IOTA Explorer. It is a tool that you can use to search through data recorded on the DLT. The figure shows

that we uploaded the data and its hash value to IOTA. This implementation assumes that the data are non-confidential, so the data are uploaded directly to IOTA. If the data are confidential, only the hash value is uploaded, and both methods can verify the data integrity using our proposed algorithm.

Indexation Payload

```

Index
Benson_dht11_test
Data
{
  "Time": "04/22/2023 02:52:45",
  "Temp": 27,
  "Humidity": 34,
  "hash": "e77d55e1c5a24a8a3a60903ff087b9145bd9adbbcd9a54a170120b1a54d734f"
}

```

Figure 13. Data on IOTA.

4.2. Security Analysis

In this section, we will conduct a security analysis of the proposed method in this paper. We will analyze potential attacks on data stored on the server and further explain the integrity of the data.

A man-in-the-middle attack is a type of attack in network data transmission wherein an attacker impersonates the identities of both ends of the communication to eavesdrop, intercept, modify, or manipulate the communication content without being detected. There are various methods to conduct this type of attack, including IP spoofing, DNS spoofing, ARP spoofing, email phishing, SSL stripping, and WiFi eavesdropping, among others. This section implements ARP spoofing and verifies that using the architecture we proposed can detect data tampering and ensure the integrity of data stored on the server.

ARP spoofing, or address resolution protocol spoofing, is a malicious network attack where an attacker sends fraudulent ARP messages to link an incorrect MAC address to an IP address within a network. This allows the attacker to effectively replace the MAC address of a host on the network with their device, and potentially intercept or manipulate the traffic flowing through that host.

Figure 14 demonstrates how an attacker can use commands to target a victim with an IP address of 192.168.0.182. The figure shows that the victim machine has mapped the MAC address of 192.168.0.1 to the attacker's machine, so all transmitted data will go through the attacker's machine. Figure 15 displays all packets that go through the attacker's machine, allowing the attacker to eavesdrop on or modify any packets sent by the victim's machine. Using a proxy, the attacker can intercept and modify packets that are prepared to be sent to the server. Figure 16 shows that, if unprocessed data are uploaded directly to the server, they may be attacked and tampered with during transmission.

```

kbm2111@kbm2111:~$ sudo arpspoof -i enp6s0 -t 192.168.0.182 192.168.0.1
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24
f8:d1:11:6:3d:24 e4:5f:1:5d:e6:63 0806 42: arp reply 192.168.0.1 is-at f8:d1:11:6:3d:24

```

Figure 14. ARP spoofing implementation by the attacker.

```

pi@raspberrypi:~/benSON_SPACE/IOTA_NODEJS_PYTHON_SOCKET/python $ arp -a
? (172.20.0.2) at 02:42:ac:14:00:02 [ether] on br-e7b00735402c
dlinkrouter (192.168.0.1) at f8:d1:11:06:3d:24 [ether] on wlan0
? (172.20.0.3) at 02:42:ac:14:00:03 [ether] on br-e7b00735402c
? (192.168.0.127) at <incomplete> on wlan0
? (192.168.0.152) at f8:d1:11:06:3d:24 [ether] on wlan0
pi@raspberrypi:~/benSON_SPACE/IOTA_NODEJS_PYTHON_SOCKET/python $

