processes

MDPI

Article

IOTA Data Preservation Implementation for Industrial Automation and Control Systems

Iuon-Chang Lin [1,*], Pai-Ching Tseng [2], Yu-Sung Chang [1] and Tzu-Ching Weng [2]

[1] Department of Management Information Systems, National Chung Hsing University, 145 Xingda Road, Taichung 40227, Taiwan; yorkisinhere@gmail.com
[2] Ph.D. Program of Business, Feng Chia University, Taichung 40724, Taiwan; tpcp630@gmail.com (P.-C.T.); tcweng@fcu.edu.tw (T.-C.W.)
* Correspondence: iclin@nchu.edu.tw; Tel.: +886-422840864; Fax: +886-422857173

Abstract: Blockchain 3.0, an advanced iteration of blockchain technology, has emerged with diverse applications encompassing various sectors such as identity authentication, logistics, medical care, and Industry 4.0/5.0. Notably, the integration of blockchain with industrial automation and control systems (IACS) holds immense potential in this evolving landscape. As industrial automation and control systems gain popularity alongside the widespread adoption of 5G networks, Internet of Things (IoT) devices are transforming into integral nodes within the blockchain network. This facilitates decentralized communication and verification, paving the way for a fully decentralized network. This paper focuses on showcasing the implementation and execution results of data preservation from industrial automation and control systems to IOTA, a prominent distributed ledger technology. The findings demonstrate the practical application of IOTA in securely preserving data within the context of industrial automation and control systems. The presented numerical results validate the effectiveness and feasibility of leveraging IOTA for seamless data preservation, ensuring data integrity, confidentiality, and transparency. By adopting IOTA's innovative approach based on Directed Acyclic Graph (DAG), the paper contributes to the advancement of blockchain technology in the domain of Industry 4.0/5.0.

Keywords: IOTA; Industry 4.0/5.0; blockchain; data preservation; industrial automation and control systems

Citation: Lin, I.-C.; Tseng, P.-C.; Chang, Y.-S.; Weng, T.-C. IOTA Data Preservation Implementation for Industrial Automation and Control Systems. *Processes* **2023**, *11*, 2160. https://doi.org/10.3390/pr11072160

Academic Editor: Sergey Y. Yurish

Received: 13 June 2023
Revised: 15 July 2023
Accepted: 16 July 2023
Published: 19 July 2023

1. Introduction

The application of Industry 4.0/5.0 is experiencing exponential growth across various fields. As Internet of Things (IoT) devices are deployed, a substantial amount of data is generated, which is traditionally centralized on client servers or cloud servers. However, such a centralized infrastructure poses challenges such as a single point of failure and a lack of trust between devices [1]. One solution to address these challenges is the implementation of a distributed system based on blockchain technology. The application of blockchain in Industry 4.0/5.0 has seen diverse use cases [2].

For instance, Datta et al. [3] proposed a security scheme that enables encrypted record storage for information and energy transactions between vehicles. Hang et al. [4] utilized Hyperledger Fabric to establish a fish farm platform, ensuring data integrity. Guan et al. [5] designed a two-tier distributed energy transaction system that safeguards transaction information through blockchain consensus mechanisms. Grecuccio et al. [6] leveraged IoT devices to record the food supply chain process and established an Ethereum node as a gateway to broadcast the data to smart contracts on the blockchain network for storage.

Despite the progress made, there are still several challenges in the application of blockchain in Industry 4.0/5.0. For instance, sensor data monitoring over an extended period is limited, and the costs associated with integrating blockchain into IoT systems,

due to handling fees and scalability issues, can be substantial. Furthermore, while smart contracts have shown promise in supply chain management, the historical data from IoT devices cannot be recorded on the blockchain, and the data transmission process from the IoT perception layer to the network layer lacks openness and transparency. To address these challenges of scalability, efficiency, and handling fees, Popov introduced a distributed ledger called Tangle, which is based on the Directed Acyclic Graph (DAG) structure [7]. Tangle differs from traditional blockchain ledgers as it utilizes the DAG structure as the foundation, enabling faster transactions and comprehensive recording of the history of Industry 4.0/5.0.

According to a report by Ericsson [8], it is projected that by 2021, there will be approximately 28 billion globally connected smart devices. Additionally, more than 15 billion devices have already adopted machine-to-machine (M2M) communication [8]. With the widespread adoption of the Internet of Things in various aspects of life, Wireless Sensor Networks (WSNs) have emerged as a crucial component for the application of Industry 4.0/5.0. WSNs are characterized by their low energy consumption, small size, and ease of deployment.

In the realm of IoT environments, ensuring real-time data reception and transmission with data integrity is crucial. To address the challenges related to massive data transmission and efficiency, Alsboui et al. [9] proposed the Mobile-Agent Distributed Intelligence Tangle-Based approach (MADIT). They utilized the IOTA Masked Authenticated Messaging (MAM) protocol to ensure data privacy on the Tangle. While ledger data are publicly transparent, sensitive data may need to be uploaded to the ledger. Therefore, Zhang et al. [10] introduced LDP, which preserves data confidentiality while uploading it to the distributed ledger technology (DLT). In scenarios where data sizes exceed the single transaction limit of the ledger, J. Jayabalan and N. Jeyanthi [11] proposed a model that encrypts medical data and stores it on IPFS, while storing the IPFS-generated index on the DLT, ensuring both data integrity and confidentiality.

Moving on to Docker, it is an extensively adopted open-source platform for containerization, enabling developers to package applications and their dependencies into lightweight, portable containers. Since its release in 2013, Docker has become an indispensable tool for building, shipping, and running applications. Containerization technology has gained significant popularity due to its ability to provide lightweight, portable, and isolated execution environments for applications. Researchers have explored various aspects of containerization, including performance, security, and usability. In a study by Divya and Sri [12], the potential of containerization technology and edge-fog cloud infrastructure is showcased in enabling efficient and scalable fall detection systems in healthcare. The paper emphasizes the importance of considering specific application workload requirements when selecting a containerization platform and infrastructure architecture.

Another research by Singh et al. [13] proposes an innovative solution to address the challenges of secure and efficient task containerization in IoT systems. The paper highlights the significance of considering both security and efficiency requirements in designing containerization solutions and demonstrates the effectiveness of game-theoretic approaches in tackling these challenges.

In the context of industrial automation and control systems, the existing architecture relies on a centralized server or cluster head to manage, identify, and encrypt connections among a large number of deployed sensors in the sensing area. However, this centralized approach presents challenges in terms of scalability, security, and privacy. Additionally, ensuring the confidentiality and integrity of data during transmission from the sensor collection end to the user end is crucial.

Furthermore, the application of blockchain in Industry 4.0/5.0 brings forth several challenges. These challenges encompass the inability to monitor sensor data over an extended period, the high costs associated with importing blockchain into the Internet of Things (IoT) due to handling fees and scalability issues, the lack of IoT data recording in the blockchain, and the lack of transparency in the data transmission process from the IoT perception layer to the network layer.

