

POLITECNICO DI TORINO

Master's Degree Course
in Computer Engineering

Master's Degree Thesis

Industry 4.0 and blockchain: On the use of distributed ledgers for
supply chain management



**POLITECNICO
DI TORINO**

Supervisor
Prof. Fabrizio Lamberti

Candidate
Alberto Doglioli

October 2019

Abstract

Since the birth of Bitcoin, in early 2009, an explosion of researches and investments were unleashed towards a new disruptive technology called blockchain. This new technology, that consists in a sequence of blocks cryptographically linked one to each other, provides a fully auditable and immutable ledger, where data, once stored, cannot be modified. The blockchain aims to revolutionize the interaction between people and machines in an untrusted distributed environment without the need of a central trusted authority. From the birth of Bitcoin, most applications built on top of such technology were mainly in the FinTech sector, regarding exchange of cryptocurrencies, in the so-called permissionless blockchain network. However, in recent years, researchers and companies studied the feasibility of adopting blockchain-based solutions also in different sectors, from supply chain management systems to data provenance and data reliability. In this context were born the first permissioned blockchains, where members are known, and they must own a digital identity in order to join the network.

This thesis analyzes the adoption of a blockchain-based solution in the context of supply chain for Avio Aero – a GE Aviation Business use case in the Industry 4.0 framework. The aim is to create a system where data are securely stored, tamper-proof and easily auditable in order to eliminate paper-based documents and to make the production environment more efficient. For the purpose of this thesis a SWOT matrix analysis has been performed in order to study the feasibility and the advantages and disadvantages of adopting a blockchain-based solution for the use case provided. In addition, a proof of concept has been developed with the use of Hyperledger Fabric for building the blockchain network and ExpressJS for developing a REST web server in order to interact with the underlying network.

Acknowledgements

First of all, I would like to thank my supervisor, prof. Fabrizio Lamberti, who supported me during my period of thesis.

A special thanks goes to Davis Quirico, who believed in me and supported me during my internship at Avio Aero.

Heartfelt thanks to my family that believed in me every day of my life and gave me the opportunity to reach this fantastic goal.

Table of contents

1. INTRODUCTION	- 8 -
1.1. Chapter content	- 8 -
1.2. Avio Aero – A GE Aviation Business.....	- 9 -
1.2.1. Avio Aero use case	- 10 -
1.3. Goal of the thesis	- 12 -
2. STATE OF THE ART	- 13 -
2.1. Introduction.....	- 13 -
2.2. Supply Chain in Agri-Food domain	- 14 -
2.2.1. AgriBlockIoT	- 15 -
2.2.2. Traceability system for food supply chain.....	- 17 -
2.3. Blockchain-based solution for product’s traceability.....	- 19 -
2.4. Blockchain ready manufacturing supply chain.....	- 20 -
2.5. Digital Supply Chain transformation	- 22 -
2.6. Collaborative Business Process	- 24 -
2.7. Data Reliability and Data Provenance	- 26 -
2.7.1. DataProv	- 26 -
2.7.2. Decentralizing privacy	- 28 -
2.7.3. ProvChain: a blockchain based architecture for data provenance	- 30 -
2.8. Conclusions	- 32 -
3. BLOCKCHAIN TECHNOLOGY	- 33 -
3.1. Introduction to blockchain.....	- 33 -
3.2. Centralized, decentralized and distributed	- 33 -
3.2.1. Centralized databases	- 33 -
3.2.2. Distributed databases	- 34 -
3.2.3. Decentralized databases architecture.....	- 34 -
3.3. Relational and non-relational databases	- 36 -
3.3.1. Relational database.....	- 36 -
3.3.2. SQL	- 36 -
3.3.3. Non-relational databases	- 36 -
3.4. Distributed systems	- 36 -
3.5. CAP theorem.....	- 37 -
3.6. Hash functions	- 37 -
3.6.1. Aliasing	- 38 -
3.6.2. Birthday paradox	- 38 -
3.6.3. One-way function.....	- 38 -
3.7. Cryptography	- 39 -
3.7.1. Kerckhoffs principle.....	- 39 -
3.7.2. Symmetric cryptography.....	- 40 -
3.7.3. Asymmetric cryptography.....	- 40 -

3.7.4.	Confidentiality	- 40 -
3.7.5.	Key exchange using asymmetric cryptography.....	- 40 -
3.7.6.	Digital signature	- 41 -
3.8.	Blocks and transactions.....	- 41 -
3.9.	Chain of blocks	- 42 -
3.9.1.	Break the chain	- 42 -
3.10.	Distributed blockchain	- 42 -
3.10.1.	Permissionless blockchain.....	- 42 -
3.10.2.	Permissioned blockchain.....	- 42 -
3.11.	Byzantine fault tolerance	- 43 -
3.11.1.	Byzantine Generals' problem.....	- 43 -
3.12.	Consensus	- 43 -
3.12.1.	PoW (Proof-of-Work)	- 44 -
3.12.2.	PoS (Proof-of-Stake)	- 46 -
3.12.3.	DPoS (Delegated proof-of-Stake)	- 46 -
3.12.4.	PoA (Proof-Of-Activity)	- 46 -
3.12.5.	PoC (Proof-of-Capacity)	- 47 -
3.12.6.	PoET (Proof-of-Elapsed-Time).....	- 47 -
3.13.	Smart Contracts	- 48 -
3.13.1.	Ricardian Contracts	- 48 -
3.13.2.	Oracles	- 48 -
3.13.3.	Languages for Smart Contracts	- 49 -
3.14.	Well-known blockchain vulnerabilities	- 49 -
3.14.1.	51% attack	- 49 -
3.14.2.	DDoS (Distributed Denial of Service)	- 49 -
4.	TECHNOLOGIES	- 51 -
4.1.	Ethereum	- 51 -
4.1.1.	Introduction	- 51 -
4.1.2.	Public blockchain framework and EVM	- 51 -
4.1.3.	The Yellow Paper.....	- 52 -
4.1.4.	The Ethereum blockchain	- 52 -
4.1.5.	Summary and conclusions	- 52 -
4.2.	Hyperledger.....	- 53 -
4.2.1.	Introduction	- 53 -
4.2.2.	Hyperledger building blocks	- 53 -
4.2.3.	Hyperledger Iroha	- 54 -
4.2.4.	Hyperledger Sawtooth	- 54 -
4.2.5.	Hyperledger Burrow	- 55 -
4.2.6.	Hyperledger Fabric.....	- 56 -
4.2.7.	Hyperledger Fabric component design.....	- 57 -
4.2.9.	Transaction's trip	- 59 -
4.2.10.	Summary and conclusions	- 61 -
4.3.	AWS proposed solutions.....	- 62 -
4.3.1.	Introduction to AWS	- 62 -
4.3.2.	AWS Managed Blockchain	- 63 -
4.3.3.	AWS QLDB.....	- 64 -
5.	AS IS vs TO BE.....	- 66 -

5.1.	AS IS situation	- 66 -
5.1.1.	High-level BPMN diagram of the AS-IS situation.....	- 67 -
5.2.	SWOT matrix analysis	- 68 -
5.2.1.	Strengths	- 69 -
5.2.2.	Weaknesses	- 71 -
5.2.3.	Opportunities	- 72 -
5.2.4.	Threats	- 72 -
5.2.5.	Conclusions.....	- 73 -
5.3.	Proposed solution	- 74 -
5.3.1.	High-level perspective	- 74 -
5.3.2.	Architecture overview	- 75 -
5.3.3.	Blockchain network proposal	- 76 -
5.3.4.	REST server	- 77 -
5.3.5.	Web-based client application	- 78 -
5.3.6.	Simulator	- 79 -
6.	PROOF OF CONCEPT	- 80 -
6.1.	Introduction and scenario description	- 80 -
6.2.	Blockchain layer	- 80 -
6.2.1.	UML diagram	- 81 -
6.2.2.	Why Hyperledger Fabric?.....	- 81 -
6.2.3.	Hyperledger Fabric project	- 82 -
6.2.4.	Digital Stamping smart contract	- 87 -
6.3.	PoC simulator	- 91 -
6.4.	Backend REST server layer	- 92 -
6.5.	UI client application layer	- 95 -
6.5.1.	Query all rows function.....	- 96 -
6.5.2.	Query row function	- 96 -
6.5.3.	Updating the ledger	- 97 -
7.	CONCLUSIONS.....	- 99 -
7.1.	Summary	- 99 -
7.2.	Proposed solution	- 99 -
7.3.	Final considerations and future works	- 100 -
	Bibliography.....	- 101 -

1. INTRODUCTION

1.1. Chapter content

This chapter aims to present the Avio Aero - a GE Aviation Business company and to present the use case in which such company would like to apply a blockchain-based solution. In Section 1.2 there is brief description of the company and the provided use case. In Section 1.3 the goal of the thesis is described.

The remaining of this thesis is organized as follow.

- Chapter 2 presents a deep analysis of the state of the art of the blockchain technology and its main applications. In this context there were analyzed not only many supply chain applications but also some interesting application in different sectors in order to underline the main aspects of such technology.
- Chapter 3 reports an overview of the blockchain technology, considering its innovative behavior in order to better understand its potential in many different applications, both in a public environments and private ones.
- Chapter 4 presents an analysis of the technologies analyzed for the goal of this thesis. In particular, Ethereum, Hyperledger Fabric and AWS solutions are analyzed and compared.
- Chapter 5 provides a high-level view of the proposed solution, considering the layered approach that has been followed. This chapter reports also a SWOT matrix analysis that has been used in order to deeper study the actual benefits of adopting a blockchain based solution.
- Chapter 6 presents the proof of concept developed for the goal of this thesis. The PoC consists in a blockchain network, a simulator to populate the network, a backend REST server to interact with such network and a web-based client application for users to interact with the ledger.
- Chapter 7 concludes the thesis with final conclusions and future work

1.2. Avio Aero – A GE Aviation Business

Avio Aero is a business of GE Aviation devoted to the design, production and maintenance of components and systems for civil and military aeronautics. Avio Aero provides its customers with innovative technological solutions to quickly respond to the continuous changes required by the aeronautics market: additive manufacturing and rapid prototyping are two examples, but also technologies dedicated to the production of transmissions, turbines and combustors.

GE Avio Aero first challenge is to develop and adopt new technologies in order to reduce energy consumption and make aircraft engines ever lighter and achieve better performances.

Avio Aero achieved a globally recognized technological and manufacturing excellence thanks to a consolidated network of relationships with the main universities and international research centers around the world, with continuous investments in research and development: a goal witnessed by the partnerships signed with the main global aeronautical operators.

Brilliant factory and Industry 4.0 - A digital revolution

To better perform in the aeronautic market and to increase the productivity and competitiveness, it is necessary to take advantage of the opportunities offered by digitalization.

In Avio, what is called fourth industrial revolution has already begun, through the idea of *brilliant factory* (that is the intelligent factory), where data are analyzed in real time in order to obtain a place of production that can continuously self-improve its products and processes. The brilliant factory in Avio is visible in a faster product development cycle and improved production efficiency.

In this context, advanced manufacturing plays a fundamental role and, even more specifically, the additive manufacturing, that see Avio Aero in the forefront thanks to their Cameri plant, that is one of the biggest plants in the world completely dedicated to additive manufacturing.

GE Avio Aero and blockchain

It is within the Brilliant Factory idea and following the Avio Aero philosophy, considering their investments in research with universities, that the desire to apply the blockchain to the production phase was born, with the objective that data are securely stored and are immutable, that workers perform better and that wasted time (that naturally comes with paper-based documents) is reduced to a minimum.

Another important aspect that should be considered is that for an excellence like Avio Aero, the adoption of a blockchain-based solution, means being competitive on the market and also the reputation and trust of Avio Aero around the world could increase adopting such a new innovative technology.

1.2.1. Avio Aero use case

The goal of this thesis is to evaluate the possibility to introduce the blockchain within the production cycle, to record and trace the different operations performed on each part produced along the production cycle. The Figure 1 reports a clearer visual representation of what the production cycle is intended to be.

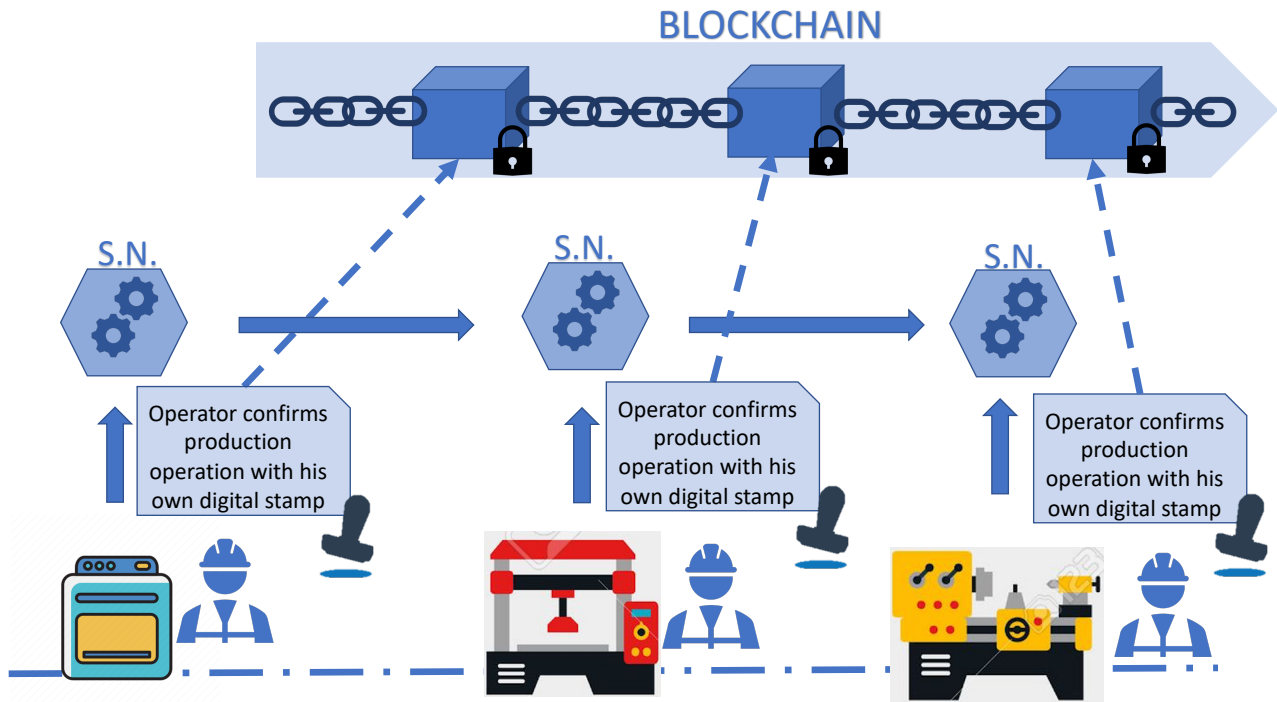


Figure 1 – A visual representation of the production cycle with the adoption of the blockchain

This simple figure schematically shows the workflow that Avio Aero intends to store in the blockchain.

In this example it is easy to observe that different operators work on different workstations of the production process. Actually, each worker, once finishes his/her job on a certain component (identified by its serial number S.N.), physically applies a stamp on the worksheet that certify the successful execution of the work on that component on that workstation by that worker. The project is to securely store in the blockchain all that information that in this way will become immutable, with a complete dematerialization of the documentation of the production process and an easy access to all the relevant information.

Table 1 represents an example of how a worksheet actually looks like, considering only the most relevant information for the goal of this thesis and without providing any sensible and private information related to the real worksheet.



Part number: PNXXX001					
Serial number: SNZZZ01					
Operation code	Workstation	Description	Worker	Date & Time	Stamp
COD111	WSZ1	DES1	OPX1	10/01/2019 10:00	
COD222	WSZ2	DES2	OPX2	14/01/2019 15:30	
COD333	WSZ3	DES3			
COD444	WSZ4	DES4			

Table 1 – A visual representation of the current worksheet

Each row of the worksheet consists in all the information of a certain production step. When a worker finishes the work, he has to apply the stamp and compile the worksheet with his/her WorkerID (that is the SSO) and with the current date and time information. The above reported sample worksheet, in the current situation, is manually compiled by workers when they finish their respective jobs and then it is brought to the next worker for the next operation.

Considering the sample worksheet reported, actually “SNZZZ01” already has crossed WSZ1 and WSZ2 and OPx1 and OPX2 worked on it. Now, OPX2 has to bring the worksheet to the workstation WSZ3 for further operations.

The information above reported in the worksheet is the one that has been used in order to design and develop this thesis.

1.3. Goal of the thesis

The goal of this thesis is to study in depth the blockchain technology, evaluating its different implementations in private blockchain solutions with a deep analysis of the state of the art of such a new technology, mainly in the Industry 4.0 framework, in order to better understand its behavior and to develop a solution for the use case provided by the company.

After a general explanation and presentation of how a blockchain works and which are its building blocks, the most widespread technologies are analyzed namely Ethereum [1], Hyperledger [2] and AWS proposed solutions [3].

Then, in order to evaluate the feasibility and the utility of adopting this kind of solution in a company like Avio Aero, a detailed SWOT matrix analysis has been adopted. The SWOT matrix analysis considers the use case provided by the company, the “Digital Stamping” case and the actual state of the company.

The core of the thesis is the Proof-of-Concept developed in order to test the Hyperledger Fabric network and to use the main commands to set up the network and to send transactions to the system, referred to the digitalization of the management of the production cycle control in Avio Aero.

The PoC consists in a smart contract written in JavaScript [4] and Go [5] programming languages, a backend server, developed using ExpressJS [6], a user interface developed using Angular [7] and Node.js [8] and a simulator in order to populate the blockchain with some sample transactions.

2. STATE OF THE ART

In this chapter of the thesis, the state of the art of supply chain management systems and blockchain based proposed solutions are analyzed considering the most relevant and recent application for the goal of this thesis. The goal of this chapter is to better analyze the actual situation of supply chains, the most important problems (in terms of trust and data reliability) related to it and to analyze the papers related to blockchain-based proposed solutions for solving such problems.

2.1. Introduction

In recent years, in the supply chain management systems area, thanks to a constant growing of the Internet era, many emerging technologies have been studied and considered for traceability mechanisms. With the explosion of the Internet, IoT devices applications, RFID systems and lately also with some blockchain based solutions, a new era for traceability of goods and supply chain management systems has started. However, most of these systems are characterized by a centralized approach, where the most critical aspects are related to data availability, transparency and auditability, security (because of the inherently point of failure of centralized systems), scalability and performances. Moreover, the centralized solutions are monopolistic, in fact there can be issues in the context of trust, data tampering, corruption, fraud and falsified information. A solution of such problems can be found in adopting a blockchain-based system.

With the launch of Bitcoin [9] in early 2009, an explosion of research and investments were unleashed, mainly in the FinTech (financial technology) industry. However, considering the inherently features of the blockchain technology (in particular its immutability, its tamper-proof feature, the enabling of trust among untrusted and potentially unknown peers), many industries believe in the application of such technology in very different areas with respect to the FinTech world. In the last few years in fact, many business areas faced with the blockchain technology and are trying to adopt this technology in many different sectors.

One of the most interesting study areas of blockchain for the gal of this thesis is SCMS (Supply Chain Management Systems).

Since many actors are involved in a supply chain, there can be many issues related to this complex network of participants. Before reaching the end user, a product crosses the supply chain network and undergoes many steps. Nowadays, the final customer has a very poor knowledge of when his product has been produced, where it was before the delivery or who worked on it. The journey of a product, from its design to its delivery to the final customer, includes many steps where suppliers, warehouses, transporters distributors and retailers work on the design phase, the production phase, the delivery phase until the sale to the end customer. Considering for example an IoT based traceability systems, it is not necessary for each device to be connected to the Internet network, but only to be connected to the nearest peer of the blockchain network. In addition, in this way, IoT devices are no more subject to external attacks and they cannot be used as a vector of attack, such as happens for example the DoS or DDoS attack.

The next sections of this chapter consist in a deep analysis of the most interesting blockchain based applications present in the most recent publications.

2.2. Supply Chain in Agri-Food domain

As explained before, in literature most of the applications of blockchain are in the FinTech area, considering the well-known scenario of an exchange of a certain digital asset (e.g. a cryptocurrency). By Investopedia [10] Bitcoin, Ether, Ripple, Bitcoin cash and EOS are some of the most famous ones.

However, many different areas are studying the feasibility and the utility of adopting such a solution.

Nowadays, in the Agriculture and Food domain, the supply chain mainly focuses on store orders and deliveries, without considering features like verifiability, transparency and traceability [11]. Many communities of researchers started to propose the adoption of IoT technology (e.g. RFID tags and wireless sensors) in order to remotely control the condition in some food delivery scenarios. Anyway, all this IoT devices still rely on centralized infrastructures where features like transparency and auditability can't find space [12].

Considering all the incidents and scandals related to the quality of food which happened in last decades, for example mad cow disease, toxic milk powder [13] or trench oil [14], customers trust in food quality and safety has dramatically fallen.

Modern Agri-Food supply chains are characterized by the cooperation of many actors, each one of them contribute to increasing the heterogeneity of such scenario. Differences can be found in terms of different business models, different levels of confidentiality and lacks in interoperability in the system.

Main actors involved in Agri-Food supply chain are reported below.

- **Provider.** This term typically identifies who provides the raw material (e.g. seeds, nutrients, pesticides and chemicals)
- **Producer.** The producer is who receives raw materials and proceed with the workflow (e.g. planting and harvesting)
- **Processor.** Is an intermediate actor who, depending on different scenarios, performs certain operations (from the simplest ones, e.g. packaging, to something more complex, like for example grape press for producing wine)
- **Distributor.** Is the actor involved in moving the produced good (from producer's plants) to the retailers.
- **Retailer.** Is who physically sell the final product to the end customers.
- **Consumer.** The consumer is the final stage of the supply chain, the one that buy the final product from the retailer.

Considering the supply chain described above, each of the involved actors (with the exception of the final consumer) must comply with standards, rules and policies provided by the competent authorities.

In order to provide a greater trust to the end consumers, many different industries and companies working in the Agri-Food domain started to study the feasibility of adopting a blockchain-based solution. In the food quality context, data immutability, impossibility to tamper with data and the inexistence of a third party are the most desirable characteristics.

In this section the most interesting use cases are reported and deeply analyzed in order to study the behavior and the usefulness of such change.

2.2.1. AgriBlockIoT

Introduction

In the context of goods traceability in the Agri-Food domain, authors of [11] provided a fully distributed traceability system that can work together with many different IoT devices and actually can run both on an Ethereum blockchain or on Hyperledger Sawtooth distributed ledger. AgriBlockIoT uses IoT devices in order to produce and consume data that are directly stored in the blockchain at an underlying level. The goal of the authors is to provide traceability of transparent and verifiable activities. To test the feasibility of the proposed solution, authors used the “from-farm-to-fork” well-known use case, in which their goal is to provide certified food traceability data along the whole supply chain, that means from agriculture (the farm) to the end consumers’ houses (the fork).

Proposed solution

In the proposed solution, the authors proposed a layered architecture for the entire system. At the bottom it is possible to find the blockchain (running on cloud) that contains all the business logic of such application. They wanted to develop a blockchain-independent solution in order to be compatible with existing traditional systems (such as ERP or CRM). The business logic is implemented in smart contracts and stored directly on the blockchain. The complexity of such contracts can vary according to the underlying selected blockchain. The smart contract in the blockchain acts also as a gateway for the blockchain itself.

On top of the blockchain layer there is the so-called “edge”, that includes all the edge devices (for example, gateways or mini-PC) that can act as full node of the underlying blockchain.

The top level of the layered proposed solution includes actors involved in the whole process.

The proposed solution consists also in a REST API exposed to other application in order to be able to use the AgriBlockIoT functionalities. The main goal of this API is to enable the integration with existing software systems. Between the blockchain and the REST API there is a software module (called “controller”) that serves to transform the API functions call to the low-level function calls of the blockchain layer.

Use case application

Following the supply chain, all the actors involved in the process must interact with the blockchain. In order to do this, each one of them must have a digital identity within the blockchain and must be registered to it. This means that everyone must have a valid public and private key pair for digitally signing their operations on the blockchain.

Authors follow a bottom-up approach, starting from the requirements of the application in order to develop it. The typical scenario is described below.

Providers store raw materials data in the blockchain, including all the relevant technical information related to them. Once providers sell goods to producer, the information about the ownership of such goods is updated in the blockchain. Producers store information related to seeding process. Producers’ sensors can automatically store data related to the growing of the seedlings and potentially also data related to the environment (temperature, humidity, etc.). farmers store information about the process of plants’ growth (irrigation, fertilizing and harvesting). Once plants are ready to be sold, the ownership of such goods is updated in the blockchain, from farmers to processors. After the processors’

information are stored on the blockchain, the ownership is again updated, from processors to retailers. Retailers store in the blockchain the amount of received products from the processors. Finally, once the product is sold to the customers, the information is again stored in the blockchain. At the end of the entire process, customers are able to have a look at the entire history of the product they bought (or they are going to buy) and transparently verify it.

Is important to underline that authors designed the system in such a way it is fully automatable. In each phase, IoT devices can automatically store information they collect on the blockchain and smart contracts can automatically register failure records when they detect anomalies.

From a more technical point of view, since AgriBlockIoT is blockchain independent, as discussed in the introduction, authors implemented two different blockchain layer, one with Ethereum and the other with Hyperledger Sawtooth. Both networks were built with six nodes and deployed in dedicated virtual machines with 4GB of RAM, 2 Intel® Core™ i5-6440HQ CPUs 2,6GHz and 20GB hard-disk, running Linux Ubuntu 16-04 as operating system. The simulation consisted in one hundred tests, each one of them consisted in setting the value of an IoT sensor and submitting a transaction to the blockchain.

Results

For each test, authors of AgriBlockIoT, measured the latency (that is the time needed to submit the value to the blockchain), the CPU load of each node and the usage of the network (byte transmitted and received).

In Table 2 are summarized their obtained results:

	Latency [seconds]	Network tx [bytes]	Network rx [bytes]	CPU load [%]
Ethereum	16.55	528108	682415	46.78
Hyperledger Sawtooth	0.021	19303	20641	6.75

Table 2 – AgriBlockIoT obtained results

Conclusion

AgriBlockIoT provide an immutable, tamper-proof and auditable system where end consumers are more confident about the product lifecycle they buy. This solution is efficient and respects all the requirements provided by the Agri-Food domain.

This use case is the typical one where industries and companies are working on in order to improve customer confidence even more. In addition, this solution is convenient for all the actors involved in the supply chain because if there is something wrong in the final product, it is easily possible to trace the culprit simply by following the history of the product. So, considering this aspect, each one of the involved participants has a strong motivation to keep the ledger updated.

2.2.2. Traceability system for food supply chain

Introduction

Remaining in the context of food traceability F. Tian proposed a real-time traceability system for food supply chain [15]. The system is based on HACCP (Hazard Analysis and Critical Control Points), IoT devices and blockchain. In addition, F. Tian used BigChainDB [16] for storage purposes.

The aim of the proposed solution is to be an information platform for all the actors involved in the supply chain, providing tamper-proof of data, transparency, reliability and auditability.

BigChainDB

BigChainDB is a software that combines both blockchain and centralized software database properties. Decentralization and tamper-proof data features are derived by the blockchain ecosystem and low latency, high transactions rate and query of structured data derive instead from the databases world. BigChainDB was first released open source in early 2016. It continuously is up to date and starting from version 2.0 it is now Byzantine fault tolerant (BFT).

BigChainDB was born from the idea that blockchain features and relational databases ones are strongly complementary to each other.

BigChainDB implements immutability by chronological blocks, each one of them hold a sequence of ordered transactions, similarly to what happens in blockchain. In addition, with prior authorization (or with the key of the asset), any user can issue an asset. Finally, the decentralized feature can be achieved through nodes of the system with a voting process, also known as super-peer P2P network.

Proposed solution

Tian proposed a distributed system, composed of IoT devices to collect and transfer data. The entire proposed system relies on BigChainDB for storage purposes.

