Article

Not peer-reviewed version

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

Iuon-Chang Lin *, Pai-Ching Tseng , Pin-Hsiang Chen , Shean-Juinn Chiou

Posted Date: 29 March 2024

doi: 10.20944/preprints202403.1859.v1

Keywords: DLT; IoT; data security; Docker; Container Technology; IOTA; Tangle

 doi:10.20944/preprints202403.1859.v1

Article

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

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

[1] Department of Management Information Systems, National Chung Hsing University
[2] Ph.D. Program of Business, Feng Chia University, Taichung, Taiwan, and Natures Bank Exchange CO., LTD.; tpcp630@gmail.com
[3] Department of Management Information Systems, National Chung Hsing University; benson0714@gmail.com
[4] Department of Mechanical Engineering, National Chung Hsing University; sjchiou@dragon.nchu.edu.tw
* Correspondence: iclin@nchu.edu.tw

Abstract: In the realm of data management, data preservation stands as a critical undertaking aimed at preserving and upholding the integrity of data. Regardless of whether it concerns personal or enterprise data, the detrimental effects of malicious alterations implemented by attackers cannot be overlooked. Particularly in conventional industrial control environments, the prevalent practice involves the transmission of data from sensors to databases for storage purposes. However, it is essential to recognize that this process exposes the data to various vulnerabilities. Thus, to ensure the long-term security and reliability of the data, it becomes imperative to implement robust data preservation strategies within these industrial control systems. However, the reliance of these databases on physical hard disks introduces inherent vulnerabilities, including the potential for data loss due to disk damage or targeted malicious attacks. Consequently, it becomes imperative to prioritize the implementation of robust data preservation measures. These measures are crucial in mitigating the risk of disruptions and protecting critical data from compromise. By establishing effective data backup systems, employing advanced security protocols, and implementing proactive monitoring mechanisms, organizations can bolster their data preservation capabilities and safeguard against potential threats to data integrity and availability. As a result, many enterprises opt to store their data with third-party providers to ensure data integrity. However, this approach carries inherent risks. If the third-party service experiences an attack or if the data is tampered with, it becomes challenging to verify the integrity of the data. To address these concerns and ensure data preservation within the context of the Internet of Things (IoT), a growing number of individuals are integrating IoT with Distributed Ledger Technology (DLT). By leveraging DLT, the integrity of data can be ensured, reducing reliance on centralized third-party storage and enhancing security in the IoT ecosystem. In this article, IOTA is the DLT, which employs Directed Acyclic Graph (DAG) to store transaction information. Compared to Ethereum or other blockchain technologies, IOTA offers notable advantages in terms of transaction verification speed, making it highly suitable for real-time IoT environments. However, the conventional transmission path from sensors to IOTA nodes entails a complex route, involving multiple hardware devices before reaching the intended destination. This complexity poses challenges in ensuring data integrity during transmission and introduces vulnerabilities such as man-in-the-middle attacks or SQL injection attacks. To address these issues, we propose a method to streamline the transmission path between sensors and IOTA, specifically tailored for industrial fields with numerous IoT devices. Our approach involves preprocessing the data stored on the server using our method before uploading, ensuring data confidentiality, and leveraging IOTA to guarantee data integrity. To achieve the shortest path between IoT and DLT nodes, it becomes necessary to establish IOTA nodes on lower-level devices, such as Raspberry Pi or IoT controllers. By simplifying the transmission path, we can reduce the potential for tampering and enhance overall data security. Implementing our proposed method enables the assurance of data confidentiality and integrity during both transmission and storage on the server, strengthening the trustworthiness of the IoT, and IOTA integration.

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

 doi:10.20944/preprints202403.1859.v1

1. Introduction

With the emergence of Industry 4.0, various innovative industrial technologies, including MES or AI technology. 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 satisfy data preservation requirements are gradually being 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 is 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 in IOTA cannot exceed 32kb. 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.

 doi:10.20944/preprints202403.1859.v1

Man-in-the-middle attacks, a common attack method, such as ARP spoofing, DNS spoofing, IP spoofing… 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 to:

- 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 it is uploaded to IOTA.
- Use containerization technology [5] to set up IOTA nodes, which can reduce the difficulty of setting up nodes, because there are lots 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 by proposed method to ensure its integrity during transmission and storage on the server. Satisfying data preservation requirements in the IoT environment.

2. Related Works

2.1. Distributed Ledger Technology

IoT covers a lot 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 point.

2.2. Transmission Path Selection from Sensors to DLT

There are three architectures for sending data from the sensor to the 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 is read by the sensors, send it to the computer, use the computer to process the data and then transfer it to the server or database, collect the data centrally, and then upload the data to the tangle.

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

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

Architecture (i) will be used in fields that require a lot 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, the computer receives the data and processes it before uploading. Architecture (iii) is the most ideal architecture in the intelligent factory, which can directly upload data to the tangle, and can also avoid the loss of transmission data caused by SPOF. Immediately send the data to the tangle.

2.3. IOTA

When the traditional DLT is applied to 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 Tangle solves the problem of gas fee and scalability. The traditional DLT needs to use miners to verify each

 doi:10.20944/preprints202403.1859.v1

transaction to propose a block and connect each block to form a DLT structure like a linked list. Tangle 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 the Tangle, to handle transactions. The Tangle enables concurrent processing of transactions and does not require miners to validate 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 the Tangle and act as validators to verify other people's transactions. When uploading data to the Tangle, the data is first verified by IOTA nodes with two transactions on the Tangle and then uploaded to the 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.

