# Assuring M2M Secure Transactions via Blockchain and Smart Contracts

Nuno Leite, Alexandre Santos, and Nuno Lopes

***Abstract*—The need to ensure the confidentiality and integrity of data generated in industrial systems and applications has been increasingly highlighted over the years, due to the clear as well as urgent requirements of not disclosing sensible proprietary information, and ensuring that data is kept immutable since it is generated. Based on these background requirements, this paper discusses a system that promotes data privacy and immutability in industrial weighing systems. In this sense, the proposed solution includes a protocol definition for a secure IoT (Internet of Things) communication and an architecture for a blockchain-based platform that: i) automates the process of complying with standardized weighing guidelines; and ii) increases data traceability, promoting both its confidentiality and immutability. Additionally, to demonstrate the proposed system's compliance with the established goals, a proof of concept was assembled, and functional tests conducted. The main conclusions withdrawn from the results obtained show that: i) the process of registering and verifying weighing guidelines is properly automated by means of Machine-to-Machine secure transactions; ii) the solution is able to ensure that the data registered is authentic and iii) that it has been transmitted without any eavesdropping, can not be erased and is kept private.**

***Index Terms*—Security, IoT, Blockchain, Smart Contracts, Communications.**

## I. Introduction

The problem discussed in this paper is framed in the landscape of industrial weighing systems. For the purpose of this discussion, it is required to define the entities that participate in this use case as well as the devices that are operated by each of those entities. Essentially, the companies promoting the development of this solution are responsible for selling weighing solutions such as load cells , which are weighing devices, and weighbridges, a physical bridge that combines multiple load cells. These devices are sold to their customers, which possess weighing stations, where multiple weighbridges, which perform the actual weight measurements, can exist. Additionally, each weighing station also owns a device that controls the weight measurements in all its weighbridges, which for the purpose of simplicity is going to be called a *SmartBox*. When a weight measurement ends, the *SmartBox* emits a sort of receipt for that weighing, which is the main asset of this use case and can be referred to as a *weighing ticket*.

The motivation for proposing this solution is mainly associated with the security of the aforementioned weighing tickets, which must: i) be private for each customer; ii) remain immutable after emission; and iii) be traceable, in order to simplify further verification processes on weighing routes (e.g. trucks that are measured from station to station and should possess a predefined weight). Since currently, there is absolutely no assurance on the three aforementioned points, the solution discussed here has the intent to mitigate those three flaws in the weighing system. From this motivation, the main goals for this solution can be established, which are: The development of a blockchain-based application capable of immutably storing the weighing tickets generated at the customers' weighing stations; The implementation of a support platform, that exposes APIs (Application Programming Interface) to enforce authentication & authorization throughout the platform and, additionally, to provide rich querying of weighing tickets, to facilitate and abstract communication with the blockchain application, increasing traceability; And, finally, the establishment of a secure protocol definition for M2M (Machine to Machine) communications.

In order to present, describe and evaluate the solution that is going to be built for the use case in study, the remaining of this paper is structured as follows: Section II discusses the SoA (State of the Art) on the relevant technologies as well as related work that present different perspectives on the application of such technologies; Section III presents and describes the proposed solution, as well as the security & privacy lifecycle of the weighing tickets; Section IV describes the experiment that was assembled and the tests that were conducted to demonstrate that the proposed solution is functionally compliant with the established goals; And, finally, section V provides a small summary on the paper, the main results obtained in it and the prospects for future work.

## II. Research Background

In order to provide some background knowledge on the work developed and presented in this paper, this section explores the SoA regarding the technologies intrinsically associated with the subject in study. Finally, it concludes by briefly overvieweing related work that presents different perspectives and solutions based on those technologies.

Manuscript received February 11, 2021; revised July 1, 2021. Date of publication August 24, 2021. Date of current version August 24, 2021. The associate editor prof. Toni Perković has been coordinating the review of this manuscript and approved it for publication.

N.Leite and N.Lopes are with the Digital Transformation CoLab, located at Campus de Azurém, Alameda da Universidade, 4800 - 058, Guimarães, Portugal (e-mails: nunoleite31@gmail.com, nuno.lopes@dtx-colab.pt).

A.Santos is with the University of Minho, located at Campus de Gualtar, 4710-057, Braga, Portugal (e-mail: alex@di.uminho.pt).

Digital Object Identifier (DOI): 10.24138/jcomss-2021-0034

### A. State of the Art

In this section, the SoA for the predominant technologies explored and used for the proposal of this solution is studied. In overview, two topics are explored to serve as the base study that ultimately generates the solution:

- *IoT*, the topic that is explored in order to establish the protocol definition for the communication system;
- *Blockchain and Smart Contracts*, which is the base technology for the application running in the cloud platform.

*1) Internet of Things:* IoT is one of the trending and growing topics of this millenium, due to the ever growing necessity of connecting *things* in order to improve information gathering as well as processes' automation. It can be defined as an Internet-enabled architecture that facilitates the interconnection of devices with multiple technologies, fostered by M2M communication protocols [1], [2]. The inherent capability of IoT technologies to digitize the physical world, i.e., communicating and generating the digital information that represents the physical state of *things* make it one of the clear enablers of *Industry 4.0* [3].

Although the application of IoT in industrial systems and applications provides numerous advantages, such as the awareness of the physical state of devices and the scalability of information gathering processes, there are also some challenges that need to be overcome in order to correctly apply it. Some of these challenges can be defined and described as follows [3], [4], [1], [5]:

- Heterogeneity - Many existent devices from many manufacturers, using different protocols and different stacks make it difficult for seamless communication between them;
- Restrained resources - Most devices operating in an IoT system are restrained in their resources, which means that deciding the way how they operate and communicate becomes of the utmost importance;
- Energy efficiency - Typically IoT devices, especially the ones belonging to industries, are required to withstand long periods of time in operation, which is harder in restrained devices;
- Security & Privacy - It is probably one of the biggest challenges an IoT environment faces, to be able to transmit data in a secure manner, abiding by standardized information security properties, such as confidentiality, immutability and authenticity. This challenge is, of course, hardened by the fact that most of the devices have low resources, thus having a hard time running heavier secure communication protocols such as HTTPS (HyperText Transfer Protocol Secure).