In this architecture, each one of the involved actors (providers, producers, manufacturers, retailers, etc.) in the supply chain, with prior registration through a registrar, can insert and update information about the product in the BigChainDB storage. In the proposed solution, each product must have a digital identity (in the form of an RFID tag) that represent a unique cryptographic identifier linked to the physical product. Also the users in the system, in order to interact with data, must have a valid digital identity, which includes all the information related to users (location, certifications, related products, etc.). After the registration phase, users are provided with credentials and a unique identity. With the public key, users are able to be identified by other users while the private key enables the signing of transactions in order to be later verified.

The rules of the whole system, according to the blockchain philosophy are stored in the BigChainDB storage and they can be modified only if broadcasted to each node and verified by the majority of them. In the proposed solution there are still some third-party authorities but with a revisited guise and function. First of all, also these authorities must have their own digital identity. The aim of such third-party authorities is to check and verify users' identity and conduct. For example, they can visit farms to inspect if they are compliant with relevant rules and regulations. Once verified, the authority can update the information related to the user, signing such information. All data related to authorities' evaluation of users must be published in the system.

Application scenario

The following application scenario is reported to better understand how the entire system work. In the following it is assumed that each user already has a unique digital identity provided by the system registration service.

In the sample application scenario new products are added by farmers, that harvest and pack the cultivated plants. Each plant is labeled with an RFID tag and then a new record is stored in the system. Plants information can include the environmental data (air temperature, humidity, etc.), growing data (fertilizers, pesticides, planting time, etc.) and packaging data. After this phase, a new trade can be initialized between farmers and producers (that means from the provider to the producer) and both parties have to sign a digital contract stored on the BigChainDB. When goods arrive to the producer enterprise RFID tag can be read by the wireless network of connected scanners and product's information can be updated by producers. Also in this phase some new information are added to the product's profile (such as producing environmental data, disinfecting, etc.). In the warehousing phase, with the use of wireless sensors, upon reception of a new good, the product's profile can be automatically read by those sensors. In addition, the use of a wireless network of connected sensors can provide real-time data storage information such as quantity, category, storage time, or environmental temperature and automatically update that information in products' profiles. Again, in the distribution phase, wireless vehicle-mounted environmental and GPS sensors can automatically update products' information during their distribution. Finally, when goods reach the retailer, customers can retrieve information about the whole life cycle of such product and, as the system is based on a blockchain-like storage layer, that information is verifiable by customers themselves. On the other side, there are also advantages for the entire system because when an incident of food safety happens, all the involved products can be easily located.

Conclusions

The solution proposed by F. Tian consists in a new real-time traceability system based on IoT devices (such as wireless connected scanners and RFID tags) and a blockchain-like storage solution. The interesting aspect of such solution is the capability of the system to auto-update products' information using data collected by the IoT devices and sensors. In the food domain, is also important also to have a transparent and auditable system to provide more and more trust to end customers.

2.3. Blockchain-based solution for product's traceability

Authors of [17] implemented a blockchain-based solution for an existing products' traceability system. Their aim was to substitute the original centralized database with a distributed one, including the blockchain technology. OriginChain's main purposes are to provide transparent tamper-proof traceability data, improve availability of such data and automate regulatory compliance controls.

As for the OriginChain's architecture, it is currently composed of a geographically distributed private blockchain that spans three different countries. The whole architecture of the main system is composed of three main entities:

- **Suppliers or big retailers** which manage product information. Actually, they can access data on the blockchain through a web server hosted by OriginChain, but in the future they will have access to blockchain data by hosting a node themselves.
- **Traceability service provider** is in charge to manage traceability information, certificates and photos. The traceability service provider doesn't store all the information on the blockchain, but it stores the hash of traceability certificates and a subset of all the required information requested by the traceability regulation (e.g. the batch number, traceability result, product's origin, and inspection timestamp). All other information is stored in a centralized MySQL database (hosted by OriginChain).
- **Labs** are entities that perform samples tests and manage sample-testing results data. Periodically they submit the sample testing results to the blockchain.

In OriginChain proposed solution, access control data can be stored on-chain or off-chain. Such data can include permissions for interacting with the blockchain or permissions of managing smart contracts. Anyway, OriginChain stores that information on-chain, in order to avoid all the problems related to a centralized solution, such as for example a single point of failure. So, OriginChain stores itself data related to controls and permissions in order to exploit the full potential of such a distributed blockchain-based solution.

Concerning what should be on-chain and off-chain, OriginChain doesn't store raw data related to traceability on the blockchain but it stores only the hashes of such information. So, first the hash of the record is evaluated and then the entire information is stored off-chain while the hash is stored on the blockchain in order to keep data safe. This mechanism allows to keep private sensible data and at the same time it provides data immutability and tamper-proof data. Also private users' data are stored off-chain, since all the participants can easily access data on the blockchain.

Authors continue the paper affirming that the blockchain should not be used only for data storage, in terms of system's adaptability; instead, some business logic must be moved to it in order to exploit the reliability that the blockchain provides as a computational platform. In the conclusion of the paper, Q. Lu et al. underline the fact that smart contracts do not belong to anyone (not even the developers who developed them) and therefore an access control mechanism would be necessary in order to check rights on functions calls of such smart contracts.

2.4. Blockchain ready manufacturing supply chain

In supply chain management context, authors of [18] proposed a blockchain-based solution that is able to collect and manage key product information of each product throughout its supply chain cycle. The limit of modern supply chain management systems is the poor knowledge, from end consumers point of view, of how and when the product has been produced and who worked on the product they are going to buy. The goal is to create a secure and auditable record for the exchange of each product, containing also all the specific information about that product.

Proposed solution

During the product's journey (from producers to the end consumers), each actor involved play an important role in the system, logging the current status of the product and the key information. Each product is assigned to a unique digital profile, containing all the information added during all the supply chain steps. Each product has attached a digital identifier, in form of RFID tag, QR code or simply a barcode, that links the physical object to its digital identity in the blockchain network. As for products, also each actor must have a digital identity in order to interact with the blockchain. Each actor can register himself to the blockchain network through the registrar service, which is in charge of providing the credentials and the certificate that represent the user's digital identity. At the time of registration, the public and private keys pair is provided to the user.

The proposed system consists of many actors. The following is a summary of stakeholders involved and a brief description of the related role.

- **Registrar service.** Is the service in charge of providing a unique digital identity to all the actors involved in the interaction with the network
- **Standards organizations and certifiers.** These are entities involved in releasing certificates of standards compliance and certificates to actors, allowing them to participate in the network.
- **Supply chain actors.** This set of actor includes producers, manufacturers, distributors and retailers which enter new products' data on the blockchain and update them.
- **Consumers.** Identify the customer, the end consumer is who buy the final product.

Besides the previous described actors, there is the software developed for the solution proposed by A. Abeyratne et al. in [18]. Each actor of the system can interact with the blockchain using a user interface, that is the software that allows user to access products' data they are associated with. The software application has been developed by a set of trusted parties and runs directly on the blockchain, enabling the execution of programmable code. The goal of such software is to make it easier the interaction with the blockchain; in particular, through the software it is possible to add new entries on the blockchain and query existing data. The system stores all data on the blockchain, and such data are accessible to anyone who has an authenticated digital identity within the network. Actors can access data respecting some rules that are stored in the blockchain as well. Such rules are written in form of code and define how the involved actors must interact with the network. In addition, storing rules in the network makes them immutable and they work exactly how they were defined unless they are broadcasted to all the nodes of the network and approved by the majority of them.

As described by F.Tian [15] in the solution previously described for food traceability, also here certifiers and standards organizations play the role of visiting sample actors of the system in order to check if they are standard compliant and they respect rules for standard programs. Again, when a user passes the certification authority's control, his profile is updated, and its products can be digitally signed by the certifiers.

Data access and management

As explained before in this chapter, each product in the system has an associated digital tag (that could be anRFID tag, a QR code or a barcode) that uniquely identifies it in the system. At a given time in the product life cycle, the product is owned by a specific actor and only this actor must have the right of updating its information and starting a new trade with another actor. During the ownership transfer of a certain product, both the involved parties must sign a digital contract in order to authenticate and validate the exchange. Upon the completion of the exchange, the system updates the ownership of the specific product.

The following is a summary of the most interesting data managed by the system.

- **Ownership.** The system stores the entire history of the product's ownership and the current owner. Every time a new trade is completed, the product's profile is updated, and a new record is created in the system.
- **Timestamp and location.** Every new record is timestamped with the current timestamp of the submitted transaction. This is useful in order to provide a chronological order of all the entries. In addition, the current physical position is kept by the system. The location data can have the form of a simple location identifier or a set of GPS coordinates.
- **Product specific information.** This is the information related to a specific product and it constitutes in the key information in the system. It includes all the product's attributes and it is used to provide a feedback to all the actors involved in the supply chain, and also to the quality controller.

All the above specified information is accessible from users with prior authentication.

Conclusions

The solution proposed by Abeyratne et al. [18] exploit the full potential of a blockchain based application in supply chain management systems. Authors extended the whole system to all the actors involved in the supply chain. The peculiarity of such solution is the software developed for the application and running on the blockchain. It provides different user interfaces depending on users' roles and rights. In addition, most operations are automated thanks to a smart contract that automatically update the product's profile when some specific conditions are met.

2.5. Digital Supply Chain transformation

The term Digital Supply Chain (DSC) represents the digitalization of the supply chains in the world. Most of them, according to K. Korpela et al. in [19], still follow an old and outdated model. Each company involved in the supply chain has its own information system and, in most cases, it is different from the ones of the other actors in the supply chain. In exchanging document, in fact, a computer-paper-computer model is followed. As result of such an old model, lots of documents are produced and exchanged from one company to another. A clearer visual representation of the actual situation is reported in Figure 2.

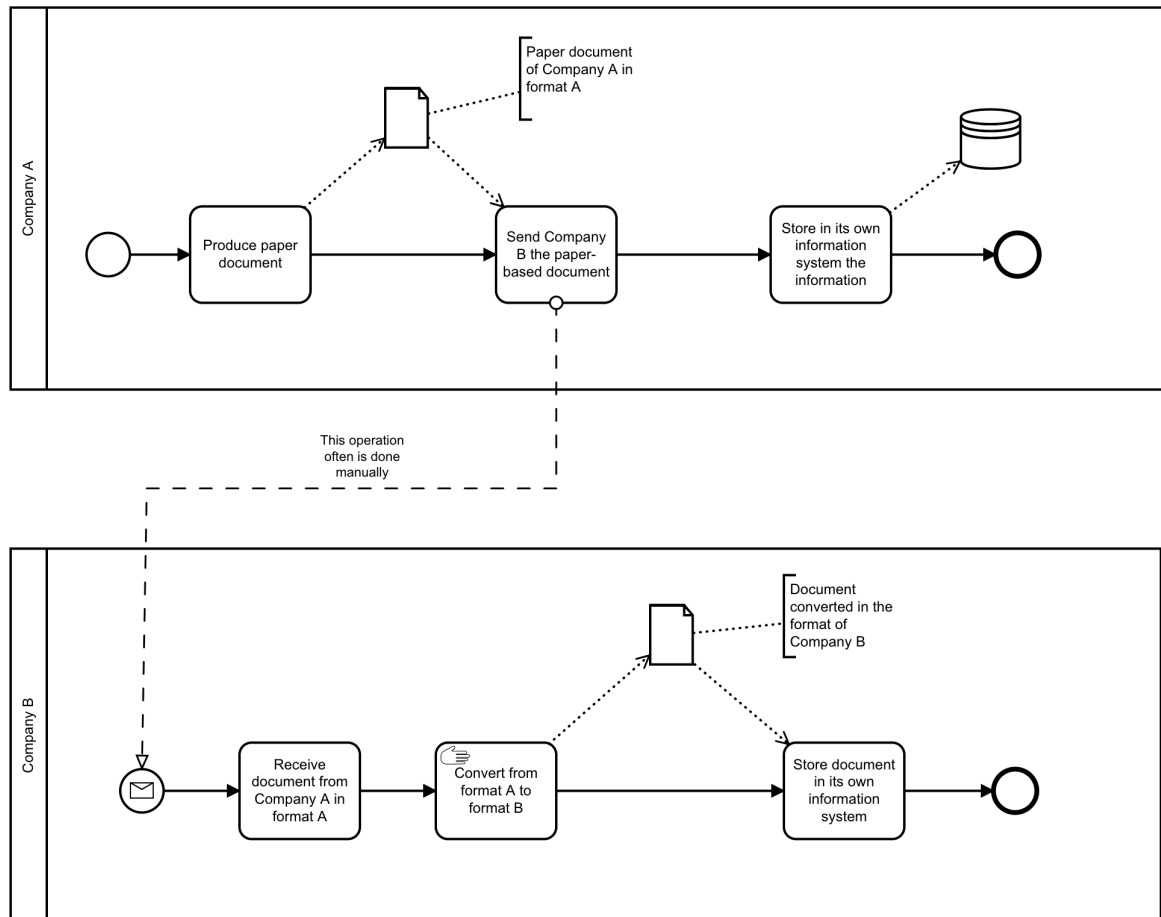


Figure 2 – A visual representation of the situation highlighted by K. Korpela et al.

As it is visible in the diagram, data exchange between different actors on the same supply chain can be very inefficient and time-consuming, due to manual operations of sending and converting information exchanged. In addition, paper-based document exchanged can be incompatible from the seller's system to the buyer's one. Digital supply chain has the aim of shifting from manual operations to digitalized ones, both inter and intra firms.

The main goal of authors was to provide a solution for integration in digitalization of supply chain in order to reduce the loss of time, the costs and to avoid trusted third parties. The major interest was to implement a common solution (with common standards and technologies) in order to better integrate business processes into a large supply chain network. The paper inspects the blockchain technology in order to check if it can be the solution for integrating different systems, hopefully increasing security levels, and supporting digital supply chain transformation.

Authors identified four main aspects that should be transformed in order to achieve a digital supply chain transformation that are blow reported.

- **Business model development.** New business models and new strategies must be adopted by companies in order to maximize innovation and digital supply chain integration.
- **Information models.** There is the need of new information models to collect, store and share data within the supply chain.
- **Business process standards for supply chain connectivity.** Companies must acquire new competences and develop new solutions with the goal to be digitally connected to business process transactions.
- **Operators for data transfers.** Intermediaries for integrating channel are required in order to integrate information from actors to systems.

Conclusions

This article identifies in the blockchain a possible solution in helping the shift from manual operations to digitalized ones, not without any limits.

Blockchain can enable the development of a common platform to be shared with all the network participants but does not provide itself the standards for electronic supply chain documents. International documents standards are required and then the blockchain can work to its full potential in terms of security, data immutability, verifiability and all the others blockchain features. In addition, adopting a blockchain-based solution often means the innovation of actors' information systems capabilities.

Finally, blockchain can enable the automation of operations that today are done manually.

2.6. Collaborative Business Process

Introduction

In this section a blockchain-based proposed solution for business processes execution is analyzed. The main goal of Weber et al. in [20] was to provide a running platform for the monitoring and the execution of collaborative business processes. In particular, this solution is adaptable to every situation where there is a cooperative business process and a lack of trust between all the participants involved in such process. As an example to better illustrate the context, a supply chain use case has been provided by the authors.

Proposed solution

In the following, the solution proposed by authors is described. The main goal of the developed solution is to provide a simpler management system of a collaborative process in either of two following ways.

1. **Choreography monitor.** This setting allows blockchain to work as an immutable data storage for the process status. The process is stored among all the involved participant in such process by observing the messages exchanged.
2. **Active mediator.** In this setting the system is able to coordinates the participants in the collaborative business process execution. Here, smart contracts are used to guide the process and contain the logic for data transformation and calculations.

The above described settings are supported by four main components shown below.

- **Translator.** This component is the main one for the first phase of the business process execution. It takes charge of translate the process from a formal process specification, such as a Business Process Model and Notation schema, to a scripting language, such as Solidity, in order to create a factory smart contract. The factory smart contract produced contains all the information required in order to instantiate the process. This component runs in the design phase.
- **C-Monitor.** The Choreography monitor (C-Monitor for short) is the smart contracts that monitors the business process execution. In this phase, variables are used to store the status of the whole process execution. The monitor is also used by the involved actors to communicate one to each other, since direct communication between them is not possible. The Choreography monitor is used when setting 1 (previously described is selected). The C-Monitor smart contract is generated from the factory smart contract produced by the translator in the design phase.
- **Active mediator.** This component is used when setting number 2 (previously described) is chosen. The aim of the mediator is to implement the collaborative business process. On the opposite side of the C-Monitor, the mediator always plays an active role. It is in charge of collecting and sending messages that respect the business logic defined in the process model of the design phase. The mediator smart contract is generated from the factory contract as well.
- **Triggers.** Interfaces and triggers are the components with the goal of connecting the blockchain with the external world and vice versa, since smart contracts cannot

directly interact with the external world with respect to the blockchain. Triggers run on full node, keeping track of the running business process execution status, and they manage sensible and confidential information. They play the role of a gateway, calling external APIs when needed and receiving APIs' calls from external actors. In addition, when a new process is created, each involved participant register itself on that process with a certain role and public key. In this way, all the involved actors can easily verify the correctness of the execution and can know which role is played by whom.

With the help of the above reported components, authors reached a solution where a business process can be executed, considering all the involved participants, in an untrusted environment. Also, the process can be advanced only if messages are right, and the system keeps an immutable ledger of all the transactions, both committed and rollbacked.

In the proposed solution, since information stored on the blockchain is accessible by everyone, two ways of storing data are evaluated. First, on the blockchain is stored the execution status of the business process. This information is not encrypted because the C-Monitor and the active mediator must have access to that information. Instead, in order to preserve users' privacy, it is possible to store messages' payloads in an encrypted form.

Authors also provided an off-chain solution to be used when the amount of data to be stored is large due to blockchain limits in storing large amounts of data [21] and because of the related costs.

Conclusions

The proposed solution is interesting in many aspects. First of all, authors provided a solution that is adaptable to many use cases thanks to the use of the translator, that from a BPMN diagram can extract a factory smart contract. In addition, two ways of executing this solution are available: an active one, that let users to interact with the system, and a passive one, that enables a secure high-level view on the execution of the whole process. The main goal of authors was to provide a system where many different use cases can be executed. The blockchain plays the role of process execution information's storage.

2.7. Data Reliability and Data Provenance

In the Big Data era, data collected by the companies are in the order of petabytes and they mainly use those data to personalize services, increase customer's satisfaction predicting future trends and to optimize corporate decisions. Current data storage systems are mainly based on a centralized storage solution where if the central server is compromised, all the data stored are compromised too. In this section the most interesting and recent applications in terms of data provenance and personal data storage solutions are deeply analyzed in order to better understand if a blockchain-based approach can be a potential solution for such problem, and which issues instead still continue to exist.

2.7.1. DataProv

Introduction

Authors of [22] focused their attention in securing the provenance related to data used in the research area. The main goal of DataProv is to securely store data related to scientific research, that is be sure of the provenance of such data but keeping them private and accessible to authorized users only.

In order to avoid frauds related to security e and truth of data, a verifiable chronological history of the provenance of each data must be maintained. To achieve a better result, also who firsts created those data, who later used and modified them, and the chronological ownership of such data can be securely stored in DataProv system.

The DataProv proposed solution provide a blockchain storage layer where data are fully auditable, and any data change can be securely tracked. The inherent characteristic of being immutable makes data temper-proof. In DataProv, each node of the blockchain network keeps an updated track of the history of data and their provenance.

Proposed solution

DataProv system is built on top of an Ethereum blockchain network and authors used Meteor framework for developing the user interface for client's modules. A voting protocol is implemented in order to avoid malicious changes to data stored in the system. Data provenance is represented in the proposed system by following the Open Provenance Model (OPM) [23]. Each operation in the system is represented as a triple containing the following information:

- **Artifact.** Artifacts are represented by files, documents, etc. both before and after a change
- **Agent.** Agents represent the initiator of the change
- **Process.** This is the set of actions that moved a document from the older version to the new one.

Authors of DataProv also considered two possible types of cyber-attacks to their system. They identified in both internal and external actors some possible threats.

Regarding external attackers, since they are supposed to not have the key to decrypt data, the solution proposed is to store encrypted files. It is the owner of each data that uses its own key to encrypt his data. In addition, access to the system and to stored data is possible

only to registered and authenticated users. Moreover, the owner of research data can grant or deny access to other users.

For what concerns instead internal attackers, the hypothesis that must be satisfied is that at least half of the network is loyal. An internal attacker has the possibility to change something in a document but since the change must be approved by the majority of the network, if that percentage of the nodes is loyal, the attack fails.

Another aspect to be considered of the proposed solution is the mechanism of versioning that allows modifications to data on the blockchain. In particular, when a change occurs, a new version of that document is created. Automatically the system considers the latest version as the current one and only this one can be modified. This mechanism follows the idea of append-only data structure of the blockchain.

The system is composed of an underlying layer that is the blockchain, while the client side is developed using Meteor [24] and MongoDB [25]. MongoDB is used only to store (client-side) documents that user is allowed to modify and the latest version of each document. Finally, a Geth [26] node (running at the client side) is used to enable communications between the client application and the smart contract.

The proposed system is mainly composed of two big components: the on-chain components and the off-chain ones.

- **On-chain.** Of course, the Ethereum blockchain platform is part of the on-chain components set. Smart contracts too are part of the on-chain components. They are stored directly on the blockchain and they are executed when called. Authors developed two different smart contracts.
 - a) **Document Tracker** is the first smart contract with the goal to keep track of all the modifications applied to stored files. It is also intended to implement policies for access control and store user access information. Furthermore, the digital signature of each initiator of a change is stored in the system as a change event. Remember that DataProv does not store any sensible information as plaintext on the blockchain since any information stored on the blockchain (including the smart contracts' code) is accessible from anyone. In addition, storing information on the blockchain has costs and limits in terms of quantity. Finally, Document Tracker smart contract also implements access controls in order to avoid malicious calls to smart contract's function.
 - b) **Vote** contract instead is the smart contract developed to implements the voting protocol. When an initiator submits the change, along with its digital signature, the vote contract verifies the received change and generate a log event that initialize the vote phase. The vote contract waits for the preset time and after the elapsed time, if the votes received are positive the change request is approved, otherwise it is rejected. In case of within the elapsed time the minimum number of required votes has not been reached, the voting phase starts again.
- **Off-chain.** Off-chain basically can be found the client modules. The **Client module** simply acts as a user interface for interacting with the smart contracts and for performing some basic operations like for example, adding a new document, tracking changes in a document, etc.

Another off-chain module is the **Event Watcher** that basically is an event listener that listen for new change event log. If the current client is tracking that document, the module decrypts the change event, verifies the digital signature of such change and then calls the **Verification module**, that is off-chain as well. The verification

script has the role to validate the change emitted by the initiator considering current and previous hashes taken in input and the latest version of that file. If such script does not find invalid changes, it can vote in favor of the user. At the end, in order to avoid further manipulation of the document, the access to the document is set only for the owner of the original document. Last off-chain module is a **Timer** that basically keep track of the current elapsed time in the voting phase. When the voting interval ends, the module triggers the termination of the voting phase for each client involved.

Conclusions

Authors of DataProv provided a system to securely track historical changes of certain types of data. They developed both software modules that runs on the blockchain and other modules that runs on client-side. Especially smart contracts must be immutable, so they must run on the blockchain, while personal data are stored locally; also user interfaces are software applications web based running client-side.

This application is interesting considering the blockchain-like behavior. Data changes are recorded as a new record in the system, following the idea of append-only data structure, as blockchain is.

2.7.2. Decentralizing privacy

Continuing within the context of Big data era, Zyskind et al. [27] proposed a blockchain-based solution for the collection, the management and utilization of private personal data, mainly for mobile applications. Authors affirm that users have a very poor knowledge about their private data managed by companies (e.g. when they are collected, where they are stored and how they are used) in order to provide personalized services. Zyskind et al. mainly focused on mobile platforms, where service companies develop applications in order to collect users' personal information. In order to provide such secure platform, they inspected three main privacy issues, below summarized.

- **Ownership.** The privacy related data ownership must belong to users. The framework recognizes ownership of such data to users while companies are guests with delegated permissions.
- **Transparency and Auditability.** Each user must have fully transparency in his/her data (e.g. when they are collected and how they are used) and such data must be easily verifiable.
- **Access Control.** Access control policies and rules should be stored in the blockchain, making them securely stored, and only users must be able to change them, each user his own ones.

Proposed solution

The proposed system is composed of three main entities: (i) **users' mobile phones**, where users download applications that collect data, (ii) **services**, that are the companies that releases applications in order to collect as much personal data as possible, (iii) **nodes** which are instead entities in charge of maintaining the blockchain correctly running and also an off-chain distributed key-value storage solution, later on better analyzed. Note that

while users remain as anonymous as possible, services' identity must be public and auditable by all users; for this reason, a digital identity of such service companies is stored directly on the blockchain.

The blockchain of the system proposed by Zyskind et al. accepts two different types of transactions.

- **Access transaction.** This type of transaction is used for access control management. Using this transaction users can also grant or deny permissions to services, modify the set of permissions to his / her personal data and modify the set of companies that now can manage that data.
- **Data transaction.** This type of transaction is used to store data in the system and to retrieve them. This operation has been developed in order to be easily integrated with mobile SDKs (Software Development Kit).

The proposed system, as previously mentioned, consists also in an off-chain storage solution. It is a key-value storage implementation of Kademlia [28], that is a distributed hash table (known also as DHT) with a LevelDB [29] persistence layer. The distributed hash table, since it is a distributed storage solution, is maintained by a set of nodes, hopefully not the same of the blockchain network, that are in charge also of approving or rejecting read and write operations. Authors designed the system in such a way that alternative off-chain storage solutions can be exploited.

Sample scenario

To better understand the potential of the proposed solution, a brief sample scenario is described below.

Consider a user that signs up in the system. When this event occurs, a new shared identity (user and service) is entered in the system and it is sent to the blockchain, along with the respective permissions, using an access transaction. Data collected by the application on the user's phone is encrypted using the shared key and submitted to the blockchain in data transactions. Such information is then stored in the key-value off-chain storage, while in the blockchain is stored a hash (that acts as a pointer to the original data) of such information. At this point, both services and users can retrieve data (using a data transaction) simply providing the has (that is the key of the storage) of the desired information.

Conclusions

The platform provided by authors has many common features with the previous analyzed solutions analyzed but also some new aspects. This developed system, enable data to be stored in an off-chain database, while the hash of the information is stored in the blockchain, that makes it immutable. Since the hash is stored in the blockchain, it is tamper-proof, and it will be very easy to understand if data have been modified outside the blockchain. In addition, this solution makes data private and accessible only to the respective user and the allowed service companies. Finally, it's interesting to notice that authors applied the blockchain technology in a revisited way: in fact, it acts as access controller and moderator for the entire system while sensible data are stored off-chain.

2.7.3. ProvChain: a blockchain based architecture for data provenance

Introduction

X. Liang et al. in [30] proposed a solution for data provenance in a cloud storage application. The solution consists in a three-layer architecture built on top of a blockchain network. Cloud data provenance is metadata that records all the changes on a certain cloud data object. Both create, update and delete operations are recorded as a data change. Adopting a blockchain-based solution for data provenance means to ensure tamper-proof and auditable records for cloud data object. Authors applied the proposed architecture to a cloud storage scenario and assumed the cloud file as data unit. They provide an architecture for collecting and verifying cloud data provenance, in parallel with enhancing users' privacy and data availability, by recording every cloud data change in the underlying blockchain. In fact, in the proposed architecture, users' identities are kept hashed in order to protect their privacy from other actors involved in the blockchain network. The cloud service provider to which they proposed this solution is ownCloud [31] and the provenance service can be activated on-demand for users who request it and pay for having it. OwnCloud is an open source application that provides both desktop client applications for managing personal files and data and both web-based cloud storage solutions. OwnCloud is flexible and allows developers to create many different applications on top of it.

ProvChain architecture

To achieve their purposes, authors follow four main targets, below reported:

- **Real-time cloud data provenance.** In order to correctly update data, users are continuously checked in real-time, looking for any change applied in cloud data file, in order to collect provenance data.
- **Tamper-proof ecosystem.** In order to provide a tamper-proof system, once data provenance records are collected, they are published to the blockchain network and each node involved in the network holds a copy of such data.
- **User privacy.** In order to protect users from other nodes of the network, ProvChain system associate provenance data to a hashed user ID. In this way, only the cloud service provider, that owns a list of all its users, is able to match each single provenance record with its real user ID.
- **Data validation.** Adopting a blockchain based solution means that once data are submitted to the blockchain, they are globally published, and a number of nodes provide confirmation for every block.