```

Figure 15. Victim of ARP spoofing.

```

ba211@ba211:~$ sudo tshark -s 512 -i enp6s0 -f 'host 140.120.40.132 and !arp and !dst net 192.168.0.1'
Running as user "root" and group "root". This could be dangerous.
Capturing on "enp6s0"
  1 0.000000000 192.168.0.182 → 140.120.40.132 TCP 74 34842 → 27017 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1665707888 TSecr=0 WS=128
  2 0.000031713 192.168.0.182 → 140.120.40.132 TCP 74 [TCP Out-Of-Order] 34842 → 27017 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1665707888 TSecr=0 WS=128
  3 0.001710715 192.168.0.182 → 140.120.40.132 TCP 66 34842 → 27017 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1665707890 TSecr=133054841
  4 0.001729100 192.168.0.182 → 140.120.40.132 TCP 66 [TCP Dup ACK 3#1] 34842 → 27017 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1665707890 TSecr=133054841
  5 0.002249454 192.168.0.182 → 140.120.40.132 SSL 325 Continuation Data
  6 0.002267940 192.168.0.182 → 140.120.40.132 TCP 325 [TCP Retransmission] 34842 → 27017 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=259 TSval=1665707891 TSecr=133054841
  7 0.004025630 192.168.0.182 → 140.120.40.132 TCP 66 34842 → 27017 [ACK] Seq=260 Ack=330 Win=64128 Len=0 TSval=1665707892 TSecr=133054843
  8 0.004043862 192.168.0.182 → 140.120.40.132 TCP 66 [TCP Dup ACK 7#1] 34842 → 27017 [ACK] Seq=260 Ack=330 Win=64128 Len=0 TSval=1665707892 TSecr=133054843
  9 0.006353638 192.168.0.182 → 140.120.40.132 TCP 200 [TCP segment of a reassembled PDU]
 10 0.006372134 192.168.0.182 → 140.120.40.132 TCP 200 [TCP Retransmission] 34842 → 27017 [PSH, ACK] Seq=260 Ack=330 Win=64128 Len=134 TSval=1665707895 TSecr=133054843
 11 0.006397040 192.168.0.182 → 140.120.40.132 TCP 74 34854 → 27017 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1665707895 TSecr=0 WS=128
 12 0.006422291 192.168.0.182 → 140.120.40.132 TCP 74 [TCP Out-Of-Order] 34854 → 27017 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1665707895 TSecr=0 WS=128
 13 0.006746889 192.168.0.182 → 140.120.40.132 TCP 74 34860 → 27017 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1665707895 TSecr=0 WS=128
 14 0.006772549 192.168.0.182 → 140.120.40.132 TCP 74 [TCP Out-Of-Order] 34860 → 27017 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1665707895 TSecr=0 WS=128
 15 0.008013671 192.168.0.182 → 140.120.40.132 TCP 66 34854 → 27017 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1665707896 TSecr=133054847
 16 0.008031928 192.168.0.182 → 140.120.40.132 TCP 66 [TCP Dup ACK 15#1] 34854 → 27017 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1665707896 TSecr=133054847
 17 0.008269849 192.168.0.182 → 140.120.40.132 TCP 66 34860 → 27017 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1665707897 TSecr=133054848
 18 0.008288105 192.168.0.182 → 140.120.40.132 TCP 66 [TCP Dup ACK 17#1] 34860 → 27017 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1665707897 TSecr=133054848
 19 0.008435939 192.168.0.182 → 140.120.40.132 SSL 325 Continuation Data
 20 0.008454528 192.168.0.182 → 140.120.40.132 TCP 325 [TCP Retransmission] 34854 → 27017 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=259 TSval=1665707897 TSecr=133054847
 21 0.008902935 192.168.0.182 → 140.120.40.132 SSL 597 Continuation Data
 22 0.008921294 192.168.0.182 → 140.120.40.132 TCP 597 [TCP Retransmission] 34860 → 27017 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=531 TSval=1665707897 TSecr=133054848
 23 0.010495988 192.168.0.182 → 140.120.40.132 TCP 66 34854 → 27017 [ACK] Seq=260 Ack=330 Win=64128 Len=0 TSval=1665707899 TSecr=133054849
 24 0.010514417 192.168.0.182 → 140.120.40.132 TCP 66 [TCP Dup ACK 23#1] 34854 → 27017 [ACK] Seq=260 Ack=330 Win=64128 Len=0 TSval=1665707899 TSecr=133054849

```

Figure 16. The captured packet by the attacker.

We propose an architecture that preprocesses or encrypts various types of data and verifies the integrity of the data before they are accessed. Our framework can detect tampering of data and identify the tampered data as Temp. After testing the framework with ARP spoofing attacks in a laboratory environment, we confirmed that our framework can prevent man-in-the-middle attacks. The integrity of the data can be verified regardless of the type of man-in-the-middle attack encountered during transmission.

A remote access attack is a form of cyber-attack that occurs when an unauthorized person gains access to a computer or network from a remote location, typically over the Internet. The attacker may use various methods to gain access, including exploiting vulnerabilities in the operating system or applications, guessing weak passwords, or using social engineering techniques to trick users into disclosing login credentials. Once the attacker has gained remote access, they can carry out a wide range of malicious activities, including stealing sensitive information, installing malware or ransomware, altering or deleting data, and using the compromised system to launch additional attacks.

After a remote access attack on a computer, the *D* or *msgID* stored on the server may be tampered with. If the data are not encrypted or preprocessed using our proposed method, both the data and *msgID* may be tampered with simultaneously, and the integrity of the data cannot be verified. However, with our proposed data preprocessing and encryption method, data integrity can be verified before data retrieval, allowing us to identify the tampered data and verify their integrity.