Having perceived the challenges that arise from the intent to apply IoT in an industrial application, some communication protocols that can mitigate them have to be studied. It must first be recognized that, although IoT environments are usually constrained in terms of resources, typical applications make use of a device which is called an aggregator. This device essentially collects all the data from the restrained devices, applies the required processing and further transmits it to where it needs to be [5], [6]. It is for this reason that, although HTTP (Hyper Text Transfer Protocol) is not quite suited for restrained environments, it is also studied in this section, due to the fact that it might proove useful to be used in the last link of communication.

TABLE I
COMPARISON BETWEEN DIFFERENT IOT COMMUNICATION PROTOCOLS

| *Attribute / Protocol* | CoAP | MQTT |
|---|---|---|
| *Transport protocol* | UDP | TCP |
| *Communication type* | asynchronous | asynchronous |
| *Communication pattern* | publish-subscribe<br>request-response | publish-subscribe |
| *Communication paradigm* | polling | event-driven |
| *Security* | DTLS | TLS |

Although there are a number of protocols that have been appearing over the years that are more or less suited to operate in IoT environments, two of them can be especially highlighted, which are *CoAP (Constrained Application Protocol)* and *MQTT (Message Queuing Telemetry Transport)* [7]. These two protocols were built with IoT in mind and, despite having some differences, they are very lightweight protocols focused on low overhead and fast transmission. Table I illustrates some of the characteristics of these two protocols and, as it can be seen, although their goal is the same, they greatly differ in how they do it. *CoAP* is an asynchronous communication protocol that typically runs over UDP (User Datagram Protocol) and communicates by polling other devices, while *MQTT* is an asynchronous, event-driven protocol that normally communicates over TCP (Transmission Control Protocol). Additionally, since *CoAP* uses UDP for the transport layer of communication, DTLS (Datagram Transport Layer Security) can be used to ensure secure data transmission [8]. On the other hand, due to the fact that *MQTT* uses TCP as its transport protocol, TLS (Transport Layer Security) is typically used for securing data transmission [9]. This essentially means that *MQTT* introduces a slightly higher overhead than *CoAP* due to its use of TCP and TLS [10], [11].

Finally, although *MQTT* only communicates via a publish-subscribe mechanism, since it is event driven, which essentially means that data is produced into a *broker* and whoever needs that data will consume it from the *broker*, *CoAP* can in some way communicate in both a typical request-response mechanism as well as in a publish-subscribe one, although it is not a typical publish-subscribe mechanism. What happens is that *CoAP* allows to abstract the requester from the burden of continuously polling the data emitter and, instead, allows the requester to *"watch"* the data emitter. This basically means that instead of the requester polling, the data emitter will continuously produce the data generated into the requester directly, without the requester asking for it [10], [11].

As was previously mentioned, although finding and using a lightweight communication protocol is of the utmost importance to enhance and render more efficient the communication between resource-restrained devices, HTTP is also a protocol to bear in mind, since it is widely used in the WWW (World Wide Web) due to its simplicity and resource-oriented architecture. If in any communication link the devices that communicate in it are resource-capable, then HTTP can, in

fact, be one of the solutions. HTTP is a synchronous protocol that typically runs over TCP in a request-response pattern with a long-polling paradigm and, it can be secured using TLS [12].

With the knowledge on the challenges that creating an IoT communication system faces, as well as the communication protocols that might mitigate, at least in part, some of those challenges, this study can be used for the proposal of the solution to the protocol definition.

*2) Blockchain & Smart Contracts:* With the introduction of Bitcoin in 2008, as proposed in [13], along came a more important concept, that of a distributed ledger that records every transaction ever made in a system in an immutable manner, the Blockchain. Throughout the years since it was introduced, researchers and the industry started to realize the immense potential of such a technology since its application was not limited to electronic cash systems. Due to its intrinsic characteristics of granting immutability and traceability of the information managed by it, the applications for the technology are immense.

Despite being a technology that grants, by design, data immutability, it does not consider mechanisms for data confidentiality, i.e., although data is immutable when stored in a blockchain, it can be seen by anyone who belongs to the network. For this reason, there are essentially two types of blockchain, the public and the private/consortium ones [14]. While maintaining the immutability and traceability of the data, a consortium blockchain possesses properties that are essential, for example, for industrial systems, where only entities with access to a certain system are able to communicate with the blockchain network, since access to it is controlled and transactions can be made private between certain nodes.

With the appearance of blockchain technology, another term that already existed gained notoriety, *Smart Contracts*, a term coined by Nick Szabo in the 1990s as a digital twin to the already known physical contract between two or more entities, which enforces a set of rules and conditions for the utilization or transfer of a given asset [15]. In a simple, clear way, smart contracts can be defined as a collection of functions and state deployed on the blockchain network, that usually abide to three properties:

- Availability - They execute on all nodes of the blockchain to where they were deployed;
- Deterministic - Multiple executions of the same transaction has to yield exactly the same result;
- Internal Operation - They typically only run in the network and do not fetch data from external web services.

The last few years have seen an increase in the existence of smart contract platforms, which provide tools to build and deploy smart contract applications in the blockchain. Some of those platforms focus on public blockchain networks, i.e., accessible to everyone, such as *Ethereum* [16], and others focus on enterprise networks, where the access to the network is controlled, such as *Quorum* [17] or *Hyperledger Fabric* [18].

*Ethereum* names itself as an open-source platform for decentralized applications, i.e., smart contract applications. Its inner workings are that of a public blockchain, everyone has access to it so long as they create an account. Essentially, this platform allows the creation of smart contracts that manage assets, which can be electronic cash or information/state created by the developer. Smart contracts in *Ethereum* are programmed using proprietary languages such as *Solidity* or *Vyper* [19]. The code programmed in either of these languages is then translated to bytecode using the EVM (Ethereum Virtual Machine) [20], which provides an abstraction between the code that will execute and the machine that will execute it, fostering program portability.