2.5. Cipher Feedback

Cipher Feedback (CFB) is a mode of operation for block ciphers that allows 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 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 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.

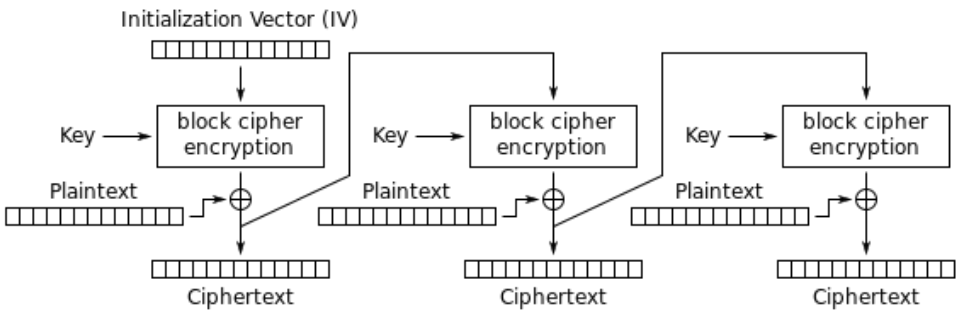

Figure 1. Cipher Feedback (CFB) mode encryption.

 doi:10.20944/preprints202403.1859.v1

2.6. InterPlanetary File System

In the IoT environment, real-time data reception and transmission with data integrity assurance are essential. Therefore, the authors Alsboui, 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 the Tangle. While data on the ledger is 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 it on IPFS, while storing the index generated by IPFS on the DLT, ensuring both data integrity and confidentiality.

3. Proposed Method

3.1. System Architecture

The proposed system architecture and method extended the previous work [13]. It ensures the integrity of data before and after uploading to the 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 the 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 have 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 that are not 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:

3. 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 it is uploaded to the chain.
4. Reducing the transmission path before uploading to the DLT: Taking the industrial control field as an example, after data is 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, it can be more challenging compared to installing it on a regular desktop computer because the Raspberry Pi uses an arm64 CPU architecture, which cannot directly install the official packages designed for the amd64 architecture. Therefore, this article utilizes containerization technology to address this issue.

Figure 2 demonstrates how to install the IOTA node on the Raspberry Pi. The main process is as follows:

- Step 1: Create container: install docker on the Raspberry Pi and create a container by docker. It 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

 doi:10.20944/preprints202403.1859.v1

increase the surrounding neighbor nodes, which is to make it synchronize with the 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.

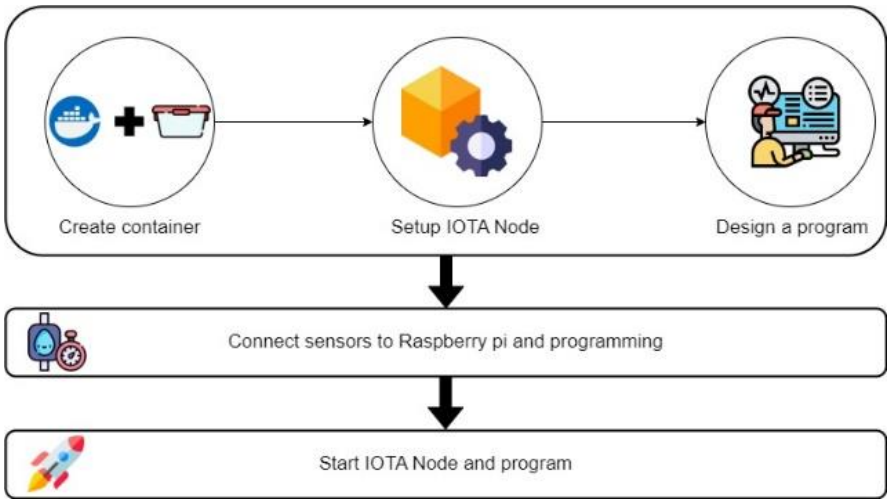

Figure 2. A flow chart of creating and setting up an IOTA Node in a container.

3.3. Proposed System Model

This section presents a secure 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 3. 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 its type using the algorithm proposed in this study. Finally, before the data is used by the data user, it will undergo integrity verification through IOTA and the algorithm proposed in this study to ensure that the data has not been tampered with.

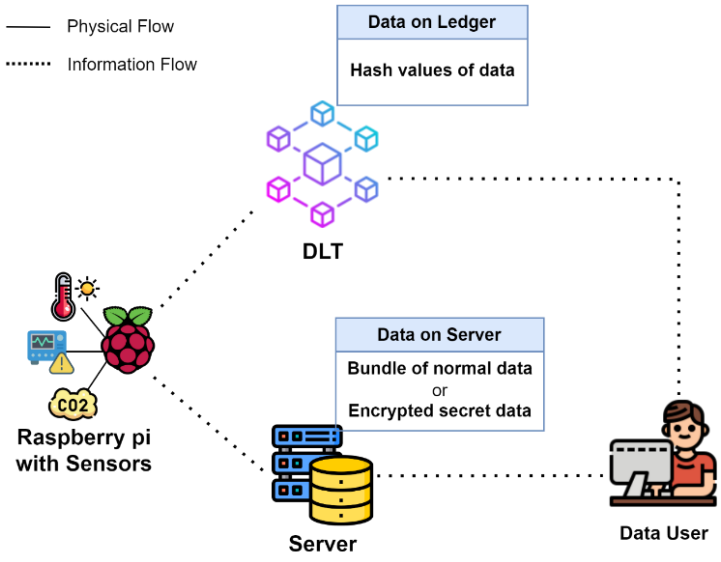

Figure 3. Proposed system architecture to ensure data integrity.

 doi:10.20944/preprints202403.1859.v1

Figure 4 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 is hashed before being uploaded to IOTA. Figure 6 shows the process of data retrieval by the data user. Using the message ID returned by IOTA, the data is 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 has not been tampered with.

