DATA RELIABILITY IN A BLOCKCHAIN TRACEABILITY NETWORK

Eng. CRISTIAN HEIDELBERG VALENCIA PAYÁN, Msc

Doctoral Thesis

Ph.D. in Telematics Engineering

Advisor:

Eng. Juan Carlos Corrales Muñoz, PhD

Universidad del Cauca

Faculty of Electronic Engineering and Telecommunications

Telematics Department

Popayán, 2022

Eng. CRISTIAN HEIDELBERG VALENCIA PAYÁN, Msc

DATA RELIABILITY IN A BLOCKCHAIN TRACEABILITY NETWORK

Thesis presented to the Faculty of Electronic Engineering and Telecommunications of the University of Cauca to obtain the title of

Ph.D. in Telematics Engineering

Advisor:

Eng. Juan Carlos Corrales Muñoz, PhD

Popayán

2022

My gratitude to God for allowing me to complete this stage of academic, professional, and personal development in the best feasible way.

I would like to thank my parents, brothers, colleagues, and friends for their support.

Dr. Juan Carlos Corrales, my tutor, was a constant source of support and contribution, and each person who participated in one way or another contributed to the route was appreciated.

Dr. David Griol, by his guidance during the last stage of this doctoral research development.

Finally, I would like to thank the University of Cauca, especially the Faculty of Electronic Engineering and Telecommunications, for providing an environment conducive to this training.

My sincere thanks go out to all of you.

Structured Abstract

Background: Traceability is the ability to identify and track the history, distribution, location, and application of products, parts and materials of a final product, so that reliability is ensured. Blockchain technology has come to significantly transform information management in the different branches where it has been applied from health care systems, cryptocurrencies, and food traceability, among others. Blockchain technology could also present new threats to traceability processes. For this reason, data reliability, human errors, and the reliability of sensor-generated measurements, must be considered.

Goals: this research will focus on data reliability before being appended to a new block in a Blockchain traceability network. So, to validate the sensed data at the time of transaction arrival in a Blockchain traceability network, this work proposes using data analysis and machine learning (ML) algorithms in conjunction with smart contracts.

Methods: The generation of semantic anomaly detection rules from machine learning models is proposed. Using historical data from the Blockchain network deployed, the developed tool can generate an anomalies classification model based on the Z-Score and Decision Tree. From this model, a function can be built in JavaScript and GO to detect semantic anomalies in incoming transaction data that using a smart contract. Additionally, the smart contract can detect syntaxis errors.

Results: A set of data collected from two production sites as part of a pilot test. Autonomous anomaly detection tool for smart contracts. An autonomous deployment tool for blockchain test networks. An anomaly detection strategy for blockchain-based traceability schemes.

Conclusions: We manage to develop a strategy to give a smart contract the ability to detect syntaxis and semantics anomalies in the data of Blockchain transactions in a traceability scheme. This strategy can translate Random Forest and Decision Tree rules into Go and JavaScript for smart contracts supported by Hyperledger Fabric. This strategy proves useful in traceability scheme in supply chains were most of the traced data is generated autonomously. Some of those data might be corrupted during transmission from the sensors to the database or during manual data conversion processes.

Keywords: Blockchain, Smart Contracts, Anomaly Detection, Reliability, Traceability

Resumen Estructurado

Antecedentes: Trazabilidad es la capacidad de identificar y rastrear el historial, la distribución, la ubicación, la aplicación de productos, piezas y materiales de un producto final, de modo que se garantice la confiabilidad. La tecnología Blockchain ha llegado a transformar significativamente la gestión de la información en las diferentes ramas donde se ha aplicado desde sistemas de salud, criptomonedas, trazabilidad de alimentos entre otros. La tecnología Blockchain también podría presentar nuevas amenazas a los procesos de trazabilidad. Por esta razón, se debe considerar la confiabilidad de los datos, los errores humanos y la confiabilidad de las mediciones generadas por sensores .

Objetivos: Esta investigación se centrará en la fiabilidad de los datos antes de ser anexada a un nuevo bloque en una red de trazabilidad Blockchain. Por lo tanto, para validar los datos detectados en el momento de la llegada de la transacción en una red de trazabilidad Blockchain, este trabajo propone el uso de algoritmos de análisis de datos y aprendizaje automático junto con contratos inteligentes.

Métodos: Se propone la generación de reglas de detección de anomalías semánticas a partir de modelos de machine learning. Utilizando datos históricos de la red Blockchain desplegada, la herramienta desarrollada puede generar un modelo de clasificación de anomalías basado en el Z-Score y el Árbol de Decisión. A partir de este modelo, se puede construir una función en JavaScript y GO para detectar anomalías semánticas en los datos de transacciones entrantes que serán utilizados por el contrato inteligente. Además, el contrato inteligente es capaz de detectar errores de sintaxis.

Resultados: Conjunto de datos recogidos de dos centros de producción como parte de una prueba piloto de desarrollo. Herramienta autónoma de detección de anomalías para contratos inteligentes. Herramienta de despliegue autónomo para redes de prueba de blockchain. Una estrategia de detección de anomalías para esquemas de trazabilidad basados en blockchain.