*Quorum* is an Ethereum-based open-source platform for decentralized applications in enterprise blockchains. This means that, although all the tooling and frameworks used by *Ethereum* are compatible with *Quorum*, the latter adds a component which allows to introduce the concept of *private transactions*. In its essence, what *Quorum* does is to introduce a privacy manager, which is responsible for allowing a member of the network to specify which members have access to the contents of a given transaction, this way it protects the confidentiality of the information. The immutability is still granted, since all nodes still work together to ensure the transactions' validity, despite some of them not being able to see the actual contents.

*Hyperledger Fabric* is a platform for enterprise blockchain-based applications, providing access control and mechanisms to ensure privacy of the data for strictly the entities that need to see it. Aside from that, smart contracts in *Hyperledger Fabric* can be viewed in two separate terms [18]:

- Smart Contract - defines the transaction logic that controls the asset's lifecycle;
- Chaincode - the packaged transaction logic in code that is deployed to the blockchain network.

The terms smart contract and chaincode can be viewed as the smart contract and bytecode in *Ethereum* or *Quorum*. Smart contracts in *Hyperledger Fabric* can be implemented in a variety of languages like, for example, Java or Javascript.

The main difference from *Hyperledger Fabric* to *Ethereum* or *Quorum* is the way how the actual blockchain network works, since the first one has an extremely modular architecture allowing the developer to substitute some of the components at their need.

### *B. Related Work*

The joint application of IoT and blockchain technologies has seen a significant increase in the last few years due to the enormous potential that both of them have.

Table II describes some of the related work studied in the preparation of the system proposed in this paper, including the year of publications its authors, as well as the focus of each of the articles.

The first three papers ([21], [22] and [23]) present more foundational perspectives on blockchain and IoT systems. Specifically:

- In [21] the authors study the possibilities and limitations of smart contracts in current applications, while exemplifying with a prototype for a system that manages automated gasoline purchases;
- In [22], the author studies a new approach for access management systems in IoT and, additionally, presents

TABLE II
RELATED WORK AND SURVEYS IN BLOCKCHAIN AND IOT INTEGRATION

| Year | Authors | Focus |
|---|---|---|
| 2018 | *Hanada et al.* [21] | Possibilities and Limitations in smart contracts for M2M communications |
| 2018 | *Oscar Novo* [22] | Blockchain-based distributed access control for IoT |
| 2019 | *Gong et al.* [23] | Proposal of an IoT and blockchain fusion model |
| 2019 | *Alladi et al.* [24] | A review of blockchain applications in Industrial IoT |
| 2020 | *Kuperberg et al.* [25] | Prototype for a blockchain-enabled railway control system |

a prototype based on blockchain technology, concluding that these types of blockchain-based access management systems can be used in scalable IoT scenarios;

- Finally, in [23], the authors propose a structure for a blockchain and IoT fusion model to enable and foster the division of devices between high performant and resource restrained. The main goal for this structure is to increase efficiency and scalability.

In addition to these three more foundational perspectives on the integration of blockchain and IoT, two additional papers are presented to provide a wider review on applications of these technologies and more pratical insights on their use.

In [24], the authors perform a wide review of several existent applications of blockchain in industrial IoT systems, such as applications for smart cities, for the automotive industry or even for the oil and gas industry.

Finally, in [25], the authors study the possibility of applying blockchain to railway control systems and, additionally, present a prototype of such a system, which enables trains to automatically find the best routes, safeguarding that choice in an auditable manner, instead of relying in central control offices or the actual drivers to perform that work.

## III. PROPOSED SOLUTION

In this section, the proposed solution for both the communication system as well as the cloud platform is shown and discussed, including the design patterns that supported the decision as well as the technologies that are going to be used. The section begins with the protocol definition by illustrating an example of a weighing station and how the devices that exist in it can communicate. The section proceeds with the illustration of the architecture of the cloud platform, coupled with the explanation on each of the components that comprise it. Finally, the security & privacy lifecycle of the weighing tickets is discussed, i.e., how the components that take part of the solution are able to ensure the required security properties.

### A. Protocol Definition

The first step in defining the protocols that are going to be used for communication is to, first and above all, clearly define the ”links” of communication that exist, i.e., how many entities communicate in the system and who communicates with whom.

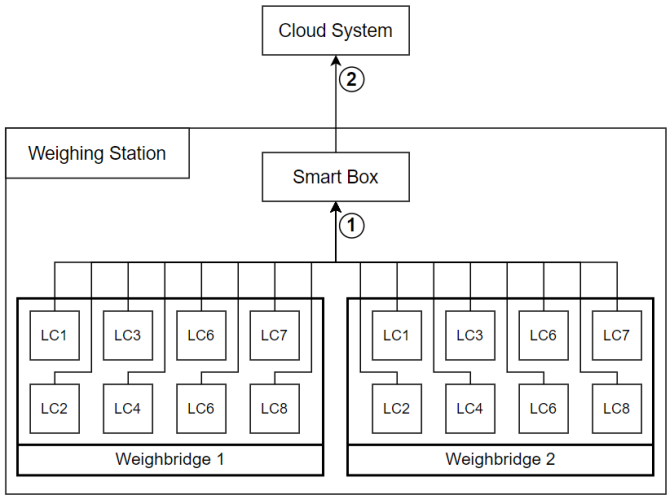


Fig. 1. Overview of the communication system in the weighing stations

Figure 1 illustrates an overview of how a typical weighing station is structured and what devices exist in it, as well as the devices that communicate with each other. From the figure, it can be noticed, as was previously mentioned, that a typical weighing station possesses one *SmartBox* and more than one weighbridge, which in turn are comprised by load cells. This way, it can be seen that there are two ”links” of communication, 1 and 2, where 1 refers to the communication between the load cells with the *SmartBox* and 2 refers to the communication between the *SmartBox* with the cloud platform.