In order to properly achieve the above reported objectives authors continuously monitor user operations by means of listeners and hooks directly on the ownCloud source code. In addition, in order to protect user identity, the proposed solution associates a hash of the user ID to transactions in the blockchain. In this way, only the cloud service provider can associate the real user to that provenance data.

In the following a summary of the main components of the ProvChain traceability system is reported:

- **Cloud user.** The cloud user is the actor that owns its data, that has some relationships with other users' data and opted for data provenance service.
- **Cloud service provider.** The Cloud Service Provider (CSP) offers a cloud storage solution and it oversees the user's registration. The CSPs can benefit from the proposed solution both in real-time users monitoring (from which they can learn which operations are most performed by their users) and both in security aspects (they can easily detect intrusion and tampered provenance data).
- **Provenance database.** The provenance database is where all the provenance data on the blockchain network are stored.
- **Provenance auditor.** The provenance auditor is who can have access to provenance data and validate them. However, PAs don't have access to owners of that data since they are hashed.
- **Blockchain network.** The blockchain network consists of nodes that globally participate to the system. Data are stored in form of blocks and verified by blockchain nodes.

ProvChain implementation

As mentioned in the first part of this section, X. Liang et al. provided a three-layer architecture for the proposed solution. The architecture is composed as follows:

- **Data storage layer.** Recall that ProvChain has been created in order to support cloud storage applications. In the proposed solution one cloud service provider is taken into account but such solution can scale to many different CSPs.
- **Blockchain network layer.** Blockchain is used to record each new provenance data entry. Since, in the proposed solution, the data unit has been identified in one file, in the blockchain is recorded each file operation with user and file name. File operations include create, modify, share or delete.
- **Provenance database layer.** This database is built locally in order to query and record the file operations.

As mentioned previously, for collecting provenance data, authors provide a set of listeners that listen for file changes. After the listener is activated, the record is generated and then submitted to the blockchain network. At the end of this phase, the provenance database is updated with the record newly generated. For storing the previously collected data, X. Liang et al. used Tierion API [32] in order to publish data records to the blockchain network, following the Chainpoint standard [33], which propose a scalable protocol for submitting data records on the blockchain. It is important to recall and underline that in the proposed solution, only the cloud service provider can have access to user related to a certain record on the blockchain since user IDs are hashed before being published on

the blockchain network. After the collection and the storage of provenance data, the validation phase can begin. In order to validate data recorded on the blockchain, the Pas (provenance auditors) previously described can request the blockchain receipt from Tierion API, which contains information of the blockchain transaction. After the validation of the blockchain receipt just requested, the provenance auditor can be able to update the data records in the provenance database.

Conclusions

X. Liang et al. provided a blockchain-based solution for cloud services that currently are widely adopted by the most important companies around the world. ProvChain aims to secure and make auditable provenance data while enhancing users' privacy and data availability. In the proposed system there is not the need of trust between participants of the blockchain network, nor of a trusted authority. However, no digital identities are taken into account as requirement for joining the network, only the registration to the cloud server provider is requested. Moreover, even if the cloud service provider, thanks to the blockchain adoption, is not trusted for keeping provenance data records, it oversees the users' registration and it has the capability of matching files with proper users.

2.8. Conclusions

In this chapter of the thesis the state of the art of the blockchain technology has been analyzed. The attention has been focused mainly on applications interesting from the goal of this thesis point of view. In supply chain management systems, from the analysis performed in this chapter, it was found that many features in common are present. In particular, the blockchain network proposed were always in the context of permissioned blockchains where participants are known but not trusted at all and must hold a digital identity in order to correctly join the network. Moreover, the systems built on top of such blockchain network followed a layered approach where each layer can interact only with neighboring layers. Finally, in product's traceability context, almost all the analyzed solutions provided an IoT-based system that can automate the entire process of recording product's and environment's data.

Besides, data reliability and data provenance blockchain-based proposed solution were analyzed in order to study the feasibility of adopting the blockchain technology in securing data reliability. Also here there are features in common with the different solutions proposed. For example, adopting event listeners that listen for changes in data stored is a common adopted solution in order to automate the update of the ledger. Moreover, both permissioned and permissionless blockchain network were adopted, depending on the purpose of the application they were working on.

3. BLOCKCHAIN TECHNOLOGY

This chapter has the aim of deeply study the blockchain technology in order to better understand its full potential and the feasibility of adopting such kind of solution in enterprises. In particular, a detailed explanation of building blocks of the blockchain components is below reported, considering also the side aspects of such ecosystem.

3.1. Introduction to blockchain

Blockchain is a technology born with the goal to store information, with a certain value, in a secure way. Blockchain was first made public in 2008 by Satoshi Nakamoto (pseudonym of an author or a group of people still unknown today) in his article titled “Bitcoin: A Peer-to-Peer Electronic Cash System” [9]. Blockchain is a subpart of what is called DLT, that stands for “Distributed Ledger Technologies”, in which data are no longer stored in a single central database but the ledger containing data is distributed around a certain number of nodes in order to build a peer-to-peer network. Blockchain, as the name suggest, it’s logically a distributed ledger in which each single building block is called block and those blocks are linked one to each other in a secure way in order to make it very difficult for an attacker to tamper with blocks. In this way, modify the blockchain is very expensive, but check if it is correct or not should be very easy for all the peer who joins the network. Moreover, in blockchain data structure are permitted append-only modifications. Blockchain is used as an alternative to centralized database in order to avoid SPOF (Single-Point-Of-Failure) and a central authority that manages data because there is no a third trusted party that have access to data but all the nodes belonging to the blockchain network has a full copy of the ledger. Another important property is that the blockchain is an immutable data structure and each modification is stored as a new immutable record.

The first real and interesting application of the blockchain was related to cryptocurrencies. In particular the blockchain was applied in 2008 to an emerging digital currency, called Bitcoin, with the aim of acting as a ledger in which are stored all payments transactions.

3.2. Centralized, decentralized and distributed

These are the three main architectures implemented in building a database. Centralized database is the classical one in which there is a single central database and all data are there, often managed by a third trusted part. Decentralized architectures provide a network in which nodes are no longer dependent in single central authority, but control functions are spread among many nodes. In the end, a distributed database is a particular architecture in which data are replicated and distributed in each node joining the database network even if the final end users continue to see one single and coherent system. Sometimes distributed systems are confused with parallel computing: in parallel computing each node perform computation simultaneously with all other nodes with the aim to achieve the final result.

3.2.1. Centralized databases

A centralized database, as the name suggests, is a database completely located in a single location. This means that there is a central authority that manages data and can perform CRUD operations (Create, Read, Update, Modify) on data. This kind of organization of data often comes with a central server or database system that contains data. Users who wants to access data can do it through a computer network which is able to give him connectivity to the central database.

Centralized databases have lots of advantages in simplicity issue to manage and keep data aligned; whoever wants to change something or simply read data he has simply to access one single database with a fixed architecture. This, obviously, can be read also as a critical aspect with respect to cybersecurity, in fact if an attacker success to gain access to the database, he can modify the data and all the end-users will see tampered data.

Anyway, let's see some advantages of a centralized database: first of all, data integrity is guarantee at the maximum possible level and data redundancy is (possibly) avoided (without considering data backups). Data are considered consistent and concurrency issues are avoided by the DBMS. In addition, the level of security in a centralized architecture is considered high in certain situation because there is one single location to protect; moreover, if it is possible to give different roles to different clients, the security level grows even more. Another important advantage is on the third trusted part side: it's easier to manage data and perform database administration if there is one single server on which data are stored and any modification is immediately visible to all the end-users. Finally, considering power supply and maintenance, a centralized architecture is more cost effective than the other architectures.

On the other side a centralized database has also some negative aspects. First of all, there is one single point of failure (called also SPOF) and data availability is strictly dependent to network connectivity. If the network goes down, all the end users are cut off. In addition, the lower the network connection is, the longer the waiting time is on client side to access data. Coming from this, also bottleneck issues must be considered during high-traffic operations. Another important aspect that must be considered is race conditions and concurrency issues that can be managed by the database administration team. In general, the more advanced tools available for database management (DBMS) solve these problems easier and in an automatic way but of course at the expense of performance. Considering unexpected network shutdowns, backup copy of all data is needed and must be continuously updated.

3.2.2. Distributed databases

In a distributed database data and computation is spread across many nodes in the network. In this way, data replication is maximized, and this is the reason why is considered very secure and makes very difficult to tamper with data. Two main processes in distributed ledger in order to keep data up-to-date and current are replication and duplication.

In replication, a specialized software is used to looks for changes in the ledger and when a change has been identified, the process makes all the instances look the same, that it means that all the nodes of the network must have the same data. This replication process can be very time consuming and expensive from a hardware resource point of view, depending on the network and database size.

Duplication, instead, has the goal of identifying one database as "master" and then duplicates this database in many copies. This is done to be sure that all the distributed copy contains same data. Important here is that an end-user who want to modify data can do it only on the master branch and then the process distribute the modification.

Both replication and duplication are used to keep data aligned between distributed locations.

3.2.3. Decentralized databases architecture

In a decentralized database there is no longer a single central storage unit, but data are spread among many servers that work together in order to provide desired information to

clients. There is no more a single central authority that can manage data on servers. Data are duplicated and distributed in each node of the decentralized network. In this kind of database organization, all nodes (called also peer) are on the same level and each node must worry about keeping data aligned and consistent with all the other peers.

In blockchain technology both terms (decentralized and distributed) are used, but the two concepts are different. The principal difference is that in a distributed system there still exists a single central authority that manage the entire system while in a decentralized solution no central authority exists.

There are two different ways in order to achieve decentralization:

- **Disintermediation** can be easily explained using a money transfer between two different bank accounts as an example. If someone wants to send an amount of money to someone else, he or she needs to go to the bank, that retaining a percentage fee, will send that quantity to the friend for you. In this example, a bank is needed, that here plays the role of the central trusted authority, for reaching customer's goal. With a blockchain solution it is possible to send money directly to the blockchain address of someone else, without the need of a third trusted party. This is the main idea of disintermediation, that is removing central authorities from the system.
- **Contest-driven decentralization** is a paradigm in which no full decentralization is achieved but can ensure that an intermediary or service provider is not monopolizing the service. In particular, in the blockchain context, with this kind of decentralization smart contracts are allowed to take decisions on which external data provider will be used, selecting it between a large number of providers, considering their quality of service, reviews, previous score and their reputation. So, providers are chosen considering the above criteria in order to achieve a partial decentralization.

In decentralized solutions is very important to understand that not only the ledger must be decentralized but also the entire environment should be decentralized in order to achieve complete decentralization. The blockchain, in fact, is a distributed ledger that is built on top of conventional systems like storage, communication mechanism and computation which follow a traditionally centralized paradigm and there is the need to decentralize also these aspects to achieve a complete decentralized system. For example, considering internet, that is the layer on which blockchain technologies based their communication, it was conceived to be a decentralized network. Anyway, nowadays the access to the internet is based on ISPs (Internet Service Providers) who are in charge of giving connectivity to end users and, in this way, they act as central hub for customers. If, for any reason, the ISP goes down, all customers of that ISP will be without internet connection according to this model. This example is useful to understand why today internet is moving toward centralization.

Thinking about computing layer instead, blockchain platforms moved toward computation decentralization with **smart contracts**, that are pieces of business logic that can run over the network in a decentralized way. A smart contract is a decentralized program that in general does not need a blockchain to be executed, even if today blockchains are becoming the main platform on which they run. Smart contracts contain some business logic and also a small amount of data. When specific conditions are met, the smart contract is triggered and execute its business logic.

More information about smart contract will be provided later in the thesis.

3.3. Relational and non-relational databases

3.3.1. Relational database

The first idea of relational model was developed by E. F. Codd in 1970.

A relational database is a collection of data with fixed relation between them. These data are organized in tables in which each row, that represent a single data, is called tuple and the columns, that might be of different types, are attributes of data.

Each row can be uniquely identified by a key, that is a special field of the table.

In relational databases, data must respect a certain schema to be inserted into the database. For instance, all the data field must respect the respective data type and, if they are not nullable, they must exist.

A very important component for the maintenance of relational databases is the Database Management System (called also DBMS). The DBMS is a software system designed to guarantee creation, modification and efficiently query the database. Typically, the DBMS is provided on a dedicated hardware machine (but is also used a simple computer).

3.3.2. SQL

To perform CRUD operations (Create, Read, Update, Delete) on data in a relational database SQL query language is used. SQL is not a standardized programming language, but it provides a basic language for each relational database on which database providers can implement their own special functions.

3.3.3. Non-relational databases

Non-relational databases, called also NoSQL, is a mechanism to access data that are not strictly stored in relational tables format.

NoSQL databases, that stands for “Not only SQL”, are designed for receive specific data with flexible schema and are used in modern real-time and big-data application. Different types of data are used like document, graph and key-value pair. Non-relational database does not follow the typical relational model and have grown in popularity because they exceed limits of relational databases, in particular in the Big Data era where applications have to do with an enormous amount of data.

One of the most important aspect of non-relational databases is the capability to scale in a horizontal way. There is, in fact, the possibility to use clusters of computers to perform operations in a non-relational database with the raising of the parallelism at the maximum level.

3.4. Distributed systems

To better understand the blockchain technology is essential to understand what a distributed system is and how it works. A distributed system is a computer system and a computer architecture in which there are at least two different nodes work in a coordinated way in order to achieve a common goal. In its best implementation, a distributed system is designed in such a way that end users logically see a single platform, while “behind” there are multiple nodes working together in order to provide the expected behavior. For example, the search engine provided by Google is based on a very large distributed system, while for end users it appears as a single platform.

In distributed systems an important role is played by each single node. Differently from the centralized approach, a distributed system is composed of many single nodes, working together in order to provide the requested services. A node, in its simplest implementation, is a process running on a device with some computational power, capable of sending and receiving messages to and from each other node of the network and performs some operations, depending on the context in which that node is deployed. The term process indicates a general entity capable of communicating with another process by means of proper messages and executing a distributed algorithm.

3.5. CAP theorem

In 1998 Eric Brewer introduced the CAP theory and then Seth Gilbert and Nancy Lynch proved it in 2002.

The CAP theorem states that a distributed system is able to satisfy at most two guarantees between Consistency, availability and Partition tolerance at the same time, but not all three.

Consistency is the property that guarantee that each node of the distributed system has the same data of each other.

Availability is the capability of the system to retrieve data and provide them to end users. Each node of the distributed system must be up and running without any failures.

Partition tolerance is the capability of the network to continue working in the right way even if some nodes are unable to communicate due to network bandwidth issues or malicious attackers or due to node's energy problems.

Blockchain seems to violate this theorem but this is not really true: in blockchain frameworks Availability and Partition tolerance are reached simultaneously while Consistency is sacrificed and is achieved over time. This is called eventual consistency, where consistency of data between nodes is reached as a result of validation from multiple nodes over time.

In a Blockchain context, CAP properties can be re-defined as following:

C – Consistency means that no forks (from the main Chain) are allowed

A – Availability means that each node of the blockchain network has a full copy of the ledger

P - Partition Tolerance means that the network must work properly even if some blocks or transactions are delayed or dropped by the physical network between peers.

3.6. Hash functions

Any function that can be used to map an arbitrary input to an output of a fixed length is called hash function. The value returned by a function like this is called hash or digest. Important is to underline that the input could be arbitrary: it could be one single character, a string of twenty characters, the null string or the entire holy bible. Anyway, the produced output will be always a value of a fixed length, depending on the algorithm and the function chosen.

These types of functions are very useful in cryptography to guarantee data integrity. In fact, a cryptographic hash function is a particular function that provide a *fingerprint* of the input. If someone try to modify data, that hash will be no more valid because it will change. Another important property is that the evaluation of the hash must be very efficient and not much time consuming. This means that the evaluation of the hash must be independently on the input message size. Another important aspect in hash functions is the **avalanche effect**, that it means

that all changes in the input text, even minimal like a single character, will produce a completely different output.

For example, think about the scenario where Alice and Bob want to exchange a message but they don't trust the network: Alice sends a message to Bob and she wants to be sure that the message delivered to Bob is her message and not a modification of it. So, Alice sends her message as plaintext (because she doesn't want confidentiality but only integrity) and also the digest of the message that she evaluated used a hash function.

When Bob receives the message, he can easily evaluate the hash of the message received (obviously he must use the same algorithm used by Alice) and if the two hashes are the same it means that the message is the original one, otherwise there is the possibility that someone tampered with it (or simply the network loose some packets).

Another important application of hash function is the implementation of data structures used in computer software in which a rapid look up is guarantee. It is very easy, in this type of hash table, to evaluate if a certain value already exists.

3.6.1. Aliasing

One important aspect of the hash functions is that should be impossible to generate collisions, that it means that does not exist until now (using for example SHA256) two different inputs that generate the same output. The parameter used to evaluate the probability of aliasing of an algorithm is the number of nits of the output. The higher the value of this parameter is, the lower is the probability of having aliasing.

$$Pa = \frac{1}{2^{Nbit}}$$

Considering Nbit equals to 256, if someone produce 2^{256} digests, then there will be aliasing

3.6.2. Birthday paradox

The birthday paradox is a particular issue where, according to probability theory, chosen a set of N people, there will be some pair of them with the same birthday. Considering, for example, 23 people, according to the theory, there will be the 50% of probability of having 2 of them with the same birthday and this value reaches 99.9% considering just 70 people.

The assumption that makes these conclusions truthful is that each day of the year (without considering February 29) is equally probable for a birthday.

From the birthday paradox derives that an N bits long hash algorithm become insecure when $2^{\frac{N}{2}}$ digests are generated because the probability of aliasing Pa becomes approximately 50%.

3.6.3. One-way function

A function is defined as “one-way function” if is very easy and not time consuming to compute on every input, but hard (ideally impossible) and time consuming to come back to the input given the output. So, from the same input it is possible to reach always the same digest but from the digest should not be possible to obtain the exact value of the input, that is the original text. Using some formulas, given x is “easy” to calculate h(x)

(where $h()$ is the hash function), but on the other hand, knowing the value of $h(x)$ it is quite difficult to go back to the value of x . So, this is an important feature of hash functions to securely certify data and make it very hard (practically impossible) to tamper with them. In fact, it is possible to evaluate the hash of some data and then if someone try to tamper with them the hash will change.

In this context, easy and hard must be interpreted in terms of computational complexity, specifically referred to the theory of polynomial time problems.

Anyway, the existence of a “real” one-way function is an open conjecture: for example, today, SHA-256 is considered a secure one-way hash function but is very likely, that due to the increasing computational power of computers, it will become deprecated in some years.

3.7. Cryptography

Cryptography is the science intended to protect sensible information in order to make very difficult for attackers to have access to that data and only the authorized people can have access to the content. Cryptography was in principle designed to provide **confidentiality** properties. Together with confidentiality, cryptography provides also some other security aspects such as **authentication**, **integrity** and **non-repudiation**.

- Confidentiality: is the property that guarantees that the content is available only for the right receiver.
- Integrity: this property guarantees that the content has not been modified by non-authorized entities.
- Authentication: is the property that provide assurance about the identity of a person (in general, an entity) and/or about the validity of a message.
- Non-repudiation: is the property that provides a final proof that something occurred. This property is very important when, for example, an entity try to deny a performed action, such as the placement of an order in an e-commerce system. Very important is to underline that a non-repudiation property has a legal value also in case of disputes in front of the law. Digital signature is an example of non-repudiation proof.

The text to be protected is also called plaintext and the message protected is called ciphertext. The transformation from plaintext to ciphertext is called encryption and vice versa is called decryption. Both encryption and decryption have in input a special parameter that is the key. To decrypt a message the receiver needs not only the key but is asked to use the same algorithm used in encryption phase.

3.7.1. Kerckhoffs principle

If the keys are kept private, are managed only by trusted systems and have a fixed length, then, is not important to keep secret the algorithms, indeed it's better to make them public in order to be studied and analyzed by as many people as possible.

3.7.2. Symmetric cryptography

The main idea of symmetric cryptography is that the key used for encryption is used also for decryption. Traditional cryptography is based on the mechanism described below.

The same key is used both for encryption and decryption. This is very common in everyday application but has many problems. While it is very simple to be managed, for a symmetric cryptography the critical point is the distribution of the keys. One possible way to exchange the keys is to meet in person or to use another communication's channel. Anyway, this is not a best practice.

Another important aspect, thinking about legal aspects, is that with asymmetric cryptography there is not the possibility of non-repudiation legal value.

3.7.3. Asymmetric cryptography

The main idea of asymmetric cryptography is that the key that is used to encrypt and the one used for decrypt are different. This mechanism is also known as public key cryptography. Both public and private keys are used both for the encryption and the decryption phase, according to which security aspects are requested.

Private key, as the name suggest, is the key that the owner must be keep private, nobody can have access to that key. The length of the private key may vary depending on the algorithm chosen.

Public key, instead, is published by the private key owner and everyone can have access to this key.

Asymmetric cryptography solves also the problem of key agreement. Each participant has a pair of keys: one public, intended to be distributed as much as possible, and the other private, that it means that must be kept in a secure place. For this reason, often is used a PSE (Personal Security Environment), both hardware and software.

Is important to underline that in asymmetric cryptography if one key is used for encryption, the only way to decrypt the message is to use the other key of the same pair.

3.7.4. Confidentiality

Considering the following example: if S wants to send a message to R, first S encrypt the message using the public key of R. In this way, since only R has his private key, he is the only one that can decrypt the message and have access to the content.

In this scenario there is no more the problem of sharing the keys, but them are distributed using a particular format (that in most cases is done using a X.509 certificate, in which a certification authority grants the pairing between the two keys).

3.7.5. Key exchange using asymmetric cryptography

After the birth of asymmetric cryptography is easy to think that the end of symmetric cryptography approached. Instead, asymmetric cryptography found an important application in the symmetric key exchange. As it is often difficult to meet in person one to each other for exchanging symmetric keys, asymmetric cryptography could be used. Is important to underline that asymmetric algorithms are slower that symmetric ones, so, for this reason, is preferable to avoid using asymmetric mechanism for encrypting a large amount of data. In this way, asymmetric cryptography can be useful to encrypt symmetric keys used for encrypting data, that are very small with respect to the entire message.

Imagine that A wants to share a secret message with B and, since the message is big, A wants to use symmetric cryptography for more efficiency. A has a symmetric key, called Ksym.

What is necessary, in addition, is that B has a X.509 certificate and B has A's public key. Now A can encrypt the secret message M using the symmetric cryptography, that is much more efficient with respect to the asymmetric one, using Ksym as input parameter ($c = \text{sym_enc}(\text{Ksym}, M)$).

After sending the ciphertext associated to the secret message, A encrypts the symmetric key using B's public key ($\text{Kencr} = \text{asym_enc}(\text{KpubB}, \text{Ksym})$) obtaining the symmetric key Ksym encrypted. After doing that, A sends the encrypted symmetric key to B.

Now B, first has to decrypt the symmetric key using its private key (associated to the public key that A used to encrypt the symmetric key). In fact, B can easily retrieve Ksym doing $\text{Ksym} = \text{asym_dec}(\text{KpriB}, \text{Kencr})$. Now, also B has the symmetric key Ksym and can simply decrypt the ciphertext obtaining the secret message $M = \text{sym_dec}(\text{Ksym}, c)$.

3.7.6. Digital signature

Another important operation allowed by asymmetric cryptography is the digital signature. In this context, if R wants to be sure that the real message comes from S, S has to sign the document using his private key (that he is the only one who knows it) and then R can verify the sign using R's public key.

In particular, S writes the message M, evaluates the digest of the message $d = h(M)$ (using for example $h = \text{SHA-256}$) and then encrypts the hash using his private key $c = \text{encr}(\text{KpriS}, d)$.

The ciphered digest is sent together with the message M and it represents the digital signature.

The receiver R can receive the message M, evaluate the hash $d1 = h(M)$ (using the same hash algorithm that the sender used), decrypt the ciphered digest received using the public key of the sender (that he retrieved from a X.509 certificate) $d2 = \text{decr}(\text{KpubS}, c)$.

If everything is ok, then $d1 == d2$ and this means that S is the right sender of the message M. If this constraint is not satisfied, then the sender is not the right one or someone tampered with the message or with the digest during the transfer on the network.

3.8. Blocks and transactions

The, so called, block, is the main actor in building a blockchain. As the name suggests, a blockchain is a sequence of "blocks". Each block contains the information that must be stored, securely certified and made immutable. In particular, in blocks are stored transactions that is the smallest unit of data allowed in the blockchain. Important is to notice that transactions are not a standard data structures but strictly depend on the application for which the blockchain has been designed. Transactions are not only related to financial world as many people believe, but they can contain a broad spectrum of data, depending on which application is built on top of the blockchain. In this scenario, transactions can be considered as a generic smallest unit of data can be submitted to the blockchain.

Moreover, blocks contain also some additional information used to build the blockchain.

3.9. Chain of blocks

Using the mechanism previously described it is possible to build a “chain of blocks”, that is not yet the blockchain.

To build this chain one more step is needed. The hash of the previous block is copied in the current block. Doing so, the evaluation of the hash of the current block is strictly related to the previous one. This is also the logical link between blocks in the blockchain and it is what guarantees data integrity (and as a consequence the integrity of the entire blockchain).

3.9.1. Break the chain

If someone tries to tamper with data in a block, the hash of the block will change. Here is where the previous hash field comes in. Since the hash of a block has been duplicated in the next block, having modified the hash of the current block implies that all the hashes of the next blocks have changed.

As it is easily imaginable, is very easy for the blockchain to realize that something has changed in one single chain and takes appropriate countermeasure.

3.10. Distributed blockchain

A blockchain, finally, is a single chain of blocks replicated in each node joining the blockchain network. The blockchain is a distributed ledger where each node runs a software to maintains data aligned and consistent with all the network. So, in this scenario, each node has a full copy of the ledger and can have access to data without a central authority.

3.10.1. Permissionless blockchain

In a permissionless blockchain network, openness and decentralization are two main aspects. In such solution, anyone (any peer) can leave and join the network whenever they want. Any peer can be both reader and writer without any restriction at any time. In addition, no central authority (no trusted third party) that manages membership is necessary in such scenario. As a consequence of this openness feature, the written data can be read by any peer joining the blockchain network.

The most important and relevant example of permissionless blockchain application is the well-known Bitcoin cryptocurrency and in general in FinTech sector.

3.10.2. Permissioned blockchain

On the contrary of what is written above, permissioned blockchain based solutions are characterized by a limited set of authorized readers and writers. It is a closed ecosystem where peers are not able to freely join and leave the network whenever they want. In this scenario a central trusted authority (that often correspond with the owner of the blockchain network) is needed in order to distribute certificates and access rights for access the blockchain and perform operations on it. The blockchain owner can also issue software updates, decide the network's structure and in general can controls everything that happen on the blockchain. An important aspect of private blockchains is that they are the contrary of the first idea of blockchain; in fact, blockchain was born with the idea to be public, free, decentralized, open to anyone and without the need of a trusted third party that manages data. One interesting difference between permissionless and permissioned

blockchains is that the permissionless ones are often slower than the permissioned ones. In this scenario, information is validated by a restricted set of peers; only approved members of that blockchain can validate transactions for that blockchain. The most famous framework for creating permissioned blockchain solutions is Hyperledger Fabric.

3.11. Byzantine fault tolerance

The byzantine fault is an issue related to the computer science world in which is known that the network may fail in delivering messages. In addition, considering a distributed system, is very difficult to understand where the communication has failed, or who tampered with it. This term takes its name from the allegory of “Byzantine generals’ problem”, conceived in 1982, in which each actor must agree a common strategy to attack the enemy and to avoid a catastrophic failure, but there are some of them who are unreliable. The key concept is trying to achieve certain type of consensus within a group of people or machines (a distributed network). Important is to note that the byzantine fault is one of the most complex and severe between all possible faults: a system to avoid this problem (a tolerance) has been needed in many applications (like nuclear power plants or airplane engine systems) where actions depend on data collected by a large amount of sensors.