4.3. Performance Analysis

This section evaluates the proposed methods in this paper by comparing the computational complexity and time required for each method and analyzing their respective strengths and weaknesses. Finally, the time required for uploading data to IOTA and Ethereum is compared, and the advantages and disadvantages of using DLT and blockchain are analyzed.

To analyze the time required for each method used in this section, we have calculated the time required for each operation used and then calculated the amount of computation required for each method. Table 2 shows the average time required for the four operations:

hashing, XOR, CFB encryption, and CFB decryption. These four operations are the methods required to be used locally in this architecture.

Table 2. Measurement of computational methods.

Computational Methods	Definition	Measurement
SHA256 hash (<i>HT</i>)	SHA256 (Secure Hash Algorithm 256) is a cryptographic hash function used to compress data into a fixed-length digital string (256 bits). It is a one-way encryption algorithm that cannot be decrypted in reverse.	6.78 ns
XOR (<i>XT</i>)	XOR (Exclusive OR) is a logical operator used to perform an operation on two binary bits. It outputs 0 when the two bits are the same, and 1 when they are different. XOR can be used for data encryption and checksums.	12.12 ns
CFB encryption (<i>ET</i>)	CFB (Cipher Feedback) encryption is a symmetric encryption mode that uses XOR operation to generate ciphertext by combining the plaintext with the output of the encryption algorithm.	41.68 ns
CFB decryption (<i>DT</i>)	CFB decryption is the process of using XOR operation to decrypt ciphertext with the output of the encryption algorithm to obtain plaintext. It uses feedback mode and can decrypt ciphertext generated by CFB encryption.	38.11 ns

This section divides the methods for uploading data into two categories: the non-confidential data upload method and the confidential data upload method. It also divides the methods for retrieving data into two categories: the non-confidential data retrieval method and the confidential data retrieval method. The calculation time required for the two upload methods uses the following equation, where n represents the total number of data to be uploaded.

Total computation time of non-confidential data upload:

$$\sum_{i=1}^n (2HT_i + XT_i) \quad (1)$$

Total computation time of confidential data upload method:

$$\sum_{i=1}^n (HT_i + ET_i) \quad (2)$$

Figure 17 illustrates the computation time required by our architecture. The non-confidential data upload method only uses hash computation. Therefore, it requires less time but lacks confidentiality. The confidential data upload method uses CFB encryption to encrypt data, which requires more computation time. Although it takes more time, it ensures data confidentiality.

The calculation time required for the two data retrieval methods uses the following equation, where n denotes the total number of data to be retrieved.

Total computation time of non-confidential data retrieval method:

$$\sum_{i=1}^n (2HT_i + 2XT_i) \quad (3)$$

Total computation time of confidential data retrieval method:

$$\sum_{i=1}^n (HT_i) + \sum_{i=1}^n \left(DT_i + \sum_{j=1}^n (DT_j) \right) \quad (4)$$

Figure 18 compares the two methods of data uploading based on the time required according to the amount of received data. The confidential data retrieval method initially requires less time to receive data compared to the non-confidential data retrieval method. However, as the amount of received data received increases, the confidential data retrieval method takes more time because it will generate a message chain, and the longer the message chain, the longer it takes to decrypt. Therefore, our architecture proposes a method of generating new message chains periodically, which is crucial for reducing decryption time.

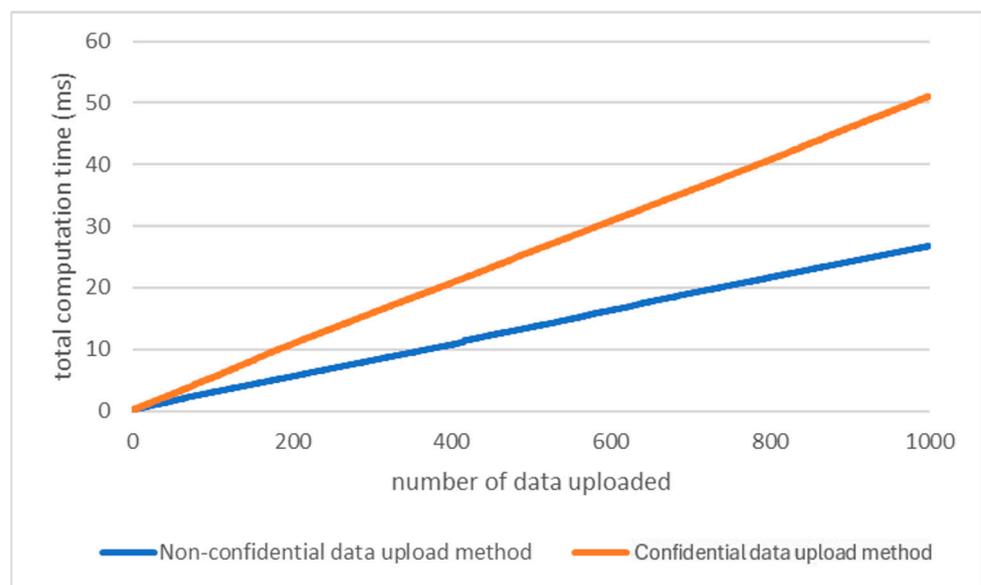


Figure 17. The computation time of data upload.