Prior proceeding with the protocol definition, it is worth remembering the characteristics of the devices which participate in each ”link” of communication, since that deeply influences the protocol that is required. There are essentially three devices/entities communicating here: i) the load cells; ii) the *SmartBox*; and iii) the cloud platform. Clearly, the cloud platform, as its name suggests, is a platform running either in a cloud provider or a proprietary server, thus it is expected a high resource availability. The *SmartBox* is an aggregator device which also applies compensation algorithms to the weighing process and, additionally, displays all the information for the operators in the stations and, thus, it is a device that cannot be considered as resource-restrained. Finally, the load cells are, in fact, resource restrained devices since their RAM (Random Access Memory) rounds the tens of KB (Kilobytes) and the CPU (Central Processing Unit) rounds the tens of MHz (MegaHertz).

From the information previously retrieved, it is now more clear that in communication link 1, a more lightweight protocol will be required so that it doesn’t negatively influence the performance of the load cells and, in communication link 2 the protocol choice is not restrained in terms of resources.

Having recognized the structure of the communication system that needs to be defined, it was chosen that communication link 2, i.e., communication between the *SmartBox* and the cloud platform, would be made with HTTP over TLS, commonly known as HTTPS. This protocol was chosen due to the fact that there are no restraints in terms of resources in

the devices that use it in this case and thus, leveraging it to communicate with the REST (Representational State Transfer) APIs that are part of the cloud platform is simpler, since REST and HTTP method formats deeply coincide.

Next, the definition of the protocol that is used in communication link 1 is required and, as reviewed in section II-A, the choice will lie between CoAP and MQTT. In this particular use case there are really no performance concerns, since the traffic that will have to be handled by the protocol is significantly low (approximately three hundred transmissions per weighbridge per day) and thus, either of the protocols will be able to perfectly deal with that throughput. So, the choice relied more on the architecture of the protocol as well as on how to further simplify the communication system that was going to be built. With that said, CoAP has an architecture in terms of communication pattern extremely similar to HTTP while, of course, greatly differing in message transmission and headers which makes it much more lightweight. This particular property that both hold make them suitable to build a clean, simple and efficient communication system which communicates in the same pattern from point to point. Additionally, CoAP transmissions can be secured by using DTLS, as previously mentioned.

Concluding, for the protocol definition of the IoT communication system, the solution designed proposes that:

1) The communication between the load cells with the *SmartBox* is done by recurring to CoAP over DTLS;
2) The communication between the *SmartBox* with the cloud platform is done by recurring to HTTP over TLS (HTTPS).

### B. Cloud Platform Architecture

Having described the communication system that will allow the emission of weighing tickets to the cloud platform, it stands to reason that now the actual components of this platform must be defined in order to perceive how they, coupled together, provide the functionality that is required.

Figure 2 illustrates the system architecture designed for the platform in question. From this figure, it can be immediately seen that it has four major components:

- The entity database, which essentially houses the data models required to represent all entities that communicate with the platform;
- The blockchain network, which is the network of nodes that are associated with the organizations that participate in the management of weighing tickets by executing smart contracts;
- The weighing tickets API, which exposes in a clean and simple REST API methods to manage weighing tickets, including its registration and querying;
- The authentication & management API, which also exposes its methods through a REST API to allow the management of entities that participate in the system.

*1) Entity Database:* From the definition of the use case, it is possible to conclude that the platform has to be able to recognize and thus, model, three entities:

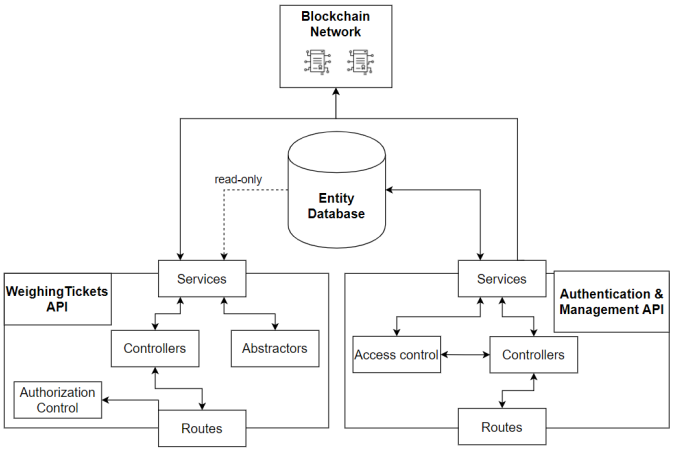


Fig. 2. Architecture of the cloud platform

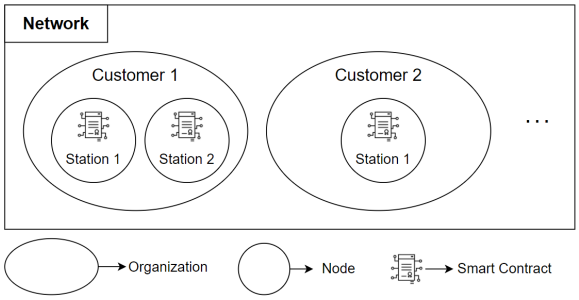


Fig. 3. Schematics of the network's structure

- Customer - the entity that buys the devices from the weighing companies;
- Station - an entity that must belong to a customer, which operates the devices bought by the customer, performing the actual weighing processes;
- User - an entity that represents a customer. A customer may have multiple users, since each one of them will represent a different authenticated person.

*2) Blockchain network & Smart contract:* For this particular problem, the platform uses a *Quorum* network due to its simplicity and the possibility of establishing private transactions between different nodes of the network.