3.11.1. Byzantine Generals’ problem

The byzantine generals’ problem is the following: each commander has an army and is located in a different place around an enemy fortress. Generals must decide whether to attack or fall back. It does not matter if they attack or not as long as all of them reach a consensus, that it means that everyone must agree on a common decision in order to execute it in a coordinated way. To complicate the matter, generals are so far apart from each other and the only way they have to communicate is using messengers. In addition, one or more lieutenants may be not fair and try to sabotage the attack sending for example a wrong message or tamper with a valid message or more don’t respect other messages and acting in a different way.

The solution to the problem is an algorithm that can guarantee that all loyal generals decide upon the same plan of action and a small number of traitors cannot cause the loyal generals to adopt a bad plan and run into a catastrophic failure.

The only way to achieve right consensus in this type of distributed systems is to have at least $\frac{2}{3}$ or more reliable and honest nodes. This means that if the majority of the network decides to act dishonestly, the system is susceptible to errors and attacks (such as the 51% attack).

The Byzantine fault tolerance is the property of a system that can resist this kind of failures. This means that a BFT system is able to continue to operate in the right way even if some nodes fail or act dishonestly. There are different approaches to build a blockchain that intends to obtain the Byzantine fault tolerance, and this leads us to the consensus algorithms, that are algorithms designed to enable all nodes of the network to work together to update the ledger securely.

3.12. Consensus

A consensus mechanism is required for a distributed system in order to not fall into pieces in a second and it is the backbone of a blockchain. The direct role of a consensus mechanism is to be sure that all the nodes of the network agree on some principles that are common for all of them. The way a blockchain reach consensus is what can distinguish one implementation to

another, since the main behavior is almost the same. There are many types of algorithms to reach consensus, not only one, and it must be selected considering also the application that will be built on top of the blockchain. This means that not every consensus mechanism is suitable for all applications. Each one of them has specific characteristics that makes the blockchain using that mechanism unique. In particular, consensus algorithms are used to “decide” which transactions are valid, legitimate and which are not. This mechanism is fundamental for the correct behavior of the blockchain. Important is to note that in a public blockchain, everyone can add new blocks to the blockchain, so it’s very important that all the transactions are steadily checked and that the blockchain is constantly audited by all nodes. Without a well-designed consensus mechanism, the blockchain is exposed to various attacks.

Consensus is the process that brings all the nodes together on the final state and value of data. Before speaking about some possible consensus mechanism is very useful to underline some important requirements that must be met to provide the desired result:

- **Agreement:** all loyal nodes agree on the same final value
- **Termination:** all honest nodes terminates the consensus process, eventually reaching a decision
- **Validity:** the final value reached by all loyal nodes must be the same as first one proposed by the first honest node
- **Fault tolerant:** The consensus process must run in the right way also if there are some malicious nodes (called also Byzantine nodes)
- **Integrity:** in a single consensus process, each node must not be able to takes multiple decisions

Types of Consensus

Consensus mechanisms are developed to manage distributed systems in order to avoid faults and to make it possible to achieve a final agreement on values of data. Consensus mechanism are divided in two big categories:

- **Traditional Byzantine Fault Tolerance based:** called also BFT-based, this mechanism is no based on big computational power operations but, instead, it is based on rounds of votes. These kinds of consensus mechanism are also known as “permissioned”. This class performs well when there is a limited number of nodes but does not scale well if the number of nodes increase.
- **Leader election-based consensus mechanism:** this is the consensus process in which nodes must compete with all the other nodes in order to win a lottery to mine new blocks. The elected leader will propose a final value.
This class of consensus mechanism is also called “permissionless”. The leader election-based consensus mechanism class scale very well also with a high number of nodes but performs slow.

3.12.1. PoW (Proof-of-Work)

Proof-of-work consensus mechanism (called also PoW) is one of the first consensus algorithms used in public blockchain and in particular is the consensus mechanism used

in Bitcoin. The Proof-of-Work process is called also mining and nodes who performs this operation are called miners. The goal is to solve complex mathematical puzzles in order to get to create a block and receive a reward for creating a new block. Only the first miner will receive a reward, so they are interested in higher and higher computational power in order to mine new blocks and receive the reward. An important aspect is that the puzzle must be time consuming and computationally complex enough, depending on each single application's requirement.

These types of "proof" have some interesting characteristics: first of all they must be asymmetric, that it means that solving a mathematical puzzle must be time consuming enough (with a certain criterion depending on application) but verifying the correctness of a solution should be very easy and not time consuming. Secondly, finding a solution must be a "brute-force" operation, that it means that who wants to mine a block has to try all possible solutions and guess the right one. Should not be possible to solve the puzzle using any other methods. This also justifies the reason why more and more computational power is required from the miners. Finally, the complexity of these mathematical puzzles could be changed to maintaining the mining time to the desired one, also if computational power increase. The higher the complexity is, the higher the time to mine a block will be.

As an example, consider the practical example in which now it is possible to see where the nonce field comes in. Let's call, for the moment, "valid" a hash that starts with a certain number of zeros (for example 6).

The Proof-of-Work here consists in evaluating a nonce that makes the block's hash valid, that it means evaluating all possible nonce from Long.MIN_VALUE to Long.MAX_VALUE that makes the hash of the current block starting with a certain number of zeros.

The algorithm (in pseudo-code) is shown in Figure 3.

```
long nonce = 0;
block.evaluateHash()
while (block.isInvalid()){
    nonce++;
    block.evaluateHash();
}
```

Figure 3 – Pseudo-code of the proof-of-work algorithm

So, for each nonce the hash is evaluated (using for instance SHA-256) of the entire block concatenated with the current nonce; if the block is valid, the algorithm finished and a new block is mined, else the algorithm continues looking for the right nonce that makes the hash valid.

Here are visible all the aspects described until now, the unique possibility to mine the block is to guess the nonce that makes the block valid.

3.12.2. PoS (Proof-of-Stake)

Proof of Stake is another mechanism to reach consensus in a distributed blockchain. Proof of Stake is more related to financial and economics world. In fact, PoS is based on the main idea that the more coins users own, the more they are interested in keeping the network working in the right way and, as a consequence, the value of the coins high.

In particular, PoS use a particular random process to choose who gets to produce the next block. Here people who generates new blocks are called validators, instead of miners.

To get a block, validators are asked to stake their tokens, that it means that they lock their tokens for a certain amount of time. The process which determines who will get the next block takes into account many factors, depending on the design of the blockchain in which this process is implemented. In general, is true that the higher the stake is, the higher the probabilities to produce a block are. Another common factor is often how long the coins have been staked. As in the Proof-of-Work, also here there's a reward for the work of validators. Again, the reward they receive depends on the blockchain design and implementation but typically, validators receive all, or a part of all, the transaction fees related to all transactions in blocks they create.

There is a big distinction between PoW and PoS mechanism: while in Proof-of-Work, miners can own no coins (that it means they act only for maximize their profits and they are not interested in improving the network), in Proof-of-Stake validators are much more interested in network behavior and right functioning because they stake their coins of the blockchain they are working on to create the next block. Finally, another important distinction between these two mechanisms is the energy consumption: Proof-of-Work needs an enormous amount of computational power in order to mine blocks, while Proof-of-Stake almost nothing.

3.12.3. DPoS (Delegated proof-of-Stake)

Delegated Proof-of-Stake is a more efficient consensus mechanism and is often referred to digital democracy. In this type of consensus mechanism each user stakes a certain amount of coins to vote for a delegate. Important is to notice that the higher is the stake, the heavier is the vote for that delegate. Considering the following example, if user1 stakes 100 coins for a delegate and user2 stakes 10 coins for another delegate, then user1's vote weighs 10 times heavier than user2's vote.

Here, a delegate is a person or an organization who is intended to generate blocks on the network.

Who gets to produce the block, that will be the delegate with the highest number of votes, will receive a reward for his job, coming from transaction fees or fixed amount of coins which are created through inflation, as in Proof-of-Stake mechanism.

Another important aspect is that, since each delegate wants more and more votes, they continuously are incentivized to create valuable things for the blockchain network in order to get more supporters.

3.12.4. PoA (Proof-Of-Activity)

This mechanism is a combination between PoW and PoS but it is more energy-efficient and follow a new idea called "Follow the Satoshi". This mechanism combines together both Pow and PoS in order to achieve consensus and also a good level of security is guaranteed. Proof-of-Work is used at the beginning, at the first stage of this process, and then it switches to Proof-of-Stake, where energy consuming is lower.

3.12.5. PoC (Proof-of-Capacity)

Proof-of-Capacity is a consensus mechanism which is simply a pre-calculation of all possible solutions and store them in a digital storage (like a hard disk). Thinking about Proof-of-Work algorithm, miners use computational power in order to guess the right solution in real time to mine a block. Here, instead, solutions are evaluated at the beginning and then stored. Obviously, the more storage capacity users have, the higher is their probability is to have stored the right solution for an arbitrary block.

Example: solution = 32 byte (sha-256 hash).

A: 1 MB storage can contain 32.768 solutions

B: 1 GB storage can contain 33.554.432 solutions

3.12.6. PoET (Proof-of-Elapsed-Time)

PoET is a special consensus mechanism provided by Intel which is based on a proper set of CPU instructions. The main idea that at each node of the blockchain network is associated a random waiting time and when this time expires the user become the leader to generate the new block. In order to correctly manages this mechanism some requirements must be satisfied. First of all, the network must be sure that every participant chooses a random time (and not an arbitrary time) and secondly, the network must be check if the node has properly waited that time. In order to respect these mechanism Intel provides an instruction set called SGX (Software Guard Extension) that allow applications to execute reliable code in a protected environment. In PoET mechanism, trusted code is what ensure that requirements are satisfied and keeps “lottery” fair.

SGX (Software Guard Extensions)

SGX provides some mechanism in order to be sure that a reliable code has been executed and nobody has tampered with it. First of all, there is a specialized hardware component which is in charge of creating a certificate that certificate that some trusted code has been executed in a protected environment. Moreover, the reliable code is executed in a private environment in order to avoid interaction with the memory space of the trusted code by some other applications.

The first point is important to communicate to all the other participant that a certain node is executing the trusted code for that network. The second point, instead, is useful to make impossible for a malicious node to tamper with code and assigning himself a shorter waiting time.

Summary

All the mechanisms here presented have the same goal: reaching consensus in a decentralized network.

Though, despite these mechanisms have the same goal, their difference in each implementation is wide. While a consensus mechanism that works correctly for each blockchain and is applicable to all the different implementations of the blockchain network does not yet exist, it is interesting to see how these mechanisms have evolved since the first blockchain was built.

3.13. Smart Contracts

The concept of Smart Contract is not new, and it is not born together with the Blockchain. A first primordial idea of smart contract can be found in the early seventies when to use a software product a software license was needed and, after the expiration date, the software product was no more available.

In '90s Nick Szabo wrote a paper in which he tried to formalize the idea of smart contracts defining them as “an electronic transaction protocol that execute the terms of a contract”. Nowadays there is not a global agreement on what is the real definition of smart contracts. The general idea is that a smart contract is a secure and unstoppable software program that is automatically executed when some conditions are met. smart contracts are written in a language that a computer or a target machine can understand and execute. An important feature for smart contract is that they should be deterministic, that it means that given a certain input they must provide always the same output, even in presence of malicious attackers. Also, to be deterministic, a smart contract should be totally bug-free, that is not so common in today software programs. Moreover, a smart contract should rely on the principle that code is law, that it means that there should not be the need of trusted third party in order to provide trust.

3.13.1. Ricardian Contracts

An important aspect is that smart contracts are different from legal prose contracts as they are not so readable in a court of law, even if they do what they are programmed to do. Following this idea, Ian Grigg developed what is called Ricardian Contracts, where he tried to solve both semantic issues and informatic code ones. The idea behind Ricardian contracts is to write a document that is readable and understandable by a court of law and, at the same time, it is executable like a computer software.

3.13.2. Oracles

It may happen, in some scenarios, that in executing a smart contract there is the need of accessing external data from some external source. In these cases, Oracles can be helpful. An Oracle is an interface that can be used to access external data from a smart contract. There are two different possibilities in using Oracles: a smart contract can pull data from Oracles or Oracles can push data into smart contracts. Even if Oracles are trusted entities it should happen that data are not coherent, or they have been manipulated.

Different types of Oracles exist. Simple Oracles are the ones provided by trusted, large and reputable third party, but here there is again the problem of centralization. In this implementation, however, to increase the reliability of Oracles provided data, data are provided not by one single data source but by many data sources, so that if all the data are congruent, they are most likely true and not modified.

Another implementation of Oracles is the decentralization one in which Oracles themselves are built on some distributed mechanism and they find source data from other blockchains, which are driven by some kind of distributed consensus mechanism. One way to prove the authenticity of data is to use what is called TLSnotary. This mechanism allows to have the proof of communication between the Oracle and the data source. The TLSnotary guarantees that the communication between the two entities occurred and in particular gives the proof that the data retrieved by the Oracle are the ones provided by the data source.

3.13.3. Languages for Smart Contracts

Many programming languages are available for writing smart contracts. In general, they are distinguished in two types: DSLs and GPLs.

DSLs (that stands for Domani-specific programming languages) has a small set of functions that are specific and optimized for the application that use this language.

GPLs (that stands for General-purpose programming languages) has a wider variety of functions, obviously not optimized for the specific use, intended to be used to build general-purpose smart contracts behaviors.

3.14. Well-known blockchain vulnerabilities

Nowadays, blockchain and cybersecurity come always together. Cybercriminals look both for network vulnerabilities and user wallet weak credentials in order to get money and bring down the network. This section analyzes the two most famous blockchain vulnerabilities. In particular the first section related to 51% attack has been studied for many years in the area of cryptocurrency and digital assets exchange, mainly in permissionless blockchain networks [34]. The second section instead is more related to cybersecurity world, but a case happened also in the blockchain ecosystem.

3.14.1. 51% attack

This attack could happen when an attacker has the control of the 51% (or more) of the blockchain network hash rate or computational power. In this way the attacker is able to fork the network and create an alternative branch that at the end will takes the place of the existing ones and will be considered the principal one. This was considered the first possible attack to the blockchain and at the beginning it was very unlikely that it could happen. However, many cryptocurrencies have suffered from this attack: Litecoin Cash (estimated damage \$1,7 million), Bitcoingold(\$17,5 millions), ZenCash(\$550.000), Coinrail (more than \$35 millions of damage), Verge, Monacoin [35].

In all the previous cases attackers managed to get enough computational power to compromise the blockchain network and steal millions of dollars, indicated between brackets.

Nowadays, the principle networks that can suffer from this attack are the smallest ones because they attract less miners so it's easier for attackers to get the computational power they need.

The 51% attack probably would not destroy the blockchain-based currencies but certainly would cause serious damage.

3.14.2. DDoS (Distributed Denial of Service)

Distributed Denial of Service attacks are not new and are not strictly related to the Blockchain world. In cyber security, a Denial of Service (DoS) attack is when one attacker tries to bring down a victim by consuming all its computational power and resources with many useless requests.

An example of DoS attack is the ping flooding attack in which the malicious entity sends many ICMP echo requests without waiting for the ICMP echo replies. In this way the victim is occupied in responding the ICMP echo requests and does not serve the “real” clients.

A Distributed DoS attacks happens when there is no one single attacker but there are many and could happen that some of them don't know they are used by the attacker to perform the attack. In particular, an attacker can create what is called a botnet, that is a network of PCs that are used to pilot the attack using phishing techniques for example. In the blockchain network, attackers perform DDoS attacks in order to bring down mining pools, e-wallets and the other financial services of the network. An example of Distributed Denial of Service in the Bitcoin's Blockchain network dates back to 2015 when 80.000 small transactions were sent at the same time on the network causing operations to be blocked [36].

4. TECHNOLOGIES

In this chapter of the thesis some of the most famous and the most advanced blockchain platforms are evaluated. Ethereum and Hyperledger Fabric have been chosen for this analysis because they are the most used in literature. AWS proposed solutions instead have been taken into account due to a strong usage of AWS in Avio Aero software solutions.

4.1. Ethereum

This section has the aim to present one of the most used blockchain framework for developing enterprises blockchain-based solutions, that is Ethereum. Ethereum has been taken into account for its advanced features and its evolved framework. In this section an overview of the Ethereum platform is reported, considering also its advantages and disadvantages.

4.1.1. Introduction

Ethereum [1] is one of the world's biggest framework for creating customized blockchain applications. Appeared for the first time in 2013, thanks to **Vitalik Buterin**, Ethereum has a strong programmable component, which enable blockchain developers to create new types of applications. The main idea of V. Buterin was in fact to develop a touring-complete language that would make it easy for developers to develop smart contracts (programs) for blockchain applications. This main idea is in contrast with the Bitcoin ecosystem, where a limited number of possible operations are available with that scripting language. At the time of this writing, the community behind Ethereum is the largest one and the most active community of blockchain developers in the world.

Ethereum has not a central authority or a centralized entity who controls it. It is maintained, improved and upgraded by a global community of contributors who work on the principal aspects of such blockchain ecosystem.

Ethereum blockchain uses a proof-of-work consensus mechanism that, as mentioned in previous chapters, can result very energy and time consuming.

Moreover, like some other blockchains, Ethereum has its own native cryptocurrency, that is Ether (ETH). Ether has many features in common with Bitcoin, it is a fully digital money, can be exchanged with someone else in the world immediately and can be spent on some other Ethereum blockchain based application. There isn't any company or government that supply Ether and its real-world value is determined by the blockchain itself.

4.1.2. Public blockchain framework and EVM

Ethereum placed itself in the context of public blockchains, appearing as a distributed computing open source platform, that can enable the creation and the management of smart contracts in a peer-to-peer way. In addition, as mentioned above, in the Ethereum blockchain network, is necessary to pay with some Ether the network itself in order to use the computational power. So, for submitting a new transaction in the Ethereum blockchain a payment is requested by the network itself.

The core of Ethereum is its Ethereum Virtual Machine (EVM), that is the engine behind this framework. In summary, the EVM represents the runtime environment for the development of smart contracts in Ethereum. Smart contracts for the Ethereum blockchain are written in Solidity. Ethereum Virtual Machine works in a protected way,

working independently of the network. The code managed by the EVM is separated from the network; in practice, smart contracts are available on the Blockchain as EVM bytecode (that is a specific binary format), written in a high-level Ethereum language and transformed in bytecode by a specific compiler.

4.1.3. The Yellow Paper

The **Ethereum yellow paper** [37] represents the formal description and written presentation of Ethereum protocol. Following the yellow paper instructions, it is possible for anyone to develop an Ethereum client that respects the Ethereum's protocol specifications.

This document is not very easy to read for people who don't have a technical background, especially in math or algebra because it contains a complex, but complete, formal description of Ethereum protocol.

4.1.4. The Ethereum blockchain

The core idea behind Ethereum blockchain is that an initial state (called genesis state) is transformed into another state (called final state) by the execution of transactions.

To better understand this main idea, consider the sample scenario reported below, where one user sends a certain amount of Ether to another user.

Sample scenario

The initial state consists in two wallets (with addresses S and R for simplicity) that contains 35 ETH each. At a certain point S wants to send 15 ETH to R. In this case the transaction consists into the information "A sends 35 ETH to R" (for simplicity gas cost, signature and many other data of transaction have not been included). The final state, resulting by the execution of the transaction mentioned above, will consists in wallet A with 20 ETH and wallet B with 50 ETH. This final state, called also **global state**, is finally stored in the Ethereum blockchain.

4.1.5. Summary and conclusions

The following is a brief summary of the most interesting feature of the Ethereum framework.

- Non pluggable component
- Permissionless (public blockchain)
- Smart contract written in Solidity
- Proof of work consensus mechanism
- Ether is used as a cryptocurrency for the execution of transactions

Since its design, Ethereum has been developed with the aim to be a point of reference in the field of public permissionless blockchain-based applications. Ethereum blockchain can perform to its full potential when used in a permissionless environment. Considering in fact also the literature analyzed in chapter 2 of this thesis, most (practically all) applications related to supply chain management (that inherently consists in a permissioned environment where actors involved in the network are known) haven't chosen the Ethereum platform for developing a blockchain-based application.

4.2. Hyperledger

This section has the goal to present Hyperledger, a global collaboration hosted by The Linux Foundation in order to provide blockchain-based framework. Hyperledger has been taken into account due to its flexible architecture that allows developers to develop many different applications in many different sectors. After an overview of the main Hyperledger's projects, the focus is on Hyperledger Fabric that is the most active Hyperledger project continuously updated.

4.2.1. Introduction

Hyperledger is an open source project powered by the Linux Foundation [38] and is one of the most popular blockchain framework for developing private blockchain applications. Since it is an open source project it has all the advantages of the open source world: anyone is free to use it, code on it, enhance it and also redistribute it. One of the most interesting feature of open source frameworks is their flexibility, modular components and standard adherence that make them easy to use in very different cases.

In the following there is a high-level description of some of the most famous Hyperledger frameworks.

4.2.2. Hyperledger building blocks

In common with many blockchain-based frameworks there are several characteristics including a shared single source of truth, security and tamper-proof, scalable architecture, auditability, confidentiality, etc.

All these features can be summarized in the following four Hyperledger building blocks.

- **Shared ledger.** Each participant has its own copy of the ledger, that is distributed among all the participants to the blockchain network. It is an append-only distributed system of record shared across a business network.
In a permissioned blockchain network, all the participants are identified, and they can have access only to appropriate transactions.
- **Privacy through cryptography.** It is fundamental to include security in blockchain since its design. It cannot be added later. It is essential for ensuring that transactions are correctly authenticated and verified.

- **Consensus or trust system.** It is fundamental to have a trust mechanism, using the power of the network to verify the transactions. It is preferable to use the term *trust* instead of *consensus* because it is an important aspect to be considered for stakeholders to invest money in any blockchain infrastructure.
- **Smart contracts.** Talking about blockchain, a smart contract is what defines the business logic of a blockchain network. It is a business agreement inserted into transactions database and executed with them. It is called smart contract because it is in a digital format, often defined as a computerized protocol. Smart contracts are deployed as code on blockchain nodes.

4.2.3. Hyperledger Iroha

Hyperledger Iroha [39] is a blockchain platform and one of the Hyperledger projects. Hyperledger Iroha is a modern framework, written in C++ and provides a unique consensus mechanism that is chain based Byzantine Fault Tolerant, called YAC (Yet Another Consensus). Iroha provides a small set of commands that let developers correctly perform most common operations in order to manipulate accounts (that are digital identities) and digital assets.

In Hyperledger Iroha, peers, that are also validation nodes, can submit half-signed transaction (as part of multi-signature transactions) using the Gossip protocol. In this framework blocks are stored in files, while PostgreSQL [40] database is used for storing the state of the ledger.

Blockchain-based applications can be developed using Python, Java, JavaScript or C++ programming languages as well as for Android and iOS mobile platforms.

Hyperledger Iroha was built with the goal to provide a permissioned blockchain framework for managing digital assets, digital identities and data in a serialized form. This framework can be used in FinTech sector, mainly in interbank regulation and payments systems, or can be used for building application in digital identities management, such as national IDs. Even if Iroha can be used in FinTech ecosystem, it does not have its own cryptocurrency (such as for example Ethereum). Since it provides a permissioned environment, data access can be monitored using a role-based mechanism.

Iroha is different from other permissioned environments thanks to its previously described consensus mechanism, YAC. This algorithm provides high performances and allows transactions finality with low latency.

4.2.4. Hyperledger Sawtooth

Hyperledger Sawtooth [41] was created with the help of Intel and provides an enterprise modular platform in order to develop, deploy and run distributed ledger applications. Sawtooth uses a consensus algorithm that is called Proof of Elapsed Time (PoET), already described previously in this thesis. However, different consensus protocols can be used. Sawtooth potentially can be applied in many areas, with the support both for permissioned and permissionless network.

Hyperledger Sawtooth make it easier to develop blockchain applications with the separation of the core system from the application itself. This approach allows developer to focus mainly on the application they are developing, defining their business rules and using their preferred programming language, without needing to know the underlying core

system. Important features of sawtooth is versatility and modularity that let major freedom to developers to build many different applications. Moreover, Sawtooth provides smart contract abstraction that let developers to use the preferred programming language for writing the logic in smart contracts. The modularity that distinguish Sawtooth from the other frameworks, allows different types of applications to run on the same underlying blockchain network.

Sawtooth was mainly built for developing permissioned blockchain network. The blockchain itself stores the permissions and roles of each peer node in such a way all actors involved in the network can easily have access to that information.

In addition, Hyperledger Sawtooth is enhanced with a parallel scheduler that enables (when possible) the parallel processing of transactions, even preventing double-spending issue. This parallel execution potentially substantially increases the performances of the whole system.

Another interesting feature of Hyperledger Sawtooth is its compatibility with Ethereum deployed smart contracts, thanks to a project called Seth (Sawtooth-Ethereum integration project).

4.2.5. Hyperledger Burrow

Hyperledger Burrow [42] too has been developed with the help of Intel initially and it is a permissioned blockchain node that is able to execute Ethereum smart contracts, that are smart contracts that follow the Ethereum specifications (and are written in Solidity as previously mentioned). The Burrow node relies on three main components.

- **Consensus engine.** Burrow uses a particular high transaction throughput proof-of-stake consensus protocol called Tendermint. It is a Byzantine fault-tolerant consensus protocol that is based on a set of known validators and protect the blockchain network from forking.
- **Permissioned EVM.** As mentioned before, Burrow is a node that is Ethereum specifications compliant. The virtual machine is built with the aim to respect the Ethereum specifications and it also checks the correctness of the permissions granted.
- **API Gateway.** In Hyperledger Burrow node, both FEST and JSON-RPC endpoints are exposed in order to interact with the blockchain. Actual tools offer the possibility to compile, deploy and link Solidity smart contracts and formulate transactions to call smart contracts stored directly on the chain.

4.2.6. Hyperledger Fabric

Hyperledger Fabric [43] is an open source blockchain framework that provides permissioned distributed ledger technology solutions. It provides a modular framework with a configurable architecture that let many different types of organizations (working in many different sectors, from banking and finance, through healthcare to supply chain) to be interested in such solution.

One fundamental characteristic is that Hyperledger Fabric does not uses domain-specific languages for developing smart contracts, instead, it allows developers to develop smart contracts in many different general-purpose programming languages. Currently, both Java, Go and Node.js are available. This feature is very important for enterprises which already have skills to start developing smart contracts, rather than first having to learn a new programming language. Smart contracts are called chaincode in Hyperledger Fabric ecosystem.

Hyperledger Fabric propose a new architecture where transactions do not follow an order-execute flow but follow a so-called execute-order-validate flow. In particular, unlike many blockchain platform that provide an architecture where transactions first are validated and ordered and then sent to peer nodes in order to be executed, in Hyperledger Fabric execute-order-validate architecture, first transactions are executed and their correctness is checked, then transactions are ordered via pluggable consensus protocol and last they are validated against an application-dependent endorsement policy. Finally, transactions are committed to the ledger. Considering that in Fabric the endorsement policies specify which (or how many) peer nodes must ensure the right execution of a given smart contracts, this proposed architecture is able to eliminate any non-determinisms since inconsistent results can be filtered before the ordering phase. This last aspect is what allows Hyperledger Fabric to use standard programming languages, avoiding binding developers to domain-specific programming languages.

In addition, since Hyperledger Fabric is a permissioned environment, actors involved in the blockchain network are known and each one of them, in order to correctly join the network, must have a digital identity, that takes the form of a public-key certificate, released by a certification authority. A permissioned environment has the goal to secure the interaction between groups of entities that have the same goal in common but maybe they don't trust one to each other.

Unlike most blockchain frameworks, Hyperledger Fabric rely on a pluggable consensus mechanism. This means that, depending on the application that developers are building, the consensus mechanism can be different. This feature is very important for companies; in this way, each company can choose the consensus protocol that best fit for the required use case. Also, differently from other blockchain based proposed frameworks, in Hyperledger Fabric there is not the need of a native cryptocurrency. This aspect can significantly reduce the risk of cyber-attacks.