These challenges underscore the significance of addressing the limitations in current architectures and exploring the potential of blockchain technology to overcome them. By developing decentralized and secure solutions, achieving scalable, cost-effective, and transparent data management in industrial automation and control systems becomes feasible. The resolution of these challenges can enhance the reliability, efficiency, and trustworthiness of Industry 4.0/5.0 applications, facilitating their widespread adoption and unlocking their full potential in various domains.

2. Related Works

2.1. IOTA

IOTA is an open-source decentralized ledger technology operating on a peer-to-peer network. It utilizes the Directed Acyclic Graph (DAG) method to store each transaction and was officially launched in around 2018. The development of IOTA is primarily driven by the Berlin-based non-profit organization, IOTA Foundation. The name "IOTA" originates from the ninth letter of the ancient Greek alphabet, symbolizing tiny things and representing the smallest unit of currency issued by IOTA.

In the era of the Internet of Everything, IOTA enables devices to conduct micropayments and exchange data without incurring additional costs. The underlying structure of IOTA, known as Tangle, employs a net-like ledger structure. Unlike traditional blockchains, Tangle allows for multiple forks and does not require transactions to be specified behind specific blocks. Instead, new transactions are randomly selected for verification by referencing two existing transactions. As a result, transactions can be generated synchronously, leading to significantly improved speed and scalability compared to conventional blockchains.

An IOTA account serves as a means to prove ownership of transactions within the Tangle. Similar to a bank account, it is created using a seed, which differentiates it from traditional name- and password-based accounts. Importantly, the seed remains solely accessible to the account owner and represents their identity on the internet. This approach ensures decentralization and maintains anonymity within the IOTA network. To generate addresses, IOTA employs Winternitz One Time Signature (WOTS) for which the seed acts as the master key. Multiple private keys can be derived from the seed, and each private key generates a unique address. It is worth noting that each address can only be used once and can hold any amount of IOTA currency. The total balance of an account is determined by the cumulative sum of currencies across all addresses within that account.

The seed serves as the sole master key for proving ownership of IOTA currency within messages or addresses. It consists of an 81-character Tryte string comprising 26 uppercase English letters and the number 9. The number of possible seeds is immense, reaching approximately 8.7×10^{115}, making the likelihood of two identical seeds extremely low. A seed can generate different private keys by varying the index and security levels. There are three security levels available, each corresponding to different private key lengths. Higher security levels feature longer private key lengths, which reduce the risk of theft. Security level 1 is suitable for storing low-value IoT device data, while security level 2 is utilized for wallet transactions and high-value IoT devices. For transactions requiring the utmost security, such as exchanges, security level 3 is employed.

There are five main steps in the transaction sending process, which are described in detail as follows:

1. Generate transaction information:

The first step is to create a single transaction. You must specify an Address and a Value for each transaction. You can also define a Tag to classify different transactions. Message is the message content of the transaction. In zero-value transactions, the Message is used to trace the root address, and a Timestamp is automatically generated by IOTA. To send a valuable transaction, at least two IOTA transaction messages are required; one with a positive value to allow the receiver to obtain encrypted currency, and one with a negative value to allow the sender to deduct the transferred amount.

2. Package into Bundle:

A Bundle is a collection of transactions. After the transaction information is generated, all transactions will be packaged into Bundles. The sum of the Value of all transactions in the Bundle must be zero, and the Bundle also has a unique address for querying transactions. After the packaging is complete, use the Seed to generate the private key, and use the private key to sign the Bundle.

3. Choose two Tips:

The POW calculation must be performed before the transaction is attached to the ledger. The object of performing POW is in the Tangle, using Markov chain Monte Carlo (MCMC) to randomly select two Tips, called Branch and Trunk, and then add the Hash of the two Tips to the Bundle.

4. Do proof of work:

Perform POW (proof of work) operation on the selected two Tips, and then append the calculated Nonce value to the Bundle.

5. Broadcast to the Tangle network:

After verifying other transactions, the Bundle is broadcast to the Tangle for storage, and the new transaction is Tips, waiting to be verified by others.

2.2. Signature and Verification

As depicted in Figure 1, the signature is generated during the Bundle generation process, where the Bundle is signed to validate ownership, and the resulting signature is appended to the Signature Fragment within the Bundle. To ensure the security and integrity of the signature, the Hash value of the Bundle is first obtained, and the private key is derived from the Seed using WOTS. The length of the private key varies depending on the chosen security level. For the lowest security level, the private key length is 27×81 Trytes. The private key is divided into segments, with each segment consisting of 81 Trytes. Unequal hash operations are performed on these segments based on the Hash value of the Bundle. By converting the Bundle Hash into its decimal representation, an integer between -13 and 13 is obtained. This integer is then subtracted from 13, and the resulting values are hashed onto the segments. The combination of these hashed values constitutes the signature. It is worth noting that higher security levels result in longer private keys and correspondingly longer signatures. As illustrated in Figure 1, the first and second Trytes of the Signed Data (Normalized Bundle Hash) are represented by the letters L and W, respectively, with their corresponding decimal values being 12 and -4. Subtracting these values from 13 yields 1 and 17, respectively, indicating that Segment 1 undergoes one sponge function calculation and Segment 2 undergoes seventeen sponge function calculations. Once all the values are integrated, the resulting combination forms the signature.

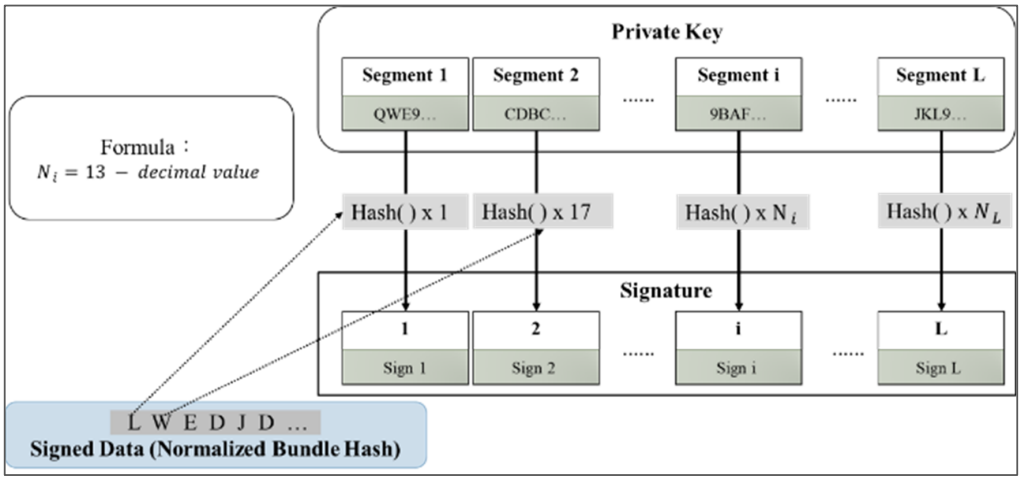

Figure 1. Transaction signature.