Figure 3 illustrates an example of how the network will be structured in terms of the entities that participate in it. Essentially a customer will be recognized as an organization inside the network, i.e., an abstract entity that is comprised by multiple nodes. Each node in an organization represents a station and each station communicates with its own smart contract. Additionally, every customer possesses an administrator node, which is essentially a node that is only associated to the customer and not to any station. In section III-C the importance of this structure will be further highlighted and explained.

In addition to the structure of the actual network, this component also comprises the definition of requirements of the smart contract, the contract between customers and their stations for the management of weighing tickets. In overview, this smart contract can be defined as a program which lever-

ages two data structures and a set of methods. The two data structures it leverages are: i) the weighing ticket, which is an object that precisely defines the attributes that comprise a weighing ticket, including their data types and names; ii) the set of weighing tickets, which holds every weighing ticket registered by that contract.

Possessing these two data structures, the methods that manipulate them can be defined and there are two major requirements for them: i) to allow the registration of a new weighing ticket; and ii) to allow the query of already registered weighing tickets. While the first requirement relates only to the implementation of one method in the smart contract, for the second requirement an improvement was considered, which is to allow the filtering of weighing tickets directly in the smart contract. Taking into account that it is expected the continuous growth of the number of weighing tickets registered and that, when querying in a large universe of information, typically the querying entity wishes to look for more specific pieces of information, the smart contract should allow the filtering of the results when queried, for all its attributes.

*3) Authentication & Management API:* As was previously mentioned, this component is a REST API which serves two main purposes:

- To manage the entities that exist in the system by allowing one to register/update/
  query/delete customers, users or stations;
- To provide the means for an user or a station to authenticate themselves and, furthermore, to be able to have their requests authorized both in this API and in the weighing tickets API.

The registration of stations is a defining operation performed in this API since it enforces the network structure defined in Figure 3. When a station is registered, the smart contract is deployed privately to the station's node and to the customer's administrator node, which means that only the station's node and the customer's node are able to interact with the smart contract. This way, a customer will always participate in all the smart contracts of its stations and each station will only participate in its smart contract.

Finally, to increase security in the confidentiality of the users' credentials, the authentication and authorization process introduces a token mechanism. Essentially, an user authenticates itself with its credentials and then receives two unique identifying tokens which he can use: i) the access token, which is the token that is actually used to authorize requests, that has a short lifespan; and ii) the refresh token, which is a token that the user can use to refresh its access token, which also has a limited lifespan, although greater than the access token. This way, this mechanism prevents the user from having to continuously authenticate themselves with their credentials in each request.

*4) Weighing Tickets API:* As earlier defined, this API is responsible for allowing entities to register or query weighing tickets, by communicating with the smart contract associated with that particular entity. The registration of weighing tickets can only be done by stations, since only they can emit them and the query of weighing tickets is limited to the users, due to the fact that the stations' devices do not have the need to consult the information on a weighing ticket, only emit them.

To put it simply, when a station attempts to register a weighing ticket, the API collects the contract address associated with it and submits a registration operation to the contract with the information that it received. This transaction is private to the smart contract, which means that only this station's node and the node associated with the customer that possesses the station have access to the contents of the transaction.

The query of weighing tickets implemented allows combined filtering, which means that the query issued can possess multiple filters regarding the attributes of the weighing tickets, such as the timestamp at when it was issued, the station it was issued from or even the weighbridge it was issued from, among others, in order to greatly facilitate the integration with other systems. Additionally, it also allows the transformation of the result by grouping the results by some attribute of the weighing tickets, such as for example, grouping all returned weighing tickets by the weighbridge by which they were emitted.

The components defined in this section that comprise the proposed solution are essential to what is the lifecycle of security & privacy in the flow of information, i.e., the assurance that the information is kept confidential and immutable since it is emitted until it is stored, as will be explained in the next section.

### C. Security & Privacy Lifecycle

The purpose of this section is to describe how the components that comprise the solution proposed in sections III-A and III-B foster data security along the lifecycle of a weighing ticket from the point it is created in the *SmartBox* until it is permanently stored in the ledger.

Prior starting the description, it is essential to thoroughly describe the lifecycle of the weighing ticket:

1) The weighing ticket is created in the *SmartBox* with the weights received from the load cells;
2) The weighing ticket is transmitted from the *SmartBox* to the platform, by calling the appropriate method in the weighing tickets API;
3) The weighing ticket is processed in the weighing tickets API and a transaction to register that ticket is issued to the smart contract;
4) The weighing ticket is stored in the ledger.

The first step in the lifecycle, in terms of security, essentially comprises the secure transmission of the weights from the load cells to the *SmartBox* for weighing ticket building. As seen in section III-A the transmission of these weights is done by using CoAP as the application protocol over DTLS, which ensures the authenticity of both the load cell devices as well as the *SmartBox* , providing, additionally, confidentiality and immutability in transit of the data transmitted between them.

After the transmission of the weights, the *SmartBox* builds the weighing ticket by completing it with the remaining information, such as the weighbridge it was measured in and at what time the weights were measured for example and then, transmits the weighing ticket to the cloud platform to be registered. This transmission step is, as seen in section III-A,

done by using HTTP as the application protocol over TLS, which ensures the authenticity of the *SmartBox* device and the weighing tickets API and, in addition to that, also grants that during the transmission, the information is kept confidential and immutable. The authentication & management API also plays an important role in this step, since it is obligatory for a station to possess a valid access token to issue the request, thus ensuring that the entity transmitting that weighing ticket, at some point, authenticated successfully and is registered in the platform.

When the weighing ticket is processed and about to be registered in the ledger using the smart contract, the weighing tickets API collects the address of the contract belonging to that station and submits the weighing ticket only to the contract in that address. It is in this step that the importance of the structure of the network enforced in the registration of entities is recognized, since as the contract for the station was deployed privately to the station and the customer it is associated with, three essential properties can be ensured, in addition to the immutability of the data provided by blockchain technology:

- The information of a weighing ticket is only visible to the station that emitted it and to the customer that possesses that station;
- Stations from the same customer cannot see each others weighing tickets;
- Entities from different organizations have no way of consulting weighing tickets from other organizations.