The modularity that Hyperledger Fabric inherently brings with it is what encourage companies to adopt a blockchain solution with the use of this framework. All the Fabric components are pluggable and customizable. In this way, Hyperledger Fabric can be configured in many different ways in order to meet many different companies' requirements and needs.

All the previously described characteristics are the reason why Hyperledger Fabric nowadays is one of the frameworks that performs better in terms of transaction processing, transaction finality, privacy and confidentiality.

Another important aspect in Hyperledger Fabric is the possibility to create many different channels on the same blockchain network. This feature is interesting when confidentiality properties are requested. In particular, consider the use case in which some consumers

have access to privileged treatment thanks to trust demonstrated over years, In such scenario, if all actors involved in the network has complete access to every data, it could result to keep different business relationships with different customers. In these cases, having different channels means the feasibility of establish different business rules with a subset of participants. In this way, only certain nodes can have access to that smart contract and data transacted, preserving confidentiality and privacy.

4.2.7. Hyperledger Fabric component design

Hyperledger Fabric has several infrastructure components that allow the blockchain to work properly with its tenets of shared ledger, encryption, trust system and smart contracts.

The main infrastructure components are:

4.2.7.1. Hyperledger Fabric certification authority

The Hyperledger Fabric Certification Authority is the implementation of a membership service, that is who provide credentials to access the network (for example the public-private key pair and related the certificates). It is essentially a module that provides right of access and acts as a vehicle to create a root of trust in the network creation.

It is not strictly necessary to use this particular CA; in fact, it is possible to use some other implementation of membership services like any X509-based KPI infrastructure that can issue EC certificates.

4.2.7.2. Peers

Peers are the components that are responsible for keeping the ledger aligned and for existing smart logic, that is the chaincode.

Peers are in charge of receiving a batch of endorsed transactions and they have to validate them and commit transactions to eliminate non-determinism.

The behavior described above is for *validating peers*.

There is also another type of peers that are called *non-validating peers*. The role of these last components is to function has a proxy, in order to connect clients (that means to issue transactions) to validating peers. A non-validating peer can't execute transactions, but it can verify them.

4.2.7.3. Ordering service

The ordering service in Hyperledger Fabric works in strict relationships with the consensus mechanism, that is pluggable.

It simply batches transactions into blocks and outputs a sequence of blocks that are linked one to each other using a hash-chained sequence and each block contains transactions.

4.2.8. Hyperledger Fabric Ledger

The Hyperledger Fabric ledger is built of two main components: a so-called world state and a blockchain. These two components are strictly related one to each other and also one defines the other.

In the following there is a clearer explanation of these two main parts.

World state

The so-called world state in Hyperledger Fabric is a key-value pairs representation of the current states of values stored into the ledger. Basically, it is a database where the current (that is the last) state of an object is stored. The world state changes every time an operation (creation, update or delete) is performed on the ledger. The main goal of the world state is to make easier for an application to retrieve the current state of an object without make the application itself to evaluate all transactions' log directly on the blockchain in order to retrieve the required information.

Blockchain

Beside the world state, there is the blockchain. As explained in previous chapters, a blockchain is a read-append-only data structure that records all the changes happened to objects in the world state. The main difference from the blockchain to the ledger is that once a record is written in the blockchain it becomes immutable, it cannot be modified or deleted.

Sample scenario

To better understand the difference from the world state and the blockchain a sample scenario is reported in which a change of ownership of a car is stored on the ledger.

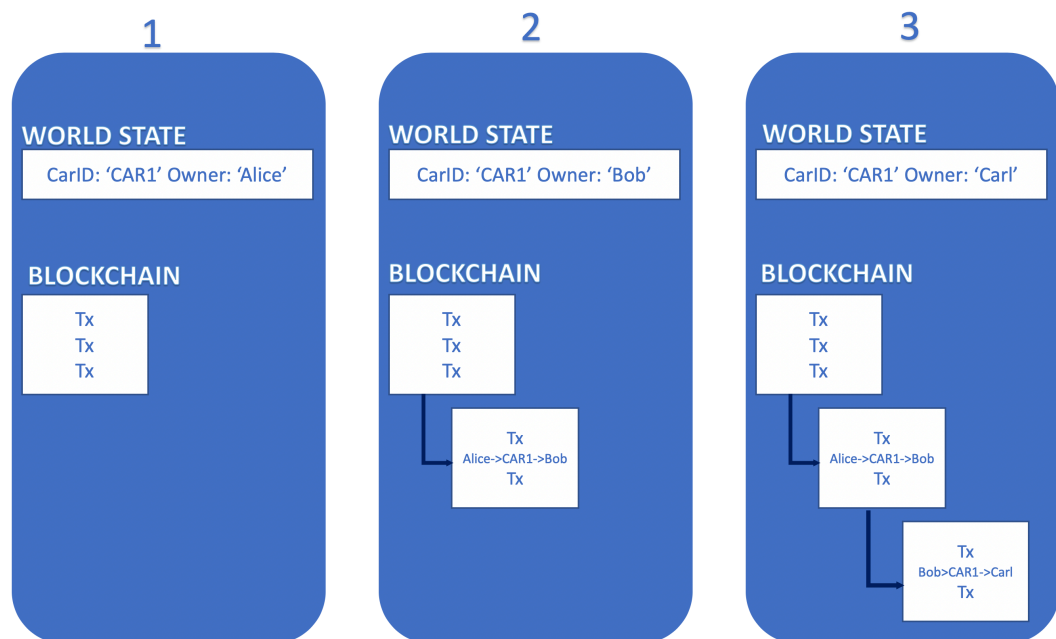


Figure 4 – A visual representation of the relation between the world state and the blockchain

As it is visible from the diagram above, in the initial situation (case 1) Alice owns a car identified by CarID = 'CAR1'. In the word state is reported the state of the object CAR1 with its owner, while in the blockchain there are general transactions. At a certain point (case 2) Alice decides to sell her car to Bob. In this case, as it is visible from the diagram, in the world state is recorder the current situation of the object CAR1, that is the new owner is Bob, while in the blockchain is recorder the change of ownership. Suppose that now (case 3) Bob decides to sell his car too. In this situation, the world state of the CAR1 object is newly updated, and in the blockchain the new change of ownership is recorded. As it is visible from the diagram above, in case 3 for example, the word state indicates that the car is owned by Carl but in the blockchain it is possible to see all the log records related to that car, that is the ownership history of that car.

Conclusions

The Hyperledger Fabric Ledger includes a blockchain and a world state. The ledger is therefore the combination of a world state, where the current version of objects is stored, and a blockchain, where the history of changes of such objects is recorded. In addition, is important to underline that the world state derives from the blockchain, that means that are information stored on the blockchain that determines the current value of the world state.

4.2.9. Transaction's trip

This section is important to better understand the Hyperledger Fabric general structure and components, that cooperate in a transaction processing. The Hyperledger Fabric model is composed of three main steps:

- **Endorsement.** In this first phase, channel's members must agree upon endorsement policies that define the approach to validate transaction proposal. Then, since peers are in charge of updating the ledger, they check the validity of the received transaction proposal and they propose their version of the updated ledger. In this first phase of a transaction's trip, only peer nodes are involved and not the orderers. In particular, client applications send transactions proposal to peers which are included in the endorsement policies. Such peers simulate the transaction in order to define the read and write set and finally send back to client application a transaction proposal response. This phase ends when the application who originates the first transaction proposal received ll the transaction proposal responses from peer nodes required by the endorsement policies.
- **Ordering.** This phase includes the consensus protocol, which since it is pluggable, it does not depend on transactions' semantic. The characteristic of being pluggable provide companies an appreciate high degree of flexibility. This phase of the transaction's trip mainly involves the orderer. After phase 1, when an application has all the transaction proposal responses, it sends them, within the original transaction proposal, to the ordering service that is in charge of collecting many transactions and orders them into a block. the block is finally sent to the all the committing peers of the network. Is important to underline in this phase that the order in which transactions are sort by the orderers may not be the same as the arrival one. The strict requirement is the order computed and proposed by the ordering service must be the same for all the committing peers.

- **Validation.** After the ordering service return the transaction to committing peers, each peer has to approve the ledger update and perform some final application-specific validations. After this phase, the ledger is correctly updated with transactions received.

Considering the journey of a sample transaction, many peer nodes are involved. The following is a summary of Hyperledger Fabric's peer nodes that work for properly update the state of the ledger.

- **Committing peers.** The committing peers are the ones in charge of keeping an update version of the ledger and its state. The committing peer, as the name suggest, is the peer that commits the transaction and, in order to do this, it can hold the chaincode.
- **Endorsing peers.** These peers are a particular variant of the previously described committing peers that are in charge of grant or deny transaction proposal's endorsement. The endorsing peers must hold the smart contract.
- **Ordering service.** The ordering service is composed of a set of nodes with the goal to accept or reject the inclusion of blocks into the ledger. Differently from other peers previously analyzed, the ordering nodes does not need a copy of the smart contract or the ledger to perform their operations.

In the following is reported a clear visual representation of the three main phases described above.

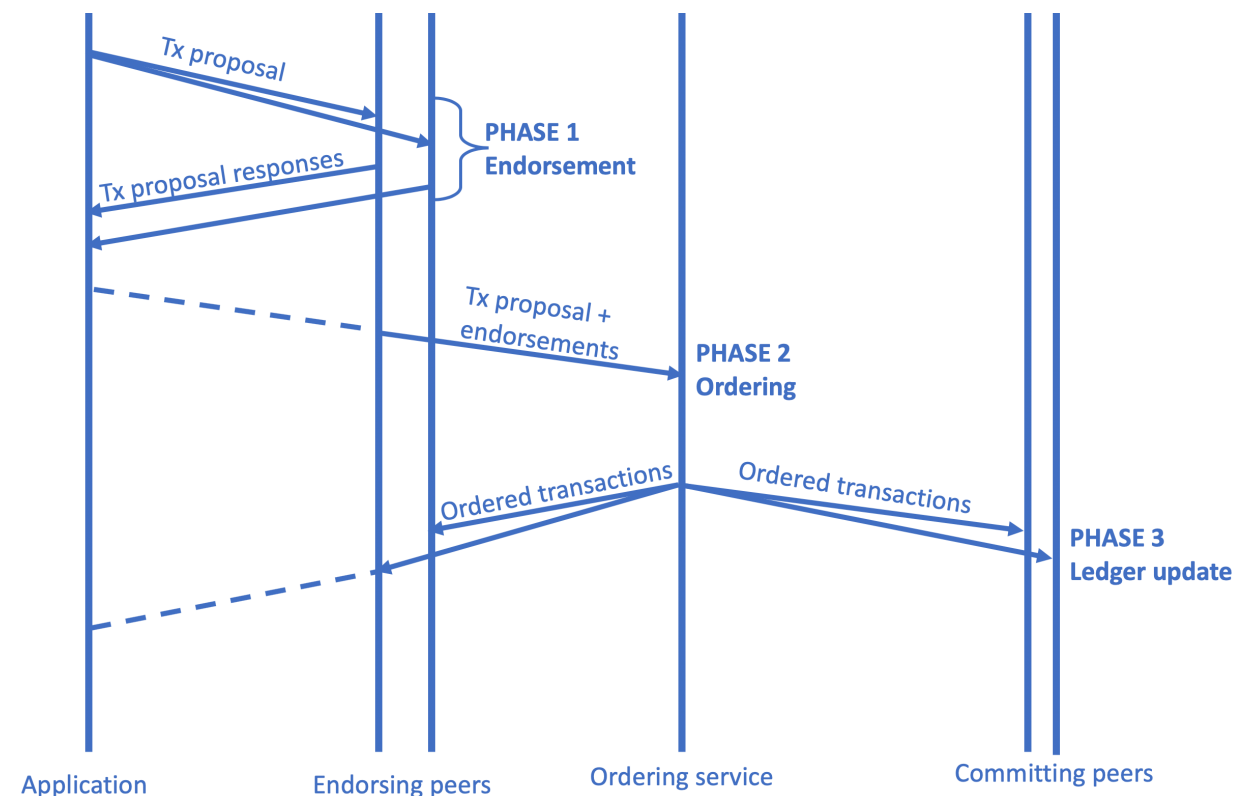


Figure 5 – A visual representation of the main phases in which a transaction is involved, from its proposal to the ledger update.

From the above reported schema, is clearly visible the three main phases in which a transaction is involved, from its proposal from the client application to the update of the ledger by the committing peers.

4.2.10. Summary and conclusions

The following is a short summary of the main aspects of Hyperledger Fabric framework:

- modular architecture;
- extendible architecture;
- pluggable components and algorithms;
- permissioned (private blockchain);
- nodes can have different roles and tasks;
- smart contract written in Go or JavaScript;
- modularity can be applied in many applications (such as supply chain).

The list below represents a summary of the pros of Hyperledger Fabric:

- hardware costs can be very low (possibly zero), depending on the current state of infrastructure where the blockchain application is deployed;
- know-how remains proprietary (internally);
- nodes are deployed on-premise (on Docker containers);
- Data can be both encrypted, both stored as plaintext;
- Less energy consumption with respect to a proof-of-work-based solution.

In the following, a list of the cons of such framework is reported:

- Higher internal expertise required;
- Higher development costs;
- Higher maintenance costs over time (both software and energy-consuming);

Conclusions

Hyperledger Fabric is one of the most used blockchain framework for developing permissioned blockchain-based solutions. Combining many different unique features, it provides a fully running platform for many different applications and requirements needed. Fabric is the most active of the Hyperledger projects. This means the community continuously improve Hyperledger Fabric features for providing the best enterprise blockchain platform as possible. On the contrary, since companies need developers for build and deploy a blockchain network based on Hyperledger Fabric, adopting this solution means to invest some resources in training developers on the framework and spend some time to learn this new and dynamically evolving framework.

4.3. AWS proposed solutions

As mentioned in the introduction of this chapter, there is no literature about AWS blockchain based solution, because it is not yet fully released. By the way, such solution can be interesting from a company's point of view. Since Avio Aero strongly uses Amazon Web Services for its software solutions and it is one of the major customers, the company is interested in evaluating also the AWS proposed solution in order to enlarge the use of AWS introducing also AWS managed blockchain. In this section, both AWS managed blockchain and AWS QLDB are analyzed and compared.

4.3.1. Introduction to AWS

Amazon Web Services (AWS) [3] it's the most used cloud platform in the world and the most complete. It offers more than 165 fully running services. The biggest companies in the world, the start-ups in strong growth and also many government agencies rely on AWS for their infrastructures, becoming in this way more agile and fast with a substantial costs' reduction.

Among the more than 165 complete services offered by AWS, 40 are exclusive: nobody else provide those functionalities.

The ecosystem created by Amazon Web Services is one of the largest in the world. It can count millions of active customers and more that tens of thousands of partners all around the world. Moreover, the customers are from very different business areas.

AWS has been designed with the target to be the most flexible and secure cloud computing platform on the market nowadays. The central infrastructure has been developed to satisfy the highest security requirements from company; in fact, there are customers from military sector, banks and also manufacturing company like Avio Aero.

AWS has released more than 1957 between new services and new functionalities in 2018, reaching speed and innovation level without precedents, especially in machine learning and artificial intelligence areas.

With Amazon Web Services it is possible to take advantage of the latest technology to bring innovation in companies.

AWS managed blockchain is an example of these services.

4.3.2. AWS Managed Blockchain

Amazon managed blockchain [44] is a fully managed service that make as easy as possible the creation and the management of scalable network using the blockchain technology with the use of the most popular open source framework that is Hyperledger Fabric (Ethereum will be available soon).

Nowadays, the most difficult step for who faces for the first time with the creation of a blockchain network is the complexity of the first configuration and management over time.

For creating a blockchain network, without the use of AWS, each member of the network has to manually provision the hardware, install the software and create and manage the certificates for access control. Once the network has been created, there is a continuously need of monitoring and eventually adapt to changes, if any.

AWS Managed Blockchain it's a fully managed service that enables to configure and manage a blockchain network with very few steps. AWS transparently and automatically manages the work for the first configuration and dynamically adapt the network to satisfy the requests from the applications. It provides certificates, it manages them for users, it enables to easily invite new member in the network, and it controls all the technical parameters (like CPU, RAM etc.).

AWS managed blockchain is potentially a good compromise between costs and internal expertise requirements. All the development and management aspects are delegated to AWS, that it means less internal costs, but there are fixed monthly costs to adopt this solution. At the time of this writing, AWS Managed Blockchain is available only in the US East region (N. Virginia).

With the use of AWS managed blockchain it is possible to build private permissioned blockchain network that requires access control and authorizations for a restricted set of known participants. From a more technical point of view AWS managed blockchain provides a selection of types of instances that include many combinations of CPU and memory to allow users to choose the best possible combination for workloads.

Moreover, AWS uses a particular service, AWS Key Management Service (KMS) to protect the network certificates and keys, eliminating the need of a secure storage solution for cryptomaterials.

AWS, differently from Hyperledger itself, use a particular mechanism to store the chronology of transactions, making easier to retrieve information about the “history” of transactions and the restoration of an “old state”. AWS Managed Blockchain uses AWS QLDB (Quantum Ledger Database) in order to do this and to enable these operations. AWS QLDB keeps a chronology of whatever happens to the ledger, ensuring the long-term storage of blockchain data.

A more detailed explanation of AWS QLDB can be found in the next paragraph.

The solution proposed by AWS has its pros and cons.

Pros:

- Quickly creation of a blockchain network on different AWS accounts (allowing a group of members to perform transactions and data sharing without a central authority)
- Elimination of the need of manual hardware provisioning, software installation and security configuration

- Automatically adaption adapt to new needs of continuously evolving applications
- It makes it possible to vote if let a new participant to join the network or not and to remove some other participants
- The internal development costs and effort of keeping everything running are very low, possibly zero.

Cons:

- The costs of development, deployment and maintenance over time to pay to AWS
- The loss of internal know-how, since the set-up of the network is delegated to AWS
- The nodes are physically external

4.3.3. AWS QLDB

AWS Quantum Ledger Database (AWS QLDB) [45] is an immutable, ciphered and verifiable database of a ledger that provides access to transactions belonging to a central trusted authority. Amazon QLDB monitors each data change of an application and store the history of that data, verifiable over time. Often, in parallel to a ledger, the companies keep a chronology of data using tables in relational databases; however, this is a time-consuming solution, exposed to human error. It requires a personalized development and design because relational databases are not inherently immutable, so data tampering could result very difficult to discover.

AWS QLDB is a new frontier of databases that avoid a strong effort for creating ledger applications. QLDB has been designed to be immutable and the chronology of the application can't change over time. In addition, using cryptography (in particular hash codes) it becomes very easy to verify that no unintentional changes have been made to the application data.

AWS QLDB uses an immutable ledger of transactions, called "journal" that continuously look for changes on applications data and store a complete and verifiable history of changes over time.

From a more technical point of view, AWS QLDB provide an SQL API to query the ledger, a flexible model for data and a complete support for transactions. QLDB is also serverless, in this way it automatically recalibrates itself to support application requests. it uses a hash function (SHA-256) in order to provide immutability and verifiability properties. There are no servers to manage and no read or write limit to configure.

Below are reported some important summary aspects of AWS QLDB:

- Fully managed by AWS
- Continuous monitoring of application data

- Cryptographically secured
- Immutability and transparency
- Keeps a ledger of changes in data over time
- Less complexity with respect to AWS Managed Blockchain
- Can be used in parallel with AWS Managed Blockchain with the function of storage of history
- Provide an SQL API to query the history of the ledger
- Serverless
- Higher scalability
- Easy to use

Amazon AWS QLDB is a possible alternative to blockchain when there is a central authority that owns and manage data and wants to share them with other participants. It provides an immutable ledger, fully managed by AWS, in which data can be only pushed into it and not modified or deleted. It is very easy to configure, scalable and can be queried using an SQL API. It uses a document-oriented model for data so that both structured and semi-structured data are accepted. It provides also ACID properties for transactions management.

In addition, AWS QLDB, like AWS managed blockchain, is unfortunately available only in the US East region (N. Virginia) at the time of this writing.

Advantages and disadvantages of the solution based on AWS QLDB, from a high-level point of view, without considering the technical implementation, are practically the same of AWS managed blockchain. The only big difference is that AWS QLDB does not provide a blockchain-based solution, so all the benefits of such distributed ledger could be lost.

5. AS IS vs TO BE

This chapter will focus on the difference between the AS IS situation, that is the way that is currently adopted for the “Digital stamping” use case, and the TO BE situation, that is the new way introduced in case of adoption of a blockchain-based solution. Recall that in the provided use case, only the internal actors of the supply chain (that is the Avio Aero workers) have been considered. That is, the blockchain proposed backbone only manages transactions from within the company. However, the system has been designed in order to be shared among all the involved actors in the whole supply chain.

The Camunda Modeler [46] software (installed version 2.2.4) has been used in order to better represent the two different situations in a clearer visual way.

The description of the AS IS situation was developed during the internship period in Avio Aero, following the information provided by actors involved in the “Digital Stamping” process in the company.

5.1. AS IS situation

The current situation in the company is strongly paper-based documents dependent. Each worker, once the processing of a specific serial number, on a specific machine is finished, has to apply a stamp by hand on a worksheet and then brings this worksheet to the next operator that has to perform the next work on that component on a different machine.

This operation is very time consuming and not very effective from a performance point of view. Operators spend a lot of time in compiling these worksheets and probably they have to cross the entire plant in order to deliver sheets to another operator for the next operation.

In this first approach to the blockchain technology the case of external processing on the components produced has not been considered.

In addition, following the research conducted by Pavlović et al. [47], is clear that electronic data collection and management is highly more convenient than collecting and managing paper-based documents.

Having warehouses full of paper is not the best solution in terms of efficiency in data analysis and this process could result very slow and inefficient. Also, the quality controls, in case of bad piece found in the market or in case of an incident in an aircraft or something else related to security of flights are not optimized using a solution based on paper documents.

Paper is intuitive, people are able to work better with paper and they know what to do, but nowadays, in order to remain competitive in the market, a digital revolution is needed, also in data storage.

5.1.1. High-level BPMN diagram of the AS-IS situation

In the BPMN (Business Process Model and Notation) diagram reported below there is a clearer visual representation of the simplified whole process, from a high-level point of view. It's important to underline that in the real-world workflow this process could be modified in order to inspect the correctness of the process or for sample quality controls.

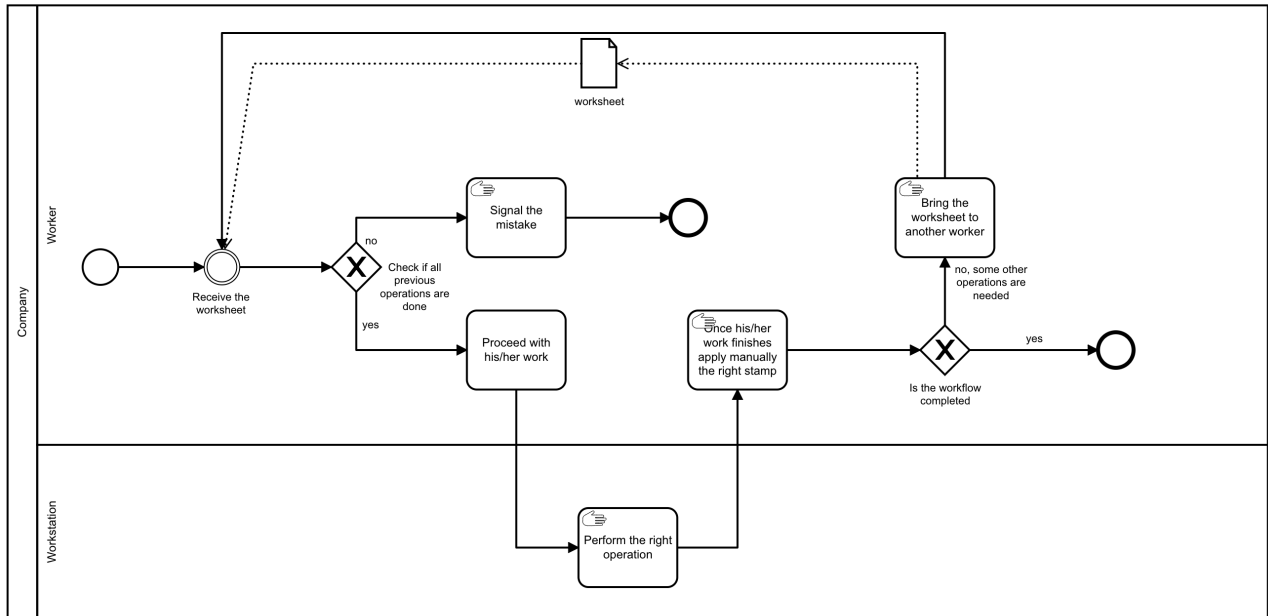


Figure 6 – High-level BPMN diagram of the current situation

As it is clearly visible by the BPMN diagram above, excluding the production operations that are obviously physical operation, also the completion of the work is a manual operation. As explained before, it consists of manually filling out the worksheet, physically apply the stamp and then bring the worksheet to the next workstation. All of these operations are very time consuming and not as much efficient as possible from a performance point of view.

5.2. SWOT matrix analysis

In order to better understand the feasibility of adopting a blockchain-based solution, a SWOT matrix analysis has been performed. In this analysis, pros and cons, both from an internal and an external point of view, are reported considering main aspects and features of the blockchain.

	POSITIVE	NEGATIVE
INTERNAL	<p><u>Strengths:</u></p> <ol style="list-style-type: none"> 1. Traceability of the history of serial numbers 2. Immutability of data 3. Resilient to cyber attacks 4. Authenticity of data 5. Eliminate paper-based document 6. Automation of digital stamping 7. Provide Big Data platform 8. Technological improvement 9. Regulatory aspects 10. Less effort for workers 11. Scalability 12. Flexibility 13. Can be integrated with any existing systems 14. Decentralized approach 15. Robustness (no SPOF) 16. Hardware 17. Proprietary know-how 18. Development, maintenance and upgrade costs 19. Physically internal nodes 20. Smart contract 	<p><u>Weaknesses:</u></p> <ol style="list-style-type: none"> 21. Loss of staff power 22. Loss of human interaction 23. Recent technology 24. Business rules change frequently, blockchain doesn't 25. Potentially in conflict with existing regulatory aspects (ex. GDPR) 26. Energy consumption 27. Concept not easy to understand 28. Hardware 29. Loss of know-how 30. Development, maintenance and upgrade costs (expertise required/loss) 31. Physically external nodes
EXTERNAL	<p><u>Opportunities:</u></p> <ol style="list-style-type: none"> 32. Smart contracts 33. Elimination of trust necessity 34. Compliance and history verification 35. World is becoming more digital 36. Producer and data owner coincide 37. Sharing information 	<p><u>Threats:</u></p> <ol style="list-style-type: none"> 38. Uncertainty about the impact 39. Quantum computing in near future 40. Some cyber-attacks still feasible 41. Sharing information

Table 3 – SWOT matrix analysis

5.2.1. Strengths

1. **Traceability of the history of serial numbers:** through the registration in the blockchain of every step that characterizes the production chain of the goods, it is easily possible to reconstruct the whole history of each produced component (steps, workers, quality control measurements, etc.).
2. **Immutability of data:** Thanks to the immutable nature of the blockchain, once data are pushed into the blockchain they cannot be modified
3. **Resilient to cyber-attacks:** one of the most appreciated blockchain properties is its high resilience to cyber-attacks such as tampering of data.
4. **Authenticity of data:** transactions in the blockchain are digitally signed using public key cryptography (public plus private key pair); each component produced must have the serial number printed permanently on it.
5. **Eliminate paper-based documents:** Since is desired to provide **legal value** to data through the adoption of blockchain (if implemented taking this issue into consideration), it is possible to eliminate paper-based documents. Each stamp can be mapped to a private key that can enable digital signature instead of applying a stamp (in this way the actual stamp is substituted by its digital equivalent with the same security but an easier and faster verification).
6. **Automation of digital stamping:** this is a bases for the elimination of paper-based documents. Smart contract can be used for the automation of the entire process.
7. **Provide Big Data platform:** Data stored in the Blockchain can be used by Big Data algorithms in order to perform statistical analysis.
8. **Technological improvement:** Avio Aero is a high-tech company and it is very important that it perceive the state of the art of technology, with respect to “**Brilliant factory**” idea that guides Avio Aero.
9. **Regulatory aspects:** By implementing within the blockchain all the verifications required by the quality control systems and quality certifications it is possible to easily check that all the regulatory aspects are satisfied, by means of appropriate smart contracts.
10. **Less effort for workers:** There is no need for additional training for workers that already use digital stamping mechanism. In addition, workers who still have to apply stamps on paper will be asked for a less effort with the adoption of the blockchain (the procedure is faster and easier because there is no need to apply the stamp on the paper documents and to manage these documents).
11. **Scalability:** If there are few transactions per day, scalability is not needed, with a favorable impact on the costs, being scalability application dependent.
12. **Flexibility:** Flexibility is one of the most appreciated blockchain property since data in the block can be of multiple types (textual, images, sounds, etc.).
13. **Can be integrated with many existing systems:** Blockchain works in the backend so on top of it many different applications can be easily implemented (for example: in current digital stamping just add calls to new APIs in which data will be submitted to the Blockchain).

