High-performance blockchain system for fast certification of manufacturing data

Diogo Costa¹ · Miguel Teixeira^{2,3} · Armando N. Pinto^{2,3} · José Santos¹

Received: 24 August 2021 / Accepted: 30 November 2021

Published online: 18 December 2021

© The Author(s) 2021

Abstract

Integration of blockchain systems into industrial applications show promise in increasing security, trust, and transparency along the value-chain during product and process tracking. However, current solutions suffer performance deficiencies, or often disregard legacy devices still in operation. We propose a blockchain system built upon an IoT architecture that is secure, modular, easily scalable, and deployable for fast certification of manufacturing data, compatible with current industrial landscapes. First, the proposed architecture is presented along with elements required to manage network functions. Second, easing integration with existing manufacturing solutions, custom APIs are created and subsequently explained. This grants the platform plug-and-play capabilities, requiring minimal hardware and software configuration for deployment. Lastly, a prototype is designed to validate the solution, concluding the viability of the proposed architecture as a fast and secure certification method of manufacturing data.

Keywords Blockchain · Digital certification · Industry 4.0 · Internet of Things (IoT) · Traceability

1 Introduction

Full traceability of products and processes is one of the main goals of the Fourth Industrial Revolution (Industry 4.0), as most of its driving forces, such as a greater emphasis in resource efficiency, and a higher need for flexibility in product development and production place additional strain in Supply Chain Management (SCM) activities [1]. One technology that has shown great promise for efficiently improving the transparency and integrity of the value chain is *blockchain* [2]. Proposed in 2008 by Satoshi Nakamoto, the first widespread implementation of blockchain technologies was the Bitcoin cryptocurrency [3]. Blockchain is based on a software network fully distributed by its numerous comprising peers (*nodes*), being each able to track, verify, and create cryptographically

protected data to add to their shared ledger [4]. This *distributed ledger* effectively becomes a database containing all gathered data by all of the chain's nodes. Each one of the data entries, called *blocks*, are stored and linked in sequential order through the use of a hash-function [5].

Potential uses for blockchain outside of financial applications and cryptocurrencies were quickly found, particularly in fields where assuring authentication and certification of processes and items along the supply chain is paramount. In [6], blockchain is suggested as a mechanism for increasing supply chain efficiency through the reduction of logistic costs. Examples showcasing the use of blockchain in supply-chain management can be found in the pharmaceutical [7], alimentary [8], or in the luxury goods industries [9].

✉ Diogo Costa, d.costa@ua.pt | ¹Department of Mechanical Engineering, Universidade de Aveiro, 3810-193 Aveiro, Portugal. ²Department of Electronics, Telecommunications and Informatics, Universidade de Aveiro, 3810-193 Aveiro, Portugal. ³Instituto de Telecomunicações, Campus de Santiago, 3810-193 Aveiro, Portugal.

An increasingly common threat of cyber-attacks [10], enhanced concerns regarding data security and privacy in Industry 4.0 applications [11, 12]. Various types of attacks (e.g., DDoS, ARP, spoofing attacks, data rate alteration, and network congestion) that might violate data integrity, justify the need for a robust security architectures [13]. This holds especially true for platforms heavily dependent on Internet of Things (IoT) devices, often plagued by security, and privacy issues [14]. The operation of tracking a singular event within the value chain is a time and resource consuming action that is heavily hampered if information is skewed or absent along the intermediary steps taken in storing it. Traditional tracking schemes are typically based on centralized, client-server architectures, which introduce single points of failure, possibly compromising the entire system [13]. Therefore, digital security of manufacturing industries could clearly benefit from the application of blockchain as shown in [15, 16]. However, the success of these technologies is not only dependent on security, but also in its ability to be compliant with the current industrial landscape and performance criteria, as data must be processed and added to the blockchain in, or near to, real-time, as well as allowing for the swift consultation and verification of information. For the latter, sustainable blockchain applications include machine's digital twins [17], mixing real devices (such as sensors) with digital real time representation of themselves and the digitally representation of the logical devices and the enterprise system(s) [18].

Decentralized architectures are seen to be a better choice to achieve a high flexibility of manufacturing processes [19], typically stratified into multi-layered platforms for efficiency and scalability reasons [10, 16, 20]. These solutions allow for high modularity, with layers loosely-coupled to one another, and network reconfiguration is done easily, reducing system-down events in case of failures. However, this raises problems regarding user and device registration. In [21], these issues are partly addressed, together with other fundamental parameters when integrating with enterprise solutions, such as the hardware dependency of existing blockchain platforms. Scalability still remains as one of the major challenges for the adoption of blockchain [22], requiring additional study. Further adding complexity, the majority of Industry 4.0 implementations will be done into existing manufacturing operations, which might not be network-enabled, or compatible with modern software solutions [23], hindering direct application of novel technologies. Furthermore, replacing legacy equipment with modern, *green-field*, machinery might not be a viable economic decision. As such, studying how legacy devices can be integrated into new blockchain frameworks is also of high importance, being discussed in some detail in [20].

However, none of the studied designs can properly fulfill underlying needs. Performance insufficiencies, high latency times, extended use of third-party solutions or clear lack of scalability and flexibility enabling mechanisms, hamper viability of existing propositions for blockchain network deployment. For instance, popular blockchain implementations like Bitcoin, Ethereum, and Hyperledger, show block-interval values (mean time between the creation of two blocks) of 600, 15, and 20 s, respectively [24]. In a real-time scenario, machinery might be generating enough data for the creation of several data blocks per second. The use of microservices through Application Programming Interfaces (APIs) in blockchain systems has also been shown to improve architectural scalability [25], making the integration of new entities easier [26]. APIs normalize access to data, allowing different cross-platform technologies to call the blockchain and retrieve information or request services from the network [13]. In [24], an interoperability API is created to allow communication between several different blockchain systems, and enable the exchange of information and assets. The benefits of using APIs are twofold: (1) the communication between each layer of the blockchain follows a set of standards regarding message format and data, while not being limited to any one type of equipment, facilitating integration with the existing enterprise technical stack; (2) the use of APIs reduces the need for detailed knowledge regarding the networks's technical implementation, easing deployment and future network modifications. This allows for the creation of a robust and secure system [26].