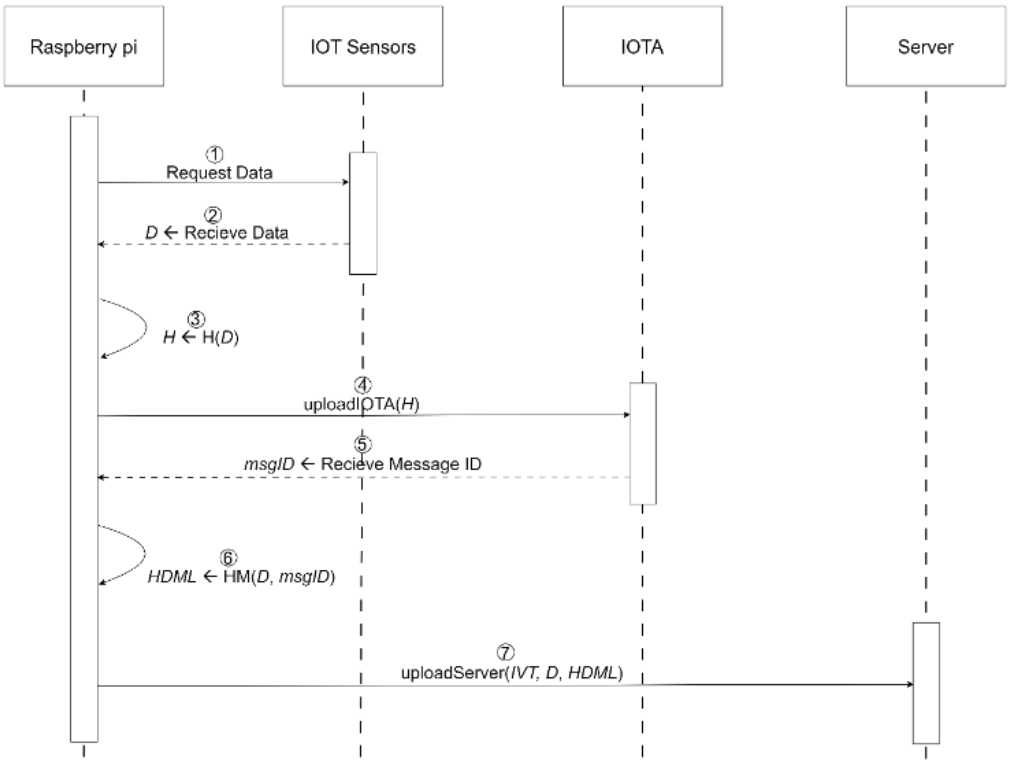

Figure 4. Sequence diagram for non-confidential data upload.

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 secret data, respectively.

 doi:10.20944/preprints202403.1859.v1

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( )	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( )	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( )	The algorithm to verify integrity of *D* (Algorithm 3)
E( )	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*

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 needs to undergo. Figure 4 illustrates the data transmission path when data is transmitted from IoT sensors to the server. The processing of data on a 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, the data can ensure data preservation on the server.

The entire process of uploading non-confidential data, with four roles involved: Raspberry Pi, IoT sensors, IOTA, and Server. The 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 7 steps:

- Step 1-2: Raspberry Pi collects raw data from sensors using a program.
- Step 3: The original data is 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 is 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 5, 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 use XOR with the plaintext. These steps can allow all data to have relatedness. If any data is tampered with, the modified data can be easily identified during verification. To avoid spending a lot of time verifying data, a timeout is set in advance on the third line of Algorithm 1, new *IVT* is generated periodically and preventing data from forming excessively

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1859.v1

long chains, it 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*.

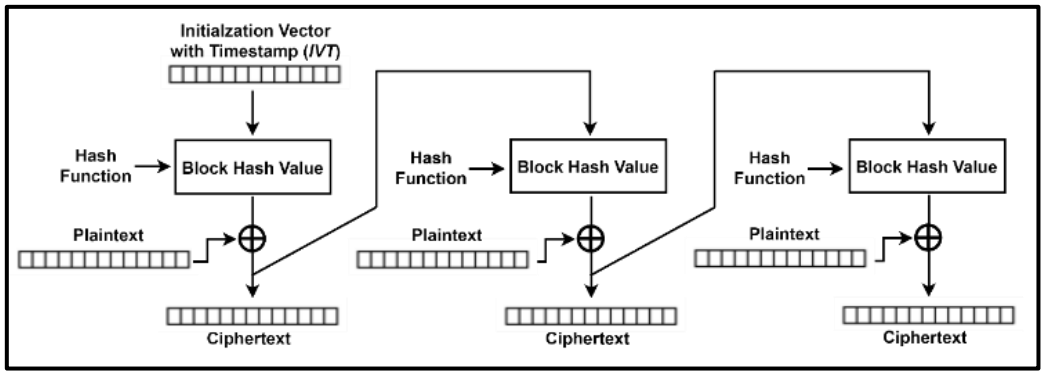

Figure 5. Proposed method based on CFB encryption.

