IoT based Blockchain for Manufacturing Process Monitoring and Logistics within an Organisation

Julio Aguilar Jimenez, Aswin Ramasubramanian, and Georgios Psimenos

***Abstract*—With the rise of the fourth industrial revolution, there has been a dramatic change in the manufacturing industries, from the integration of cyber-physical systems and IIOT for automation to the deployment of Artificial Intelligence systems. The concern over the secure monitoring of manufacturing processes from the beginning to the point where the product has reached the client continues to be a significant challenge today. In this paper, we discuss the use of blockchain technology in secure industrial process monitoring and tracking, within a manufacturing enterprise. We also discuss secure tracking of goods from the manufacturer until they reach the client's premises. The core objective of this paper is to introduce a securely contained distributed ledger, accessible by particular people within an organization, to track complex logistical processes in a trustworthy manner. Relevant data is stored in a blockchain to safeguard it against fraudulent manipulation. The primary consensus method considered in this paper is Proof-of-Authority (PoA) with Identity. In such as system, tampering with blockchain data is impossible without revealing one's identity to the network. We herein present a prototype implementation of this system, and discuss its characteristics.**

***Index Terms*—Blockchain, manufacturing process, industry 4.0, proof-of-authority, logistics.**

I. Introduction

Blockchain technology was introduced around a decade ago as an innovative way to maintain a distributed database. A blockchains decentralized structure and use of strong cryptography, among other features, provides unsurpassed availability, transparency and trust. Although the most widely known application of blockchains today is cryptocurrencies, blockchain technology has vast potential to drive revolutionary developments across countless fields. One field that is very likely to benefit is manufacturing and industrial automation.

Conventional supply chains and manufacturing processes are becoming incredibly complex and inefficient. It is increasingly difficult to manage them using traditional methods, and there is a significant lack of transparency. In the current scenario, integration of cyber-physical systems and IoT into process monitoring and logistics enable real time tracking of materials, resource management and transport handling. Going forwards, a blockchain can offer exactly what a modern supply chain requires. Distributed consensus enables absolute integrity and eliminates disputes while providing complete transparency into the origins and chain of ownership of assets. In a blockchain-powered supply chain, any individual can trace products back to their source. Ideally, an entire products lifecycle will be recorded in a blockchain, from the processing steps in its production to the route it follows from the time it leaves the factory to the time it reaches the consumer. Success of a manufacturing organisation relies on factors which gives value addition to the product. With the rapid growth in the manufacturing industries, if a company supplies goods that are trustworthy, in the right time, quality and place it can sustain longer in the area. This implementation of blockchain in manufacturing industry provides tracking, subsequent repairs and change of ownership which are recorded throughout, enabling full traceability of the product.

In this report, we are proposing a private blockchain-enabled IoT platform for industrial automation and optimization of supply chains. A software implementation/simulation of such a platform is presented.

II. Motivation

With ever-increasing competition in the field of smart manufacturing and connected industries, businesses invest more in the fields of IoT, AI and machine learning to optimize their production system. Industry 4.0 incorporates many modern technologies. This has paved the way for a more digitalized network in the manufacturing sector, involving a huge amount of data flow. Many industries want to achieve transparency and inter-dependability, in terms of collaboration between various departments of the organization, reliable sourcing of information, product quality and standardization. Our intention is to conceptualize and implement a secure way of recording the processes involved in a manufacturing company from the beginning to the end and provide easier traceability using IoT, AI, and blockchain technology in the context of Industry 4.0.

III. General Overview of the Idea

The idea is centered around establishing a secure private blockchain which can keep track of the production process and record each manufacturing step. Manufacturing is defined as the process of converting raw materials into complete products that meet the desired specification. There are several sequential processes involved in the manufacturing of a product. Let us consider a production facility; Each raw item is given a unique product ID, order ID, and design requirements. These are tagged on the item with RFID technology. Now a block is created indicating that the raw material has entered the production line. When the raw material reaches the production line, either automated machines or the human operator scans the item and validates the block. So, the first block is added to the blockchain. Proof-of-Authority is used as the validation method in this system. The raw item has reached the first processing station. Let's asume a drilling operation is performed on the item. A new node is created. This node consists of