In this study, we achieve a deep integration of blockchain technologies with existing enterprise systems and layouts, via a comprehensive use of APIs. The architecture is lightweight, modular, and high-performing with a large data throughput and real-time access. Low-level device communications are secured through an efficient symmetric key encryption scheme.

This document is divided into four sections: Section 1, is the introduction; Sect. 2, explains in detail the several layers in which the proposed architecture is built upon, as well as other network supporting elements; Sect. 3, illustrates preliminary testing subjected to the proposed architecture, and result assessment; Sect. 4, concludes discussion regarding the shown architecture and provides direction for future research.

2 Proposed blockchain framework

2.1 Architecture

To better manage network scaling, the modular architecture is divided into a four layer platform, as shown in Fig. 1.

Fig. 1 Depiction of the proposed framework's four layer architecture, network nodes, and the inter-layer communication APIs

The four layers are: the *Field Sensing Layer*, the *Blockchain and Storage Layer*, the *Network and Security Layer*, and lastly, the *User Layer*. Each one of the layers is loosely coupled, working independently from one another. Inter-layer communication is done through well defined APIs, allowing for an easier integration of new devices onto the network and coupling with other existing enterprise systems. APIs can be grouped in three manners: *Sensor/Gateway*, *Field/Blockchain*, and *Blockchain/User*, and are explained in more detail in Sect. 2.3.

By default, transactions are confirmed in real-time without the need for the creation of transaction batches following a one-transaction one-block approach. This greatly increasing speed with which the network can store data in the blockchain, effectively negating the need for a predefined block-interval, making the time between each block dependent only on the time required for establishing the secure connections between devices themselves. As data are certified promptly, a critical part or process that requires constant certification can be validated right-before, or during, immediate subsequent manufacturing stages, limiting possible bottlenecks introduced to production when implementing this blockchain certification system. However, batching may still be employed

under circumstances where non-critical information is to be stored, or when the benefits of quickly adding transactions to the blockchain are not applicable, thus limiting overall load to the network induced by non-essential tasks.

2.1.1 Field sensing layer

The lowest level of the network, and the one with highest user abstraction as end users do not have immediate access to details regarding this layer. It deals in data collection from the field using sensors and ID devices, where information regarding product, process, identification, and localization are gathered allowing for full traceability. Identification is a sensible matter as a strong bond must be established between the physical asset and the digital representation. For instance, signatures using chemical methods can be used for unitary authentication, after which twinning these signatures into the blockchain network, strengthens the trust for customers and stakeholders, even allowing, if needed, for these historical events to be trailed on the product life cycle also by third party regulators and verifiers [27]. These embedded nanotechnology tags, e.g., carbon dots based chemical signatures, associated with physical QR, may answer the vulnerability between the linkage between the physical asset and the cyber-physical digital twin representation that is not solid enough itself.

Each sensor needs to identify itself using an ID appointed by the network administrator, and attribute a timestamp to the collected data. Sensors or IoT devices that possess additional intelligence and are capable of more demanding edge computing tasks can also be used, as the developed APIs are flexible in the data types and formats that they can transmit. Although outside the scope of the current work, this enables the possibility of performing on-edge analytics that could further decrease the total amount of stored data, by performing pre-processing stages directly on data gathering devices. Furthermore, autonomous decisions could also be made at the edge level.

As a measure to increase the overall security of the architecture, each device acting in the lower level of the *Field Sensing Layer* is not connected to exterior networks, but instead operating in a local intranet. Each one of these intranets consists in one or more sensing units, forming a *Business Unit*. Connection with outer networks, and by extent, to the blockchain network, is only allowed by first interfacing with *Edge Staging Gateways*. These devices bridge communication between the lower-level sensing equipment and the decentralized ledger. *Gateways* have a wide range of interfacing capabilities, accommodating several different sensors and IoT. If required, they must interface the existing *brownfield* equipment, effectively

granting them wireless capabilities, bridging the gap closer to a *greenfield* industrial landscape.

To access the *Blockchain and Storage Layer*, a connection to external networks is required, as the physical location of devices operating as *Mining Nodes* might vary greatly. These external connections have higher security risks, therefore the successful deployment of *Gateways* is dependent on several security features. Namely, *Gateways* are capable of establishing secure connections (e.g., TLS and Hypertext Transfer Protocol Secured (HTTPS)), and use a firewall software responsible for maintaining and regulating access control of incoming and outgoing traffic to the network [13].

Appropriate hardware must be selected with enough compute power to allow a high inflow of input data coming from lower-level devices, while maintaining secure connections with other higher-level network elements. As data come from multiple sources, with different formats and at varying collection rates, if required, the *Edge Staging Gateways* may temporarily store data and normalize it, effectively functioning as a buffer or a storage proxy agent. As each node in the network is required access to all information, *Gateways* securely transmit the collected and normalized data onto every node in the blockchain network.