The node verifies the signature in the transaction through the Bundle Hash and address. The method is similar to the signature. First, obtain the value of the Bundle Hash, and then convert it to a decimal. Then add 13 to each Tryte value, and then each signature fragment in the signature corresponds to the result of the Bundle Hash operation, and the number of hash operations is different. Finally, combine the hash values of the signature fragments and perform two hash operations to obtain the transaction address. Verify that the address of the Bundle is consistent with this address to know whether the signature is correct. Because the address generation method is also WOTS, in the step of the private key, the segment is subjected to 26 sponge function operations, and the 13 − d (decimal value) operation is performed when signing, and 13 + d (decimal value) is performed again during verification. The result of 13 − d + 13 + d is 26 sponge operations. This method can achieve the purpose of signature verification without leaking the private key and seed. As shown in Figure 2, the decimal places of the first two Trytes of Signed Data are 12 and −4, and the values after adding 13 are 25 and 9. After 25 and 9 operations, the private values of Segment 1′ and Segment 2′ are obtained. Key hash value. After all the fragments are merged and hashed, the address can be obtained. If the addresses match, the transaction is valid.

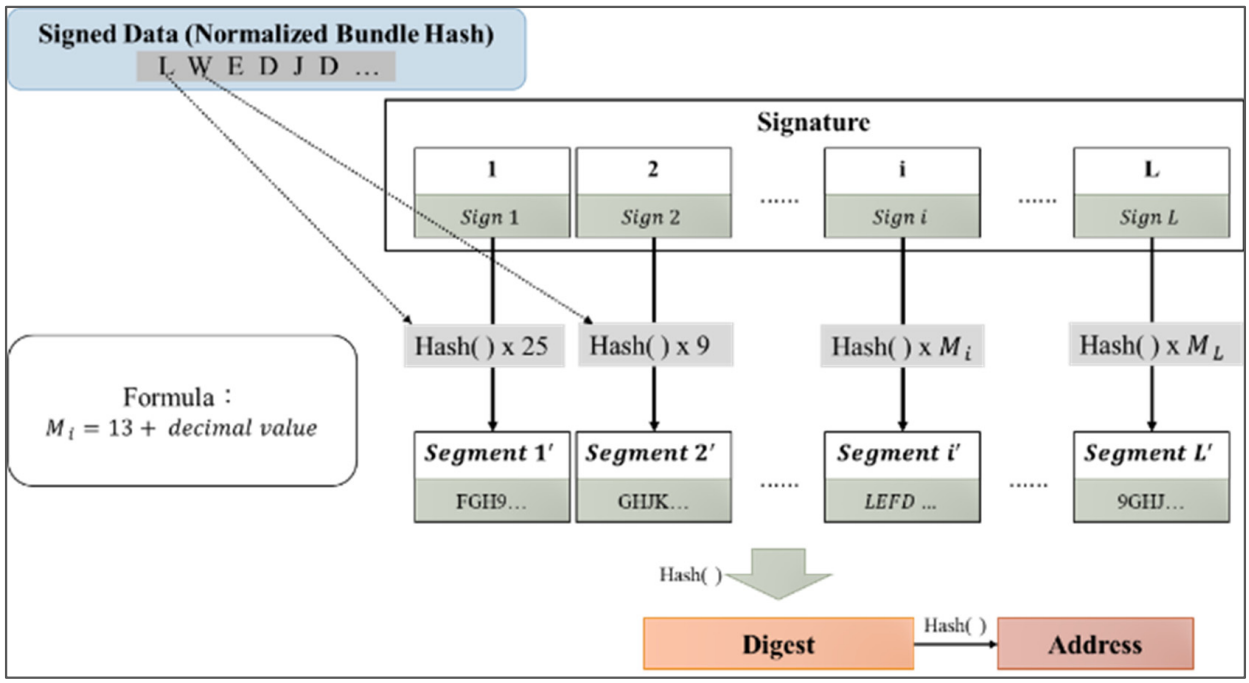

Figure 2. Signature verification.

2.3. Proof of Work

Proof of Work (PoW) is a cryptographic process used to demonstrate that a problem has been solved, granting the right to write data into the ledger. These problems are challenging to solve, but their correctness can be easily verified. PoW is employed to safeguard the network against spam attacks, as the associated computational effort incurs a certain cost. In IOTA, PoW is not performed by miners but rather delegated to nodes for execution by individuals initiating transactions. Prior to writing data into the ledger, every node must execute PoW. As illustrated in Figure 3, the Nonce value consists of 81 Trytes within the transaction message. To carry out PoW, the Nonce value must be randomly filled in, and the transaction message, including the Nonce, is converted into a Trits format for hashing operations. The resulting Minimum Weight Magnitude (MWM) of the final Trit sequence must be zero. The difficulty of achieving MWM varies based on the network. In the main network, the MWM is set to 14, meaning that the next 14 Trits after PoW execution must be zero. In the test network, the MWM is nine. When the transaction initiator node

completes PoW and broadcasts the transaction to other nodes for storage, these nodes independently perform the calculation using the provided Nonce value. If the calculation result satisfies the MWM requirement, the transaction can be written into the Tangle.

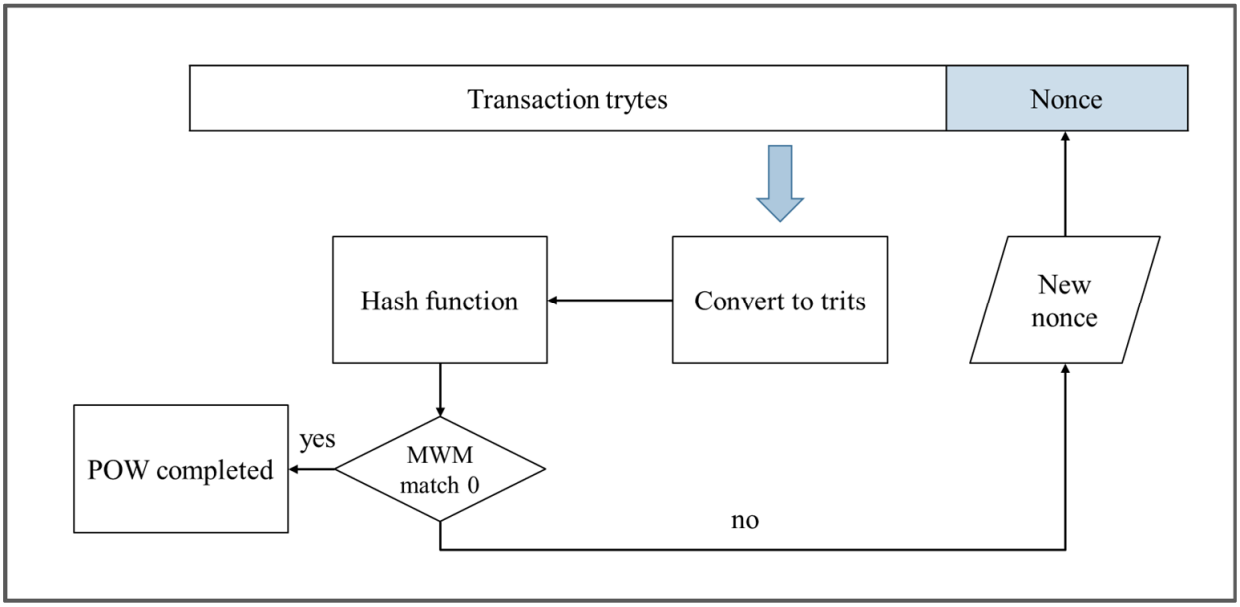

Figure 3. Proof of work.