Algorithm 1 A method to hash data based on CFB

1: **Input :** IVT, 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** $timeout <= t$ **then**
6: $CT \Leftarrow$H$(PD) \oplus HDM$
7: $HDML$.append(CT)
8: **return** $HDML$
9: **else**
10: Generate IVT with timestamp
11: $CT \Leftarrow$H$(PD) \oplus IVT$
12: $HDML$.append(CT)
13: **return** $HDML$
14: **end if**

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 it to ensure that the data has not been tampered with. Figure 6 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 6 steps:

- Step 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 includes *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, where true indicates successful verification and false indicates verification failure.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1859.v1

- Step 4-5: Obtain the hash value of the entire data and its corresponding milestone on the IOTA using the *msgID* that has integrity. Verify if the milestone indicates whether the data has been uploaded in order, or 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, where true indicates successful verification and false indicates verification failure.

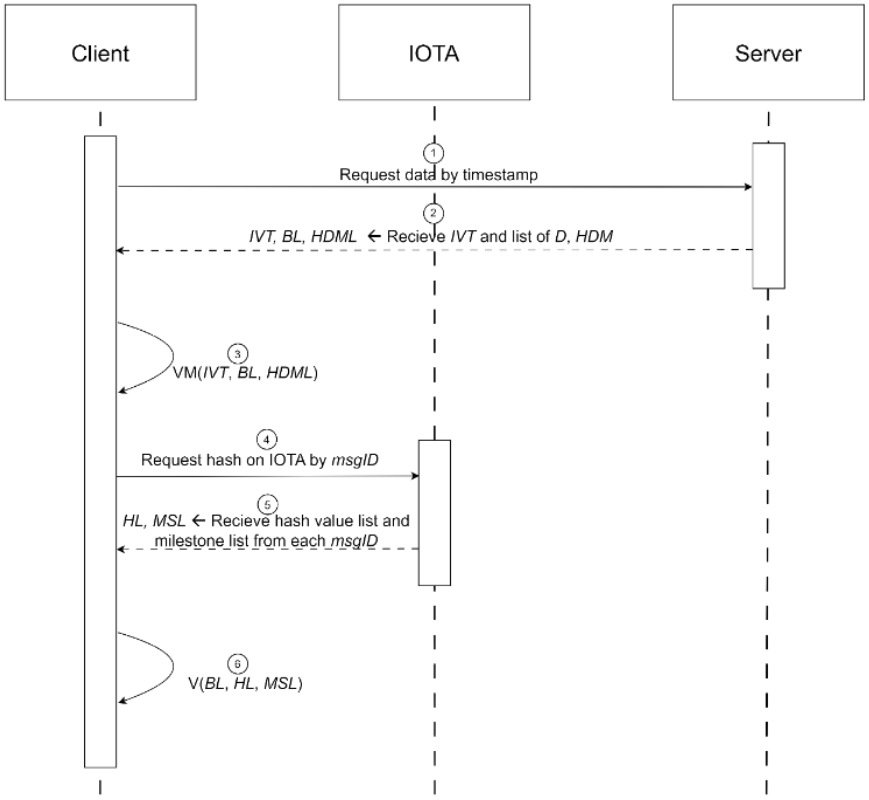

Figure 6. Sequence diagram for non-confidential data retrieval.

In step 3 of Figure 6, 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 for loop in line 8 will compare all remaining *msgID* or data in BL to check if they are correct. Finally, true or false will be returned, where true indicates that the *msgID* has not been tampered with, while false indicates that the data has been tampered with.

 doi:10.20944/preprints202403.1859.v1

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 is correct and has not been tampered with. false indicates that the data has been maliciously modified.*/
4: $FD \Leftarrow \mathrm{H}(BL[0]_{IVT} \oplus BL[0]_{msgID} \oplus BL[0]_D)$ //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** $\mathrm{H}(BL[i-1]_D \oplus BL[i]_{msgID} \oplus BL[i]_D) \neq HDML[i]$ **then**
12: Data has been tampered with someone
13: **return** false
14: **end if**
15: **end for**
16: **return** true

In Step 6 of Figure 6, Algorithm 3 is used to verify the integrity of the data by comparing it 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 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 checks whether the milestones on the IOTA Tangle are sorted in order. If there is an issue of milestones being out of order, it means that the *msgID* has been replaced, and the data does not have integrity. Finally, this algorithm will return whether the data has been tampered with. If the data has been tampered with, it will return false, otherwise it will return true.

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** $\mathrm{H}(BL[i]_D) \neq HL[i]$ **then**
7: Data has been tampered with someone
8: **return** false
9: **else if** $MSL[i-1] >= MSL[i]$ **then**
10: Data has been tampered with someone
11: **return** false
12: **end if**
13: **end for**
14: **return** true

3.6. Secret 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 7 shows the entire process of uploading confidential data, involving four roles: Raspberry Pi, IoT sensors, IOTA, and Server. Raspberry Pi is used for receiving and transmitting information,

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1859.v1

IoT sensors are used for receiving various signals, IOTA is used for verifying the integrity of the data, and 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 sensors to the 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 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 generated by Algorithm 4 to the server and store it in the corresponding message chain using the timestamp of the *IVT*.