2.1.2 Blockchain and storage layer

Responsibilities of this layer are twofold: (1) validation of data by the *Mining Nodes*; (2) storage of the validated data. As the blockchain network aims to be as lightweight as possible, only necessary information for certifying data existent on enterprise system is retained on-chain. As such, the *Simplified Distributed Ledger* will only contain the hash information characterizing each transaction. If the hash-value of a database entry does not correctly correlate with the hash-value stored in the blockchain network across its multiple nodes, then it is possible to dismiss the entry as incorrect or as being tampered. The network is not rigid, in the sense that it must be capable of removing or deploying new nodes, conforming the architecture to current enterprise needs. Thus, it is vital to establish a synchronization mechanism across nodes, as the information kept on-chain is only relevant if it is possible to trace it back onto every node element in the network (or a representative majority).

Both the *Field Sensing Layer* and the *Blockchain and Storage Layer* work side-by-side with the existing traceability solutions at the shop floor. In fact, it is rather likely that most of the existing infrastructure can be salvaged for use with the new proposed framework, as only the *Gateway* devices will likely be added to existing systems. Furthermore, if gateway-like devices are already present,

it is merely required to interface the blockchain network via one of the designed APIs, as no rigid data format is imposed for transmission. The existing conventional database is maintained, and contains a record of all of the stored data, which means that other enterprise operations that are fed by this data remain unaltered.

2.1.3 Network and security layer

The *Network and Security Layer* involves all underlying technologies that concern secure operation of the network, like the implementation of cryptography tools and transaction encryption. All communications, except for ones occurring in the lower levels of the *Field Sensing Layer*, are protected using the TLS protocol (version 1.3) with both server-side and client-side authentication required, establishing secure HTTPS connections. A *Certification Authority* (CA) must be used to sign the X.509 certificates that grant access to the network via TLS. The CA has to be established as a trusted institution that guarantees the identity of the network's participants, and will be responsible for validating all of used certificates [28]. Devices that do not establish connections with insecure networks, but use wireless interfaces (i.e., IoT) are still susceptible to attacks such as snooping, and require some sort of protection. Due to the limited computational capabilities possessed by most IoT, lighter symmetric-key encryption algorithms like Advanced Encryption Standard (AES) are used.

Other management roles credited to this layer include establishing and enforcing rules dictating the blockchain's validation mechanisms (i.e., consensus algorithm) for the creation and approval of new blocks. Furthermore, information restriction to external users of the network must be regulated, as it is expected that in-house users of the network are granted a higher level of intelligence detail than third party entities.

2.1.4 User layer

This layer provides validated users access to the records gathered, and enables real-time monitoring of the network through the front-end interface that constantly receives data from the APIs. Two types of user permissions are considered, with each rendering a different version of the front-end interface: private permissions and public permissions.

Private permissions are only granted to in-house users, having a higher level of access to information stored on the network. Alternatively, public permissions can be granted to external agents, allowing them access only to the required data, blocking out additional features. This selection comes from enterprise decisions to assure

process confidentiality. In-house or internal users might be internal plant departments, external plants, or corporate users. External users may be suppliers, partners or even the final consumers.

2.2 Characterization of network nodes

Network nodes can serve one of two functions: administrative responsibilities, or block mining. *Mining Nodes* are responsible for creating (*mine*) blocks, and can be deployed across multiple cells in a production line, different production lines, departments, factories or even organizations. As the number of *Mining Nodes* increases, so does the overall trust of the network, as more entities exist to validate and share the common ledger. On the other hand, *Administrator Nodes* are reserved for the sovereign entity that fundamentally holds legislative power over the network—true for a private blockchain system.

2.2.1 Blockchain administrator

Serves as a mediator between all the devices acting on the blockchain network, whether they are *Mining Nodes*, *Gateways*, or even front-end users. The main roles performed by the *Blockchain Administrator* include managing message flow between the several layers and the nodes; and validating front-end users' access to the network, regulating information to which they have access to. Fundamentally, the administrator will "produce, maintain, and configure the decentralized applications" [13], taking decision actions proactively, based on the visual synoptic representation in real time and align optimized manufacturing processes, according to behavior, without the need of time-expensive complete physical mock-ups [17, 18]. Communication and access logs are also created and curated by the *Administrator Node*, allowing for future fail analysis to be conducted, and the creation of user log-in analytics.

Lastly, the *Administrator Node* allows access to the system administrator's Human Machine Interface (HMI), where it is possible to conduct such tasks as enrolling a new *Mining Node* or *Gateway* onto the network—admittedly, of course, that the to-be enrolled entities have provided valid certificates. This, however, does not mean that the *Blockchain Administrator* has influence over the actual data that circulate within the network, as administrators only have a data-viewing privilege.

2.2.2 Mining nodes

Mining Nodes are responsible for creating new data blocks, according to the governing rules deployed on the network. To increase the solutions modularity, they are easily deployed with minimal configuration, in a *plug-and-play*

fashion. Likewise, the network is self-sustaining, requiring little in the way of maintenance: *Mining Nodes* are required to solve any conflicts that may arise in terms of block acceptance as well as network synchronization between nodes, autonomously.

Each node will keep two copies of the simplified shared ledger: one is kept in RAM for fast data access and quick response to inquiries, done for instance by an user or *Administrator Node*; while another is kept in a auxiliary database as a backup. This allows for both fast data transfers and calculations, while assuring data integrity in case of system failure or power loss. The simplified shared ledger consists only of the block headers, as the bare minimum information for reconstructing the transaction-tree and later provide digital certification to transactions is stored in a decentralized system, mitigating storage overhead. With block bodies kept within traditional centralized services, necessary to avoid added interference with current manufacturing routines, the required information to reconstruct all of the collected data and transactions is maintained, yet still enabling the detection of possible tampering attempts to past records.