14. **Decentralized approach:** Since blockchain is a decentralized system, no trusted third parties are needed for data management.
15. **Robustness (no SPOF):** Since there are multiple nodes, even if one of them shut down the network still continue working properly and data are no lost.
16. **Hardware:** hardware can be both a strength and a weakness point depending on the technology chosen. For example, considering an AWS proposed solution, hardware costs are zero because nodes are physically owned by AWS. Instead, adopting a solution like Hyperledger or Ethereum means having to own the required infrastructure devices.
17. **Proprietary know-how:** the know-how can be both a strength and a weakness point depending on the solution chosen. Like for the hardware, but opposite, delegating the set-up and the management of the blockchain network to AWS means a significant loss of the know-how, while developing the network internally can be seen as a proprietary know-how.
18. **Development, maintenance and upgrade costs:** similarly to the point above reported, development, maintenance and upgrade costs depend on the chosen solution.
19. **Physically internal nodes:** the physical location of nodes belonging to the network depends on the technological solution chosen too. With Hyperledger Fabric for example nodes are deployed by means of Docker containers, while with AWS proposed solution nodes are physically owned by it.
20. **Smart contract:** can be implemented for many features. For example, a smart contract can be deployed that once the piece exits a working station, the digital stamp is automatically applied (if the piece is compliant), and the piece can move to the next stage.

5.2.2. Weaknesses

21. **Loss of staff power:** Some research has brought to light that eliminating human interaction not always is seen as a positive thing. For example, applying smart contracts for automating payments can be seen as loss of power by someone. A paper about this has been published by **Senato della Repubblica – 11^a Commissione Lavoro, previdenza sociale** [48]
22. **Loss human interaction:** Eliminating the paper is a way to reduce human interaction
23. **Recent technology:** Since private blockchains are an emerging technology in industries there are no standards to follow (lack of standards) and it is not yet a mature technology that is 100% safe to adopt.
24. **Business rules change frequently, blockchain doesn't:** In industries business rules may change frequently due to market change for example. In a blockchain, once is designed for an application, can be very difficult (require some effort) to change its behavior
25. **Potentially in conflict with existing regulatory aspects:** It's important to verify all "legal" aspects before implementing a blockchain solution for an industry such as Avio Aero. For example, is possible that some data cannot go out as plaintext but need a form of encryption.
26. **Energy consumption:** must be taken into account for a Blockchain application, it is dependent on the number of nodes and transactions
27. **Concept not easy to understand:** The concept can be difficult to understand for people involved in software development.
28. **Hardware:** hardware can be both a strength and a weakness point depending on the technology chosen. For example, considering an AWS proposed solution, hardware costs are zero because nodes are physically owned by AWS. Instead, adopting a solution like Hyperledger or Ethereum means having to own the required infrastructure devices.
29. **Proprietary know-how:** the know-how can be both a strength and a weakness point depending on the solution chosen. Like for the hardware, but opposite, delegating the set-up and the management of the blockchain network to AWS means a significant loss of the know-how, while developing the network internally can be seen as a proprietary know-how.
30. **Development, maintenance and upgrade costs:** similarly to the point above reported, development, maintenance and upgrade costs depend on the solution chosen.
31. **Physically external nodes:** the physical location of nodes belonging to the network depends on the technological solution chosen too. With Hyperledger Fabric for example nodes are deployed by means of Docker containers, while with AWS proposed solution nodes are physically owned by it.

5.2.3. Opportunities

32. **Smart contracts:** smart contracts can be implemented for many features. Smart contracts contain the business logic of the system and they can automatically trigger some operation when some conditions are met. For example, is it possible to trigger automatically payments when an external supplier finishes his work on a component and sends it back to Avio Aero.
33. **Elimination of trust necessity:** With the adoption of blockchain there is no need for a third trusted party that manages the system. Each involved actor keeps a full copy of the entire ledger. Consider also that in permissioned blockchain solutions each peer node must have a valid digital identity (that consists in a public key certificate) in order to join the network and interact with the ledger.
34. **Compliance and history verification:** if a buyer has access to the blockchain data he can easily check the identity of the purchased goods and be able to verify all the quality control steps it has undergone in the supplier production chain.
35. **World is becoming more digital:** The introduction of a blockchain-based solution can be seen as a positive aspect in the global reputation of Avio Aero around the world. Investing in new technologies is fundamental for companies that want to remain competitive in the market.
36. **Producer and data owner coincide:** There is no need for a trusted third party that manage sensible data. In addition, blockchain applications make very difficult to modify data “on the fly”, tampering data can happen but it requires a lot of time.
37. **Sharing information:** the possibility that blockchain provides of sharing data across the entire network and to the all involved participants can be seen both as a positive aspect and a negative one. Blockchain is able to provide a secure data storage solution in a network of untrusted actors. In addition, data payloads can be encrypted in such a way nobody, with the exception of who owns the right key, can have access to those data.

5.2.4. Threats

38. **Uncertainty about the impact:** Since blockchain is an emerging technology is very difficult to predict its impact on the business. Up to now, almost all application have brought innovation and brand improvement.
39. **Quantum computing in near future:** Quantum computers are increasingly a reality, it's difficult to estimate now the impact on cyber security with that computational power.
40. **Some cyber-attacks still feasible:** In spite of being very secure, some blockchain attacks are still feasible (51% attack and DoS are two examples of blockchain attacks).
41. **Sharing information:** the possibility that blockchain provides of sharing data across the entire network and to the all involved participants can be seen both as a positive aspect and a negative one. Since there can be sensible information, companies may not want to share their sensible data with other participants in the network. In addition, exposing personal data outside can be a weak point, in terms of cyber security aspects.

5.2.5. Conclusions

After the critical analysis conducted with the use of the SWOT matrix, is clearly visible that positive aspects are substantially more than the negative ones, and also the weight associated to positive improvements is higher than weaknesses ones.

Since many weak points are strongly technology-dependent (points 29,30 and 31), choosing the right technology, the right platform and using it at its full potential means possibly avoid them. Moreover, many other weak points are related to human world (see points 21, 22 and 27). These weaknesses can be avoided, and also can become a strength, with the right teaching to human people involved in such new ecosystem.

On the other side, there are positive aspects both from a technological point of view, where paper-based documents are eliminated and there is an improvement and innovation in company's information systems, and both from a more physical perspective, since workers are asked for a less effort during the process.

After this analysis is it possible to states that the adoption of a blockchain-based solution can be the right solution for the use case provided and, considering the mission and the vision of the company, this adoption could be positive in Avio Aero's reputation around the world.

5.3. Proposed solution

As mentioned in the introduction chapter of this thesis, the proposed solution is based on a Hyperledger Fabric network, a REST server developed with ExpressJS and a web-based client application developed with Angular. This chapter aims to provide a high-level view of the proposed solution in order to address the specifications given by the company, considering both the network infrastructure built with the use of Hyperledger Fabric framework and both considering the applications built on such infrastructure.

5.3.1. High-level perspective

For the goal of this thesis, considering the use case provided by the company, a sample scenario has been taken into account, without providing any sensible information related to the real workstations, components and the real workflow. The aim is to prove that a blockchain based solution can be adopted in such scenario and evaluate the pros and the cons of adopting such solution.

The TO BE situation is no longer characterized by paper-based documents. Instead, a more digital approach is considered, introducing the blockchain as a backbone of the production environment, and user interface for enabling workers to interact with the system and submit the transaction, that is the information of their work, to the blockchain. In the BPMN diagram below is reported a BPMN diagram of the proposed solution from a high-level point of view, without considering all the side aspects but focusing on the main workflow. The goal of this thesis is not to drastically modify the whole production process but to integrate it with a blockchain-based system for securely store information related to components produced and to eliminate paper-based documents.

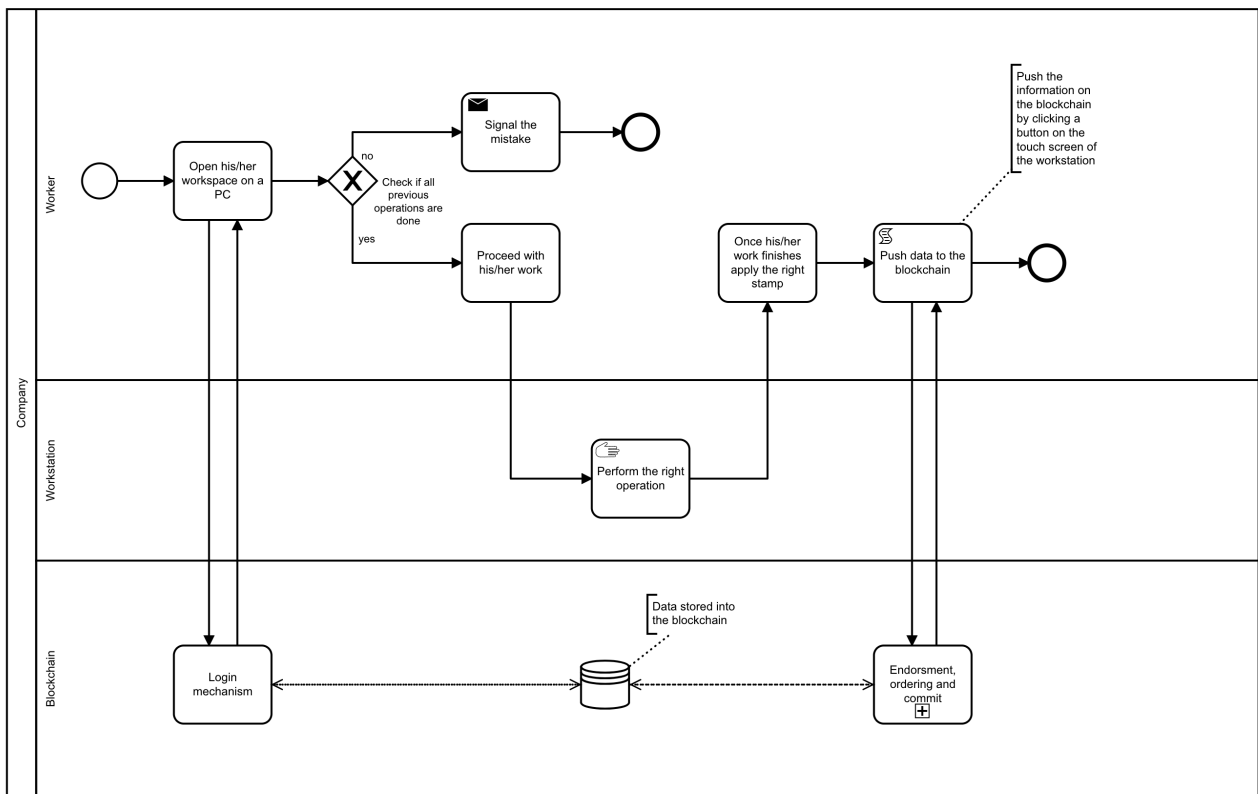


Figure 7 – High-level BPMN diagram of the TO BE situation with the adoption of the blockchain

As it is easily visible from the BPMN diagram reported in Figure 7, excluding production works (that inherently remain manual works), the final phase (from a worker point of view) is characterized no more by manual operations and paper-based worksheets but everything is done digitally, with the introduction of the blockchain network.

In particular, when a worker finishes his/her job on a certain workstation for that particular serial number, he/she simply fulfill a digital form, in a web-based application, with the required information and then simply submit this form to the system.

This operation triggers a call to the developed backend server that act as a gateway for the interaction with the blockchain. The server mainly receives requests by the users and wrap them to calls to the chaincode. In this phase, the entire mechanism of the blockchain is started and the server waits for the completion of the submit request.

From the diagram reported above, another aspect can be easily identified. In this first proposed solution and proof of concept, the blockchain is currently owned by the company, which has full access to data.

5.3.2. Architecture overview

The proposed solution, from a technical point of view, follows a layered approach, as seen in the state of the art. At the underlying level there is the running Hyperledger Fabric blockchain network. It has been developed by means of a smart contract, that contains the business logic, written in Go and JavaScript. The smart contract is installed in peer nodes of the network. A more detailed explanation of the network architecture is in the next section of this chapter.

On top of it there is the application developed to interact with the underlying blockchain network. The application consists in a backend REST server, written with the use ExpressJS and a web-based user interface on one side, and a simulator on the other side. The server acts as a mediator between users and blockchain. In fact, it receives the HTTP requests and wrap them to calls to the smart contract of the blockchain. Since it follows the REST guidelines provided by R. Fielding in [49], many different applications can be built to interact with it, from web-based applications to mobile ones.

On top of this layered stratified approach there is the user interface that provides visual way to display data.

On the other side instead the simulator component has the role of emulating the behavior of automated populating the blockchain submitting new transactions.

A clearer visual representation is reported in Figure 8.

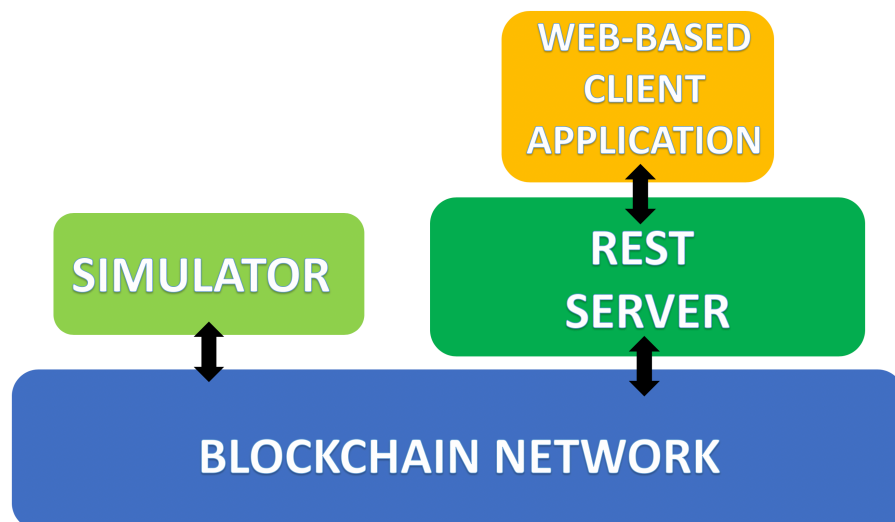


Figure 8 – A visual representation of the layered architecture of the proposed solution

As it is clearly visible from Figure 8, a layered approach has been followed for developing the proposed solution. Each layer can interact only with its neighboring layers.

In particular, user interfaces call the backend server's APIs through HTTP methods which wrap them into calls to the related smart contracts of the blockchain network. The web-based client application has been developed with Angular framework and using Typescript programming language [50], the REST backend server instead has been developed using ExpressJS framework while the blockchain consists in the following described Hyperledger Fabric blockchain network. Choosing a REST approach for the backend server allows developers to develop many different types of application (e.g. mobile apps, web-based apps, etc.).

On the other side, the simulator consists in a JavaScript file that it has been developed for populating the blockchain, emulating what would be the real use of the system.

Next section represents a more technical and detailed view of the blockchain backbone.

5.3.3. Blockchain network proposal

This section has the goal to provide a more technical and detailed explanation of the blockchain network set-up for the development of this proof of concept.

The proposed solution has been deployed by means of a Hyperledger Fabric network. The proposed blockchain network is composed two organizations (respectively called org1 and org2), two peer nodes for each organization (respectively called peer0.org1, peer1.org1, peer0.org2 and peer1.org2), one membership service provider for each organization (called respectively ca_peerOrg1 and ca_peerOrg2) and one ordering service (called orderer). All the previously described actors are deployed by means of Docker containers and actually runs locally on developing machine.

In Figure 9 there is a clearer visual representation of such blockchain network.

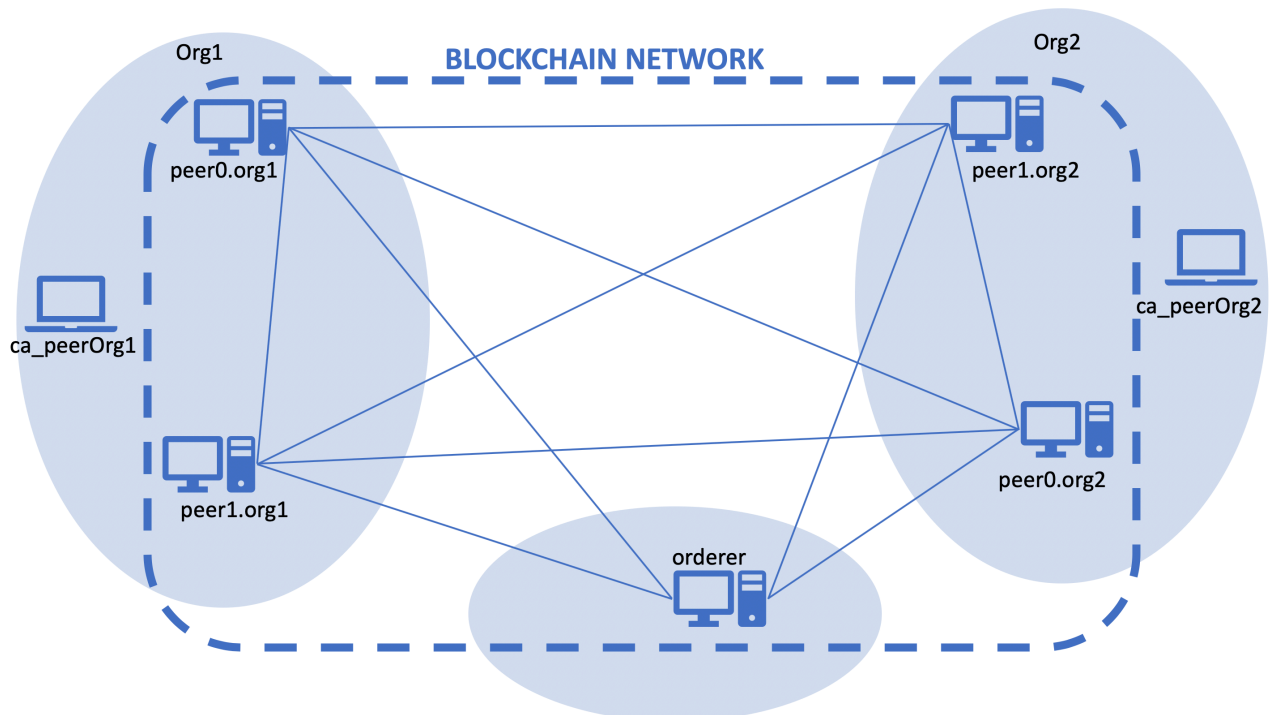


Figure 9 – The architecture of the blockchain network in the proposed solution

As it is clearly visible from Figure 9, the blockchain network consists in the previously described main components. The orderer node can be either owned by org1 or org2 or owned by a third organization joining the network. In all cases, even the ordering service must own its own digital identity provided by a membership service provider.

Note that even if it is not necessary to have to different organization joining the blockchain network, this approach has been chosen first to test the high potential of the Hyperledger Fabric network, and second to prove that this system, in future works, can be extended to other participant in order to make the system itself more complex and let the blockchain to work at its full potential.

5.3.4. REST server

In this section, the REST server built on top of the Hyperledger Fabric blockchain network is presented.

The server has been developed with the use of ExpressJS framework, that is a flexible Node.js web application framework that provide a robust set of features for web and mobile applications development. ExpressJS provide a large set of HTTP utility methods and middleware in order to enable developers to build a robust set of APIs in a fast and easy way.

For the goal of this proposed solution, the REST server developed acts as a middleware between client applications and the underlying blockchain network.

Basically, it receives HTTP requests from clients and wrap them into calls to Digital Stamping smart contract functions. In order to do this, this software module connects to a peer using the before enrolled 'user1' credentials (a more detailed explanation of users' registration can be found in next chapter).

HTTP verbs are correctly accepted by the software and responses produced are consistent within data stored on the network and they respect the REST guidelines as described previously.

In Figure 10 there is a high-level view of the architecture of the proposed solution and how components interact one to each other.

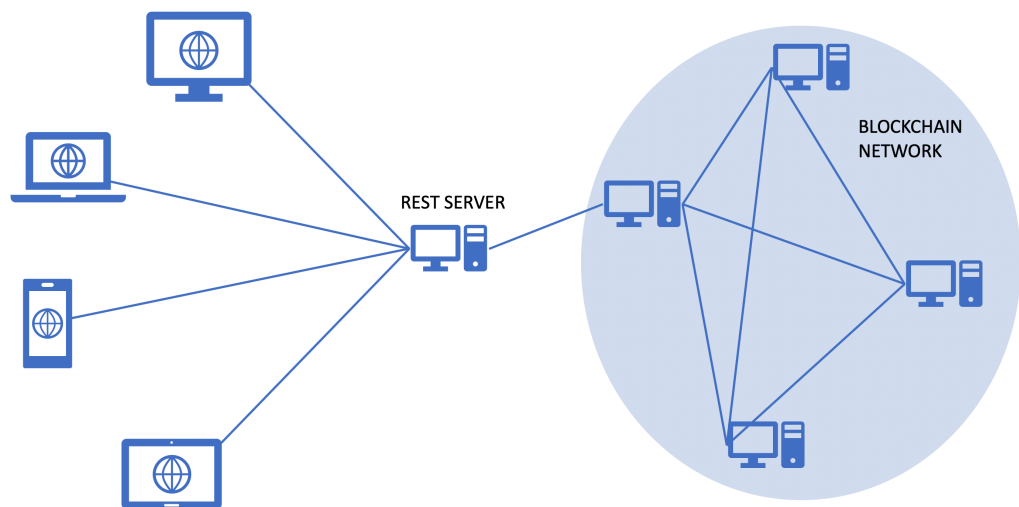


Figure 10 – A visual representation of the architecture proposed in order to interact with the blockchain network through the REST server.

As it is clearly visible from Figure 10, the REST server developed acts as a middleware between the blockchain network and client applications. As mentioned above, since it follows the REST guidelines, many different applications can be developed to interact with it, from browser-based applications to mobile ones.

5.3.5. Web-based client application

The aim of this section is to present the web-based client application built at the top of the stratifies approach followed for the development of the proposed solution.

Since the underlying server follows the REST guidelines, it is possible to build many different applications to interact with it. A web-based approach has been followed for creating a client application with the goal of querying the ledger and display data stored on the blockchain.

The web-based user interface has been developed with Angular framework, that enables developers to build application for many different types of platforms in an easy and fast way.

Also the web-based client application proposed follows a stratified approach. With the use of Angular, many of the so-called dummy-components have been created in order to simply display data on the browser. Such components rely on a service (called ‘data.service’ in the proposed solution) that has the purpose of interacting with the REST server in order to send HTTP requests and receive responses.

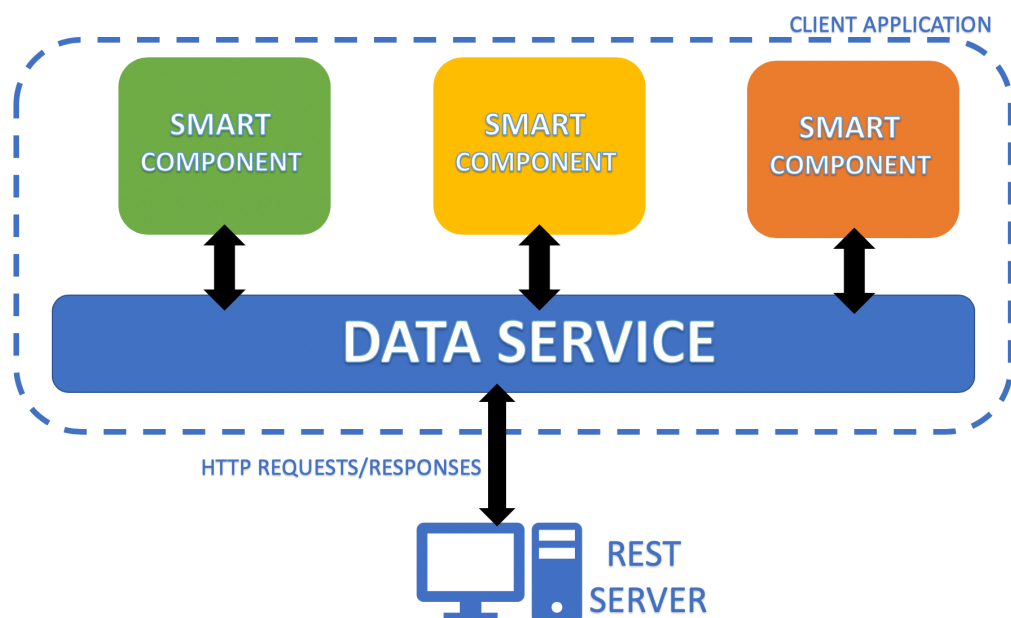


Figure 11 - A visual representation of the client application architecture

As it is visible Figure 11, the components do not exchange HTTP messages directly with the REST server, but the data service does it for them. In this compartmental way of producing client applications, each software module can focus mainly on its functions, delegating all the other to right components. In this way, also, developers can mainly focus

on one operation at a time, without creating intricate systems that are difficult to understand and to manage.

5.3.6. Simulator

This section aims to better describe, from a high-level point of view, the simulator developed for the proof of concept of this thesis. As mentioned previously in this chapter, the simulator consists in a JavaScript program with the goal to populate the blockchain with some sample transactions.

The simulator first connects to a peer, using the credential of the user, namely user1, previously registered. After this phase, the simulator is able to retrieve information about the Digital Stamping smart contract of the network and to start submitting sample transactions.

Figure 12 reports a high-level diagram of the workflow of the simulator in order to populate the blockchain.

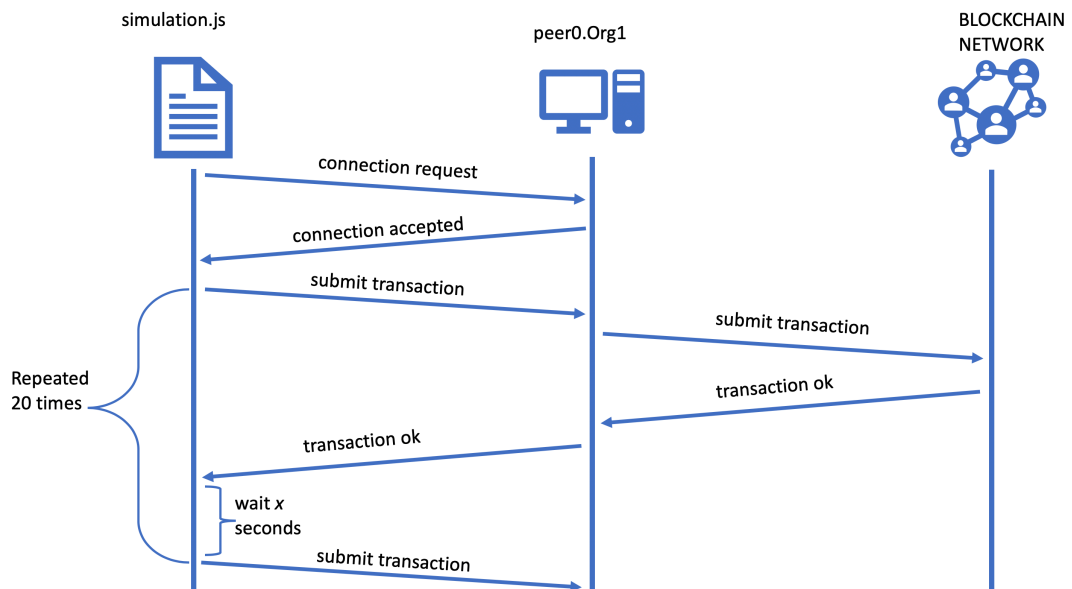


Figure 12 – A visual representation of the whole functioning of the simulator in submitting new transactions to the blockchain.