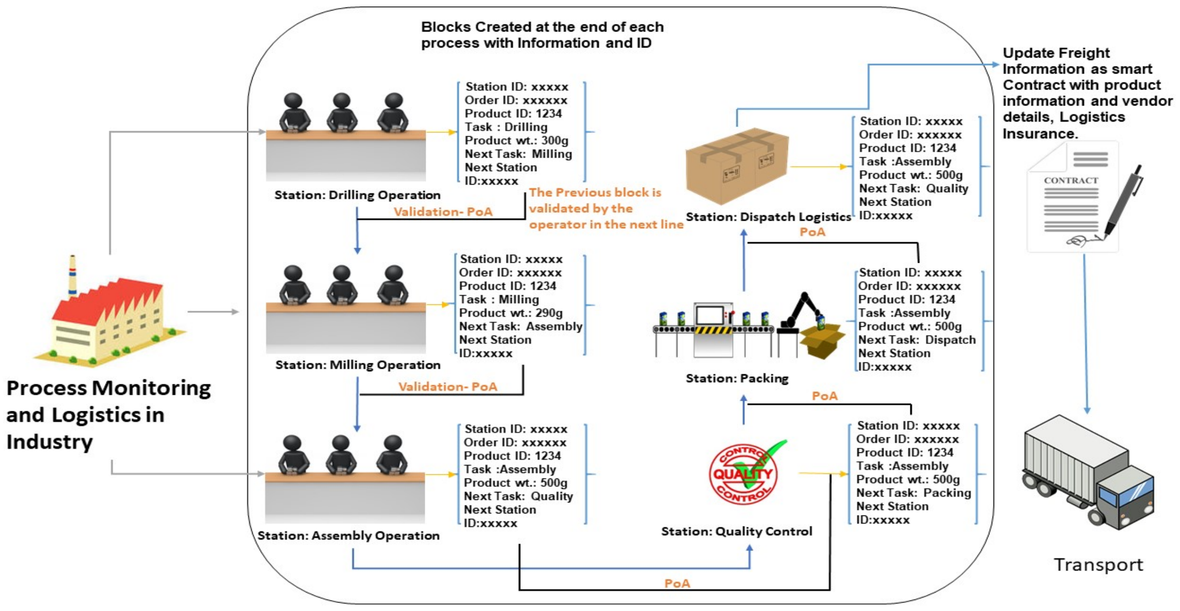

Fig. 1. Overview of Process monitoring and production control system trying to achieve using blockchain in this project

various information such as workstation ID, order ID, product ID, task information, product weight, next task, and finally the ID of the next station. The product is then conveyed to the next station. The operator in the next station validates the previous block. For the proper validation of the block, the next station ID and next task from the previous block should match the current station ID and task. If this condition is true, the previous block is "mined" and added to the blockchain. As the item advances through subsequent stations, previous blocks are validated and further operations are performed on the item. Each procedure the item undergoes is monitored and recorded in the blockchain. The third operation is the assembly, followed by the quality inspection, packing and finally logistics. In terms of logistics, there is a digitally authorized smart contract which encapsulates freight information, insurance, customer details and sender information. The logistics container is transported through predetermined checkpoints, allowing the system to track and update the package information. Furthermore, packages could be fixed with IoT sensors that provide information on package movement for real-time tracking. Thus, a secure, trusted network of freight management with distributed consensus can be achieved using blockchain technology within the industry. The network is readily scalable, and facilitates product quality assurance as well as reliable auditing of manufacturing processes, materials used and spare parts.

IV. Objectives

Our proposed system aims to provide the following core advantages:

Decentralized trust: The blockchain is distributed across a peer-to-peer network. Instead of relying on a central authority, the nodes in the network all contribute in verifying transactions and reaching consensus among them. This removes the requirement for a universally-trusted authority. It also means that no single entity can take control of the blockchain.

Resilience: A distributed blockchain is resistant to failures. The network is run by multiple nodes, and thus has no single points of failure. This peer-to-peer structure also safeguards the blockchain from fraudulent alterations. The information that is recorded in the blockchain cannot be modified or removed after it has propagated through the network and been verified by the majority of nodes.

Security: Strong cryptography is an integral part of blockchain technology. The authenticity of recorded information can be indisputably proven, as each transaction is cryptographically signed. In the context of supply chains, this facilitates auditing of individual assets or products, and adds integrity to quality assurance procedures.