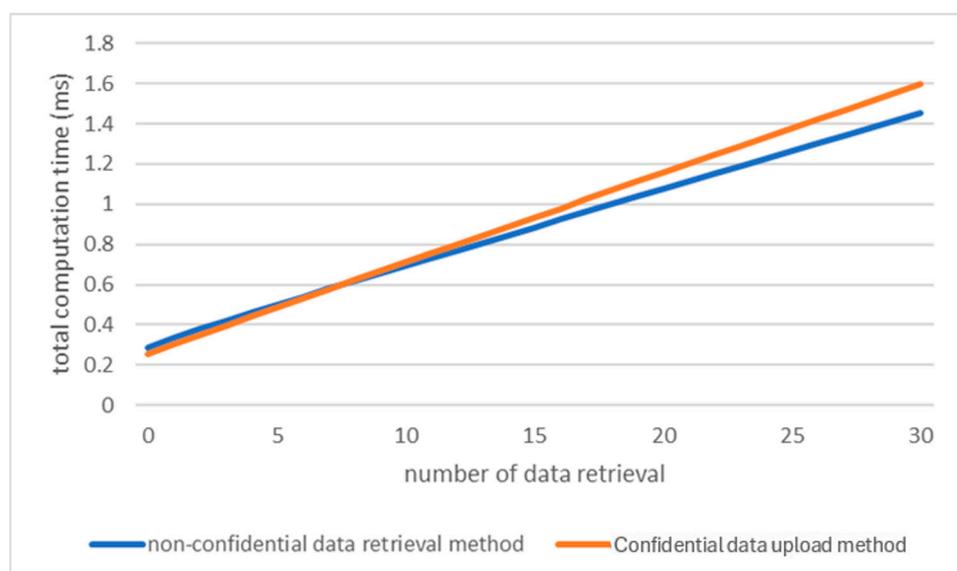


Figure 18. The computational time of data retrieval.

This section compared the computation time required for different methods. As can be seen from the analysis, if data need to be kept confidential, more computation time is required. Depending on the needs of different fields, different methods can be used to ensure data storage. For the uploading and receiving of confidential data, our method successfully reduced the time required for data decryption by updating the message chain.

By addressing specific vulnerabilities and offering a comparative analysis with established DLTs, this paper underscores the importance and relevance of the proposed solution in mitigating real-world challenges encountered in industrial control systems. Through proactive measures and innovative solutions, organizations can navigate the intricate landscape of data management, fortify data security, and foster an environment conducive to innovation and progress.

5. Conclusions

To ensure data preservation in IoT networks, we proposed a system architecture that guarantees data integrity and confidentiality from sensors to the IOTA Tangle. Once the sensors generate data, they are immediately uploaded to Tangle via our proposed method. This method is designed to reduce transmission paths and ensure that data integrity and confidentiality meet data preservation requirements. CFB encryption and DLT were used in this study to ensure that data were not tampered with. By testing the architecture against common attack methods, we were able to verify the security of the system, detect any malicious modifications, and ensure data integrity and confidentiality. As a result, we established a comprehensive data preservation framework.

Furthermore, our proposed architecture is versatile and can be applied to various fields such as manufacturing execution systems (MESs), supply chain management, AI training, environmental social governance (ESG), machine as a service (MaaS), and intellectual property (IP) management, among others. In traditional MES, data including product tracking and historical records are stored in a database. By using our proposed architecture, data integrity and confidentiality can be ensured, meeting data preservation requirements and reducing the possibility of tampering. In intelligent factories, large amounts of IoT-generated data are often used for AI training. Our architecture ensures the integrity of the data used to train the model, so the accuracy of the model will not be affected by data tampering. Our proposed architecture can also be applied to ESG by using it to validate data generated by CO₂ sensors, which makes it impossible for enterprises to forge data. It is difficult to verify the integrity of intangible assets, but with our architecture, the immutability of the data can be proven through DLT when disputes arise, providing better management of intangible assets. In traditional supply chains, expensive IT platforms are necessary to provide customers with comprehensive and transparent information about product and process quality. Incorporating our architecture enables the tracking of the source and quality of every component, as well as eliminating unnecessary quality control processes because DLT ensures data authenticity, thus ensuring the quality and security of the entire supply chain.