Conclusiones: Logramos desarrollar una estrategia para dar a un contrato inteligente la capacidad de detectar anomalías de sintaxis y semánticas. Esta estrategia es capaz de traducir las reglas de dos algoritmos de aprendizaje máquina para su uso en contratos inteligentes. Esta estrategia demuestra ser útil en el esquema de trazabilidad en las cadenas de suministro donde la mayoría de los datos rastreados se generan de forma autónoma y pueden estar dañados durante la transmisión o durante los procesos manuales de conversión de datos.

Palabras clave: Blockchain, Contratos inteligentes, Detección de anomalías, Confiabilidad, Trazabilidad

Content

Tables

Figures

Equations

Chapter 1

Introduction

1.1. Problem Approach

Traceability is the ability to identify and track the history, distribution, location, application of a final product's products, parts and materials to ensure reliability (United Nations Global Compact & Business for Social Responsibility, 2014). This definition can suffer variations depending on the field of application. In recent years it has gained significant importance in supply chains, regardless of the type of product that needs to be traced (Haleem et al., 2019). Traceability arose in the 1930s when in some countries of the European region it was wanted to have proof of the origin of high-quality products such as French champagne (Setboonsarng et al., 2009). Its use has been growing due to the increasing quality control conducted by both consumers and the public sector to improve the safety and reliability of the products marketed.

Traceability has algo been defined as an organization's ability to determine its provenance, visibility in terms of what it is doing both upstream and downstream in the supply chain, and transparency in terms of how it is communicating its upstream activities and products to the public by (Sodhi & Tang, 2019). A supply chain is the network comprised by all resources, organizations, and activities involved in the creation and/or sale of a product. Traceability is a key element in Supply Chain Management (SCM), a goal of SCM is to improve efficiency by coordinating the efforts of the various entities in the supply chain. This can result in a company achieving a competitive advantage over its rivals and enhancing the quality of the products or services.

Blockchain technology has come to significantly transform information management in the different branches where it has been applied, from healthcare systems (Griggs et al., 2018) to cryptocurrencies (Scott, 2016) and food traceability (Tian, 2016) among others. Blockchain emerged as an innovative technology in 2008 as part of the Bitcoin cryptocurrency (Bhardwaj & Kaushik, 2018). This technology is based on a distributed ledger of financial accounting. On this ledger, every new piece of information is stored in a block and cannot be modified or eliminated from the block containing it. Its use has expanded rapidly to other areas due to the advantages that this technology offers, such as transparency in the management of information, its decentralization, its security, and the ability to be auditable at any time, among others (Zheng et al., 2018), and the increasing use of smart contracts in Blockchain networks.

However, this technology also presents several challenges, including the scalability of the network, the cost of network transactions, reliability, and security failure. Reliability has been focused on how reliable the information on the network or the source of information is, but not on the report before being entered into the network (D. Liu et al., 2019; M. Wang et al., 2018; Xiang et al., 2019). Though, these processes have a basis that the information collected does not present any anomaly. Nevertheless, network information must be dependable from the moment it has been appended.

Blockchain technology could also introduce new threats to traceability processes, as authors (Grover & Sharma, 2016) mention, and third-party interactions with the network. For this reason, data reliability, human errors, and the reliability of sensor-generated measurements, due to the lack of mutual trust mechanism in the data collection and transmission processes, must give place to multiple security threats (J. Wang et al., 2020) be checked. Even if the data recorded on the Blockchain is secure and tamper-resistant, the chance of an altered input must be considered (Sheldon, 2021). Hence, this research will focus on data reliability before being appended to a new block in a Blockchain traceability network.

But what can Blockchain technology brings to traceability to improve an already tested and accepted process? In recent years, it has gained significant importance in supply chains, regardless of the type of product that needs to be traced, due to the increasing quality control conducted by both consumers and the public sector to improve product safety and reliability. Nevertheless, the information is centered, and traceable partners may not have access to certain information about the product in real time. Due to Blockchain technology's distributed feature, all transactions are stored in every node of the Blockchain network and are available at any time, making them transparent, secure, and auditable (Zheng et al., 2018).

Smart Contracts are used to validate transactions on the Blockchain Network. Once deployed on the Blockchain network, these contracts are software that runs autonomously (Sánchez et al., 2018) when a series of agreed-upon conditions are met. Since their implementation is done over a Blockchain network, Smart Contracts are immutable. They can be reviewed entirely, so they can be used to validate transactional data related to supply chain traceability. Organizations like Ethereum have proposed standards for the packaging of Smart Contracts to facilitate the reuse of contracts developed and have also made efforts to offer standards throughout the construction of the Blockchain network(*EIP-190: Ethereum Smart Contract Packaging Standard*, n.d.)

To validate the sensed data at the time of transaction arrival in a Blockchain traceability network, this work proposes using data analysis and machine learning

(ML) algorithms in conjunction with smart contracts. In this way, an almost resilient anomaly data detection will be ensured.

1.2. Motivation

The rise of Industry 4.0 must be considered a challenge for traceability in supply chains due to the supply chain's further segmentation. Nowadays, individual businesses cannot compete while being independent; they must participate as members of a global supply chain (GSC) (Ben-Daya et al., 2019; Koberg & Longoni, 2019). However, the ever-changing environment made the supply chains vulnerable at many levels, given that the traceability of a product must be done carefully to avoid future problems.