Scalability: The peer-to-peer nature of a blockchain network makes it very easy to scale up or down depending on the required throughput. As a supply chain grows, or the production volume of a factory increases, the additional load can be accommodated by adding more nodes to the network. New nodes can seamlessly join at any time without any disruption to the network.

Consensus: For rapid mining of blocks within a private environment, we use a *Proof-of-Authority* based consensus mechanism with identity at stake [7]. In this method of consensus, each individual who is allowed to validate blocks is associated with an identity in the form of a personal identification number. The identity of the person validating a node is therefore known, motivating the individual to be trustworthy and honest in the task of validating blocks.

V. Protocol

A. Hashes

The hash implemented uses a SHA-256 algorithm, which is a member of the SHA-2 cryptographic hash functions designed by the NSA. It's a one-way hash, i.e., a hash can be generated from any piece of data, but the data cannot be generated from the hash. The output is a 256 bit chain, or, using a hexadecimal representation, a 64 digits long, as follows:

$$\begin{aligned} hash(hello) = {} & 2cf24dba5fb0a30e26e83b2ac5b9e29e1 \\ & b161e5c1fa7425e73043362938b9824 \end{aligned}$$

B. Signature

The signature process follows Public Key Cryptography Standards (PKCS-v1_5) developed by RSA Labs. The process is described below:

1) Generation: It requires signer's RSA private key and the message to be signed (in octet string format). The message is encoded using the EMSA encoding operation and transformed to an integer representation, upon which the RSAP1 signature algorithm is applied. Finally, the outcome is given by a signature string of length k octets, where k is the length of the modulus. In this work, a 1024-bit modulus length was chosen.

2) Verification: For verification, the signer's public key, message and octet string signature are required. The first step is to check the length of the signature, which must match the modulus length. An inverse process is then applied. It is transformed to an integer representative by means of RSAVP1 algorithm, then to an encoded message. At this point the EMSA algorithm is used to produce a second encoded message. If both are the same then, it's a valid signature. Othewise, it fails.

C. Addresses

In order to create public and private key addresses, RSA algorithms need the size of the modulus and a random object. The size of the modulus is 1024 bits and the object is created using *Crypto* library from RSA Labs. This procedure returns a pair of keys (private and public).

D. Transaction

A transaction is formed by 4 attributes; sender public key, sender private key, receiver public key and value. Value property refers to the report to be submitted. The report has a device identifier, the process that the device performs and details of any failure. Additionally, a transaction has the following 3 methods.

- *__getattr*. Retrieves a specified attribute.
- *to_dict*. Builds an ordered dictionary with the value and public keys of sender and receiver.
- *sign_transaction*. Signs the transaction using the sender's private key.

E. Block

Given the immutable characteristic of a block, it is not a class as in the case of transactions but it is created by a call to the parent class, i.e., the Blockchain class. Each block has an identification number, timestamp of creation, transactions array, nonce value and the hash of the previous block.

F. Blockchain

The Blockchain class is the core of this project and handles block creation, verification, conflict resolution etc. Its attributes are as follows:

- *transactions*. Array containing the broadcast transactions.
- *chain*. Array of blocks.
- *nodes*. Data structure containing no repeated values of the different nodes.
- *node_id*. A 128 bit unique identifier number (UUID).
- *mining_reward*. In this case it works as a counter of validations.

The methods that handle all the operations are described below:

1) Genesis block: This procedure is executed only once by the first node and represents the creation of the first block in the block chain. A call is made to the $create\_block$ method.

2) register_node: Check the url or path from a node and add it if the address is correct.

3) verify_transaction_signature: Verify the signature using the procedure explained in section V-B.

4) submit_transaction: If the verification was successful, it adds the broadcasted transaction to the transaction pool. Otherwise, it rejects the transaction.

5) create_block: This function fills the block attributes described in section V-E, removes the transactions added to the block from the pool and appends a the new block to the chain. This function automatically calculates the timestamp and the block identification number.

6) hash: Create a SHA-256 hash of the block. Before hashing the function, it sorts the keys in a dictionary to prevent inconsistencies in the hashes.

7) proof_of_authority: Grab the previous block, extract, compare and validate the hash of the block; validate station number, i.e., the material or product station tag must match with the current station; validate previous validator signature. If some validation failed, the product returns to the relevant station or it is set aside as defective; success otherwise.