Seminal blockchain consensus algorithms such as *Proof-of-Work* (PoW), were considered for application in the proposed architecture, however, due to the permissioned nature of the network, the trade-off offered between performance and security is not advantageous [29]. Therefore, blocks are approved according to a variation of the *Proof-of-Authority* (PoA) consensus algorithm, taking advantage of the high-level of established trust that inherently can be found in a permissioned private network, and yielding much higher data throughput. Any given miner cannot mine two consecutive blocks, with the miner that has not mined a block the longest, first in line to forge a new block. A schematic of the operation of the proposed consensus mechanism is shown in Fig. 2. *Mining Nodes* must agree upon an order for which

Fig. 2 Diagram of the node mining order used to achieve consensus in the network

to mine the blocks. This order is automatically adjusted each time a new *Mining Node* is added or removed from the network, and if the order is not maintained by a given node, the forged block will not be accepted. The use of PoA does not only facilitate network scaling, as it is less resource intensive for the network, it completely removes the long wait times between block creation, allowing for a much higher block cadence. Please note, however, that the terms *mining* and *Mining Nodes* are maintained from the current literature in a broad sense to describe the act of creating a block and the elements responsible for block creation, respectively, even-though the traditional mining process (which involves solving a difficult computational problem) is not present in this implementation of PoA, where no competition is introduced between nodes with very heterogeneous capacities.

Algorithm 1 depicts the steps taken by the *Mining Nodes* in the process of creating and appending a new data block to the blockchain, based on the above shown consensus algorithm.

Algorithm 1: Creating data blocks using the proposed consensus algorithm

Input: Data payload from gateways. Ordered *node_list* of all nodes on network.

Output: New blockchain data block.

begin

└ Administrator creates a new "mine" order;

forall nodes do

└ **if** *internal chain in sync* **then**

└ *temp_block* ←

└ *payload, prev_hash, timestamp*;

└ Calculate hash value of *temp_block*;

└ **if** *node_ID == next_sender_ID* **then**

└ Sign *hash_temp_block*;

└ Send *hash_temp_block* to all nodes;

└ **else**

└ Check signature of received

└ *hash_temp_block*;

└ **else**

└ Request sync;

if *signature OK in > 60% of nodes* **then**

└ **forall nodes do**

└ Append *hash_temp_block* to

└ *simplified_distributed_ledger*;

└ *next_sender_ID* = next in *node_list*;

└ Append *temp_block* to *blockchain*;

2.3 Deployed API

The created set of APIs are categorized according to the layers for which they serve as an interface.

2.3.1 Sensor/gateway API

Used only for data transport between IoT and sensing devices with wireless capabilities and the *Gateways*. As most of these low-level devices are not capable of establishing TLS connections, information payload is encrypted beforehand using symmetric encryption, and decrypted at the *Gateway*, through the method shown in Sect. 2.7.

2.3.2 Field/blockchain API

APIs acting on this group are responsible for maintaining the blockchain's functionality. For instance, and following the procedure shown in Algorithm 1, *Gateways* must send payload values—*/submit/new*—to *Mining Nodes*, initiating the block mining procedure. The *Blockchain Administrator* will then send a mining request—*/mine*—and wait for confirmation from the remaining network nodes - */mine/confirm*. In case a block is not accepted by the nodes, an error response is returned identifying the dubious node.

Nodes need to maintain a synchronous state between them, both in terms of internal ledgers and of the mining order. To achieve this, whenever a new node is added or removed from the network, a

/nodes/synchronize request must be sent.

2.3.3 Blockchain/user API

This group of APIs allows users to control and retrieve information from the blockchain network. Both *Mining Nodes* and *Gateways* need to be notified of modifications done to the network—e.g., */gateways/add* or */nodes/add* routes add elements, and */gateways/remove* or */nodes/remove* routes remove elements from the network and notify existing participants. Furthermore, *Mining Nodes* are also capable of receiving a */check/hash* request containing hash-values wished to cross-check against their internal ledger.

Other deployed APIs serve to feed real-time data to the user via the front-end dashboard, manage user accounts, and real-time data analytics. Requests can be done to */graph/node/data* or */graph/time/data* routes to retrieve information regarding the distribution of blocks created per *Mining Node* or the number of blocks created per hour and the time taken between creating each block, respectively. To access users' network activity logs, the */get/user/data* call can be performed. For data-analysis operations, requesting the */get/dashboard/data* route will generate a response with the following information: *id* or hash-value of last block on chain; timestamp of when the final block on chain was created; number of blocks in the blockchain; average block size [MB]; total space taken by the blockchain [MB]; total number of traced parts; number of nodes

on network; number of gateways on network; etc. These APIs can be accessed in one of two ways: either by calling them directly as an user and expecting a response (e.g., through a web-browser), assuming the user has a sufficiently high clearance level; or through internal enterprise data-gathering processes that feed data into other user interfaces.

2.4 Database structure

One of the architecture's main requirements is flexibility, and as such a malleable approach to handling data is crucial. With that in mind, a non-relational, or also commonly referred to as non-SQL or NoSQL, database is used. This brand of databases does away with the tabular relations used in relational databases in favor of other data modeling means. Three different types of collections, comparable to tables in a relational database, can be identified as in use in the designed framework. These include: (1) the main collection of data making up the entirety of the blockchain information; (2) the nodes' simplified distributed ledger backup; (3) and an user information collection, storing users' logs onto the network and usage statistics.

Although some drawbacks are associated to non-relational databases, such as the cumbersome querying mechanisms resulting from the lack of tabular relations, and the potentially higher usage of disk space when compared to a Structured Query Language (SQL) database, this does make the proposed architecture more flexible, accommodating a larger variety of scenarios without any reconfiguration whatsoever.