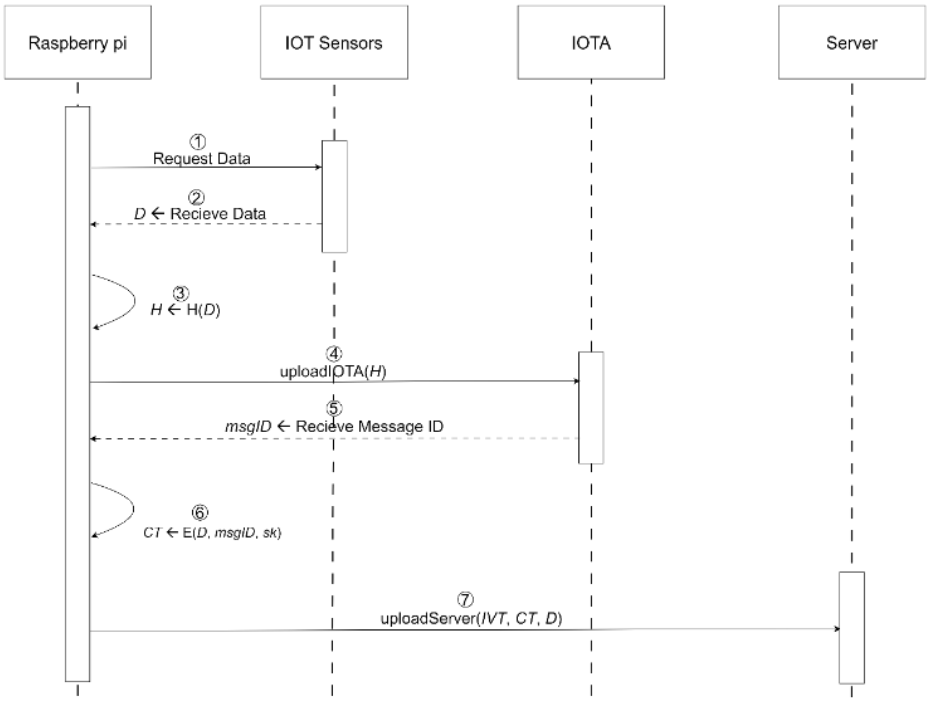

Figure 7. Sequence diagram for uploading secret data.

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. Because of 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 is encrypted, and use XOR operation 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 lot of time, so the *IVT* is regularly updated at intervals to generate a new message chain.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1859.v1

Algorithm 4 Encrypt data based on CFB

Input : $D, msgID, key$
Output : CT
/* Return the ciphertext(CT) generated by using CFB encryption and the initialization vector with timestamp(IVT).*/
$PD \Leftarrow$ Search previous encrypted data on raspberry pi
if $PD \neq null$ **or** $timeout <= t$ **then**
    $CT \Leftarrow \mathrm{E}_{key}(PD) \oplus (D \oplus msgID)$
    return CT
else
    Generate IVT with timestamp
    $CT \Leftarrow \mathrm{E}_{key}(IVT) \oplus (D \oplus msgID)$
    return CT, IVT
end if

3.7. Secret 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 8 shows the detailed process of data retrieval, which involves three roles: 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 7 steps:

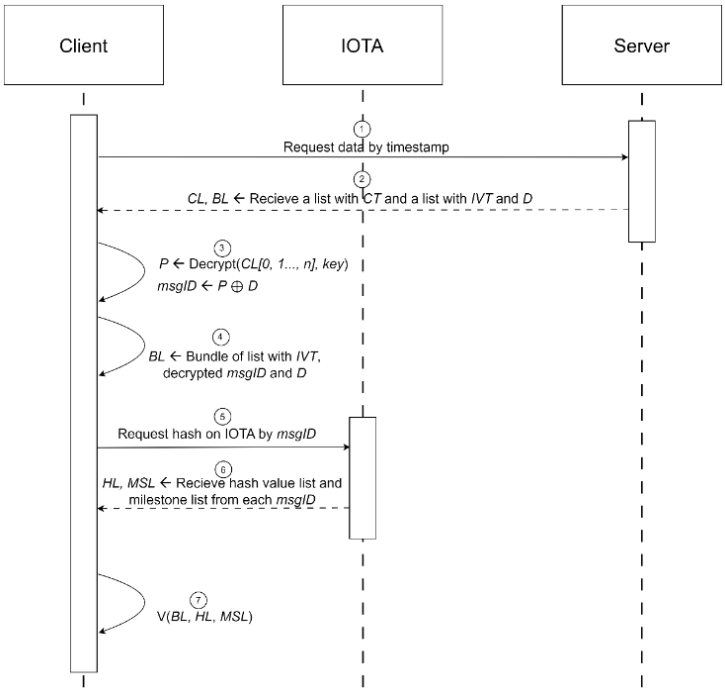

Figure 8. Sequence diagram for secret data retrieval.

- Step 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,...,n]D obtained from the server is used to XOR each P to obtain the *msgID* with integrity.

 doi:10.20944/preprints202403.1859.v1

- 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 checked to ensure that the hash value of the data is consistent. The order of the milestones is also checked to detect any abnormalities in the milestones, which indicate that the *msgID* has been replaced.

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 where larger data such as photos or videos need to be uploaded, the 3MB 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 9 shows the detailed process of data retrieval. The process of uploading data involves 7 steps.

- Step 1-2: Retrieve data from IoT devices and return it to the Raspberry Pi.
- Step 3-4: Upload the data to IPFS and retrieve the IPFS *CID*, which can be used to locate the file on IPFS.
- Step 5-6: Upload the IPFS *CID* to IOTA and retrieve the corresponding *msgID*.
- Step 7: Upload the obtained *msgID* to the server for storage.