2.4. Channel

MAM's Channel is like streaming media. Publishers can publish messages regularly, viewers can receive messages through subscriptions, and only the owner can publish messages. In IOTA, this owner is the seed owner. If the Seed is stolen, the attacker can publish messages at will. There are three modes in the Channel to control the message flow, namely Public, Private, and Restricted. In the Public mode, everyone can directly obtain the message content while in the Private message content, only the owner of the Seed can unlock the encrypted content. In the Restricted mode, encrypted content can be unlocked by using a key called Sidekey. By sharing the Sidekey, confidential information can be easily opened by specific people for viewing. The address generation methods of the three modes are not the same. Although they all use the Merkle-tree Signature Scheme, they are different in the last Root. The detailed address is generated as follows:

1. Public: Address = Root.
2. Private: Address = hash (Root), hidden messages are decrypted using Root.
3. Restricted: Address = hash (Root), hidden messages are decrypted using Sidekey.

2.5. Sponge Function

The sponge function is a cryptographic algorithm. It uses a limited state to receive input bit streams of any length. After the data are "absorbed" into the sponge, the desired result is "squeezed out", and then it can satisfy any Length of the output. Sponge function can be divided into two stages, namely Absorbing and Squeezing. The information is first input and compressed repeatedly, and then the result is repeatedly extruded. As shown in Figure 4, the values M0~M3 need to be input; after inputting M0, go through the calculation of XOR and function, then input the value of M1, and repeat the calculation until all the values are counted. Then when the value is taken out, it will go through the function calculation again to obtain the value of Z0. If the length is not enough, continue the calculation until the required length is obtained.

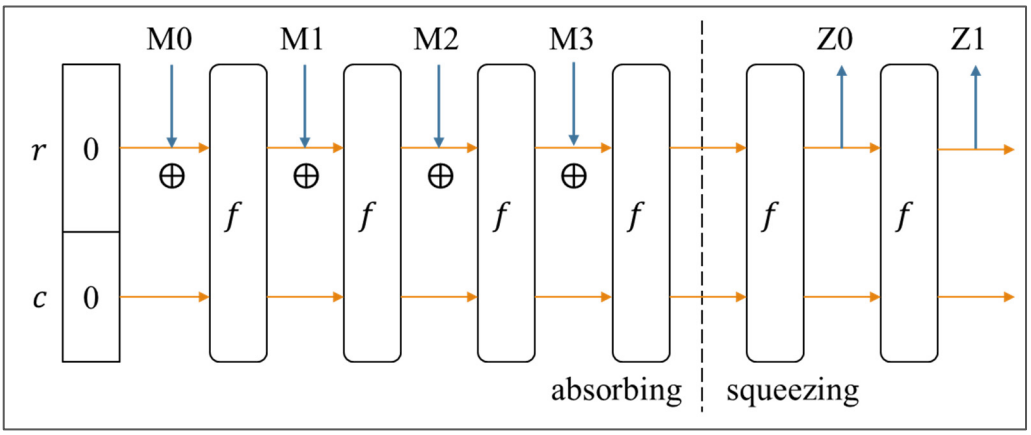

Figure 4. Sponge function.

3. Implementation

3.1. Method

The proposed study will employ a research design focused on evaluating the implementation and execution of data preservation from industrial automation and control systems to the IOTA DAG technology within the context of Industry 4.0/5.0.

1. Research Design: The research design will be based on a practical implementation and execution approach, aiming to assess the feasibility and effectiveness of utilizing the IOTA DAG technology for data preservation in industrial automation and control systems. The design will involve setting up a testbed or simulation environment to simulate real-world scenarios and evaluate the performance of the proposed solution.

2. Data Collection: Data collection will involve capturing sensor data from industrial automation and control systems. The specific environmental parameters and performance metrics to be collected will be determined based on the objectives of the study. The collected data will be securely transferred and stored using the IOTA DAG technology, ensuring data integrity throughout the process.

3. Implementation and Execution:

The proposed solution for data preservation using the IOTA DAG technology will be implemented and executed within a testbed or simulation environment. This will involve designing and deploying the necessary infrastructure, configuring the sensor networks, and integrating the IOTA DAG technology. The execution phase will focus on monitoring and evaluating the performance of the data preservation process, assessing factors such as scalability, efficiency, and cost-effectiveness.

By adopting a practical implementation approach and focusing on the evaluation of the IOTA DAG technology for data preservation, the study aims to provide insights into the feasibility and effectiveness of this solution in industrial automation and control systems within the Industry 4.0/5.0 context. The system designed in this paper is divided into three main members: Base Station, Cluster Head, and Sensor Node. The hardware configuration of the system is as follows.

3.2. Base Station

The base station is divided into two parts, namely Gateway and IOTA nodes. The Base Station runs on the Windows 10 operating system and is equipped with Intel's i7 processor and 16G of memory. The Proxy Server run by Gateway is written in node.js language, while IOTA nodes are set up using Hornet, and only the 15,600 Port is open for external node synchronization. In the data storage part, use Chronicle to store the data in Scylla's NoSQL database.

Because MAM data upload must choose a website with SSL/TLS standards, the Base Station must apply for a Domain Name and generate a certificate. To set up a node, you

must first download the public account snapshot from the IOTA website, then select a fixed node, and wait for the node to complete synchronization before uploading data to the node.

3.3. Cluster Head

Cluster Head uses a Raspberry Pi 4/8G device with a network card, as shown in Figure 5. The operating system uses a Linux system and uses the version of ubuntu-20.04.2-preinstalled-server-arm64+raspi. Cluster Head also uses node.js to write, uses mosca's suite to run MQTT Server, and uses IOTA's node.js Client Library to encrypt and sign data.

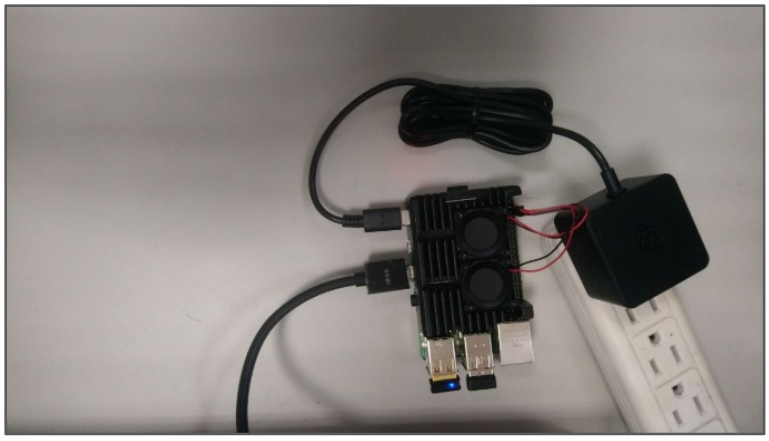

Figure 5. Raspberry pi 4.