Recently there has been an increase in the use of Blockchain technology for multiple purposes; one of them has been traceability. This technology has increased quality control, safety, and reliability. So, the producers are looking for better ways to trace the products at any chain state to ensure their quality. To survive in the current and complex supply chain environment, the business must be agile, with high resilience and risk mitigation (Koberg & Longoni, 2019). The barriers most considered when implementing sustainable supply chains are higher costs, coordination and complex effort, and insufficient or lack of communication between the different actors in the chain (Seuring & Müller, 2008).

Using Blockchain technology, we could improve the traceability of the processes in multiple environments so that users, both consumers, and businesses, have a greater capacity to respond to the different events that may occur in the production and sale process and increase the scope of business by offering reliable methods regardless of the process using smart contracts with data validations and even include management recommendations based on expert knowledge.

In the above context, in this doctoral thesis, we proposed a strategy to provide a smart contract with the tools to perform a certain degree of data validation to ensure that the data related to a traced product is reliable and that no problem has been detected during its passage through the supply chain. A smart contract with data validation will contribute to the sustainability on a supply chain, specifically to the traceability of products in favor of potential protection for consumers and traceability partners, considering that to reduce the uncertainty and ensure the chain’s sustainability, it is pertinent to strengthen the quality of the relations between the participants.

1.3. Objectives

1.3.1 General Objective

Autonomous anomaly detection on a Blockchain traceability network transaction through smart contracts.

1.3.2 Specific Objectives

1. To characterize the anomalies in a traceability network.
2. To propose an anomaly detection strategy.
3. To update smart contracts in real-time according to the anomaly detected.
4. To validate the autonomous anomaly detection in a simulated environment.

1.4. Contributions

In this doctoral thesis, the following main contributions are made:

- A dataset with all the data collected in two production sites as part of a pilot test of the development.
- A tool to detect anomalies in a data store on a Hyperledger Fabric Blockchain network.
- A tool to autonomously update smart contract rules for anomaly detection.
- A tool to autonomously deploy a test network using Hyperledger Fabric and deploy a smart base contract.
- A strategy to perform data anomaly detection in a Blockchain traceability scheme.
- **Article**: **A Smart Contract for Coffee Transport and Storage with Data Validation** published in the journal **IEEE Access**, vol. 10, pp. 37857-37869, 2022. (Qualified as A1 by MinCiencias Publindex – SJR Q1)
- **Article**: **Smart Contract to Traceability of Food Social Selling** accepted to be published in the journal CMC-Computers, Materials & Continua ISSN:1546-2226. (Qualified as A1 by MinCiencias Publindex – SJR Q2)
- **Article**: **Blockchain for Supply Chain Traceability with Data Validation** presented and to be published in the 17th International Conference on Soft Computing Models in Industrial and Environmental Applications SOCO'22.

1.5. Contents of the Dissertation

This document is divided as follows.

Chapter I describes the problem approach, the primary motivation, general and specific objectives, and the research contributions.

Chapter II presents the base concepts for this development such as Blockchain, traceability, and more, also the different data anomalies that may arise in a supply chain traceability scheme. Finally, a state-of-the-art study about data reliability in a Blockchain-based traceability scheme, with their contributions and gaps.

Chapter III describes the developed strategy for syntaxis anomaly detection using Smart Contracts, network selection, and deployment. Finally, the Smart Contract test using different approaches, including a production site deployment.

Chapter IV contains the development and test of the proposed smart contract for semantic anomaly detection, the ML model training, and the usage of the anomaly detection rules on the smart contract. Also, the autonomous Smart Contract anomaly detection rules process is described and tested using production site data.

Chapter V presents the conclusions and future work based on the experience gained through the development of this doctoral research.

Chapter 2

Background

This project has a comprehensive documentary reference, which allows for forming the necessary knowledge base for the development of the project and its orientation in what it seeks to achieve: Blockchain, Smart Contracts, and Data Reliability, framed within the concept of Traceability. The associated concepts are found in section 2.1, in which the techniques used in the topics mentioned above are presented individually. Finally, section 2.2 presents the related works corresponding to each main topic.**Key Concepts**

2.1.1 Traceability

Traceability of a product is the ability to identify a product at any stage of the supply chain or follow the historical process, application, or location of a product according to ISO 9000:2015 (ISO 9000 Sistemas de Gestión de La Calidad — Fundamentos y Vocabulario, 2015). The identification and traceability of a product is the primary form of traceability alongside other comprehensive product information components and activity records that can only be accessed if the product is traced (Florkowski et al., 2009).

Traceability can be simple and inexpensive through a "One-up One-Down" implementation, i.e., immediately knowing who the supplier is and to whom the product is targeted. Or it may be a little more complex depending on the level of information you want to have about the product (Karlsen & Olsen, 2016).

The GS1 group (GS1, 2020) defined the standard procedure for traceability to be used across industries. In this standard every traceability partner will report the product information following a set of rules for the labeling and description of the product.