Our main objectives are to ensure the integrity of data storage, confidentiality and security of data transmission, minimize data transmission paths, etc. With the proposed architecture, we have successfully achieved these objectives. Furthermore, we have successfully constructed a data preservation architecture.

In future research, we will adopt a user-centric approach to explore usability issues and ensure the practical applicability of our proposed architecture. Additionally, we will take an interdisciplinary perspective by collaborating with experts from various fields to address legal, regulatory, ethical, and social implications, thereby enhancing the robustness and effectiveness of our data preservation framework.

Author Contributions: Conceptualization, I.-C.L.; Methodology, P.-H.C.; Software, P.-H.C.; Validation, I.-C.L.; Formal analysis, P.-C.T.; Writing—original draft, P.-H.C.; Supervision, S.-J.C.; Project administration, P.-C.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [National Science and Technology Council, Taiwan] grant number [112-2218-E-005-007].

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to specify the reason for the restriction.

Conflicts of Interest: Author Pai-Ching Tseng was employed by the company Natures Bank Exchange Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Zheng, X.; Sun, S.; Mukkamala, R.R.; Vatrappu, R.; Ordieres-Meré, J. Accelerating Health Data Sharing: A Solution Based on the Internet of Things and Distributed Ledger Technologies. *J. Med. Internet Res.* **2019**, *21*, 1–12. [CrossRef] [PubMed]
2. Lamtzidis, O.; Gialelis, J. An IOTA Based Distributed Sensor Node System. In Proceedings of the 2018 IEEE Globecom Workshops, Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
3. New MESA Model: A Framework for Smarter Manufacturing. MESA International. 2022. Available online: <https://mesa.org/topics-resources/mesa-model/> (accessed on 5 May 2023).
4. Nakamoto, S. Bitcoin: A Peer-To-Peer Electronic Cash System. Bitcoin.org. 2008. Available online: <https://bitcoin.org/bitcoin.pdf/> (accessed on 5 May 2023).
5. Kumar, K.; Kurhekar, M. Economically Efficient Virtualization over Cloud Using Docker Containers. In Proceedings of the 2016 IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore, India, 19–21 October 2016; pp. 95–100.
6. Soltani, R.; Saxena, L.; Joshi, R.; Sampalli, S. Protecting Routing Data in WSNs with Use of IOTA Tangle. In Proceedings of the 19th International Conference on Mobile Systems and Pervasive Computing, Niagara Falls, Canada, 9–11 August 2022; Volume 203, pp. 197–204.
7. Silvano, W.F.; Marcelino, R. Iota Tangle: A Cryptocurrency to Communicate Internet-of-Things Data. *Future Gener. Comput. Syst.* **2020**, *112*, 307–319. [CrossRef]
8. Popov, S. *The Tangle*; Computers & Industrial Engineering, 2019; Volume 136, pp. 160–172.
9. Mani, V.; Manickam, P.; Alotaibi, Y.; Alghamdi, S.; Khalaf, O.I. Hyperledger Healthchain: Patient-Centric IPFS-Based Storage of Health Records. *Electronics* **2021**, *10*, 3003. [CrossRef]
10. Alsbouei, T.; Qin, Y.; Hill, R.; Al-Aqrabi, H. Enabling Distributed Intelligence for the Internet of Things with IOTA and Mobile Agents. *Computing* **2020**, *102*, 1345–1363. [CrossRef]
11. Zhang, K.; Tian, J.; Xiao, H.; Zhao, Y.; Zhao, W.; Chen, J. A Numerical Splitting and Adaptive Privacy Budget-Allocation-Based LDP Mechanism for Privacy Preservation in Blockchain-Powered IoT. *IEEE Internet Things J.* **2023**, *10*, 6733–6741. [CrossRef]
12. Jayabalan, J.; Jeyanthi, N. Scalable Blockchain Model Using Offchain IPFS Storage for Healthcare Data Security and Privacy. *J. Parallel Distrib. Comput.* **2022**, *164*, 152–167. [CrossRef]
13. Lin, I.C.; Tseng, P.C.; Chen, P.H.; Chiou, S.J. Securing Industrial Control Systems: Enhancing Data Preservation in IoT with Streamlined IOTA Integration. In Proceedings of the 4th IFSA Winter Conference on Automation, Robotics & Communications for Industry 4.0/5.0, (ARCI' 2024), Innsbruck, Austria, 7–9 February 2024.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.