3.4. Sensor Node

Sensor Node uses the development version of Arduino Nano 33 IoT with 256 KB of CPU Flash Memory and 32 KB of SRAM. And the part of the sensor that uses the RFID Reader of RC522, as shown in Figure 6, uses the C/C++ language to write the program.

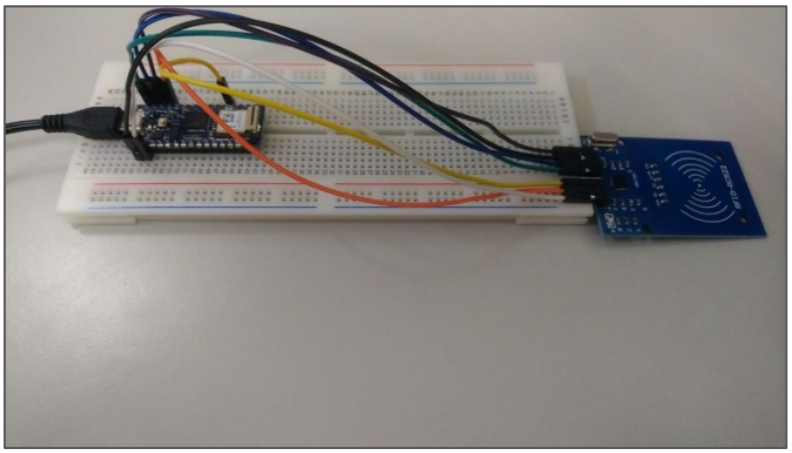

Figure 6. Arduino nano 33 IoT.

3.5. Gateway

The Gateway in the Base Station is responsible for configuring new identity data for the new device, as shown in Algorithm 1. When a new device wants to join the network, it will first indicate whether it is Cluster Head or Sensor Node, and then obtain the latest data from Channel 3, and determine whether the new device is in the blacklist. If everything is correct, Gateway will configure a new AES key and ID for the new device and obtain the whitelist of the network from Channel 2, and then append the ID of the new device to the whitelist and upload it to Channel 2.

Algorithm 1: New Device.

Input: field, ch, information
Output: device_id, encryption key
$count = 2$
$tag =$ SECONDCHANNEL
$mac \leftarrow get\ address\ from\ information$
$blacklist \leftarrow fetch\ mam\ data\ from\ channel\ 3$
If *mac in the blacklist* **then**
    $return\ failed$
else
    $fetchdata \leftarrow fetch\ mam\ data\ from\ channel\ 2$
    $ID \leftarrow generate\ random\ number\ for\ new\ device$ **If** $field\ is$ cluster **then**
        $generate\ cluster\ key\ by\ hash(BSkey\ ||ID$
    else if $field\ is\ sensor$ **then**
    $generate\ sensor\ key\ by\ hash(BSkey\ ||ch\ ||ID)$
    end if
    $channelState \leftarrow read\ channel\ detail\ from\ channel2.json$
    $channelState.count = count$
    $newdata \leftarrow pack\ ID\ to\ fetchdata$
    $message \leftarrow createmessage(channelState, newdata)$
    $mamAttach(message, 3, 14, tag)$
    $Strore\ the\ mam\ root\ to\ channel2.json$
end if
$return\ ID,\ key$

Assuming that the wireless sensor network has a detection tool for malicious devices, when a malicious device is detected, the Blacklist program can be called, as shown in Algorithm 2. Pass in the ID and MAC address of the malicious device, then obtain the latest blacklist from Channel 3 and obtain the latest network identity list from Channel 2. Then record the ID, MAC, and time of the malicious device on the blacklist and upload it to Channel 3. Finally, the malicious device is removed from the list of Channel 2 and then re-uploaded to Channel 2, and the latest root is stored in channel2.json.

Algorithm 2: Blacklist.

Input: id, mac
Output: root
$count = 3$
$tag =$ THIRDCHANNEL
$blacklist \leftarrow fetch\ mam\ data\ from\ channel\ 3$
$fetchdata \leftarrow fetch\ mam\ data\ from\ channel\ 2$
$newlist \leftarrow pack\ id\ and\ mac\ to\ blacklist$
$channelState \leftarrow read\ channel\ detail\ from\ channel3.json$
$channelState.count = count$
$message \leftarrow createmessage(channelState, newlist)$
$mamAttach(message, 3, 14, tag)$
$Strore\ the\ mam\ root\ to\ channel3.json$
If $id\ is\ in\ fetchdata$ **then**
    $newdata \leftarrow remove\ id\ from\ fetchdata$
    $channel2 \leftarrow read\ channel\ detail\ from\ channel2.json$
    $channel2.count = 2$
    $msg \leftarrow createmessage(channelState, newdata)$
    $mamAttach(msg, 3, 14, "SECONDCHANNEL")$
    $Strore\ the\ mam\ root\ to\ channel2.json$
end if
$return\ mam\ root$

As shown in Algorithm 3, when the Cluster Head wants to register the connection identity data, it will send the Cluster Head ID and encrypted data to the Gateway. Then Gateway will use the Cluster Head ID and its own key to perform MD5 operations and use the first 16 Digest as the decryption key and the last 16 Digest as the IV of the AES CBC. Then use the Key and IV to unlock the encrypted content to obtain the timestamp and the identity ID. After the Gateway verifies that the Cluster Head identity is correct and the Channel 2 list exists, a temporary certificate and the validity of the certificate will be generated. Then use the key and IV of the Cluster Head to encrypt the certificate data and send it back to the Cluster Head.

Algorithm 3: Register.

Input: CHID, encryptdata
Output: BSID, return_data
return_data = ""
concat_id ← *BSkey concatenate with* CHID
SKey ← *MD5 hash*(*concat_id*)
key ← *get first* 16 *digest from SKey*
iv ← *get last* 16 *digest from SKey*
content ← *AES CBC decrypt*(encryptdata, *key*, *iv*)
*TS*1 ← *content XOR CHID*
If *TS*1 *is timeout* **then**
    return failed
else
    fetchdata ← *fetch mam data from channel* 2
    If *CHID is not in fetchdata* **then**
        return failed
    else
        TE ← *compute the* overdue time
        Pi ← *MD5 hash the value of* (*CHID*||*BSID*||*TE*)
        TC ← *MD5 hash the value of* (*Pi*||BSkey)
        phrace ← *package TC*, *Pi*, *TE*, *and the timestamp TS*2
        return_data ← *AES encrypt*(*phrase*, *key*, *iv*)
    end if
end if
return return_data

The logged-in Pseudocode is shown in Algorithm 4. The Cluster Head will send the certificate-related data to the Gateway for identity verification and at the same time send the first root data of the MAM and the decrypted sidekey to the Gateway. When the Gateway receives the message, it will verify whether the certificate has expired and whether the Cluster Head has the correct certificate data. Then the Gateway will generate the AES CBC Key and IV and decrypt the encrypted content and, at the same time, verify whether the timestamp is within a reasonable transmission time. Finally, the MAM information of the Cluster Head is recorded in Channel 1, which is used to unlock the MAM data stream uploaded by the Cluster Head. Then Gateway stores the latest root of Channel 1 in channel1.json and records the IP, *sidekey*, and *TE* of the Cluster Head in whitelist.json.