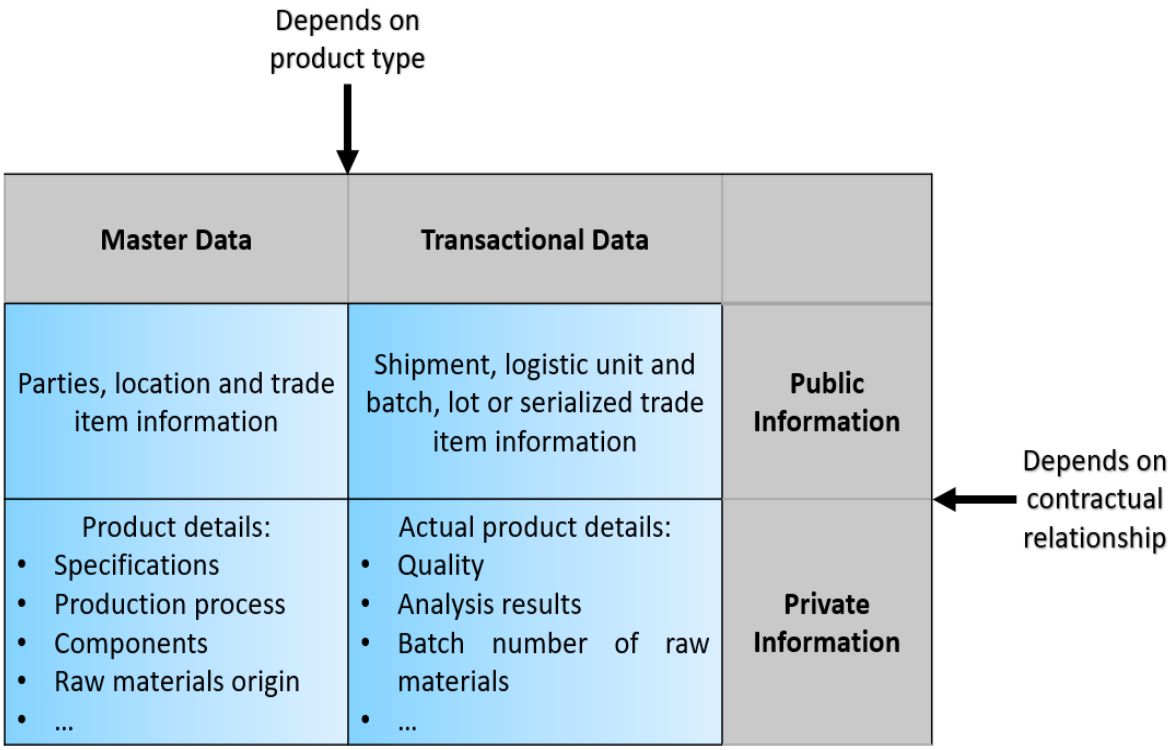


Figure 1. Traceability Matrix adapted from (GS1, 2012)

Figure 1 shows the traceability matrix with some examples of the information related to each stage and how to classify it on a public or private level. Public information will be available to all kind of consumers while confidential information will be available only to the publisher itself and control entities. As can be seen traceability, is a chain of information related to a product and its history across manufacturing process. The GS1 has also been working in the development of a traceability standard for the Blockchain technology inclusion (GS1, n.d.).

2.1.2 Blockchain

Blockchain or distributed secure ledger (Pal et al., 2019) is a technology that has had a great boom in recent years, thanks to the possibility of not having the intervention of third parties to validate transactions in peer-to-peer networks or P2P. This technology arises in 2008 as part of the Bitcoin currency proposed by Satoshi Nakamoto (Bhardwaj & Kaushik, 2018). The most significant advantage of this currency with Blockchain is that it does not require a central authority to authenticate, control or validate transactions. Due to its distributed feature all the transactions are stored in every node of the Blockchain Network and are available at any time making them transparent, secure, and auditable. These features made Blockchain technology a vital and advisable companion to processes like traceability, monetary transactions and more without the need for a third party involved

This technology is still in development. However, it is possible to find examples of use in multiple fields such as the food industry, IoT devices, banking network, etc. The block network or Blockchain consists of multiple nodes or points that maintain a partial or full copy of all transactions performed over the network. Most network members must validate these transactions before being entered into the network with their respective time tags and verify that the block to be created contains a valid transaction and refers to the immediately preceding block (Azzi et al., 2019).

To support and operate with this technology each network pair has to provide the following basic functionalities: routing, storage, wallet, and mining services (Reyna et al., 2018). According to the Blockchain peer or node's capabilities, it will be classified into one of the different node types.

Table 1. Basic node types and functionality (Reyna et al., 2018)

Wallet	Storage	Mining	Router	Node Type
X	*X*	*X*	*X*	**Core**
	X		*X*	**Full**
	X	*X*	*X*	**Miner Only**
X			*X*	**Light Wallet**

Basic node functions used in BlockChain

Table 1 shows that the routing function is present on each node, this because it is required to be able to participate in the P2Pnetwork. The storage function is used to keep a copy of the BlockChain network on the node. The function of the wallet is to provide security keys for network users to transact. Finally, the mining function is responsible for the creation of new blocks in the network (Reyna et al., 2018). A Core node has all the functionalities offered by the BlockChain network; a Full node only store the data of the BlockChain network. A Miner Only node perform mining activities and store the data of BlockChain network. Finally, a Light Wallet node obtains all the information needed about the payments in cryptocurrencies without downloading all the BlockChain network.

Blockchain technology can currently be used in multiple business models; the most common is the ***Utility Token*** model, where tokens or symbolic objects are used to obtain functionality on the network. ***Blockchain as a Service (BaaS),*** where a third party offers a specific client service while keeping functions such as cloud management and server deployment. ***Security*** is the most recent; it is sought that the tokens represent a legal property or a physical or digital object. ***Development Platform*** for this case, the technology is used for the development of applications known as decentralized applications (Dapps). ***Blockchain software-based products*** where the developers sell or rent their software solutions (Dapps) to thirds. The Fee Charge model charges a fee for using the

Blockchain network to end users or for using Dapps. Finally, the **Professional Services** model is where Companies offer their services for developing applications, consultancies, audits, etc., to companies starting or showing interest in using Blockchain.