2.5 Block topology

Each block is comprised by two main components, the block header and the block transactions, as seen in Fig. 3. The block header contains the calculated hash-value corresponding to the "block transactions" portion of the block, as well as a timestamp of when the block was forged. The block headers are used to create the *Simplified Distributed Ledger* in the nodes. As for the block transactions, it encompasses the sensor payloads wished to be stored. As a non-relational database is being used, there is a certain freedom as to which format the stored data follows. However, for an example use-case, it is of reasonable assumption that the metrics shown in Fig. 3 are of crucial importance.

2.6 Integration with enterprise systems

The proposed framework works alongside conventional enterprise traceability technologies, as illustrated in Fig. 4. In fact, the proposed architecture aims to maintain part

Fig. 3 Block topology in the proposed framework, depicted with an example payload

of the current infrastructure, with equipment pertaining to the *Field Sensing Layer* being unaltered from existing systems. The only caveat is that these systems must now also interface the higher level blockchain network through the devised APIs. This process is aided by the plug-in-play nature of the nature, requiring no action to be done to the code-base for new deployments, with all tasks being configured through a simple configuration file.

Data collected from the *Field Sensing Layer* will still have to be stored in the existing corporate databases, fundamentally translating into storing block bodies within the existing traceability systems. Additionally, information in the traditional systems will have to be cross-checked with the records found in the distributed ledger of the blockchain, via the use of one of the *Blockchain/User APIs*. The way in which data are saved has to remain similar between existing traceability systems and the proposed blockchain-based system, ensuring that hash-values can be easily determined and cross-checked between the two systems, however, due to the flexibility in chosen database storage engines, the proposed blockchain system is flexible to different data types without much hassle. Furthermore, the blockchain system is capable of working at the speed which the centralized services already operated, therefore a new bottleneck will not be introduced, in spite of the added benefits of digital certification being possible. Additionally, as both centralized and decentralized services are concurrent and loosely coupled, a system down state in

Fig. 4 Framework integration with current enterprise systems

either of them does not compromise the other in terms of data security or reliability.

2.7 Securing low-level hardware and IoT

Even though IoT and sensing devices with wireless capabilities are only connected to local intranets, these are still susceptible to data tampering or snooping attacks, and thus upstream data payloads are required some sort of encryption. Due to the computational constraints of these devices, efficient encryption tools must be used. Novel schemes for assuring secure communication in IoT through TLS have been studied and proposed in [30–32], however it is not assured that they are applicable to the most constrained of devices. As such, the encryption system shown in Fig. 5 is proposed. It is based on AES symmetric key encryption, which can be implemented even in 8-bit processors, as it only uses simple bitwise XOR and byte-shifting operations [33]. The mode of operation selected for AES is Cipher Block Chaining (CBC).

Fig. 5 Diagram of proposed encryption system used to secure IoT payloads

The encryption key is obtained for each one of the devices using their ID, with both the IoT device and the Gateway being aware of this key. Gateways will only accept as being valid encrypted payloads sent by white-listed low-level devices: after decryption, the message must contain a reference regarding the device that authored it. To further increase security, encryption keys should be renegotiated between the devices periodically.

3 Implementation and testing

3.1 Blockchain framework implementation

A wide range of technologies are used in the proposed architecture's prototype implementation, both in terms of hardware, as well as software solutions. Figure 6 summarizes used tools and exemplifies application in an existing production line. To simulate operations, an Arduino Uno R3, ESP8266 and ESP32 were used. This also demonstrates how the framework is programming-language agnostic, with both C/C++ and MicroPython being used for the sensing devices. Gateways were based on a Raspberry Pi Model 2 B+ running Ubuntu Server 18.04 LTS.

Mining Nodes are built on top of the Flask web micro-framework with secure TLSv1.3 connections enabled. A custom *Certification Authority* was created using the OpenSSL v1.1.1f TLS/SSL toolkit to validate the X.509 certificates. Lastly, a MongoDB non-relational database is employed as the storage solution.

3.2 Testing the framework

For testing, two distinct methodologies were used: firstly, the prototype platform was assembled and used to test the network's functionality; secondly, a simulation program was devised to test response times and stress-test against possible shortcomings of the architecture. As the platform is based on physical hardware, it requires input signals to operate, therefore testing done by software allows for a greater exploit of the architecture's limits.

Fig. 6 Framework implementation schematic, including used tools, hardware, and software

3.2.1 Performance testing

Performance was tested for a varying amount of *Mining Nodes*, by measuring response times of the database querying feature of the front-end interface. Seeing that the network querying operation not only depends on the acting speed of the *Blockchain Administrator*, but also of the *Mining Nodes* and of the MongoDB database, it becomes representative of the latency seen across the network. To see how well adding additional *Mining Nodes* scale in the network, a test sample of 3, 5, 10, 15, and 20 nodes is used, representing in this case a factory setting with up to 20 production lines, each acting as a network node. Each simulated production line will have a *Gateway* pinging data at a random interval between 0.5 and 3 s. The *Gateways* have a payload of five sensor value readings. Request Timings, i.e., the time taken for a request to be sent and a response to be received, was measured using Mozilla's *Firefox Developer Tools*.

Queries sampled for testing were of three types: by block number; by hash-value; or using each part's ID. Each manner of blockchain query produces a different impact on the network. For instance, queries that use a block number or hash-value directly (as seen in Figs. 7 and 8,

Fig. 7 Request time relative to total number of nodes operating in network, when querying by Block Number

Fig. 8 Request time relative to total number of nodes operating in network, when querying by Hash Value

Fig. 9 Request time relative to total number of nodes operating in network, when querying by Part ID

respectively) are comparably much faster than those done using a part ID, Fig. 9. This is due to the amount of data being gathered—it is expected that a part ID returns more than one match on-chain versus one exact result for the hash-value—and differing amounts of accesses required to the enterprise or centralized data-storage systems. In fact, these queries contemplate requesting and comparing information from across all participating nodes in the blockchain to determine if in fact it is an approved and

validated transaction across a majority of the network. This level of redundancy is essential as there may be several acting entities that require certification across multiple organizations displaying some level of distrust in each other. For example, an external auditor may request the digital seal of quality for a given critical part, and is thus unwilling to trust only one node of the network (which, hypothetically, could have been tampered by one of the participating entities). As we need to cross-check all nodes, this will certainly lead to an increase in the network latency, however, it is still marginal in effect for users. All of the shown results were based on the average of 100 samples for each one of the number-of-nodes combinations. Figure 9, contains the worst scenario found, which is when queries are done by part ID, in which three possible matches are found on-chain and presented to the user.