As it is clearly visible from the diagram reported in Figure 12, the simulator first connects to the peer node, that is part of the network, and then, every x seconds it submits a new transaction to the blockchain. X is a parameter of the simulator and, for the development of this thesis, it has been set to 5 seconds. Also the number of loops can be customizable: it has been set to 20 in order to test the right functioning of the simulator itself but it can assume different values. Consider that in the use case provided, the rate with which transactions will be submitted is in the order of hours.

In the sample diagram reported above, for simplicity, is supposed that everything worked properly, and no exception management is needed.

6. PROOF OF CONCEPT

This chapter describes the proof of concept developed for the use case provided by Avio Aero, the so-called Digital Stamping use case. The chapter first provides an overview of the developed proof of concept from a high-level point of view and then, each single component is deeply described and documented. First the set-up and the execution of the blockchain network is described (with also the description of the chaincode developed), then the chapter provides a description of the simulator, that directly runs on top of the blockchain network. Finally, the REST server and the client application are presented.

6.1. Introduction and scenario description

The Proof-of-Concept has been realized with the use of Hyperledger Fabric framework for the blockchain network, ExpressJS for the REST backend server and Angular and Typescript for the web-based client application.

The “Digital Stamping” application has been designed and developed considering the requirements provided by Avio Aero and it has been developed considering the state of the art analyzed in Chapter 2. It is important to underline that differently from the literature, in designing this proof of concept, not all the actors involved in the supply chain have been considered but only the ones related to the company. Anyway, the proposed blockchain network, can easily be extended to all the involved participants in future work. A sample simulator has been developed using JavaScript and Go programming languages and provides the functionality required by the company; every x seconds (x can be evaluated considering some samples production operations in the stable of Rivalta di Torino) the simulator submits a new transaction on the blockchain network containing data described in next sections. It simulates what the worker has to do in order to use this application; when an operator finishes his/her work, he/she simply press a button on the touchscreen and this operation will trigger the script that will create the transaction. From this moment the information has been pushed into the blockchain and it becomes immutable and securely stored.

6.2. Blockchain layer

The following are some simplifications that have been made in order to develop the proof of concept.

- 1) Data inserted in transactions are the most interesting one, not considering every detail of production operation but a sample customization of them for the realization of the proof of concept. Below such data are summarized.
 - a. “rowID”, is an identifier of the transaction. It is automatically evaluated by the system.
 - b. “workerID” is the SSO of the worker who performed the operation
 - c. “stampID”, the substituted physical stamp has a corresponding digital code
 - d. “machineID” is the identifier of the workstation
 - e. “tstamp” that identify the date and the time when a transaction has been submitted to the system

6.2.1. UML diagram

In this section, a more formal representation of data managed by the system is shown. The following diagram has been developed using StarUML software application [51].

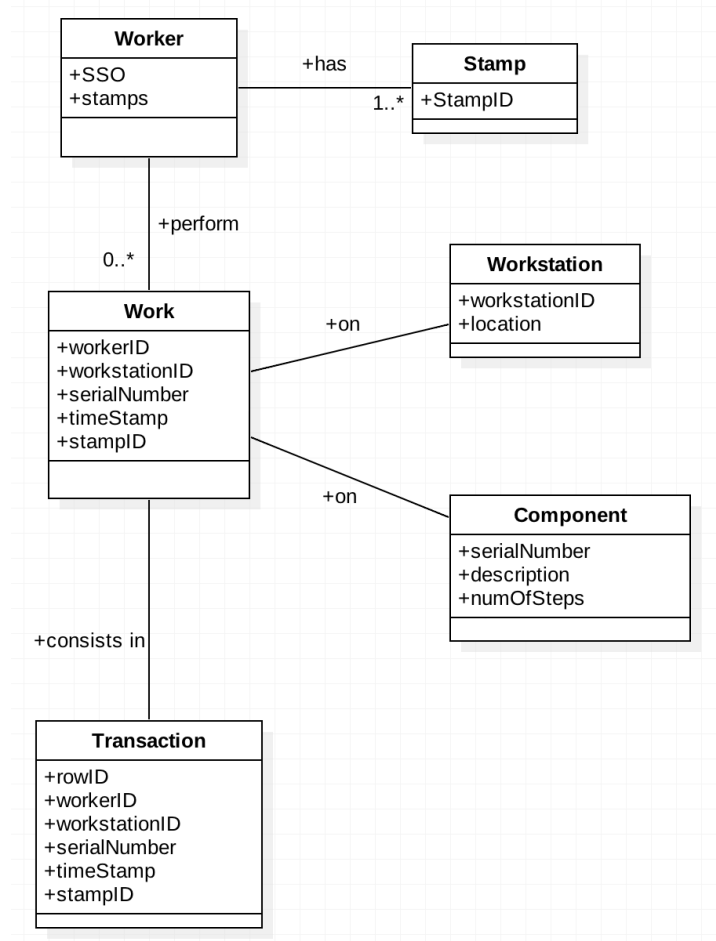


Figure 13 – UML diagram of the simulator in the proposed solution

As it is visible from Figure 13, the simulator takes in input a work and submit a transaction to the system. The simulator consists in the defined operations allowed by the digital stamping smart contract, such as query one single row given the rowID, query all rows, create a new row and so on. As input for creating a new transaction, the simulator requires information related to a work.

6.2.2. Why Hyperledger Fabric?

For the development of the proof of concept of this thesis, Hyperledger Fabric blockchain framework has been chosen. Hyperledger Fabric is the most used framework in developing permissioned blockchain ecosystems. It allows to block users who don't are authorized to access data, without spreading sensible data of registered and authorized users. Hyperledger Fabric identify authorized members by means of public and private key pairs and respective certificates. So, only users with a valid certificate can interact with the

ledger, query and update it. So, referring to this specific case, only Avio Aero entities can have access to sensible data in the blockchain.

Beside the access control of the blockchain there is another important feature that should be considered. Hyperledger Fabric allows the cooperation between different organizations. Considering this feature, it is possible, for a manufacture company like Avio Aero, to extend the blockchain network to the other companies that work for it so that the blockchain can work to its full potential in an untrusted environment. This feature is reached without the need of a third trusted party which controls the whole process.

Finally, Hyperledger Fabric allows to develop fully customizable smart contracts in order to better satisfy the requirements of each application.

The following is a brief summary of the most interesting Hyperledger Fabric features:

- **Permissioned blockchain network:** as explained before, only authorized members can have access to ledger's data and interact with them. Sensible users' data are not widespread.
- **Many organizations involved:** in future, it will be possible to extend the blockchain network to other companies in order to provide end customers the best data reliability as possible and to let the blockchain works to its full potential.
- **Disintermediation:** like other blockchain network, Hyperledger Fabric does not need for a central authority who manages data and oversees to the entire process.
- **Smart contracts:** with the use of Hyperledger Fabric it is possible to write smart contracts that fit for each different application. In particular, as explained later in this chapter, also data submitted into transactions can be customized as the requirements request.

6.2.3. Hyperledger Fabric project

This section will expose the main steps to follow in order to start working with Hyperledger Fabric blockchain framework, from the installation of the software needed to the deployment of the code and the use of it. This is not an official documentation of Hyperledger Fabric and for the technical description of how to install Fabric and how to work on it, readers have to follow the official documentation written by the community [43].

6.2.3.1. Setting up of working environment

The proof of concept for this thesis has been developed on a MacBook Pro 13", running macOS Mojave (version 10.14.6), with a 2,7 GHz Intel Core i7 CPU and 16 GB of RAM (LPDDR3).

Prerequisites

Hyperledger Fabric and all its required tools and software modules are open-source and they are downloaded for free from the Internet. The following is a list of required

components that must be installed in order to properly run and work with Hyperledger Fabric:

- Docker (installed version 19.03.1, build 74b1e89)
- Docker-compose (installed version 1.24.1, build 466789b)
- Go programming language (installed version go1.11.5 darwin/amd64)
- Node.js (installed version 8.16.0)
- Npm (installed version 6.10.3)
- Hyperledger Fabric Examples [52]
- Hyperledger Fabric tools (in the fabric-samples folder as binaries) including:
 - cryptogen
 - configtxgen
 - configtxlator

In the Hyperledger Fabric examples folder can be found a set of sample network configurations with the aim to test the correct installation of Hyperledger Fabric.

6.2.3.2. How to set up a running network

In order to start working on the blockchain network, multiple complex operations must be executed. In the following there are the main operation to follow:

1. Generation of cryptographic material
2. Definition of configurations
3. Deployment in Docker containers

Generation of cryptographic material

As shown in previous sections Hyperledger Fabric is composed by means of multiple entities. Peers, orderers and clients are one example of them, and they must have valid credentials (a certificate) to join the network.

Here **cryptogen** tool can be used in order to generate the right crypto material. This tool reads from a file (**crypto-config.yaml**) in which is written how many organizations, how many peer nodes per organization and how many orderers there are in the system). In other words, the crypto-config.yaml file contains the topology description of the network. Moreover, cryptogen allows developers to generate all the needed certificates and keys, both for organizations and for parties belonging to those organizations. Each organization is given a unique root certificate (called ca-cert) that links that organization's components to it. This tool will create a folder, called **crypto-config**, in which there will

be stored a set of public key certificates, including all the keys necessary for the authentication and authorization of the participants (both private and public ones). The aim of this phase is to simulate and reproduce the real behavior of a typical network, where a participant must be fitted with its own certificate provided by its certification authority (CA).

Definition of configurations

With the use of Hyperledger Fabric it is possible to have many organizations that cooperate in order to exchange data. The **configtxgen** tool is used to set up all the necessary to have different channels and organizations and also for creating the *anchor-peers*, that enable the interaction between different organizations.

In this sample PoC, two different organizations are created, with two peers per organization) in order to simulate a more advanced behavior of the Hyperledger Fabric blockchain network.

All these configurations are specified in a specific file (**configtx.yaml**) in which is possible to identify three main sections:

- Organizations. This section is used to specify how many organizations are included in the blockchain application. Here name, id, policies and anchor peers must be defined.
- Orderer. This section has the aim to describe the behavior and the features of the orderers of the system. In this section also information like hostname and port are specified. Moreover, this section allows developers to define some parameters about the blocks that will be created in the blockchain (maximum / minimum size, max number of transactions accepted, etc.).
- Profiles. This section defines the set of organizations (also called consortium) that can interact one to each other and the channels (with their names).

Deployment in docker containers

All the participants described until this point are deployed by means of Docker containers, one container for each entity described above. The configuration parameter of the blockchain network are in the **docker-compose.yaml** file, in which is possible to define the containers settings for each entity of the network.

The main containers used are:

- Peer container settings. For each peer node the hostname, IP address and ports are defined. In addition, many other information can be retrieved in this section, such as the address of the anchor-peer or the information about the CouchDB state database (that is optional), etc.
- Orderer container settings. For each orderer must be defined the hostname, IP address, ports, etc.

- Certification authority container settings. This is the container that host the CA service that releases certificates to enable secure communications through TLC channel.

6.2.3.3. Run the Hyperledger Fabric network

In this thesis proof of concept, one blockchain network has been set up with two organizations and 2 peers per organization.

Due to the high complexity of the overall procedure to set up all the necessary modules in order to have the blockchain up and running, a bash script is used.

startFabric.sh is a script provided in the fabric-samples folder downloaded directly from the official GitHub repository of Hyperledger Fabric [52].

To run the script it is possible to execute the following command:

```
./startFabric.sh
```

In the first section of the script some configuration properties are set: for example, it is possible to set the runtime language, according to a parameter of the command run previously.

After this first step, there are the commands to install the smart contract related to Digital Stamping application on the peers. Recall that each peer node of this PoC is running on a Docker container, so there is one container for each peer node, one container for each membership service provider and one container for the orderer service. Internally, is called the command

```
peer chaincode install
```

A more detailed explanation of the chaincode can be found in next sections of this thesis. In the next section of the “startFabric.sh” script, as shown in Figure 14, the smart contract related to the Digital Stamping application is instantiated on the channel. Internally is called the command

```
peer chaincode instantiate
```

```

echo "Instantiating smart contract on mychannel"
docker exec \
  -e CORE_PEER_LOCALMSPID=Org1MSP \
  -e CORE_PEER_MSPCONFIGPATH=${ORG1_MSPCONFIGPATH} \
  cli \
  peer chaincode instantiate \
    -o orderer.example.com:7050 \
    -C mychannel \
    -n digitalStamping \
    -l "$CC_RUNTIME_LANGUAGE" \
    -v 1.0 \
    -c '{"Args":[]}' \
    -P "AND('Org1MSP.member','Org2MSP.member')" \
    --tls \
    --cafile ${ORDERER_TLS_ROOTCERT_FILE} \
    --peerAddresses peer0.org1.example.com:7051 \
    --tlsRootCertFiles ${ORG1_TLS_ROOTCERT_FILE}

echo "Waiting for instantiation request to be committed ..."
sleep 10

```

Figure 14 – A code snippet of startFabric.sh script where the smart contract is instantiated on the channel

At the end of the startFabric.sh script the initLedger() method of the chaincode is called on the channel created. A more detailed explanation of the chaincode can be found in the next section.

At this point, a fully running blockchain network composed by two organizations, two peer node per organization, two ca per each organization and an ordering service has been created.

Creating an admin user

During the creation of the network (that is the execution of the startFabric.sh script) an user (named 'admin') was created as registrar for the CA (the Certificate Authority). At this point is fundamental to generate the admin's certificate and his private and public keys pair. To do this it is possible to execute the enrollAdmin.js script launching the command

```
node enrollAdmin.js
```

The output of the execution of this command can be found in a sub-directory called 'admin' of the directory 'wallet'. Here, the private key, the public key and the certificate of the admin users can be found and inspected.

The process of enrolling a new admin follow a CSR mechanism. In Certificate Signing Request mechanism (CSR), first the private and the public keys are generated locally. Then, the public key is sent to the certification authority that returns the encrypted certificate to be used by the application.

```

// Enroll the admin user, and import the new identity into the wallet.
const enrollment = await ca.enroll({ enrollmentID: 'admin', enrollmentSecret: 'adminpw' });
const identity = X509WalletMixin.createIdentity('Org1MSP', enrollment.certificate, enrollment.key.toBytes());
await wallet.import('admin', identity);
console.log('Successfully enrolled admin user "admin" and imported it into the wallet');

```

Figure 15 – A code snippet where a new admin user is enrolled in the system

Creating a user to interact with the ledger

In order to interact with the blockchain, now that an admin user has been created, it is possible to register a new user (named 'user1' in this example), that will be able to query and update the ledger. In order to do this, a similar approach to what was done for the admin user can be done here. The following command can be used in order to register the new user:

```
node registerUser.js
```

At the end of this phase, in the wallet directory can be found another sub-directory called 'user1', where the information about private key, public key and certificate are stored, as happened for the admin user.

```
// Create a new gateway for connecting to our peer node.
const gateway = new Gateway();
await gateway.connect(ccpPath, { wallet, identity: 'admin', discovery: { enabled: true, asLocalhost: true } });

// Get the CA client object from the gateway for interacting with the CA.
const ca = gateway.getClient().getCertificateAuthority();
const adminIdentity = gateway.getCurrentIdentity();

// Register the user, enroll the user, and import the new identity into the wallet.
const secret = await ca.register({ affiliation: 'org1.department1', enrollmentID: 'user1', role: 'client' }, adminIdentity);
const enrollment = await ca.enroll({ enrollmentID: 'user1', enrollmentSecret: secret });
const userIdentity = X509WalletMixin.createIdentity('Org1MSP', enrollment.certificate, enrollment.key.toBytes());
await wallet.import('user1', userIdentity);
console.log('Successfully registered and enrolled admin user "user1" and imported it into the wallet');
```

Figure 16 – A code snippet of the registration of a new user into the system

This phase of the setup of the network is very important in order to interact with the ledger. Recall that Hyperledger Fabric provide a permissioned blockchain network. So, in order to properly interact with the network (that is not only submit new transactions but also query the world state of the ledger) it is necessary to be authenticated with valid credentials.

This phase therefore is essential for building the upper layers of proof of concept of this thesis. In fact, every component previously mentioned (both the simulator and the REST server) first connect to the peer node using the credential of the newly registered *user1* user, and then they can correctly interact with the ledger.

6.2.4. Digital Stamping smart contract

This section represents a deep analysis on what concerns the smart contract of the Digital Stamping application and the operations allowed by that chaincode.

The chaincode is where the actions that peers can perform on the blockchain are defined. As previously mentioned, there are three versions of the digital stamping smart contract, one written in Go programming language and the other two in written in JavaScript (in order to be compliant also with lowest versions of such programming language). The chaincodes are organized in a sub-directory of 'chaincode' directory called 'digitalStamping'. The three main operations allowed by such smart contract are related to query the ledger, create a new element in the ledger, by means of submitting a transaction, and change some values in a previously stored element. Recall that, even if the ledger must be immutable, modifications are allowed and they are view, from a system perspective, as a new transaction with the new value of an existing element. The previous information is

not deleted or overwritten but simply, the system considers as the last version of that element is the last one. This is what in Hyperledger Fabric is called **world state**.

```
type Row struct {  
    WorkerID string `json:"workerID"`  
    StampID string `json:"stampID"`  
    MachineID string `json:"machineID"`  
    Tstamp string `json:"tstamp"`  
}
```

Figure 17 – The definition of the Row struct that represents data submitted to the blockchain

Figure 17 reports the Row struct definition, taken from the Go version of the Digital Stamping chaincode, in order to better view the structure of each element submitted to the blockchain by means of transactions.

Is important to underline that a smart contract's function can be called also with prior authentication. Since Hyperledger Fabric provide a permissioned network, in order to interact with such network, is fundamental for a peer to have its own digital identity released by a membership service provider of one of the organizations involved in the blockchain network.

Querying the ledger

In the following code snippets, the versions written in JavaScript of the smart contract are reported.

Recalling some more technical aspects, each peer of the blockchain network keep an updated version of the ledger. An application can inspect the ledger and query the most recent version of such ledger. This means that the actual state of the ledger (called world state in Hyperledger) is shown after a query operation invoked from the smart contract. The world state of the ledger takes the form of a set of key-value pairs and given a key (or a set of keys) the corresponding record (or records) are returned and displayed.

In the digitalStamping.go chaincode, two ways of querying the ledger are present:

- **'queryRow(rowID : string)'** is the function enables to query the ledger and given a specific rowID, the corresponding value is returned. Remember that values are stored in form of key-value pairs.


```

async queryRow(ctx, rowID) {
  // get the row from chaincode state
  const rowAsBytes = await ctx.stub.getState(rowID);
  if (!rowAsBytes || rowAsBytes.length === 0) {
    throw new Error(`${rowID} does not exist`);
  }
  console.log(rowAsBytes.toString());
  return rowAsBytes.toString();
}

```

Figure 18 – `queryRow()` smart contract's method used in order to query the ledger and retrieve the information about one single object

- **'queryAllRows()'** is the function that returns the world state of the ledger, that is the set of all the elements in the ledger. Since a Row object may have been modified and updated during its life cycle, the `queryAllRows()` method returns the so called world state of the ledger, that is the last version of each element currently stored in the ledger.

```

async queryAllRows(ctx) {
  const startKey = 'ROW0';
  const endKey = 'ROW999';
  const iterator = await ctx.stub.getStateByRange(startKey, endKey);
  const allResults = [];
  while (true) {
    const res = await iterator.next();
    if (res.value && res.value.value.toString()) {
      console.log(res.value.value.toString('utf8'));
      const Key = res.value.key;
      let Record;
      try {
        Record = JSON.parse(res.value.value.toString('utf8'));
      } catch (err) {
        console.log(err);
        Record = res.value.value.toString('utf8');
      }
      allResults.push({ Key, Record });
    }
  }
  if (res.done) {
    console.log('end of data');
    await iterator.close();
    console.info(allResults);
    return JSON.stringify(allResults);
  }
}

```

Figure 19 – `queryAllRows()` smart contract's method that is used in order to query the ledger and retrieve the information about all the stored objects

As shown in Figure 19, all elements with a rowID between a certain range (that in this case is between 'ROW0' and 'ROW999') are first printed on `console.log` and then returned.

The ledger query is the simplest operation allowed by a blockchain-based system. Since it does not imply the consensus mechanism, it is also very efficient. Recall that, since Hyperledger Fabric provide a permissioned environment, also for querying the network users must connect to an authenticated peer node, that is a peer node with a valid certificate for that blockchain network.

Updating the ledger

Another important and fundamental operation that can be performed on a ledger is the update operation. Potentially there are many update operations that can be performed but for the goal of this thesis the creation of a new row is the most interesting one. This operation (the creation of a new record) is a simple operation, from the application point of view. The application simply has to submit a transaction to the blockchain network. Once it has been validated and committed, a notification is sent back to the application in order to signal the success of the operation.

From a network point of view instead, this process starts the consensus mechanism, where the components of the blockchain network must work together in order to ensure that the proposed update of the ledger is valid and executed in an agreed and consistent order.

In order to do this, the transaction (that contains the proposal to update the ledger) is sent to peer nodes of every organization involved in the blockchain network. Each peer generates a signed transaction response using the proposal received by executing the requested smart contract and return it to the origin application SDK. Then all these responses are collected by the SDK and sent to the orderer into a single transaction.

The orderer is in charge of collecting and sequencing all transactions from every application into a block containing many transactions. Finally, the ordering service distributes blocks, containing validated and committed transactions, to every peer in the blockchain network and notify the SDK which returns control to the application.

All these operation are wrapped into the **'submitTransaction()'** function that call the **'createRow()'** function of the chaincode, reported in Figure 20. The submitTransaction() function is analyzed later in this chapter when the REST backend server is presented.

```
async createRow(ctx, rowID, workerID, stampID, machineID, tstamp) {  
    console.info('===== START : Create row =====');  
  
    const row = {  
        machineID,  
        docType: 'row',  
        workerID,  
        stampID,  
        tstamp,  
    };  
  
    await ctx.stub.putState(rowID, Buffer.from(JSON.stringify(row)));  
    console.info('===== END : Create row =====');  
}
```

Figure 20 – createRow() smart contract's method used in order to add a new object into the ledger

In the update context, as mentioned before, potentially many operations are allowed. For the goal of this thesis, for sure, the most interesting one is the one described above.

However, the DigitalStamping smart contract allows some other update on the ledger. Since each element stored on the blockchain takes the form of a key-value pair, given the key, that is the rowID in the proposed solution, each field of the associated value object can be updated.

In Figure 21, the changeStampID(rowID: string, newStampID : string) is reported.

```

async changeStampID(ctx, rowID, newStampID) {
    console.info('===== START : changeStampID =====');

    // get the row from chaincode state
    const rowAsBytes = await ctx.stub.getState(rowID);
    if (!rowAsBytes || rowAsBytes.length === 0) {
        throw new Error(`${rowID} does not exist`);
    }
    const row = JSON.parse(rowAsBytes.toString());
    row.stampID = newStampID;

    await ctx.stub.putState(rowID, Buffer.from(JSON.stringify(row)));
    console.info('===== END : changeStampID =====');
}

```

Figure 21 – *changeStampID()* smart contract's function used to update the value of the ledger related to the stampID field

This function allows to change the current value of the stampID associated to the rowID key object. This could happen, for example, when a worker mistakenly applies the wrong stampID to his work record. At the end of this operation the updated record is retrieved from the blockchain and returned to the caller function. As mentioned above, is important to underline that this operation, from a blockchain-based system point of view, is seen as an append operation, that is this information is appended in the blockchain as a new transaction and the world state is updated. Anyway, the previous information is not deleted or overwritten; simply the system considers for the world state the last version of each element stored in the ledger.

6.3. PoC simulator

The simulator of the proof of concept for the goal of this thesis has been developed in Avio Aero and for Avio Aero use case. The core of the PoC, as explained in previous sections, consists in simulate a sample behavior of the workflow of the process analyzed in this thesis. From a more technical point of view it emulates such behavior by means of a while loop that every x seconds a new transaction is submitted to the blockchain. In the submitted transaction there are sample information related to a work performed by a worker (with a certain workerID), on a certain workstation (identified by a machineID), terminated on a certain date and time (represented in the tstamp parameter) and with an applied digital stamp (that here is called stampID). The transaction is submitted to the blockchain network with the call of the submitTransaction() method of the Digital Stamping smart contract.

The simulator consists of a JavaScript file named **simulation.js** that emulates the real behavior of the Digital Stamping application.

As Hyperledger Fabric provides a permissioned blockchain network, first the simulator has to connect to an authenticated peer node, using the previously described credential of 'user1'. After this, the simulator is able to retrieve the information about the channel and the digital stamping chaincode, that are needed in order to submit a new transaction to the network.

The core of the simulator is reported in Figure 22.

```

let i = 0
while (i < 20){
  let rowID = 'ROW'+i
  let stampID = 'stampABC'+i
  let workerID = 'worker00'+i
  let machineID = 'machineXXX'+i
  await sleep(5000);
  await contract.submitTransaction('createRow', rowID, workerID, stampID, machineID, '123456789');
  console.log('Transaction ' + i + ' has been submitted');
  i++
}

```

Figure 22 – The core of the simulator consists in a while loop that submit transactions at regular intervals in time to the blockchain

As it is clearly visible from Figure 22, the x parameter of the simulator has been set to 5000ms for development purposes, but it is fully customizable. Considering that the **sleep()** function accepts a parameter that represent the milliseconds, the code fragment reported above submit a new transaction every 5 seconds. Consider that in the use case provided by the company, the average elapsed time between one operation and another one is the order of hours. Another parameter set for developing the simulator is the number of submitted transactions. This value has been set to 20 but also in this case it is fully customizable. As output, the code reported above, after having called the **submitTransaction()** function, perform a query on the ledger in order to correctly check if the new transactions has been committed, using the **evaluateTransaction('queryAllRows')** method, and print it on the console log.

After the simulator has been run, the REST server can be used in order to retrieve information stored on the blockchain.

6.4. Backend REST server layer

This section aims to presents the backend REST server developed for the goal of this thesis. The developed software module, as mentioned in the previous chapter, acts as a middleware between client applications and the underlying blockchain network. It consumes HTTP requests and produce HTTP responses according to the REST guidelines.

The server has been developed with the use of ExpressJS framework and Swagger in order to document APIs [53]. ExpressJS framework provides developers a fast and easy way to develop flexible Node.js web applications.

The methods allowed are described by the Swagger documentation reported in Figure 23, automatically generated by the annotation in server source code.