Given the characteristics of the Blockchain network, it is ideal for traceability processes since the information stored in the network cannot be altered. It can be determined with certainty when an alteration was made in the knowledge of the traced product and what was the original information of the product. It also always allows access to the product information available to all network members, keeping the public and private levels of knowledge. This technology will also raise the level of trust in the information in the traceability network. Adding smart contracts to the traceability process will help to ensure that the changes or updates on the product information will be made by those who are authorized to and will be registered on the Blockchain when and who made the change or the update. Also, it could be used to validate the product information before being annexed to a new Blockchain block.

2.1.3 Smart Contracts

Nick Szabo initially coined this term in the 1990s (Szabo, 1996). However, its use began with the introduction of Blockchain technology in 2008. A smart contract is software that, once deployed on the Blockchain network, runs autonomously (Sánchez et al., 2018) when a series of conditions agreed upon by the parties that make up the network are met.

Since their implementation is done over a Blockchain network, smart contracts are immutable and can be reviewed in their entirety. Smart contracts are stored on the network and can be used by anyone who is part of this. However, these cannot be modified by their creators or network administrators. So, before its implementation and deployment, the parties involved must fully agree on the contract. The nodes of the Blockchain network that have the mining function are responsible for executing these contracts.

Currently, Blockchain platforms support smart contracts execution due to the multiple possible applications like the Internet of Things, E-commerce, and more.

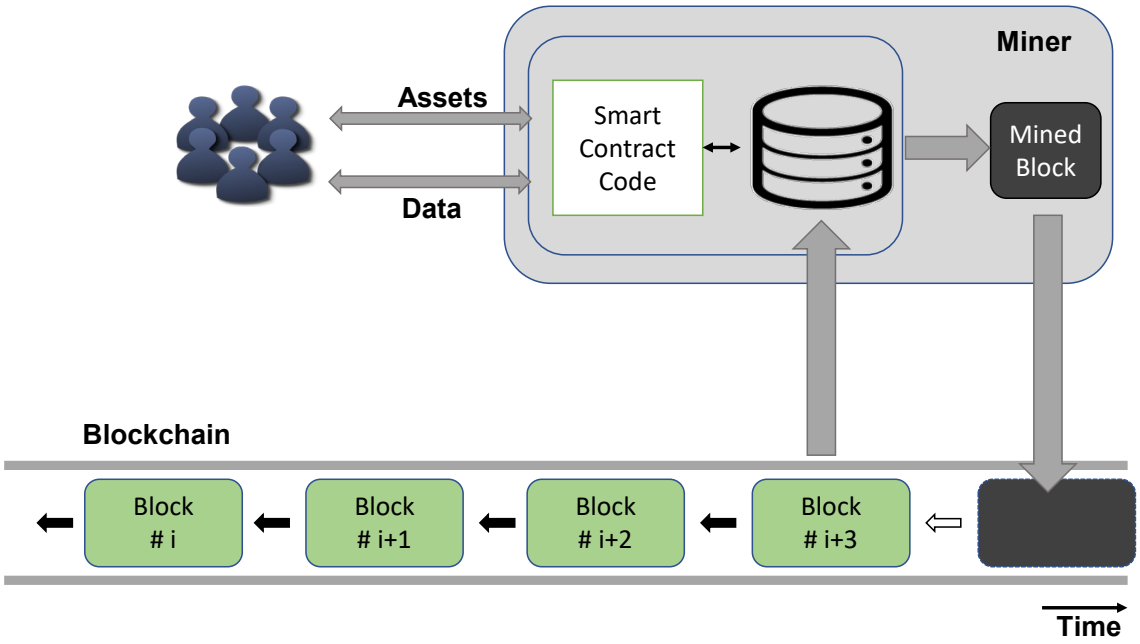


Figure 2. Smart contracts system, based on (Alharby & Moorsel, 2017).

Figure 2 shows the smart contracts system with account information to assign assets to the user and recollect data, private storage, and an executable code. The smart contract is executed by a Miner that stores the transaction data on a new block on the Blockchain. Finally, smart contracts have access to the information generated on the Blockchain.

A smart contract is a computer program that considers all the parties agreed-upon conditions. When it is deployed on the Blockchain network, it can be called by any user with access permissions and changes a set of state variables on the network using the transaction information. According to definition 12 on (Hu et al., 2020), the smart contract needs to satisfy the following equation.

$$C(S_i, Tx_i) = (S_j, R_j)$$

Equation 1. Smart contract definition

Where

$$S = \{S_{i \in \mathbb{N}^*}\}$$

Equation 2. States on the contract

Equation 2 represents all the possible states in the smart contract C.

$$T = \{tx_{i \in \mathbb{N}^*} = (t, in, out, s, pld)_{i \in \mathbb{N}^*}\}$$

Equation 3. Transaction on the network

Equation 3 exposes a transaction on the Blockchain network where *t* is the transaction's timestamp, *in* represent the data sent, *out* represent the transaction response, *s* represents the transaction signature and *pld* is the payload data.

$$R = \{R_{i \in \mathbb{N}^*}\}$$

Equation 4. Smart contract response

Equation 4 represent the possible responses that the smart contract will give. If a valid transaction calls the contract, the smart contract will produce a new state S_j and a corresponding response.