8) valid_chain: Validate a blockchain. It iterates through the entire chain starting from block 1. At every stage it takes the block in the current index and the block before; the first filter hashes the last block and compares the result to the $previous\_hash$ attribute of the current block. The second filter takes all transactions in the current block, sorts the transactions and uses them along with $previous\_hash$ and $nonce$ to validate the signature and station. If a successful result is obtained, the filter moves on to the next block. Otherwise, it fails.

9) return_transactions_to_pool: After solving a conflict (chains of same length), the blocks that were invalidated return their respective transactions to the pool and the mining reward transactions are removed.

10) resolve_conflicts: Solve conflicts between chains by replacing the current chain in the node by the longest one in the network. First, it computes the length of the local chain and gets the list of all nodes in the network; then grabs the chains in the nodes through a web request. Finally it checks if one of the chains is longer and valid by means of $valid\_chain$.

If successful, the chain is replaced and the transactions of invalidated blocks are returned to the pool using the previous function.

VI. Application

A. Back-End

Python is an interpreted language that is easy to use, powerful, and versatile. Its great popularity has placed it in the first place according to the IEEE survey [1]. It is largely used both academically and industrially. Also, a large number of optimization libraries have been developed allowing it to compete with compiled languages such as Java. These reasons led us to choose it for the back-end implementation of this work.

B. Front-End

The web programming triangle (HTML5, CSS and JavaScript) was chosen given its versatility and great compatibility. JavaScript has libraries that allow presenting information in a simple way in the web interface and help us stay focused on the application.

C. Frameworks

Flask is called a micro framework because it does not require particular tools or libraries. It works as the server instance allowing the communication between the front-end and the back-end.

For visual interface purposes, Bootstrap, Ajax, jQuery and DataTables are used in this work. All of them work through the JavaScript language.

D. Features

The Blockchain implemented in this project has server, client and two web interfaces or dashboards for miners and users. Below, the main characteristics of each are presented and will be explained in depth in the following sections.

1) Client:

- Identification generator (RSA-based).
- Encrypted transaction generator (RSA-based).

2) Blockchain:

- Proof of Authority (PoA).
- Transaction encryption (RSA cryptosystem).
- Addition of multiple nodes.
- Conflict resolution between nodes.

3) Web Interface:

- Blockchain miners/validators interface.
- Users interface.

E. Client

The client allows navigation and is responsible for handling routes and communication with the user's web interface. It has a default configuration to listen to local host address 127.0.0.1 on port 8080. The port is an argument and can be changed by the user.

The client allows the creation of public and private keys for the user/validator. Furthermore, it contains the transaction class and therefore it is responsible for generating, ordering and signing each transaction that will be broadcast to nodes in the Blockchain.

F. Server / Node

The Blockchain file acts as the server, however given the decentralized nature of the Blockchain, the term node is more accurate. It works on the local address 127.0.0.1 on port 5000. The port is an argument and can be changed. Additionally, this implementation supports different nodes working simultaneously. The tasks performed by the node use the protocol in section V-F and are listed below.

1) *Validate* broadcasted transactions, all fields must be filled; if successful the transaction is sent to the transactions pool.
2) *Get* transactions in the transactions pool.
3) *Return* the current chain and its length when asked for a web request.
4) *Mine/Validate* a new block. It executes $proof\_of\_authority$, forges a new block and return the information.
5) *Register* nodes that want to join to the blockchain network.
6) *Solve* problems through consensus using *resolve_conflict* method.

The node maintains constant communication with the user through the client, so the responses to the web requests are accompanied by the following messages and web codes.

- 400. Missing values.
- 406. Invalid transaction.
- 201. Transaction created.
- 200. Ok / Successful.

G. Web Interface

The interface that is handled by the *Client* has as its main window the key generator. In it, each user can generate their key pair (private and public). In the *New Transaction* window the user can broadcast transactions to a node by filling all the fields (private, public and receiver key along with the data). After filling, a new confirmation window appears with an additional field which accepts the node address that will receive the transaction. Finally, the last window *History* connects to a node to request the history of the transactions carried out.

In the *node/server* side the main window has two tables, each with a search field to filter transactions and a selector to choose the number of transactions to display per table. Table at the top shows the transaction in the transaction pool, i.e, waiting to be mined/validated; the second table show information about the mined/validated transaction and the block number in which they are. The configure window allows the addition of nodes to the network. In each window of the interface there is a visual section to display *error messages*, as feedback to the user.