The data shows a roughly linear increase of request processing times, with the number of deployed nodes. This linearity is expected to hold true even for an extremely large number of nodes, as querying all nodes in a network is done in $O(n)$ time. The processing time taken by each node for the same payload is approximately the same across all other nodes. Nevertheless, for future iterations of this work, this relationship will have to be studied in further detail for larger networks.

For this network, which is somewhat large in terms of permissioned blockchain standards, latency times are below those that are noticeable for the end user. To see where most of the time is being spent, for each request, distribution of time between each stage of a HTTP request was plotted for the most extreme scenarios—3 and 20 nodes, Fig. 10. It becomes clear that in both cases, the *Waiting* procedure is the biggest bottleneck, contemplating the time taken by the server, i.e., the *Blockchain Administrator*, to gather the queried data and check hash-values with each one of the *Mining Nodes* individually. Remaining of stages stay consistent with each other, regardless of number of nodes, which is to be expected.

Querying the network by part ID is the most work-intensive task that can be requested. Not only are all nodes always asked to cross-check their internal ledgers with given hash-values, as in the case of query by Part ID, typically, more than one hash-value has to be validated, with this operation also taking roughly three times as much to render results, compared to other queries. Nonetheless, user experience of the front-end is still perceived as quick. TLS connections were found to introduce very little overhead in the final, implemented architecture.

Testing with multiple *Gateways*, also shows little to none influence over query times. However, due to the synchronous process nature of the Flask framework, complications could arise when two *Gateways* submit a transaction for mining at roughly the same time. For commercial deployment, instead of using the built-in web-server of the Flask micro framework, a commercial grade server should be used. This transaction process is eased, as Flask's main objective is to be deployed within a Web Server Gateway Interface (WSGI) server, such as Heroku, Microsoft Azure, Google App Engine or Apache. Nonetheless, to manage multiple *Gateways* submitting data, within a very short interval of each other, an asynchronous server should be deployed. Another alternative would be the use of a queuing method at the *Mining Nodes*, to avoid having two nodes mining blocks at an almost simultaneous time, or employing an automatically adjustable buffer functioning as a temporary database proxy at the *Gateways*.

As *Mining Nodes* keep two simplified distributed ledgers, one on RAM and another as a backup in a MongoDB collection, RAM usage was measured. To test this, a simple Python script was devised to mimic procedures in the *Mining Node*. For a total number of 17 million blocks, roughly 0.147 GB were occupied in RAM. Choosing MongoDB for data storage has both advantages and disadvantages. For one, not requiring a strict data format is an obvious advantage, allowing to miss-match different information all within the same collection. This does come at the cost of storage space and query

Fig. 10 Request timing subset stages division for a 3-node and 20-node network, when querying by a Part ID

speed. If a SQL database were to be used, total storage space would be considerably smaller. However, during all the tests performed, database read and write speeds were never an issue; even when a very specific query was made (e.g., a certain payload value within a block). To reduce storage size, small alterations were done, such as using hash-values as *_id* fields (required by default by MongoDB), and by shortening the name of each field (e.g., *pl* instead of *payload*).

3.2.2 Security analysis

Wireshark was used to test if correct network protocols are in place, namely if connections being established are in fact TLS encrypted. As all software procedures for communication follow the same routines, testing was done mostly on what might be considered one of the most data-sensitive operations: user input of the login password onto the front-end interface. Both Wireshark and Mozilla's *Firefox Developer Tools* detect that connections are correctly being established with TLS 1.3. The TLS protocol sees widespread adoption across multiple Internet applications [34]. The used version, 1.3 is an improvement of the previous 1.2 version, benefiting from stronger security guarantees with increased identity protection of both client and server [35]. We can encounter several examples in literature that assure the soundness of the TLS protocol, namely the security of the handshake stage [36, 37]. IoT devices communicating in wireless connections also have encrypted payloads, albeit in a symmetric-key scheme, which were also confirmed as hidden from snooping attacks.

Exposure to outside networks is a possible point of attack, with exploits including malicious data injection or Distributed Denial of Service Attacks (DDoS). To combat this, these devices are only allowed contact to trusted IP addresses, with this operation being managed by the firewall software. Furthermore, a request limiter is implemented, which throttles the number of requests per IP address.

This same principle is also applied to the web front-end, with an artificial timeout being created for the login page, avoiding bruteforce password generation attacks to gain access to the blockchain. This timeout limits access attempts to one per session per user per second. In case of a successful attack, user information is stored not in plain-text form, but as an hash value, meaning that if a malicious agent gained access to this data, it would be of no value.

4 Conclusions

In this paper, we propose a blockchain architecture for fast traceability of manufacturing data, with high emphasis on assuring backwards compatibility with

existing enterprise solutions. To achieve this, the network was created from scratch using open-source technologies built upon an IoT stack, with abstraction regarding technical solutions being assured through the implemented APIs.