In the part of secure communication, because the Cluster Head has been authenticated during the previous registration and login, the secure communication does not verify too much information; it only confirms whether the IP of the Cluster Head is in the whitelist and whether the validity of the certificate of the Cluster Head has expired. If it is correct, it will be transferred to the IOTA node of the Base Station, as shown in Algorithm 5.

3.6. Raspberry pi

Algorithm 6 is the Pseudocode of the Cluster Head. The Cluster Head will receive the data passed by the Sensor Node and use the ID of the Sensor Node and its own Key to perform the MD5 hash function calculation. Then the first 16 Digests of the generated value are used

as the Key, and the last 16 Digests are used as the IV. Then the Key and IV are calculated by the AES CBC to obtain the encrypted content of the Sensor Node. When it is confirmed that the encrypted content of the Sensor Node can be correctly unlocked and the timestamp is in line with the upload delay, the Cluster Head will package the sensor data and send it to the 3000 Port of the Base Station so that the Gateway will forward the packet to the IOTA node for upload. Finally, the Cluster Head will store the latest Root data in channel.json.

Algorithm 4: Login.

Input: CHID, encryptdata, C, TE, R, P
Output: BSID, confirm
$IP \leftarrow get\ ip\ address\ from\ TCP/IP$
If TE *is timeout* **then**
    $return\ failed$
end if
$P^* \leftarrow concatenate\ BSID,\ CHID,\ TE$
If $P \neq P^*$ **then**
    $return\ failed$
end if
$Compute\ TC\ by\ MD5\ hash(P\ ||BSkey)$
$C^* \leftarrow\ MD5\ hash(CHID\ ||TC\ ||R)$
If $C \neq C^*$ **then**
    $return\ failed$
else
    $concat\_id \leftarrow BSkey\ concatenate\ with\ CHID$
    $SKey \leftarrow MD5\ hash(concat\_id)$
    $key \leftarrow get\ first\ 16\ digest\ from\ SKey$
    $iv \leftarrow get\ last\ 16\ digest\ from\ SKey$
    $content \leftarrow AES\ CBC\ decrypt(\text{encryptdata},\ key,\ iv)$
    $TS3 \leftarrow get\ timestamp\ from\ content$
    If $TS3$ *is timeout then*
        $return\ failed$
    else
        $sidekey \leftarrow get\ sidekey\ from\ content$
        $root \leftarrow get\ first\ root\ from\ content$
        $Store\ IP、sidekey、TE\ to\ whitelist.json$
        $\text{root}Key \leftarrow MD5\ hash(CHID\ ||sidekey)$
        $mam\ publish(root,\ sidekey)$
        $Store\ mam\ publish\ root\ to\ channel1.json$
    end if
end if
$return\ \ confirm \leftarrow true$

Algorithm 5: MAM publish.

Input: mam transaction data
Output: forward transaction to hornet node
$IP \leftarrow get\ ip\ address\ from\ TCP/IP$
If *IP is not in the whitelist.json* ***then***
    $return\ failed$
else
    $TE \leftarrow get\ TE\ from\ the\ whitelist.json$
    If *the TE is timeout* **then**
        $return\ failed$
    else
        Forward to 14265 port where the hornet is located
    end if
end if

Algorithm 6: Cluster Head.

Input: SID, mqtt_data
seed ← *get mam seed from config*
sidekey ← *get mam key from config*
TC ← *get temporary confirm from storage*
TE ← *get time expired from storage*
If *not register* **then**
    call base station port 3001 *to register*
end if
If *not login* **then**
    create new channel by using seed, sidekey
    create mamMessage by channelState, TC, TE
    call base station port 3002 *to login*
else
    If *TE is timeout* **then**
        login again
    else
        concat_id ← *CHkey concatenate with SID*
        SKey ← *MD*5 *hash*(*concat_id*)
        key ← *get first* 16 *digest from SKey*
        iv ← *get last* 16 *digest from SKey*
        content ← *AES CBC decrypt*(mqtt_data, *key*, *iv*)
        channel ← *read channel detail from channel.json*
        msg ← *createmessage*(*channelState*, *content*)
        mamAttach(*msg*, 3, 14, "*CLUSTERHEAD*")
        Strore the mam root to channel.json
    end if
end if

3.7. Arduino Nano

Algorithm 7 is the upload process of the Sensor Node. The Sensor Node can collect data on a regular basis, but because the experimental device is an RFID Reader, the Sensor Node will passively obtain the data. When the Sensor Node obtains the data, it will obtain the Key and IV from the configuration file and convert the sensor data to hexadecimal and then encrypt it with AES CBC. Then it is sent to the Broker of the Cluster Head through the MQTT protocol. After the Subscriber of the Cluster Head obtains the Sensor Node data from the Broker, the data will be packaged and uploaded to the Tangle ledger.

Algorithm 7: Sensor Node.

Input: sensor data
Output: SID, mqtt_data
iv ← *get AES iv from config*
key ← *get AES encryption key from config*
If *WiFi is not connected* **then**
    reconnect()
end if
If *MQTT broker is not connected* **then**
    reconnect()
end if
While *read the RFID tag from reader* **do**
    hexdata ← *change data to hex string*
    *TS*4 ← *get timestamp of sensor node*
    encdata ← *AES CBC encrypt*(*TS*4, *hexdata*, *iv*, *key*)
    publish(*SID*, *encdata*) *to cluster head*
end while

4. Execution Results

The execution results depicted in Figures 7–12 provide valuable insights into the implementation and functionality of the proposed system. Figure 7 showcases the verification screen for Gateway registration and login, displaying the successfully logged-in IP of the Cluster Head. Subsequently, the MAM message is forwarded to the IOTA node. In Figure 8, the Cluster Head execution and login screen illustrate the encrypted message generated during registration and login, along with the screen displaying the first MAM message stream sent after a successful login. Figure 9 demonstrates the process in which the Cluster Head receives the message from the sensor node. Upon receiving the sensor node message, the Cluster Head repackages it and uploads it to the Tangle.

Figure 7. Base station operation screen.

Figure 8. Cluster head login and registration screen.

Figure 10 portrays the sensor node reading the sensor data. After receiving the sensor data, the Arduino Nano encrypts the packet and forwards it to the Raspberry Pi via MQTT. Once the sensing data are successfully uploaded to the Tangle, users can utilize the IOTA Tangle Explorer to query the recorded data. Furthermore, Figure 11 presents the First Root, certificate information, certificate validity, and hashed sidekey of the Cluster Head in WSNs. Figure 12 showcases the Dashboard of the node in the Base Station, enabling users to view

the current running status. The "Synced" indication on the node signifies that it has been operating synchronously with the IOTA network.