VII. Results and Analysis

The application proved to be able to perform monitoring and tracking of logistic processes within a company in a secure and immutable way. The creation, editing, signing, hashing, broadcasting, verification, conflict resolution and consensus, among other processes stipulated in the protocol, were satisfactory and sufficient to carry out the required tasks.

Taking into account the Bitcoin blockchain as a reference (given its great security and standardized protocol), an analysis of the chain implemented in this paper is presented below.

- The user public key is used as its address. Bitcoin uses a more complex system of addresses; however, given the aims and objectives of this application a more complex system is not required, unless intended to operate across several connected companies.
- The signature process in Bitcoin is covered by Elliptic Curve Digital Signature Algorithm (ECDSA), whereas this implementation uses RSA. Both are very secure, and although ECDSA slightly outperforms RSA in terms of security and space optimisation, the verification process is more than 30 times slower.
- Transactions in Bitcoin require a script language. For this project a simple transaction with one JSON object per output is enough. If it is required to have multiple data types per output, a stack-based script will be required.
- Bitcoin uses a Merkle tree to save disk space. The amount of information in this project is not as big as in Bitcoin, therefore this optimisation can be skipped unless the system is intended to serve a large number of users.
- The SHA-256 hashing is performed twice in Bitcoin, but just once in this implementation. We consider one hashing round to be sufficient, as this presents a good trade-off between security and speed.

VIII. Limitations

The application was created taking into account scalability and application of good practices. Nevertheless, given the available time and resources, it lacks the flexibility and robustness necessary to enter a production environment. The client interface lacks a tool that allows users to manage their contacts and not have to remember the public keys; also, the transaction generation form is complicated to use due to the manual filling of information, making it prone to human errors.

On the server side, layers of security and authentication must be added. The communication between client and node has been implemented with *Flask*. However, this micro-framework has many limitations and it is necessary to use a full-featured, flexible framework such as *Node.js*, to make the system production-ready.

The most important topic in Blockchain and its raison d'etre is security. The main security concerns in this application are listed below.

- Use of floating point arithmetic can lead to many problems (e.g. loss of precision).
- The implemented JSON serialization is not reproducible across platforms.
- The implemented mining method can loop forever due to integer overflow.

We have considered the case of a basic sequential process involved in production. The scale of complexity in large corporations is not fully known. Many factors might require reconsideration while implementing the system in companies involving parallel tasks, dependency based tasks, and multiple processes. Linking of different simultaneous processes at different locations, and secure validation of the blockchain is crucial and requires further work and understanding to be deployed more robustly.

IX. Further Work

An extended and standardized version of the protocol is required where the type and size of data for each field is established, the number of transactions or size in bytes per block is limited, a different JSON serialization method is used, the mining algorithm is optimized and the use of floating point is avoided.

This work can be expanded to accommodate multiple tasks and processes within a company. A reliable help desk system can be implemented and the use of smart contracts can be integrated for greater flexibility in financial processes. Bonuses can be awarded to workers who work on a commission basis and employee contracts can be integrated in the system. The possibilities are many and this project can be used as the nucleus and engine for all of them.

Furthermore, extending this project, we could manage to automate the whole manufacturing process to automatically verify data, validation and updating blockchain with integration of more sophisticated IoT sensor. This can reduce the even the minimal error that are prone due to human involvement. One of the basic advantage of this method is that it has got less mining time compared to that of other consensus mechanism and does not require more computational power. Hence these system can be implemented in small scale industries with basic structure at low cost.

References

[1] S. Cass, "The 2017 top programming languages," Jul 2017. [Online]. Available: https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf."

[3] "Pkcs-1: Rsa cryptography specifications version 2.2." [Online]. Available: https://tools.ietf.org/html/rfc8017#section-8.1.1

[4] "Script." [Online]. Available: https://en.bitcoin.it/wiki/Script

[5] "Pkcs-1: Rsa cryptography specifications version 2.2." [Online]. Available: https://tools.ietf.org/html/rfc8017#section-5.2.1

[6] "Protocol documentation." [Online]. Available: https://en.bitcoin.it/wiki/Protocol_documentation

[7] P. Network, "Proof of authority: consensus model with identity at stake," Nov 2017. [Online]. Available: https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256