The architectural prototype performed adequately, accomplishing all initially set-out objectives. Mining and access of data happens at a fast pace, allowing for fast transactions and certification. By choosing a highly supported general-use programming language, Python, for the framework's foundations, it not only allowed for compatibility across multiple platforms, as also supports future expansion of the architecture by easily adding or improving functionality. Compatibility with multiple different interfacing devices, such as sensors or IoT, was also assured through the above-mentioned APIs.

With minimal alteration, this framework could be used to highly strengthen the security of IoT powered network architectures, typically associated with serious security shortcoming. Security measures were implemented all throughout the solution, from device-to-device communications; personal user-data; to the securing of used databases. Added overhead from the implementation of the blockchain is not negligible, however this is considered a fair trade-off between performance and security.

Node scalability and performing real-time changes to the network layout is also guaranteed, as special care was given into adding a degree of abstraction between each network component. Lastly, this is a cost-effective way of strengthening any existing network, with the only expenditure being associated with cost of hardware used for testing and time required for development, as all software is open source.

5 Future work

In future iterations of the platform, scaling and testing for even larger networks with more nodes is required. Furthermore, the performance could still be improved upon through more optimizations of the *Mining Nodes* to accept a larger number of simultaneous connections from *Gateways*.

In spite of security advantages associated with using blockchain platforms, security remains a primary concern. Recent academic development focuses on the chain structure, the consensus algorithms and the cryptographic techniques, usually classified with three levels: process, data, and infrastructure. Many researchers addressed these approaches; however, the work remains open for future conclusions and development from other perspectives, including signatures schemes and privacy protection [38].

Acknowledgements This work was partially funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020-UIDP/50008/2020 (actions QUESTS and QuRunner)

Author contributions Diogo Costa and Miguel Teixeira contributed to the study, conception and design. Material preparation, data collection and analysis were performed by Diogo Costa and Miguel Teixeira. The first draft of the manuscript was written by Diogo Costa and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Lasi H, Fettke P, Kemper HG, Feld T, Hoffmann M (2014) Industry 4.0. *Bus Inf Syst Eng* 6(4):239–242. <https://doi.org/10.1007/s12599-014-0334-4>
- Chod J, Trichakis N, Tsoukalas G, Aspegren H, Weber M (2020) On the financing benefits of supply chain transparency and blockchain adoption. *Manage Sci* 66(10):4378–4396. <https://doi.org/10.1287/mnsc.2019.3434>
- Nakamoto S (2009) Bitcoin: a peer-to-peer electronic cash system. Cryptography Mailing list at <https://metzdowd.com>
- Zheng Z, Xie S, Dai H, Chen X, Wang H (2017) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: *Proceedings—2017 IEEE 6th international congress on big data, BigData congress 2017*, pp 557–564. <https://doi.org/10.1109/BigDataCongress.2017.85>
- Restuccia F, D'Oro S, Kanhere SSDaS, Melodia T, Das SK (2019) Blockchain for the internet of things: present and future. *arXiv* 1(1):1–8. arxiv.org/abs/1903.07448
- Perboli G, Musso S, Rosano M (2018) Blockchain in logistics and supply chain: a lean approach for designing real-world use cases. *IEEE Access* 6:62018–62028. <https://doi.org/10.1109/ACCESS.2018.2875782>
- Soltanisehat L, Alizadeh R, Hao H, Choo KKR (2020) Technical, temporal, and spatial research challenges and opportunities in blockchain-based healthcare: a systematic literature review. *IEEE Trans Eng Manag.* <https://doi.org/10.1109/tem.2020.3013507>
- Casado-Vara R, Prieto J, La Prieta FD, Corchado JM (2018) How blockchain improves the supply chain: case study alimentary supply chain. *Procedia Comput Sci* 134:393–398. <https://doi.org/10.1016/j.procs.2018.07.193>
- Choi TM (2019) Blockchain-technology-supported platforms for diamond authentication and certification in luxury supply chains. *Transp Res Part E Logist Transp Rev* 128(May):17–29. <https://doi.org/10.1016/j.tre.2019.05.011>
- Hang L, Kim DH (2019) Design and implementation of an integrated IoT blockchain platform for sensing data integrity. *Sensors (Switzerland)*. <https://doi.org/10.3390/s19102228>
- Fraga-Lamas P, Fernández-Caramés TM (2019) A review on blockchain technologies for an advanced and cyber-resilient automotive industry. *IEEE Access* 7:17578–17598. <https://doi.org/10.1109/ACCESS.2019.2895302>
- Makhdoom I, Abolhasan M, Abbas H, Ni W (2019) Blockchain's adoption in IoT: the challenges, and a way forward. *J Netw Comput Appl* 125:251–279. <https://doi.org/10.1016/j.jnca.2018.10.019>
- Bodkhe U, Tanwar S, Parekh K, Khanpara P, Tyagi S, Kumar N, Alazab M (2020) Blockchain for Industry 4.0: a comprehensive review. *IEEE Access* 8:79764–79800. <https://doi.org/10.1109/ACCESS.2020.2988579>
- Yu W, Liang F, He X, Hatcher WG, Lu C, Lin J, Yang X (2017) A survey on the edge computing for the Internet of Things. *IEEE Access* 6:6900–6919. <https://doi.org/10.1109/ACCESS.2017.2778504>
- Noizat P (2015) Blockchain electronic vote. Elsevier Inc., New York. <https://doi.org/10.1016/B978-0-12-802117-0.00022-9>
- Wan J, Li J, Imran M, Li D (2019) A blockchain-based solution for enhancing security and privacy in smart factory. *IEEE Trans Ind Inf* 15(6):3652–3660. <https://doi.org/10.1109/TII.2019.2894573>
- Leng J, Ruan G, Jiang P, Xu K, Liu Q, Zhou X, Liu C (2020) Blockchain-empowered sustainable manufacturing and product life-cycle management in industry 4.0: a survey. <https://doi.org/10.1016/j.rser.2020.110112>
- Leng J, Ye S, Zhou M, Zhao JL, Liu Q, Guo W, Cao W, Fu L (2021) Blockchain-secured smart manufacturing in industry 4.0: a survey. *IEEE Trans Syst Man Cybernet Syst* 51:237–252. <https://doi.org/10.1109/TSMC.2020.3040789>
- Leng J, Yan D, Liu Q, Xu K, Zhao JL, Shi R, Wei L, Zhang D, Chen X (2020) Manuchain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing. *IEEE Trans Syst Man Cybernet Syst* 50:182–192. <https://doi.org/10.1109/TSMC.2019.2930418>
- Bahga A, Madiseti VK (2016) Blockchain platform for industrial Internet of Things. *J Softw Eng Appl (JSEA)* 1:533–546. <https://doi.org/10.4236/jsea.2016.910036>
- Latif S, Idrees Z, Ahmad J, Zheng L, Zou Z (2021) A blockchain-based architecture for secure and trustworthy operations in the industrial Internet of Things. *J Ind Inf Integr* 21(September 2020):100190. <https://doi.org/10.1016/j.jii.2020.100190>
- Chanson M, Bogner A, Bilgeri D, Fleisch E, Wortmann F (2019) Blockchain for the IoT: privacy-preserving protection of sensor data. *J Assoc Inf Syst* 20(9):1271–1307. <https://doi.org/10.17705/1jais.00567>
- O'Neill A (2012) Open source software. In: Chadwick R (ed) *Encyclopedia of applied ethics*, 2nd edn. Academic Press, San Diego, pp 281–287. <https://doi.org/10.1016/B978-0-12-373932-2.00063-6>
- Scheid EJ, Hegnauer T, Rodrigues B, Stiller B (2019) Bifröst: a modular blockchain interoperability API. In:

- Proceedings—conference on local computer networks, LCN 2019–Octob, pp 332–339. <https://doi.org/10.1109/LCN44214.2019.8990860>
25. Oktian YE, Lee SG, Lee HJ (2020) Hierarchical multi-blockchain architecture for scalable internet of things environment. *Electronics (Switzerland)* 9(6):1–27. <https://doi.org/10.3390/electronics9061050>
26. Sousa PSD, Nogueira NP, Santos RCD, Maia PHM, Souza JTD (2020) Building a prototype based on Microservices and Blockchain technologies for notary's office: An academic experience report. In: Proceedings—2020 IEEE international conference on software architecture companion, ICSA-C 2020 (i):122–129. <https://doi.org/10.1109/ICSA-C50368.2020.00031>
27. Leng J, Jiang P, Xu K, Liu Q, Zhao JL, Bian Y, Shi R (2019) Maker-chain: a blockchain with chemical signature for self-organizing process in social manufacturing. *J Clean Prod* 234:767–778. <https://doi.org/10.1016/j.jclepro.2019.06.265>
28. Bralić V, Stančić H, Stengård M (2020) A blockchain approach to digital archiving: digital signature certification chain preservation. *Rec Manag J* 30(3):345–362. <https://doi.org/10.1108/RMJ-08-2019-0043>
29. Bouraga S (2021) A taxonomy of blockchain consensus protocols: a survey and classification framework. *Expert Syst Appl* 168(June 2020):114384. <https://doi.org/10.1016/j.eswa.2020.114384>
30. Bideh PN, Sönnerup J, Hell M (2020) Energy consumption for securing lightweight IoT protocols. *ACM Int Conf Proc Ser* 10(1145/3410992):3411008
31. Kondoro A, Ben Dhaou I, Tenhunen H, Mvungi N (2021) Real time performance analysis of secure IoT protocols for microgrid communication. *Futur Gener Comput Syst* 116:1–12. <https://doi.org/10.1016/j.future.2020.09.031>
32. Restuccia G, Tschofenig H, Baccelli E (2020) Low-power IoT communication security: on the performance of DTLS and TLS 1.3. In: 2020 9th IFIP international conference on performance evaluation and modeling in wireless networks, PEMWN 2020. <https://doi.org/10.23919/PEMWN50727.2020.9293085>, arXiv: 2011.12035
33. Stallings W (2014) *Cryptography and network security principles and practice*, 7th edn. Prentice Hall, Hoboken
34. Holz R, Hiller J, Amann J, Razaghpanah A, Jost T, Vallina-Rodriguez N, Hohlfeld O (2020) Tracking the deployment of TLS 1.3 on the web: a story of experimentation and centralization. *SIGCOMM Comput Commun Rev* 50(3):3–15. <https://doi.org/10.1145/3411740.3411742>
35. Arfaoui G, Bultel X, Fouque PA, Nedelcu A, Onete C (2019) The privacy of the TLS 1.3 protocol. *Proc Privacy Enhancing Technol* 4:190–210. <https://doi.org/10.2478/popets-2019-0065>
36. Dowling B, Fischlin M, Günther F, Stebila D (2015) A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: Proceedings of the ACM conference on computer and communications security 2015–Octob(May), pp 1197–1210. <https://doi.org/10.1145/2810103.2813653>
37. Cremers C, Horvat M, Hoyland J, Scott S, Van Der Merwe T (2017) A comprehensive symbolic analysis of TLS 1.3. In: Proceedings of the ACM conference on computer and communications security, pp 1773–1788. <https://doi.org/10.1145/3133956.3134063>
38. Leng J, Zhou M, Zhao LJ, Huang Y, Bian Y (2020) Blockchain security: a survey of techniques and research directions. *IEEE Trans Serv Comput*. <https://doi.org/10.1109/TSC.2020.3038641>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.