In addition to the enforcement of the network structure, both the authentication & management and weighing tickets APIs play an important role in securing information vital for communication with the smart contracts, by implementing a type of encryption at rest mechanism. Whenever one of the APIs stores blockchain-related information in the entities database associated to a station or a customer, such as the address of the contract or the URL (Uniform Resource Locator) to connect to the node, they store it encrypted and authenticated. This means that, when written to the database, the information is confidential and any change in it can be detected through the authentication code stored with it. So, if one of the APIs needs to read that data, they have to first verify the authentication code and only then, decrypt the data to obtain the original information, through a key shared by both at boot-up.

Finally, after the weighing ticket is registered in the ledger, the characteristics of blockchain technology and the way how the smart contract is built ensure that the weighing ticket remains immutable, since: i) a transaction cannot be altered unless it is accepted in the networks' consensus protocol, which implies the permission of a significant number of nodes in the network; and ii) the smart contract does not possess methods to "legally" alter the contents of a weighing ticket. So, since the original transaction cannot be changed and there is no way of issuing a transaction for the smart contract to alter the weighing ticket, it is stored as received permanently in the ledger.

These four steps in the weighing ticket lifecycle clearly highlight the importance that all the components defined in the proposed solution have on the security & privacy of the weighing tickets, allowing to provide an end-to-end secure transmission and registration of them.

## IV. Proof of Concept

The purpose of this section is to provide a functional demonstration on some of the aspects of the solution, particularly the ones related to the correct and secure transmission of the weighing tickets as well as maintaining privacy between different customers. Additionally, there is also an intent to provide some illustration on the results obtained by consulting existent weighing tickets.

### A. Experiment Setup

In order to provide such functional demonstrations, an experiment was set up to represent an example of the use case in study that implements the flow of information required.

For the purpose of this experiment two different customers were assumed, where each one of them possesses one weighing station. Each of those weighing stations are comprised by two weighbridges and each weighbridge has four load cells.

The experiment was assembled using three devices, one laptop and two raspberry pi. The laptop was responsible for running the cloud platform, i.e., the blockchain network, the weighing tickets API, the authentication & management API and the entities database. In this experiment the blockchain network is comprised by five nodes (one for the network administrator, two for the customer administrators and two for the weighing stations) coupled their respective privacy managers, i.e., the components responsible for ensuring data privacy when required.

Figure 4 illustrates the structure assumed for a weighing station in the experiment. Each of the raspberry pi devices was responsible for running the necessary components to simulate the behaviour of one weighing station. Accordingly, each of them ran:

- Eight load cell communicators, four per weighbridge, the software components responsible for securely communicating weights from the load cells to the *SmartBox*;
- One *SmartBox* communicator, the software component responsible for securely communicating with the cloud platform, either to register new weighing tickets or to request authentication.

With the infrastructure used to run the experiment ready, the purpose was now to simulate the emission and registration of weighing tickets. To do this, an already existent dataset was used which held 600 weighings, 300 per weighing station. The elements in this dataset were split, put into the according station's *SmartBox* communicator and, additionally, the weights were removed from the original weighing, placing them in the load cell communicators so that they could be transmitted by them. After this step of preparation the load cells were made to transmit weights periodically, starting a continuous transmission process of weighing tickets.

### B. Results & Discussion

With the transmission of weighing tickets from the simulated weighing stations in execution, what's left to obtain is

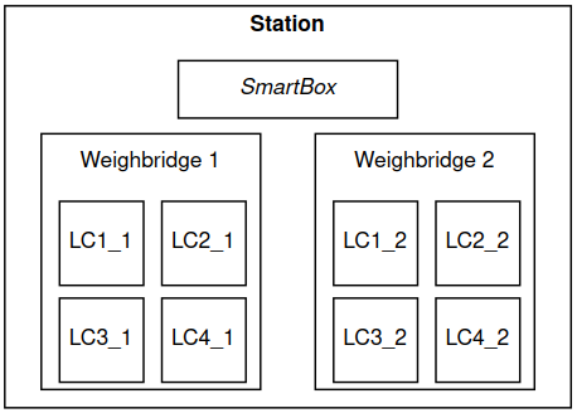


Fig. 4. Structure of a station in the experiment


Fig. 5. Information saved on the *SmartBox*

some illustrations that allow to conclude that the functional requirements were met. Essentially, what has to be functionally demonstrated is the correct registration of weighing tickets, the secure communication between the devices at the weighing stations and finally the correct separation of data between customers.

Figure 5 illustrates the object that is stored in the *SmartBox* after a successful weighing ticket registration. There are two parameters, one that uniquely identifies the weighing ticket (*ticketID*) and another that uniquely identifies the transaction that registered the weighing ticket in the ledger (*transactionHash*). The presence of the *transactionHash* value demonstrates that the transaction was issued and validated in the blockchain network.

Although the aforementioned parameters assure that the weighing ticket was, in fact, registered in the ledger, it also has to be seen that its construction and transmission was made in a secure manner by ensuring that: i) the weights transmitted from the load cells to the *SmartBox* were secured; And ii) that the transmission of the weighing ticket to the API was also secured. This was done by placing a packet sniffer, *Wireshark*, listening to the communication between the devices and acquiring the packets that are transmitted, in order to check if the data that was sent could be seen or not.

Figures 6 and 7 illustrate secure communication between a *SmartBox* device with the weighing tickets API. The first figure shows an excerpt of communication between both devices, where a TLS exchange can be identified and the second figure illustrates an example of a packet captured where it can be clearly seen the existence of encrypted data, not permitting the visualization of the original information.

Figures 8 and 9 illustrate secure communication between load cell devices in a weighbridge with the associated *SmartBox*. In the first figure, an excerpt of communication between them can be seen, where it is possible to identify a DTLS exchange between different load cell devices and the *SmartBox*.