2.1.4 Data Analysis

Statistical and logical techniques systematically describe, illustrate, condense, and evaluate datasets (*Data Analysis*, 2013). Data analysis involves the processes of inspection, cleaning, transformation, and modeling of a dataset to find helpful information that allows one to draw conclusions and support decision-making processes. This process also involves verifying or validating the information contained in the database for reliability purposes.

In the stages of inspection and cleaning the data, we can identify problems that can be corrected or allowed so that the developed models can detect these errors and alert the inconsistent data found in future applications.

Among the methods that can be used in the modeling stage are rule-based models (Seines et al., 1998) and ML algorithms. Among these, we have the Neural Networks (Jordan & Bishop, 2004) and Decision Trees (Loh, 2011) among others, which also have methods that improve the results obtained by the applied models. Techniques such as Bagging (Breiman, 1996), Boosting (Skurichina & Duin, 2002), Incremental Learning (Xie et al., 2009), and others.

- **Data Reliability**

Data reliability is a crucial foundation when building data on an interesting topic. It means that the data is sufficiently complete and accurate (U.S. Government Accountability Office, 2019). It is related to the quality of the captured data. Reliability can also be expressed as the number of failures over time (Romeu, 2003).

2.1.5 Data Anomaly

An anomaly is an observation different from what is usual or expected. These observations have inspired scientists to reconsider or modify theories or hypotheses, so they play a crucial role as observations that are inconsistent with the model in the academic paradigm (Foorthuis, 2021). Identifying,

understanding, and predicting data anomalies is a crucial pillar of data mining (Chandola et al., 2016).

2.1.6 Anomaly Detection

The problem of anomaly detection refers to finding anomalies in data. In different application domains, "anomalies" may also be known as outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants (Chandola et al., 2016). This has become a field of interest in data mining in multiple application domains such as credit card fraud detection, faults diagnosis in industry, or intrusion detection in cyber security. Also, anomaly detection can improve the understanding of the studied system.

Currently, multiple techniques to perform anomaly detection can be split into two groups: Point Anomaly Detection and Anomaly Detection for Complex Data. Next, we will discuss these groups, their approaches, and their most common techniques.

A. Point Anomaly Detection

In this group, the goal is to identify points in a dataset that do not behave as average points do. Usually, there is no previous knowledge about a data point's average or abnormal behavior. There are multiple methods to perform this task; some of them are presented next.

- **Statistical Method**

  This is the first and most studied method to detect anomalies. It considers that if the difference between the data point and the statistical distribution is more significant than a particular value or range, then that data point is an anomaly (H. S. Wu, 2017). Statistical methods also estimate a parametric or nonparametric model from the dataset and apply a statistical test on each data point to assign an anomaly score based on the probability of it being an anomaly. These methods are effective is the standard data points can be modeled by a statistical distribution (Chandola et al., 2016).

- **Clustering Method**

  Anomaly detection based on data clusters use current clustering algorithms such as DBSCAN, K-Means, Gaussian Mixture Model, etc., to learn sets from a given dataset, and a point that cannot be clustered is considered an anomaly. Nevertheless, these algorithms are not developed to detect anomalies, but as a subproduct of its clustering process, the researchers can find abnormal points on the dataset (Chandola et al., 2016; H. S. Wu, 2017).

- **Distance Method**

This method estimates the distance between data points in data space using a distance function selected by the researcher. The greater the distance, the greater the probability that the issue is an anomaly in the data set. The critical assumption is that average points lie in dense clusters so that distances will be shorter and anomalous points lie in sparse groups, so distances will be more considerable (Chandola et al., 2016; H. S. Wu, 2017).

- **Classification Method**

In this method, the ML algorithm learns how to separate anomalies from average data points using a labeled or unlabeled dataset. Based on the available training data, the assumption is that one can learn a classifier in the given feature space to distinguish between normal and anomalous points (Chandola et al., 2016). Some algorithms used for this are Random Forest (Breiman, 2001), Neural Networks (Stergiou & Siganos, n.d.), and Decision Tree (Loh, 2011), among others.

B. Anomaly Detection for Complex Data

In some cases, data points are related, so anomalies can only be found when analyzing the relationship between data points in a dataset. Next, we will review methods developed to consider these cases and detect anomalies within complex data.

- **Kernel-based Methods**

Analogous to point-based anomaly detection techniques, these techniques analyze the entire test sequence as a unit. They typically apply proximity-based point anomaly detection techniques by defining an appropriate similarity kernel for the sequences (Chandola et al., 2016).

- **Window-based Methods**

Within the test sequence, these techniques analyze a short subsequence of symbols. Subsequences within test sequences are treated as unit elements for these techniques. By analyzing the subsequences within a test sequence, these techniques can determine whether the whole test sequence is anomalous (Chandola et al., 2016).

- **Time Series Methods**

These methods work by forecasting the following observation in the time series, using the statistical model and the previous comments, and comparing them with the actual word to detect anomalies (Chandola et al.,

2016). These methods manage univariate time series but can also be extrapolated to multivariate ones.