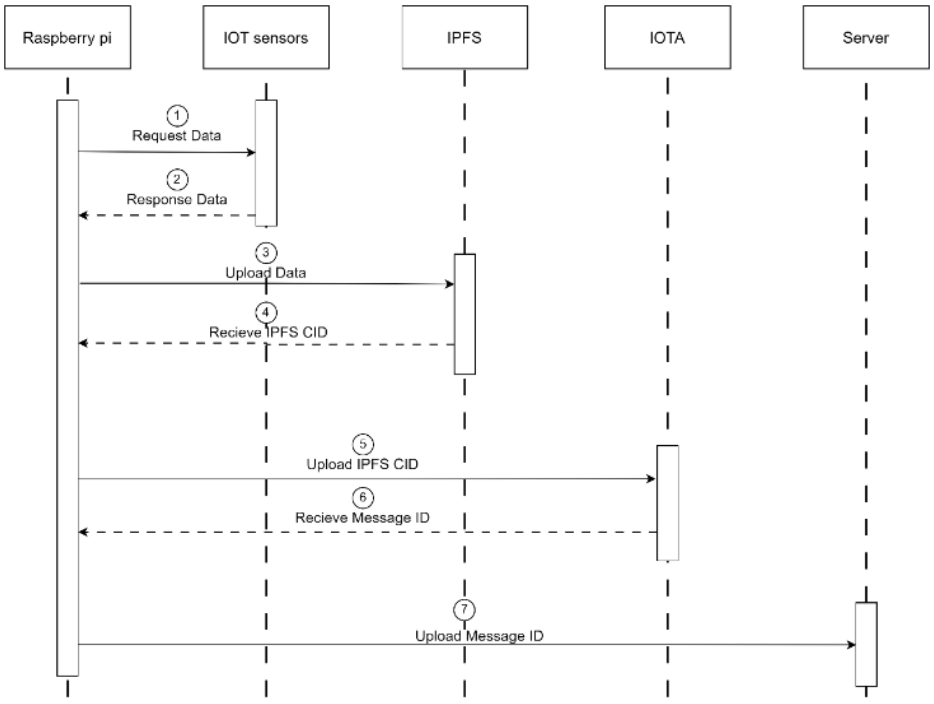

Figure 9. Sequence diagram for large data upload.

3.9. Large Data Upload Method

When receiving data, IPFS may be used to retrieve the data as it is stored on different nodes. However, this may not be more efficient than traditional data retrieval due to the scattered storage of the data. Figure 10 illustrates the entire receiving process, which is divided into 6 steps:

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

 doi:10.20944/preprints202403.1859.v1

- 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.

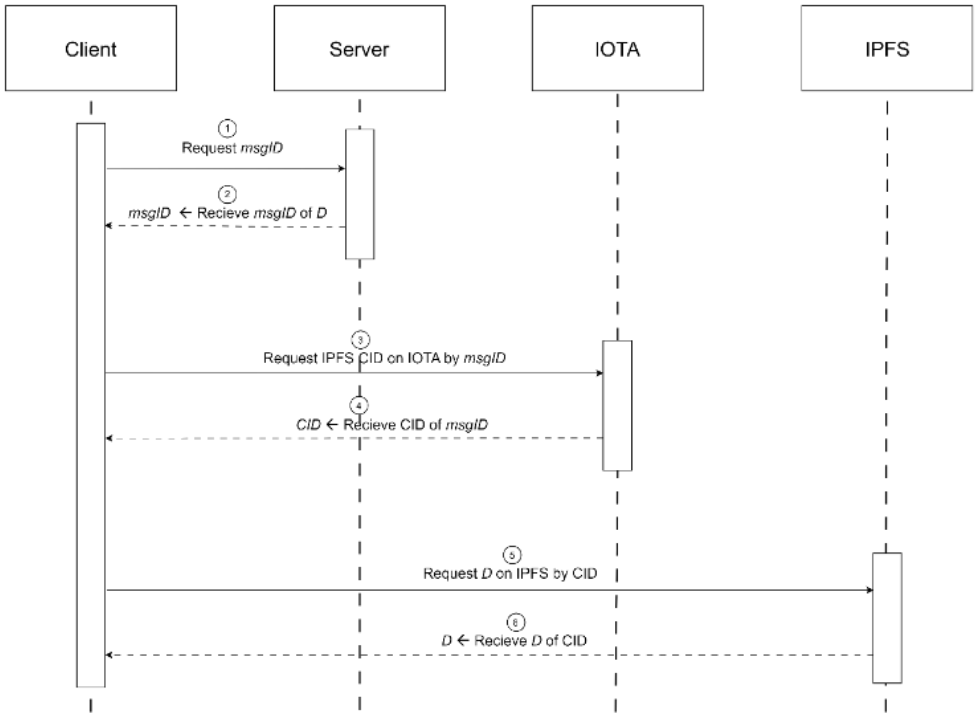

Figure 10. Sequence diagram for large data retrieval.

4. Implementation and Experimental Results

We implemented that the data received by the sensor is immediately uploaded to the IOTA. Our proposed architecture can be applied in the current industrial environment, it can guarantee the integrity of the data. This article will implement the use of Docker to set up an IOTA node 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, 8GB 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.