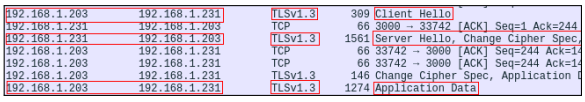


Fig. 6. Excerpt of communication between a *SmartBox* with the weighing tickets API

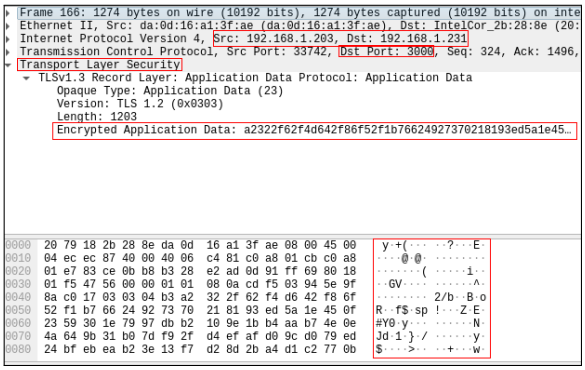


Fig. 7. Example packet exchanged between a *SmartBox* with the weighing tickets API

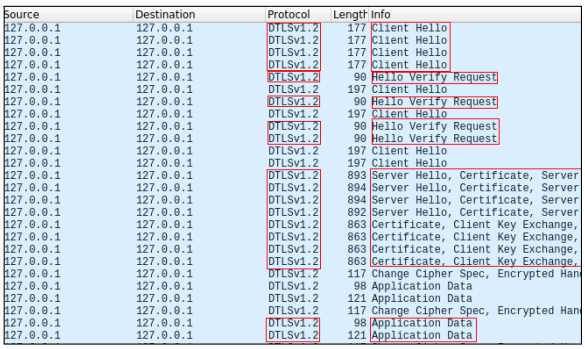


Fig. 8. Excerpt of communication between the load cells with a *SmartBox*

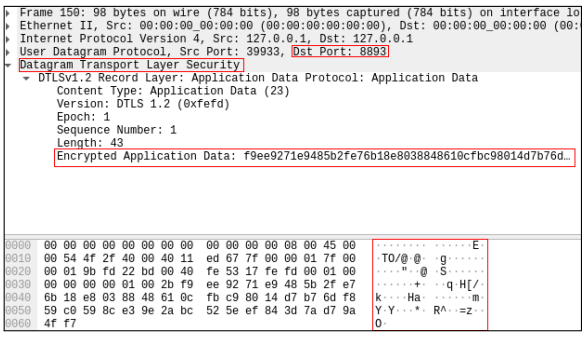


Fig. 9. Example packet exchanged between the load cells with a *SmartBox*

The second figure illustrates an example of a packet captured in this communication where it can also be seen the indication of DTLS being used and, of course, the existence of encrypted data.

Having collected the aforementioned illustrations and ensuring that the communication between all devices is secured, the next and final step is to verify that the customers cannot see other customers' data, ensuring their weighing tickets are kept private.

The first step in doing this is ensuring that a customer can

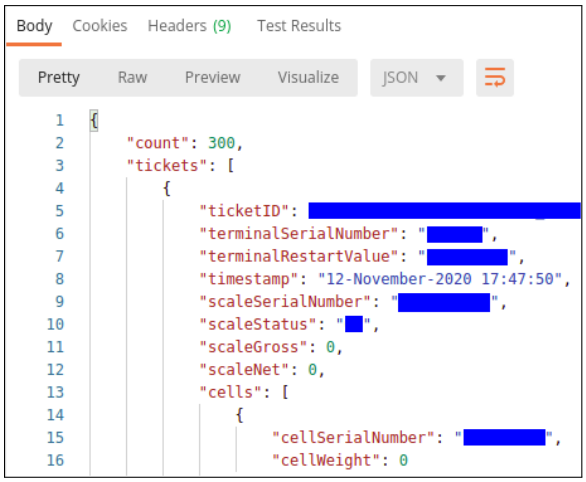


Fig. 10. Result of a customer requesting to see its weighing tickets


Fig. 11. Result of one customer requesting to see other customer's weighing tickets

actually access its own weighing tickets. Figure 10 illustrates the result received when a customer requests to see its weighing tickets. This result is obtained when issuing a request to the weighing tickets API to collect the weighing tickets with an authorization token that uniquely identifies the customer performing the request.

While validating that a customer can access its own weighing tickets, it is also required to verify that one customer cannot access other customer's weighing tickets, thus ensuring that they are kept private.

Figure 11 illustrates the result obtained when issuing the same request made in figure 10, with a slight change. In figure 11 the request is performed with an authorization token belonging to one customer, while specifying a weighbridge identifier that is associated with another customer and, as it can be seen, nothing is returned, although it is known from figure 10 that the customer possesses 300 weighing tickets.

With the illustrations previously shown it can be concluded that, functionally, the solution that was proposed and developed complies with the security requirements that were identified, since weighing tickets are transmitted securely between all devices and, additionally, they are registered in the ledger privately to the customer associated with the weighing station.

## V. Conclusion

In this paper, a novel solution for securing weighing receipts using blockchain technology was studied and proposed. The paper first explores the state of the art regarding the technologies necessary to design the solution, namely, IoT for the communication between the devices that operate on the weighing stations, and blockchain to immutably secure weighing tickets once they are registered. After this research, a solution is proposed which is comprised by a protocol definition for an IoT communication system and a cloud platform capable of: i) allowing the registration and query of weighing tickets; ii) ensuring data privacy between different customers; And iii) simplifying the communication with the smart contract application through the use of REST APIs.

Having proposed the solution, the security & privacy lifecycle of the weighing tickets is briefly but thoroughly explained in order to detail the security properties that are ensured and how they are ensured by the components that comprise the solution.

Finally, a functional proof of concept is presented to illustrate and demonstrate that the functional requirements initially defined are met by the solution that was developed.