2.1.7 Anomalies in Supply Chain Traceability

Anomalies in a supply chain traceability can occur due to multiple sources because supply chain management involves various actors and processes. So, every actor must be accountable for explaining or justifying their data or actions in the supply chain (Khalfaoui et al., 2013). They must be able to detect anomalies in their processes and perform the necessary steps to avoid those future anomalies. An anomaly along the supply chain, such as temperature fluctuations, can affect population safety and the environment. Supply chain actors often ignore anomalies for economic reasons, or there is no clear path to deal with the anomaly (Khalfaoui et al., 2013). Therefore, traceability has gained much popularity in supply chains. With the explosion of IoT sensors, increasingly wireless technology has been used in recent implementations (M. Chen et al., 2017), allowing for easier anomaly detection and tracking of products.

IoT sensor use has made problems like supply chain infiltrations, retail service copycatting, and factory overruns (D'Amato & Papadimitriou, 2013) harder, but they can still happen. These problems often introduce anomalous data into the supply chain traceable process. Although wireless sensors have improved the amount and reliability of traced data, they are also susceptible to security issues (Grover & Sharma, 2016; Tomić & McCann, 2017). Security issues are one of the reasons behind anomalous data in the supply chain traceability (Jan et al., 2019). Considering this, the anomalies typically found in supply chain traceability are based on the problem description made by the following authors (Baumgartner Data et al., 2021; Beteto et al., 2022; M. Chen et al., 2017; Jindal et al., 2020; Konovalenko & Ludwig, 2022; N. Liu et al., 2022; Masciari, 2012; Tsolaki et al., 2022; Zunic et al., 2020), are described next.

- **Missing data anomaly**

  This type of anomaly occurs when no data value is stored for a variable in observation. So, no information was provided, the information was lost during the transmission process, mistakes were made if the information was manually recollected, security issues, etc. Missing data anomaly could be at random or not at random, depending on the issue producing the missing data anomaly

Table 2. Missing data anomaly

Variable 1	Variable 2	Variable 3	Variable 4
15.26	Complete	1500	Good
16.25	Incomplete		Good
28.24		1600	Bad

Example of missing data anomaly

Table 2 shows an example of what a missing data anomaly would look like, Variable 2 and variable 3 presents white spaces on instance 2 and 3, respectively. The terms NULL or NAN also represents sometimes

- **Unknown data anomaly**

This type of anomaly occurs when the data stream has values for an unknown variable or a variable that has not been seen before. The information provided does not correspond with any of the expected variables under the expected normal conditions.

Table 3. Unknown data anomaly

Variable 1	Variable 2	Variable 3	
15.26	Complete	1500	Good
28.24	Complete	1600	Bad

Example of unknown data anomaly

Table 3 shows an example of a unknown data anomaly, in this case we can see a nameless variable. A solution to solve this issue the system uses a predefined variable name or to ignore the data related to a unknown data anomaly.

- **Duplicate data anomaly**

This type of anomaly occurs when the data stream has identical values for two or more observed variables and/or two or more instances. In this case, the information of a process is replicated multiple times due to human mistakes, data transmission problems, security issues, etc.

Table 4. Duplicate data anomaly

Variable 1	Variable 2	Variable 3	Variable 3
16.45	Incomplete	1800	1800
16.45	Incomplete	1800	1800

Example of duplicate data anomaly

Table 4 shows an example of a duplicate data anomaly, in this case we can see that the variable 3 is repeated and that instances 2 and 3 are the same. Normally, to solve this type of anomaly only one of the repeated variables or instances is preserved.

- **Syntaxis data anomaly**

This data anomaly refers to information that does not correspond with the expected data type. In other words, this anomaly occurs when the system tries to convert the information received to store the corresponding variables, but the data cannot be restored.

Table 5. Syntaxis data anomaly

Variable Name	Type	Correct Data	Receive Data
Temperature	Numeric	18.5	18.5C
Location	Varchar	Popayan	19001
Date	Date Time	15/08/2022	08/15/22

Example of syntaxis data anomaly

Anomalies in syntax data are illustrated in Table 5. The first example shows data from the temperature variable. As shown in the Correct Data column, the variable should be numeric, but the system receives a value similar to the Receive Data column. Depending on how the database is implemented, the system will generate an error or warning if this value is converted to numeric. The same considerations can be used when expected and received data types differ. In the last example, the expected and corrected data are dates, but the formats are different, resulting in a syntax error.

- **Semantic data anomaly**

Semantic anomaly is when one or more data or instances do not follow the usual pattern. In this case, the information shows values over or under the expected average or values that correspond with the sensed variable but on a different scale. Typically, semantic data anomalies are considered outliers or noise in the data.

Table 6. Semantic data anomaly

Variable 1	Variable 2	Variable 3	Variable 4
15.26	Complete	1500	Good
16.25	Incomplete	200	Good
58.24	Complete	1600	Malo

Example of semantic data anomaly

Examples of semantic data anomalies are shown in Table 6. Variable 1 has a value that is several times greater than the other values in the column, and this behavior is also apparent in Variable 3 where there is a value several times lower than the others. Lastly, Variable 4 shows another type of semantic anomaly; in this case, the expected value is Bad, but the system receives Malo; the meaning remains the same, but the system may not be able to process it.