The implementation of using Docker with containerized technology to set up nodes is shown in Figure 11 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.

 doi:10.20944/preprints202403.1859.v1

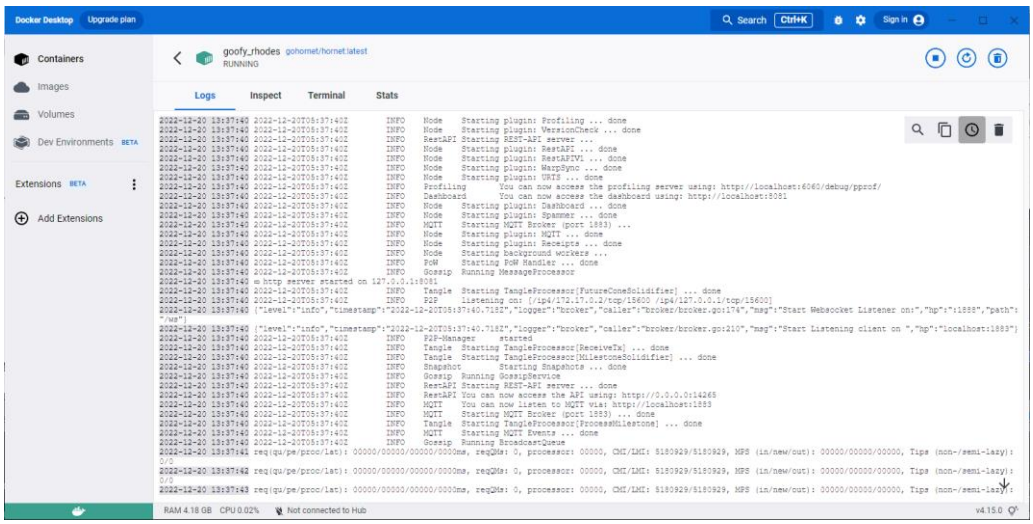

Figure 11. IOTA node built with Docker.

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

Figure 12 shows the uploaded data from the dht11 sensor query using IOTA Explorer. It's 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 is non-confidential, so the data is uploaded directly to IOTA. If the data is secret, only the hash value is uploaded, and both methods can verify the data integrity using our proposed algorithm.

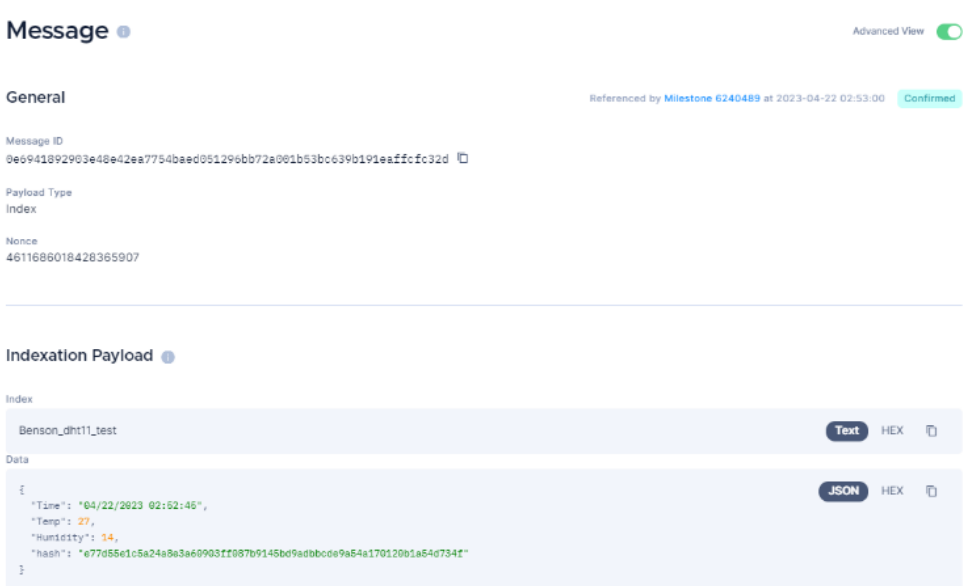

Figure 12. Data on IOTA.

4.1. 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 where 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

 doi:10.20944/preprints202403.1859.v1

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 own device, and potentially intercept or manipulate the traffic flowing through that host.

We propose an architecture that preprocesses or encrypts various types of data and verifies the integrity of the data before it is accessed. Our framework can detect tampering of data and identifies 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 is 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 its integrity.

4.2. 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.

 doi:10.20944/preprints202403.1859.v1

Table 2. Measurement of computational methods.

Computational methods	Definition	Measurement
SHA256 hash (*HT*)	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 (*XT*)	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 (*ET*)	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 (*DT*)	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 method divide into two categories: the Non-confidential data upload method and the Secret data upload method. It also divides the methods for retrieving data into two categories: the non-confidential data retrieval method and the Secret data retrieval method. The calculation time required for the two upload methods using the following equation, where n represents the total number of data to be uploaded.

Total computation time of non-confidential data upload:

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

Total computation time of secret data upload method:

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

Figure 13 illustrates the computation time required by our architecture. Non-confidential data upload method only uses hash computation, therefore, it requires less time but lacks confidentiality. The secret data upload method uses CFB encryption to encrypt data, which requires more computation time. Although it takes more time, it ensures data confidentiality.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1859.v1