These experimental results provide visual representations of the system's functionality and data flow, highlighting the successful execution of key processes such as registration, login, message transmission, encryption, and data recording. The figures offer a comprehensive understanding of the system's operation, further supporting the proposed methodology's effectiveness and demonstrating the feasibility of preserving data from industrial automation and control systems using IOTA technology.

Figure 9. The cluster head receives the packet screen.

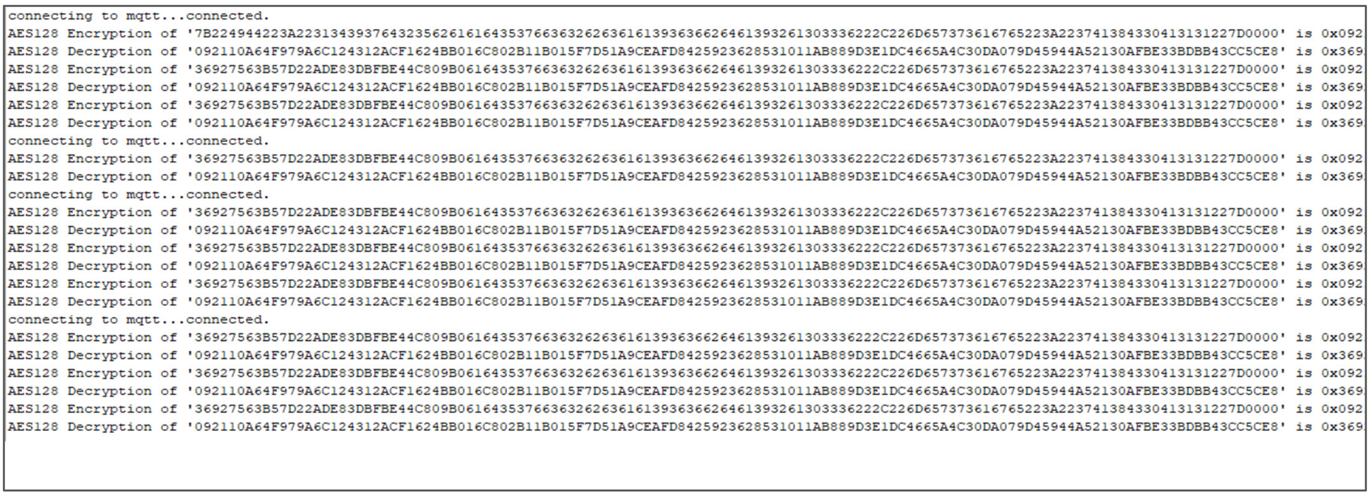

Figure 10. Sensor node upload package screen.

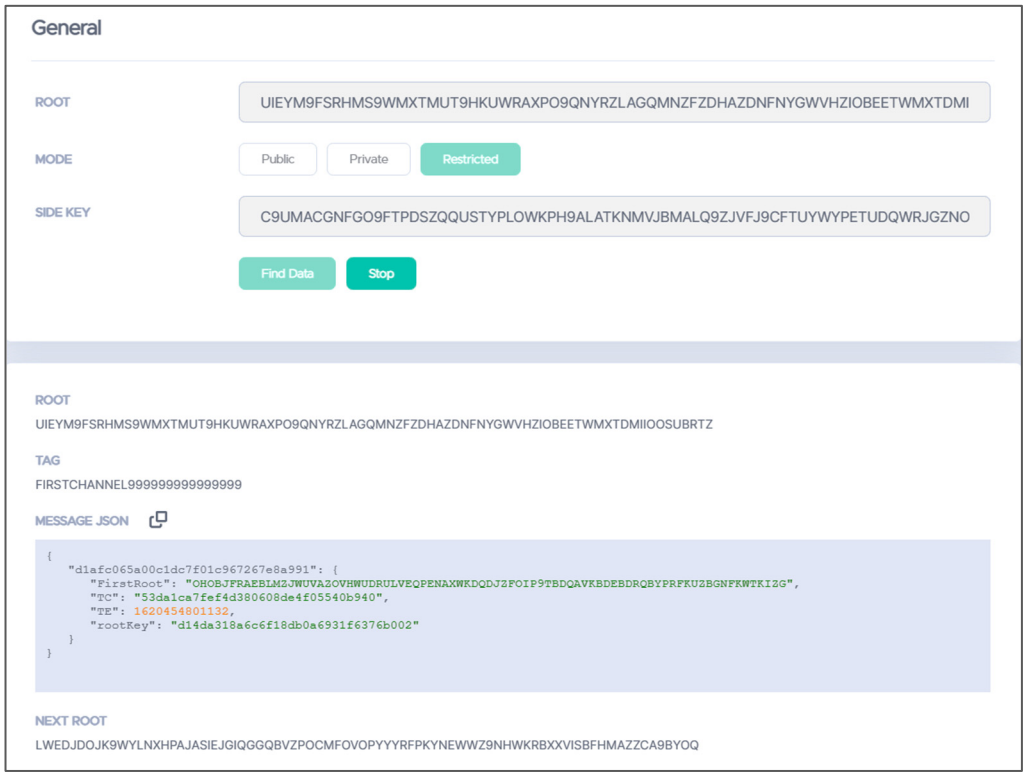

Figure 11. The content of the message stored in the base station channel 1.