### A. Future Work

Although the project has reached a good maturity level, there is still some work to conclude it. There are essentially two major aspects which will be taken into account from now on, which are:

- To foster a better integration of the cloud platform with existing systems of the companies by, for example: i) adapting the authentication mechanism to an already existent one; And ii) by offering a managed blockchain solution to release the burden of network management, which is high.
- To test the solution in a real environment instead of a simulated one

These two last items are the essential aspects were the work will be directed to in order to conclude and present the final solution capable of being completely integrated with the systems that already exist while, of course, providing all the new functionality to securely transmit and manage weighing tickets.

## Acknowledgment

This work has been partly supported by national funds through FCT—Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2020 and by the European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 039334; Funding Reference: POCI-01-0247-FEDER-039334] and also by NORTE-06-3559-FSE-000018, integrated in the invitation NORTE-59-2018-41, aiming the Hiring of Highly Qualified Human Resources, cofinanced by the Regional Operational Programme of the North2020, thematic area of Competitiveness and Employment, through the European Social Fund (ESF).

## References

[1] R. H. Weber, "Internet of Things - New security and privacy challenges," *Computer Law and Security Review*, vol. 26, no. 1, pp. 23–30, 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
[3] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.
[4] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," *Proceedings - IEEE Symposium on Computers and Communications*, vol. 2016-Febru, no. July, pp. 180–187, 2016.
[5] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10–16, 2017.
[6] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
[7] S. Jaloudi, "Communication protocols of an industrial internet of things environment: A comparative study," *Future Internet*, vol. 11, no. 3, 2019.
[8] IETF, "DTLS - Datagram Transport Layer Security," https://tools.ietf.org/html/rfc6347, [Online; Accessed 10 - December - 2020].
[9] Internet Engineering Task Force, "TLS - Transport Layer Security," https://tools.ietf.org/html/rfc8446, [Online; Accessed 10 - December - 2020].
[10] IETF, "The Constrained Application Protocol," https://tools.ietf.org /html/rfc7252, [Online; Accessed 10 - December - 2020].
[11] OASIS, "MQTT - Message Queuing Telemetry Transport," http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, [Online; Accessed 10 - December - 2020].
[12] IETF, "Hypertext Transfer Protocol version 2," https://tools.ietf.org/html/rfc7540, [Online; Accessed 10 - December - 2020].
[13] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Journal for General Philosophy of Science*, vol. 39, no. 1, 2008.
[14] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, pp. 557–564, 2017.
[15] Nick Szabo, "The Idea of Smart Contracts," https://nakamotoinstitute.org /the-idea-of-smart-contracts/, [Online; Accessed 11 - December - 2020].
[16] Ethereum, "Proof of Stake - Ethereum," https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/proof-of-stake/, [Online; Accessed 11 - December - 2020].
[17] Quorum, "Quorum - The proven blockchain for business," https://www.goquorum.com/, [Online; Accessed 11 - December - 2020].
[18] Hyperledger Fabric, "Smart contracts and chaincode," https://hyperledger-fabric.readthedocs.io/en/release-1.4/smartcontract /smartcontract.html, [Online; Accessed 11 - December - 2020].
[19] Ethereum, "Developer Resources," https://ethereum.org/developers/getting-started, [Online; Accessed 11 - December - 2020].
[20] The Ethereum Foundation, "The Ethereum Virtual Machine," https://ethereum.org/en/developers/docs/evm/, [Online; Accessed 11 - December - 2020].
[21] Y. Hanada, L. Hsiao, and P. Levis, "Smart contracts for machine-to-machine communication: Possibilities and limitations," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, 2018, pp. 130–136.
[22] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.
[23] X. Gong, E. Liu, and R. Wang, "Blockchain-based iot application using smart contracts: Case study of m2m autonomous trading," in *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, 2020, pp. 781–785.
[24] T. Alladi, V. Chamola, R. M. Parizi, and K.-K. R. Choo, "Blockchain applications for industry 4.0 and industrial iot: A review," *IEEE Access*, vol. 7, pp. 176 935–176 951, 2019.
[25] M. Kuperberg, D. Kindler, and S. Jeschke, "Are smart contracts and blockchains suitable for decentralized railway control?" 2019.

**Nuno A. L. Leite** is a final year student of the Integrated Masters in Informatics Engineering at the University of Minho, and a Research Technician in the Software & Information Systems department at the Digital Transformation CoLab. His research interests are currently focused towards information security, particularly in novel blockchain applications, as well as software engineering.

**Alexandre J. T. Santos** is Associate Professor at the Informatics Department, University of Minho, and Senior Researcher at Centro Algoritmi. He received his PhD in Computer Communications from the University of Minho in 1996. He is in the lead of the Computer Communications and Pervasive Media (CCPM) research group and has several participations in European and other international R&D projects. He has authored more than eighty scientific papers at international refereed Journals and Conferences and has served as Journal Editor and TPC member for several IEEE, ACM and IFIP sponsored events. Recognized as an IEEE Senior Member (SM'10) and member of the IEEE CS and IEEE VTS Society.

**Nuno Lopes** is the Chief Technology Officer (CTO) of the National Lab on Digital Transformation in Portugal and professor on a part-time basis at the University of Minho. He has two Postdoctoral courses, one on Computer Science at the University of Coimbra and another on Electronic Governance at the United Nations University. Previously, he worked at the United Nations University, where he founded and coordinated the research line on Smart Cities. During his working life, he has been involved in several national, European and international projects, such as Electronic Governance for Context-Specific Public Service Delivery, Knowledge Society Policy Handbook, Policy Monitoring on Digital Technology for Inclusive Education, Intelligent Computing for Internet and Services, Internet of Things for Disabled People, Smart Defense and Smart Cities for Sustainable Development. Editor of the Springer Book on "Smart Governance for Cities: Perspectives and Experiences", Editor of the IGI Global Book on "Developing Knowledge Societies for Distinct Country Contexts", co-author of the United Nations Report on Smart Sustainable Cities for Developing Countries and of the UNESCO Handbook on Knowledge Societies.