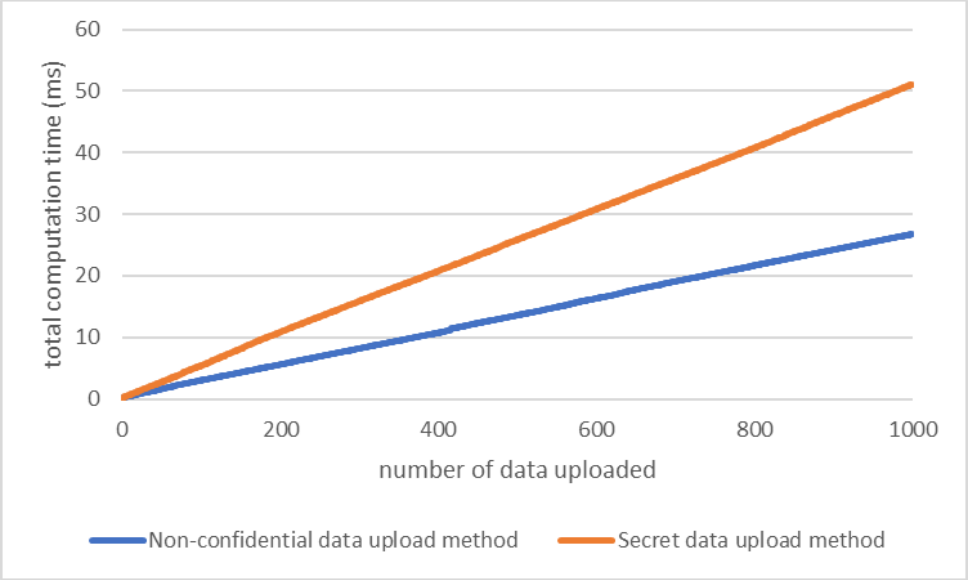

Figure 13. The computation time of data upload.

The calculation time required for the two data retrieval methods using the following equation, where n denotes the total number of data to be retrieved.
Total computation time of non-confidential data retrieval method:

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

Total computation time of secret data retrieval method:

$$\sum_{i=1}^{n} HT_i + \sum_{i=1}^{n}(DT_i + \sum_{j=1}^{i-1} DT_j) \tag{4}$$

Figure 14 compares the two methods of data uploading based on the time required according to the amount of received data. The secret data retrieval method initially requires less time to receive data compared to the non-confidential data retrieval method. However, as the amount of data received increases, the secret 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 14. The computational time of data retrieval.

 doi:10.20944/preprints202403.1859.v1

This section compared the computation time required for different methods. As can be seen from the analysis, if data needs 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 secret data, our method successfully reduced the time required for data decryption by updating the message chain.

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, it is 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 was 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.

Our proposed architecture is versatile and can be applied to various fields such as Manufacturing Execution Systems (MES), 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 won't 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's 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 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.

References

1. X. Zheng; S. Sun; R. R. Mukkamala; R. Vatrapu; and J. Ordieres-Meré. “Accelerating Health Data Sharing: A Solution Based on the Internet of Things and Distributed Ledger Technologies,” *Journal of Medical Internet Research*, **2019**, vol. 21, no. 6, pp. 1-12.
2. O. Lamtzidis; and J. Gialelis. “An IOTA Based Distributed Sensor Node System,” *Proceedings of 2018 IEEE Globecom Workshops*, Abu Dhabi, United Arab Emirates, December 9- December 13, pp. 1-6, 2018.
3. “New MESA Model: A Framework for Smarter Manufacturing,” MESA International. Available online: https://mesa.org/topics-resources/mesa-model/, 2022. (accessed May 2023).
4. S. Nakamoto, “Bitcoin: A Peer-To-Peer Electronic Cash System,” Bitcoin.org. Available online: https://bitcoin.org/bitcoin.pdf/, 2008. (accessed May 2023).
5. K. Kumar; and M. Kurhekar. “Economically Efficient Virtualization over Cloud Using Docker Containers,” *Proceedings of 2016 IEEE International Conference on Cloud Computing in Emerging Markets*, Bangalore, India, October 19-October 21, pp. 95–100, 2016.
6. R. Soltani; L. Saxena; R. Joshi; and S. Sampalli. “Protecting Routing Data in WSNs with Use of IOTA Tangle,” *Proceeding of The 19th International Conference on Mobile Systems and Pervasive Computing*, Niagara Falls, Canada, August 9-August 11, vol. 203, pp. 197–204, 2022.
7. W. F. Silvano, and R. Marcelino, “Iota Tangle: A Cryptocurrency to Communicate Internet-of-Things Data,” *Future Generation Computer Systems*, **2020**, vol. 112, pp. 307-319.
8. S. Popov. “The Tangle,” White paper, 2018, vol. 1, no. 3.
9. V. Mani; P. Manickam; Y. Alotaibi; S. Alghamdi; and O. I. Khalaf. “Hyperledger Healthchain: Patient-Centric IPFS-Based Storage of Health Records,” *Electronics*, **2021**, vol. 10, no. 23, pp. 3003.

 doi:10.20944/preprints202403.1859.v1

10. T. Alsboui, Y. Qin, R. Hill, and H. Al-Aqrabi. "Enabling Distributed Intelligence for the Internet of Things with IOTA and Mobile Agents," *Computing*, **2020**, vol. 102, no. 6, pp. 1345–1363.
11. K. Zhang; J. Tian; H. Xiao, Y. Zhao; W. Zhao; and J. Chen. "A Numerical Splitting and Adaptive Privacy Budget-Allocation-Based LDP Mechanism for Privacy Preservation in Blockchain-Powered IoT," *IEEE Internet of Things Journal*, **2023**, vol. 10, no. 8, pp. 6733-6741.
12. J. Jayabalan; and N. Jeyanthi. "Scalable Blockchain Model Using Offchain IPFS Storage for Healthcare Data Security and Privacy," *Journal of Parallel and Distributed Computing*, **2022**, vol. 164, pp. 152–167.
13. I. C. Lin, P. C. Tseng, P. H. Chen, and S. J. Chiou. "Securing Industrial Control Systems: Enhancing Data Preservation in IoT with Streamlined IOTA Integration," *Proceedings of 4th IFSA Winter Conference on Automation, Robotics & Communications for Industry 4.0 / 5.0, (ARCI' 2024),* 7-9 February 2024, Innsbruck, Austria.