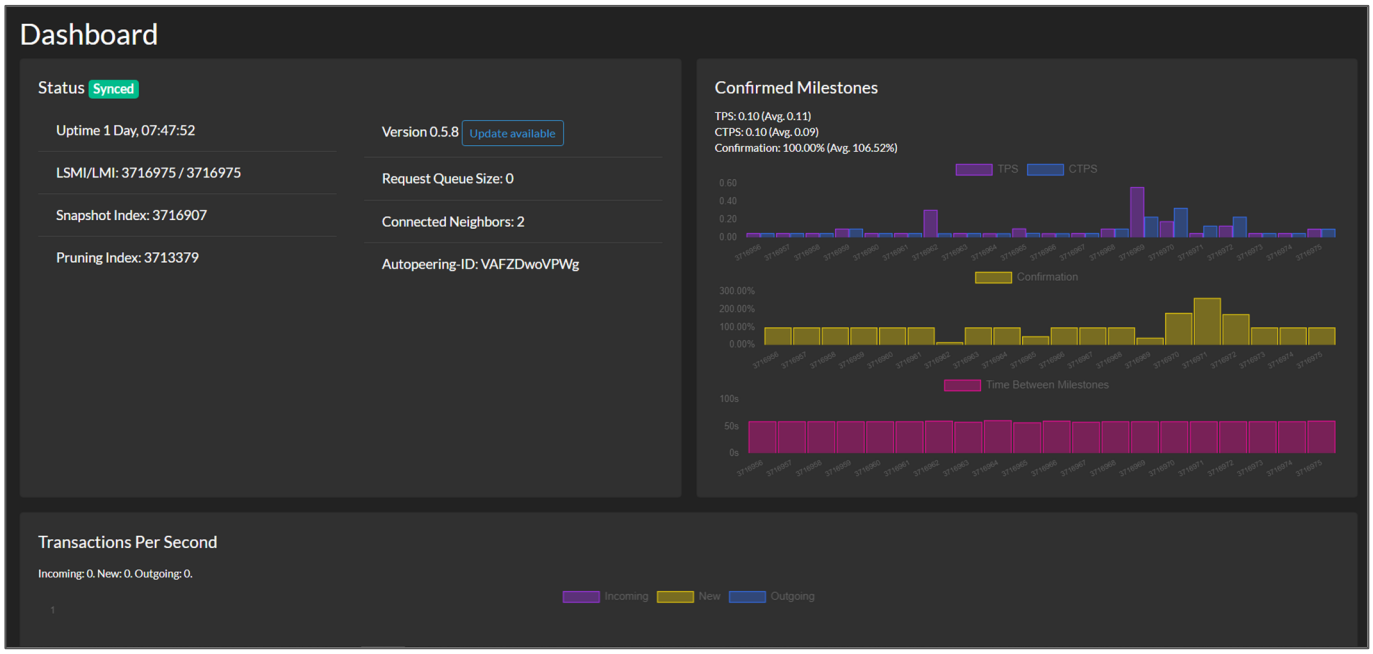

Figure 12. Node dashboard screen.

5. Conclusions

Numerous achievements have been made in the application of IOTA in industrial automation and control systems, with several papers proposing solutions to integrate IOTA into these systems. However, these solutions often overlook the constraints faced by wireless sensor networks, where many sensor nodes have limited computing power and storage capacity. Particularly in industrial control networks, sensing data must traverse multiple network layers before being uploaded to the distributed network, necessitating

the protection of security and privacy in the intermediate stages. The IOTA Foundation has proposed solutions for IoT devices, including the design of hardware devices and firmware such as CryptoCore and STM X-Cube-IOTA1. Nevertheless, the lack of hardware support poses a hindrance to the development of industrial automation and control systems as these devices still require substantial computing power. Although lightweight Bee nodes have been enhanced by IOTA for data upload actions, not all hardware is capable of supporting these nodes. Therefore, this paper focuses on conducting identity authentication for resource-constrained sensing devices and securely uploading their sensing data to the Tangle for storage. The implementation of IOTA in industrial automation and control systems is demonstrated in this paper, with satisfactory execution results.

By addressing the challenges of scalability, security, and privacy in existing centralized architectures, the proposed solution holds promising implications for various stakeholders in this domain.

1. Enhanced Scalability: The utilization of the IOTA DAG technology allows for improved scalability in managing a large number of sensors deployed in industrial automation and control systems. The decentralized nature of the DAG structure enables efficient communication and verification among the sensors, fostering a more scalable and resilient network.

2. Strengthened Security and Privacy: The integration of the IOTA DAG technology provides enhanced security and privacy measures for data preservation. The utilization of a distributed ledger ensures data integrity and confidentiality during the transmission process, mitigating vulnerabilities associated with centralized approaches. This has significant implications for ensuring the confidentiality and integrity of sensitive data in industrial automation and control systems.

3. Transparent and Immutable Data Record: The adoption of the IOTA DAG technology enables the establishment of a transparent and immutable record of the collected data. This has implications for auditability, compliance, and trustworthiness in various domains, such as supply chain management and regulatory compliance, within the Industry 4.0/5.0 context.

4. Potential Cost Reduction: The implementation and execution of data preservation using the IOTA DAG technology have the potential to reduce costs associated with importing blockchain technology into the Internet of Things (IoT). The decentralized nature of the DAG structure eliminates the need for intermediaries, resulting in potential cost savings for the stakeholders involved.

Ultimately, these implications can drive the widespread adoption of decentralized and secure approaches in the industrial automation and control systems domain, unlocking their full potential and benefits.

Author Contributions: Methodology, P.-C.T.; Software, Y.-S.C.; Validation, T.-C.W.; Project administration, I.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, grant number 112-2218-E-005-007.

Data Availability Statement: Not applicable.

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

References

1. Agrawal, R.; Verma, P.; Sonanis, R.; Goel, U.; De, A.; Kondaveeti, S.A.; Shekhar, S. Continuous security in IoT using blockchain. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 6423–6427.
2. Wang, Q.; Zhu, X.; Ni, Y.; Gu, L.; Zhu, H. Blockchain for the IoT and industrial IoT: A review. *Internet Things* **2020**, *10*, 100081. [CrossRef]
3. Datta, S.K.; Haerri, J.; Bonnet, C.; Da Costa, R.F. Vehicles as connected resources: Opportunities and challenges for the future. *IEEE Veh. Technol. Mag.* **2017**, *12*, 26–35. [CrossRef]

4. Hang, L.; Ullah, I.; Kim, D.-H. A secure fish farm platform based on blockchain for agriculture data integrity. *Comput. Electron. Agric.* **2020**, *170*, 105251. [CrossRef]
5. Guan, Z.; Lu, X.; Wang, N.; Wu, J.; Du, X.; Guizani, M.J. Towards secure and efficient energy trading in IIoT-enabled energy internet: A blockchain approach. *Future Gener. Comput. Syst.* **2020**, *110*, 686–695. [CrossRef]
6. Grecuccio, J.; Giusto, E.; Fiori, F.; Rebaudengo, M. Combining Blockchain and IoT: Food-Chain Traceability and Beyond. *Energies* **2020**, *13*, 3820. [CrossRef]
7. Popov, S.J. The Tangle. White Paper 2018; Version 1.4.3. Available online: http://cryptoverze.s3.us-east-2.amazonaws.com/wp-content/uploads/2018/11/10012054/IOTA-MIOTA-Whitepaper.pdf (accessed on 18 June 2023).
8. Ericsson. White Paper, Cellular Networks for Massive IoT, Uen 284 23-3278. January 2020. Available online: https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-networks-for-massive-iot{-}{-}enabling-low-power-wide-area-applications (accessed on 18 July 2023).
9. Alsboui, 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]
10. 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]
11. Jayabalan, J.; Jeyanthi, N. Scalable Blockchain Model Using Off-Chain IPFS Storage for Healthcare Data Security and Privacy. *J. Parallel Distrib. Comput.* **2022**, *164*, 152–167. [CrossRef]
12. Divya, V.; Sri, R.L. Docker-Based Intelligent Fall Detection Using Edge-Fog Cloud Infrastructure. *IEEE Internet Things J.* **2020**, *8*, 8133–8144. [CrossRef]
13. Singh, C.; Kumari, P.; Mishra, R.; Gupta, H.P.; Dutta, T. Secure Industrial IoT Task Containerization with Deadline Constraint: A Stackelberg Game Approach. *IEEE Trans. Ind. Inform.* **2022**, *18*, 8674–8681. [CrossRef]