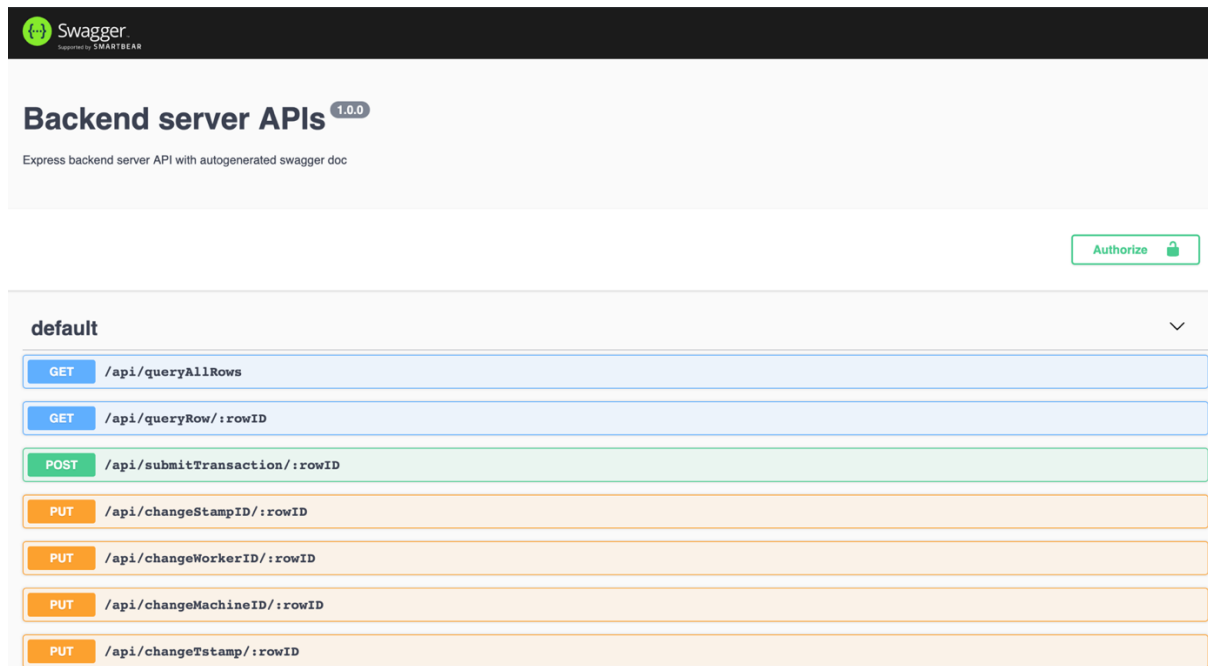


Figure 23 – REST server's APIs documented with the use of Swagger

As it is clearly visible from Figure 23, the REST server enables developer to develop applications that call those APIs in order to properly interact with the underlying blockchain network. The above reported documentation is automatically generated by means of Swagger annotations in the source code of the server developed, thanks to which developers can simply describe the behavior of each method (in the form of comments with a certain format) and the swagger parser atomically produce the related documentation.

Following the layered approach described in previous chapter, each of the above reported APIs, internally wrap a smart contract function's call. As can be guessed by the URLs, the GET request on localhost:8080/api/queryAllRows internally calls the evaluateTransaction("queryAllRows") method of the Digital Stamping smart contract.

In this sample backend REST server, there is one HTTP request for each smart contract functions. Each HTTP requests, triggers an async function that internally contains a Digital Stamping chaincode function's call.

It is important to underline that, upon each single call to the REST APIs, the server first connect to a gateway using the *user1* credentials, and once it is authenticated, it can correctly retrieves the information related to the chaincode and operates on the ledger calling the appropriate smart contracts functions. Moreover, once the response is sent back to client applications, the function, before returning the control to the main function that listens on port 8080 for new requests, closes the connection with the gateway of the blockchain network, in order to avoid some pending connections that can be used as vector of cyber-attacks.

Considering for example the POST request on /api/submitTransaction/:rowID, the server first extracts from the body the transaction to submit to the blockchain network (in form of a JSON object) and then calls the corresponding submitTransaction('createRow', ...) function of the smart contract.

For the PUT requests, the behavior is similar to the previous described one. The server first extracts from the body the corresponding JSON object (that depending on the method called has a different value with a different meaning), and then calls the respective chaincode's method considering the record with the rowID specified in the path to update. Both for POST and PUT requests the server, after having called the right function of the smart contract, perform an evaluateTransaction() operation in order to return client the created or updated resource.

Remembering that the server developed follows the REST principles, many already existing applications can be used in order to interact with it and query and update the underlying ledger. In fact, with this purpose, Postman [54], that was born with the idea to be a robust API development environment, has been used in order to test the right functioning of the developed REST APIs.

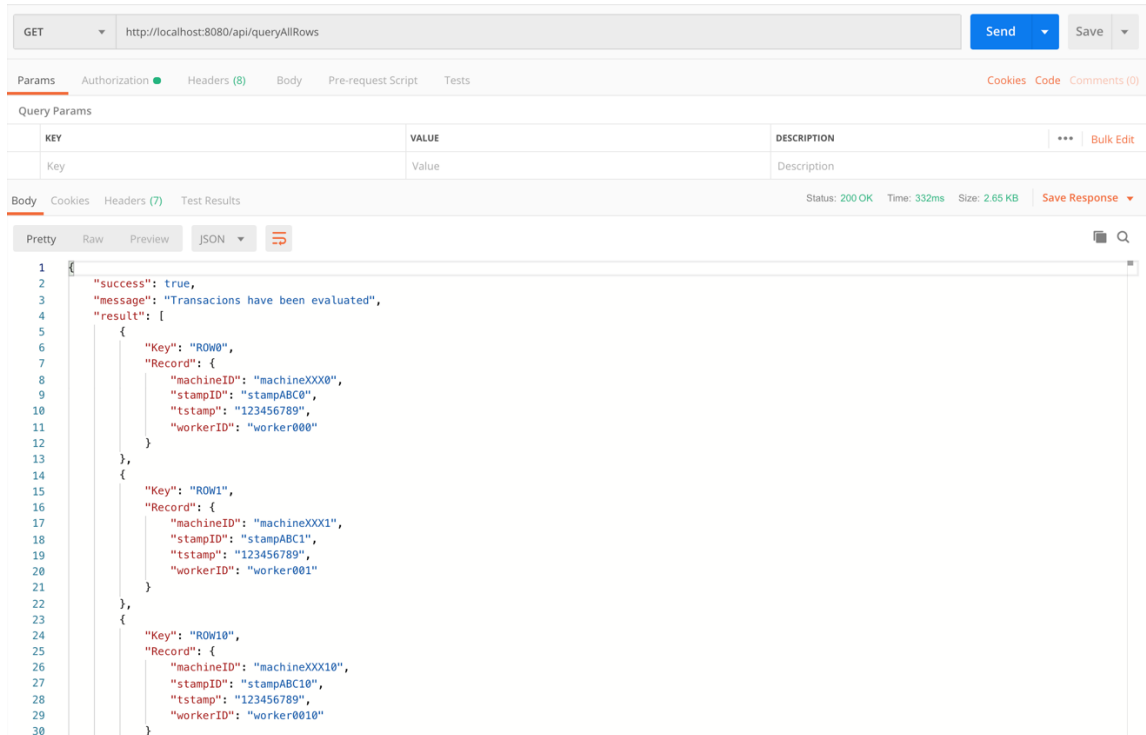


Figure 24 – Server response after a GET request shown with the use of Postman application

The Figure 24 represents the Postman interface. In particular, as it is visible from the picture, a HTTP GET operation has been performed on the URL 'http://localhost:8080/api/queryAllRows' and the returned object is printed in the above section of the image. The returned object consists in a JSON object with the following fields:

- **Success.** This field is a Boolean value and represent the status of the response. This is not the status of the HTTP response but the status of the query of the ledger performed by the REST server.
- **Message.** As the name suggests, this is a message returned by the server that indicates the exit status of the method. It can be both an error message and a positive message like the one displayed above.
- **Result.** This field represent the result of the query on the ledger. In this case, since the queryAllRows has been called, all the object stored in the ledger have been returned.

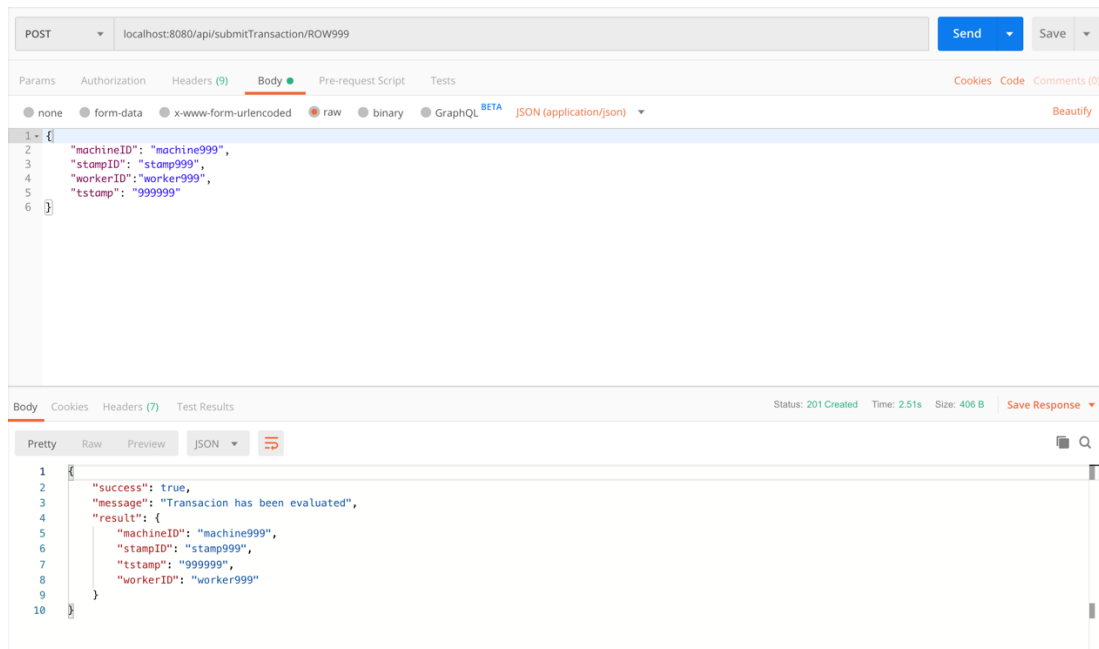


Figure 25 – Server response after a POST operation in which a new element is created and submitted to the blockchain

Another example in the use of a REST client like Postman is the one reported in Figure 25. In this case a POST operation has been performed, following the REAST APIs previously documented with Swagger in which a new object is created on the ledger. In HTTP POST body a JSON object is defined with the fields defined in the smart contract. Upon the POST request, the server contact the peer node and submit a new transaction with the provided information. After the mechanism described in chapter 4 about the transaction's trip a queryRow('rowID') operation is performed in order to properly check the new value of the ledger. The JSON object returned to the client is the one previously described with the difference that the result field is not an array of element but only the one just created.

6.5. UI client application layer

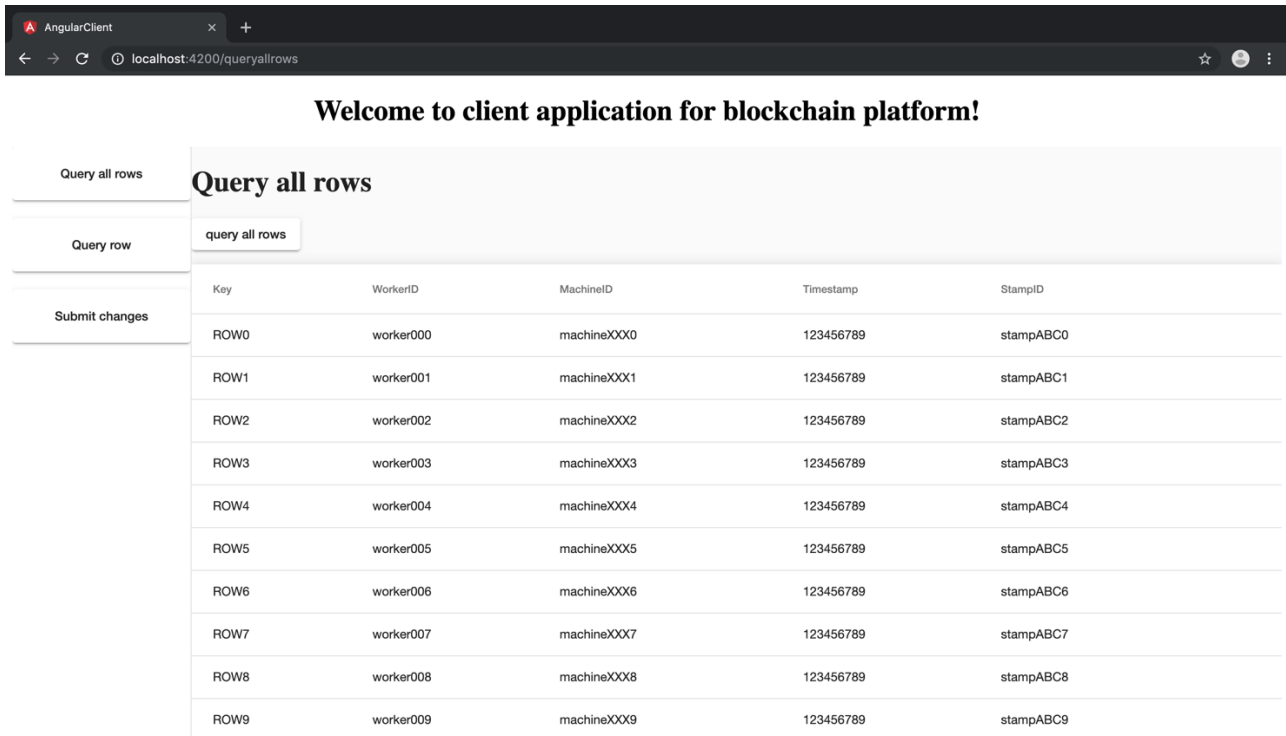
This section of the chapter aims to present the client application built in order to display and update data on the ledger.

Recall that, since the underlying server follows the REST guidelines, many different applications can be built for interacting with it and also many already existing ones can be used. As described in previous section, also an already existing REST client like Postman can be used in order to properly interact with the blockchain. Anyway, a client application more user-friendly has been developed in order to avoid typing manually the URLs or to manually compose the JSON object for the HTTP requests.

The client application has been developed with the use of Angular framework, is written both in Typescript and JavaScript. From a more technical point of view, the application follows a layered approach, fully transparent to end users. In particular, what users see on the web page are components that displays data and relies on a service in order to contact the REST server via HTTP. The developed web-based client application provides functionalities to query the ledger and update existing values, while the creation of new elements on the ledger has already been delegated to the simulator previously described in this chapter and not be reimplemented in the client.

6.5.1. Query all rows function

In this section the view where users can see all the objects present in the ledger is reported. In particular, after the queryAllRows operation, the world state is reported to clients, that is the current value of the objects stored in the ledger.



Key	WorkerID	MachineID	Timestamp	StampID
ROW0	worker000	machineXXX0	123456789	stampABC0
ROW1	worker001	machineXXX1	123456789	stampABC1
ROW2	worker002	machineXXX2	123456789	stampABC2
ROW3	worker003	machineXXX3	123456789	stampABC3
ROW4	worker004	machineXXX4	123456789	stampABC4
ROW5	worker005	machineXXX5	123456789	stampABC5
ROW6	worker006	machineXXX6	123456789	stampABC6
ROW7	worker007	machineXXX7	123456789	stampABC7
ROW8	worker008	machineXXX8	123456789	stampABC8
ROW9	worker009	machineXXX9	123456789	stampABC9

Figure 26 – The client application interface developed with the use of Angular in order to query the ledger

As it is visible from Figure 26, selecting the ‘Query all rows’ tab from the side of the page it is possible to have access to all the objects stored in the ledger. Upon the request the server provides an array of JSON object to the data service of the client application, that return this object to the component which display them on the web page.

6.5.2. Query row function

In this section is reported the view for querying a single object in the ledger given the identifier. Recall that the world state is a key-value pairs storage, so, given the key, the system is easily able to return the associated element.

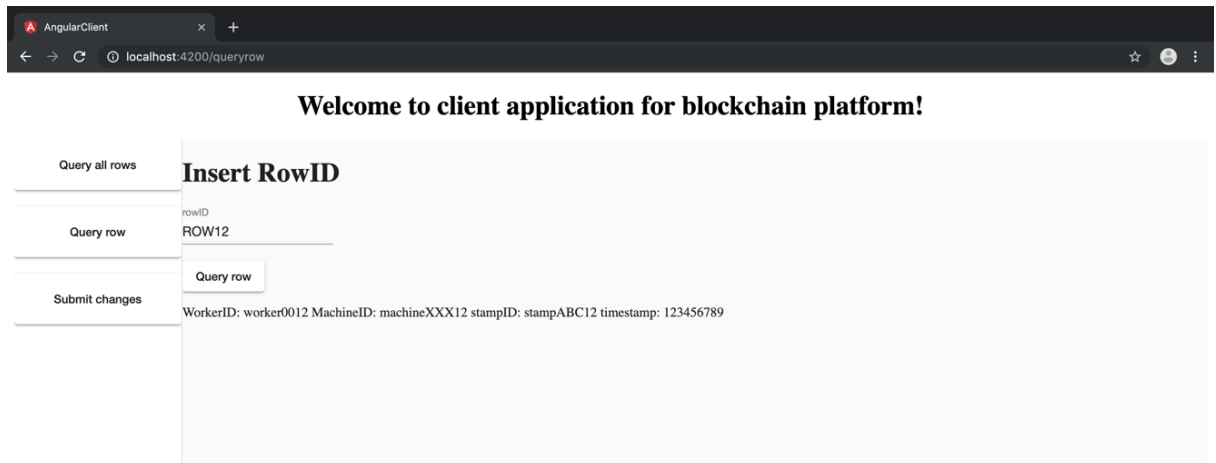


Figure 27 -- The client application interface developed with the use of Angular in order to retrieve one single element

In Figure 27 is reported the interface related to query the ledger and retrieve the information about one single element. In particular, the client is asked to insert the rowID of the desired element and the system will return and print on the web page the associated values. The click on the button triggers a HTTP GET request to the server following the APIs previously described with Swagger. As mentioned above, recall that is not the component that directly interact with the server, but this operation is delegated to a proper service layer in the client application architecture.

6.5.3. Updating the ledger

Last functionality developed for the client application is the update operation. The update operation, in REST guidelines, relies on a HTTP PUT request to the underlying server. In particular, users are asked to select the rowID element on which perform the update operation, the field they want to update and finally insert the new value.

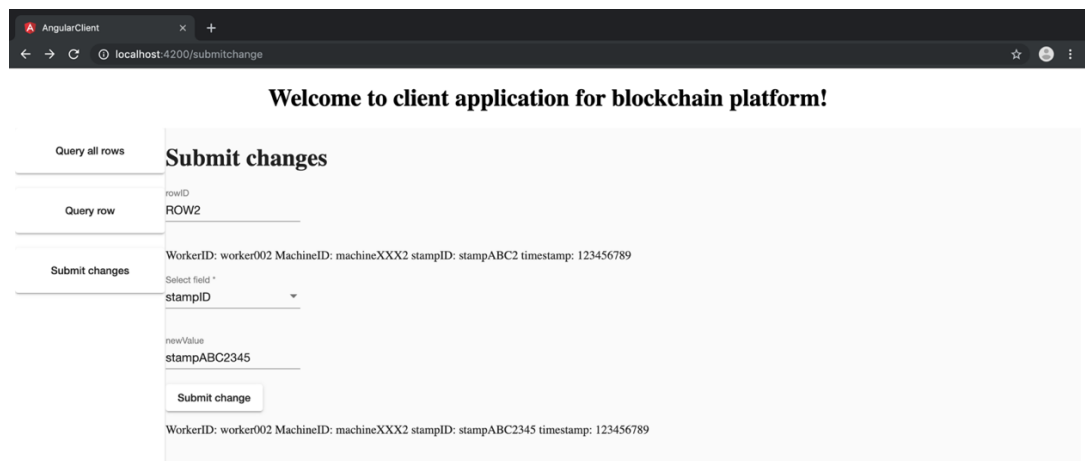


Figure 28 - The client application interface developed with the use of Angular in order to update one element stored into the blockchain

As it is visible from Figure 28, when selecting a rowID object the current values of that associated element are printed below. Then, upon the click on submit change button, the application performs a PUT request on the server, which, as described previously, calls a smart contract's function and then, in order to check the correctness of the operation execution, perform a query for that object. The final result, that is the new version of the element updated is finally printed on the web page. Recall that what is printed client-side is the world state of the ledger. This means that, as explained in chapter 4, when a update occurs, the blockchain component of the ledger records this new event in a new transaction, while the world state of the ledger is updated. So, in conclusion, after an update to the ledger, in the blockchain the old value still exists but when clients query the ledger, only the last version is shown.

7. CONCLUSIONS

This last chapter aims to conclude this thesis, summarizing what has been seen so far, highlighting most important goals achieved and also the limits of the proposed solution.

7.1. Summary

The thesis has been developed during an internship in Avio Aero which provided the use case for the application of a blockchain-based solution. The Digital Stamping use case consists in eliminate paper-based documents during the production phase in order to make more efficient and more secure the whole process, keeping however the advantages of the paper: paper is intuitive, paper is easy and workers are used to working with paper-based worksheets. The blockchain technology applied in supply chains, can make the difference from both a performance point of view and a security perspective. In particular, considering the efficiency, in the use case provided by the company, the throughput with which transactions are generated is not an issue since it is in the order of hours. Instead, from a worker point of view, the digitalization of such process, in particular the last phases of each work, consists in less effort in manually filling worksheets and then taking them to the next workstation. All these actions can be automated and, thanks to the security properties of the blockchain technology, also security features, like tamper-proof data, can be reached.

After the presentation of the use case, a deep analysis has been conducted in order to study the state of the art of such technology, in particular in supply chain management systems and data provenance and data reliability, since these are critical aspects to be considered in designing a blockchain-based application.

After an overview of the blockchain technology and its building blocks, an analysis of the most interesting and used blockchain frameworks is reported. In particular, Ethereum, Hyperledger and AWS proposed solutions have been taken into account during this analysis. Ethereum and Hyperledger are two of the most advanced and mature frameworks in building blockchain-based systems, while AWS has been considered in this analysis because Avio Aero strongly uses AWS services for its software solutions, and it is of the most important and most active AWS customers.

Afterwards, in order to study the feasibility of adopting the blockchain technology for the use case provided by the company, a SWOT matrix analysis has been adopted. During the SWOT analysis, both more technical and managerial aspects were considered. This analysis helped to better understand the strengths and the weaknesses from a high-level point of view of adopting a blockchain-based application.

7.2. Proposed solution

The proposed solution follows a layered approach. At the bottom of the pyramid there is the running blockchain network built with the use of Hyperledger Fabric framework. The proposed blockchain is a permissioned blockchain where users must hold a digital identity (in the form of an electronic certificate released by a membership service provider) in order to correctly join the network. A permissioned environment consists in a network of untrusted, but known, participants. This is the case of supply chains.

On top of the blockchain, in the proposed solution, a simulator was developed in order to populate the blockchain at regular time intervals. The simulator aims to submit a new transaction (representing a new work performed) on the blockchain every x seconds. For the development of the proof of concept for this thesis x has been set to 20, for testing purposes only.

Besides the simulator, that takes the form of a JavaScript file that runs directly on the blockchain network, a REST backend server has been developed in order to enable the interaction between users and the underlying blockchain platform. In this proof of concept, since the creation of new data has been delegated to the simulator, the server provides functionalities for querying the ledger and modify fields of existing objects previously stored in the ledger. A more detailed and in-depth explanation can be found in chapter 5 and 6 of this thesis. Following the layered approach, on top of the REST server, a web-based client application has been developed in order to enable users to interact with the backed server (and as a consequence with the underlying blockchain). As for the server, also the client application provides functionality both to query the ledger and to update it.

7.3. Final considerations and future works

Even if not necessary, the blockchain network developed for this thesis, as shown in chapter 5, is composed of two organizations and two peer nodes per organization. This decision was made in order to better study the Hyperledger Fabric network and to evaluate a more advanced configuration for the blockchain network. In the context of network configuration is where one possible limit of the proposed solution can be found from an end customers point of view. In the use case provided by the company, data stored in the blockchain network are only related to the company itself and to its own production cycle, without considering all involved actors in the supply chain of that produced components. Since all the nodes composing the network are owned by one single company, the entire blockchain ecosystem potentially can be compromised and, as a consequence, data stored in that system are compromised too. Anyway, in future work, after seeing the blockchain at work in one single company, it is expected to extend this system to other involved actors in the Avio Aero supply chain. In addition, in future works, this proposed solution can be applied to all the production lines in the company.

Bibliography

1. <https://www.ethereum.org>
2. <https://hyperledger.github.io>
3. <https://aws.amazon.com/it/>
4. <https://javascript.info>
5. <https://golang.org>
6. <https://expressjs.com>
7. <https://angular.io>
8. <https://nodejs.org/en>
9. Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).
10. <https://www.investopedia.com/news/top-5-cryptocurrencies-market-cap>
11. Caro, Miguel Pincheira, et al. "Blockchain-based traceability in Agri-Food supply chain management: A practical implementation." *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*. IEEE, 2018.
12. TongKe, Fan. "Smart agriculture based on cloud computing and IOT." *Journal of Convergence Information Technology* 8.2 (2013).
13. Aung, Myo Min, and Yoon Seok Chang. "Traceability in a food supply chain: Safety and quality perspectives." *Food control* 39 (2014): 172-184.
14. Jing, Xiao, Liu Ziyu, and Li Beiwei. "Research on a food supply chain traceability management system based on RFID." *Journal of Agricultural Mechanization Research* 2 (2012): 41.
15. Tian, Feng. "A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things." *2017 International Conference on Service Systems and Service Management*. IEEE, 2017.
16. McConaghy, Trent, et al. "BigchainDB: a scalable blockchain database." *white paper, BigChainDB* (2016).
17. Lu, Qinghua, and Xiwei Xu. "Adaptable blockchain-based systems: A case study for product traceability." *IEEE Software* 34.6 (2017): 21-27.
18. Abeyratne, Saveen A., and Radmehr P. Monfared. "Blockchain ready manufacturing supply chain using distributed ledger." (2016).
19. Korpela, Kari, Jukka Hallikas, and Tomi Dahlberg. "Digital supply chain transformation toward blockchain integration." *proceedings of the 50th Hawaii international conference on system sciences*. 2017.
20. Weber, Ingo, et al. "Untrusted business process monitoring and execution using blockchain." *International Conference on Business Process Management*. Springer, Cham, 2016.
21. Snow, Paul, et al. "Business processes secured by immutable audit trails on the blockchain." (2014).
22. Ramachandran, Aravind, and Dr Kantarcioglu. "Using blockchain and smart contracts for secure data provenance management." *arXiv preprint arXiv:1709.10000* (2017).
23. Moreau, Luc, et al. "The open provenance model core specification (v1. 1)." *Future generation computer systems* 27.6 (2011): 743-756.
24. <https://www.meteor.com>
25. <https://www.mongodb.com>
26. <https://geth.ethereum.org/downloads/>
27. Zyskind, Guy, and Oz Nathan. "Decentralizing privacy: Using blockchain to protect personal data." *2015 IEEE Security and Privacy Workshops*. IEEE, 2015.
28. Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." *International Workshop on Peer-to-Peer Systems*. Springer, Berlin, Heidelberg, 2002.
29. <https://github.com/google/leveldb>
30. Liang, Xueping, et al. "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability." *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing*. IEEE Press, 2017.

31. <https://owncloud.org>
32. <https://tierion.com>
33. <https://chainpoint.org>
34. Bastiaan, Martijn. "Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin." *Available at http://refraat.cs.utwente.nl/conference/22/paper/7473/preventingthe-51-attack-a-stochasticanalysis-of-two-phase-proof-of-work-in-bitcoin.pdf*. 2015.
35. <https://www.pavia-ansaldo.it/blockchain-e-cyberattacks-la-blockchain-e-una-tecnologia-sicura-e-a-prova-di-hacker/>
36. https://en.bitcoin.it/wiki/July_2015_flood_attack
37. <https://ethereum.github.io/yellowpaper/paper.pdf>
38. <https://www.linuxfoundation.org>
39. <https://www.hyperledger.org/projects/iroha>
40. <https://www.postgresql.org>
41. <https://www.hyperledger.org/projects/sawtooth>
42. <https://www.hyperledger.org/projects/hyperledger-burrow>
43. <https://hyperledger-fabric.readthedocs.io/en/latest/>
44. <https://aws.amazon.com/it/managed-blockchain/>
45. <https://aws.amazon.com/it/qlldb/>
46. <https://camunda.com/download/modeler/>
47. Pavlović, Ivan, Tomaž Kern, and Damijan Miklavčič. "Comparison of paper-based and electronic data collection process in clinical trials: costs simulation study." *Contemporary clinical trials* 30.4 (2009): 300-316.
48. https://www.senato.it/application/xmanager/projects/leg17/attachments/dossier/file_internets/000/002/240/documento_conclusivo_lavoro_4.0.pdf
49. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
50. <https://www.typescriptlang.org>
51. <http://staruml.io>
52. <https://github.com/hyperledger/fabric-samples>
53. <https://swagger.io>
54. <https://www.getpostman.com>