Multiple types of anomalies can be found in supply chain traceability. Each has various approaches for detection and correction, from data imputation to data elimination, depending on the importance of the data, the amount of data with problems, etc. To narrow the scope of our development, we will focus only on detecting numeric syntax and semantic data anomalies.

2.2. Related Works

For this project, the starting point was established around using smart contracts in traceability processes. With this in mind, a systematic mapping was conducted based on the methodology proposed by (Petersen et al., 2008), to obtain an overview of the research area and locate gaps and contributions in the research found; the method of the systematic review proposed in (Kitchenham & Charters, 2007) was used. These methodologies are based on a procedure that can be summarized in five steps: first, define the research question, then perform the literature search, next select the relevant studies, classify the selected studies, and finally, extract and synthesize the information. Each of these steps will be detailed next.

2.2.1 Research Methodology

A. **Research Question:** The primary goal of this work is to provide a Blockchain traceability scheme with the necessary tools to perform data reliability validations to ensure that the trace data is reliable and gives an accurate representation of the traced product on the supply chain.

   Question 1. In a traceability scheme, what kinds of anomalies can be found?

   Question 2. How do smart contracts detect anomalies in a traceability scheme using Blockchain technology?

   Question 3. How can smart contracts be autonomously updated to adjust for anomalies in a traceability scheme?

B. **Data Sources and Search Strategy:** To conduct the systematic mapping for our research, we consulted the following scientific bibliometric databases; Scopus, ScienceDirect, and IEEE Xplore. In total, we obtained

more than 3,400 scientific investigations, most of them between 2015 and 2022, since Blockchain technology and its use in traceability schemes is relatively new.

C. **Relevant Studies Selection:** The criteria used for the selection of studies were the following:

> ***Acceptance****: Those studies that use binding terms such as "Blockchain," "Smart Contracts," "Traceability," "Reliability," "Data Validation," and/or "Supply Chain." Those studies whose titles suggest using Blockchain technology to detect anomalies in a traceability scheme.*
>
> ***Rejection****: Those studies that do not use the terms mentioned before.*
>
> Only journals and conference articles written in English or Spanish were considered.

With these acceptance and rejection criteria, the search is conducted in the selected databases using the following search strings:

String 1. *"Blockchain" AND "Supply Chain" AND "Traceability."*

String 2. *"Blockchain" AND "Data Validation."*

String 3. *"Traceability" AND "Data Anomalies."*

String 4. *"Smart Contracts" AND "Anomaly Detection."*

String 5. *"Blockchain" AND "Data Reliability."*

Data anomalies are only used in the case of traceability to broaden the search in this field and to review what has been done, specifically in this area.

D. **Relevant Studies Classification:** As mentioned before, more than 3400 research were found. We reduce the number of research using the acceptance and rejection criteria to more than 1000 research documents. These documents were revised by title, summary, and conclusions to select the ones related to our investigation. Finally, 79 studies were selected and categorized after the review.

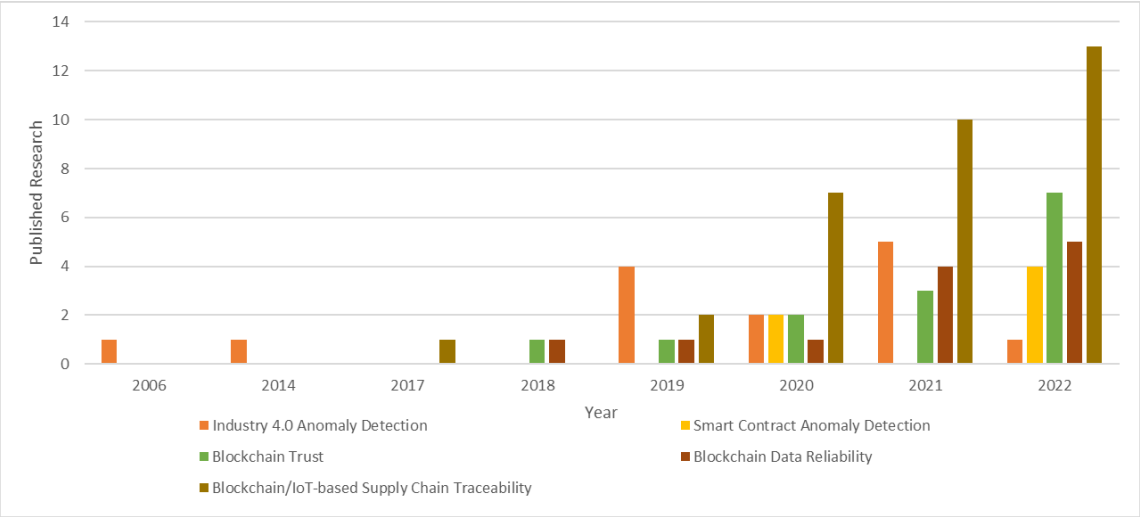


Figure 3. Categorized studies distribution.

Figure 3 shows the distribution over the years of each group of the selected studies, the figure shows how the research interest has been shifted in the past years to Blockchain data reliability and Blockchain/IoT supply chain traceability. Also, we can see that there are some approximations for anomaly detection using smart contracts.

A brief overview of some of the studies classified in each subject follows.

1. **Industry 4.0 Anomaly Detection**

In this category, the research papers highlight that data tampering is something that can happen on a Blockchain scheme before the network transaction occurs (Iyer et al., 2019; X. Liu et al., 2020) so to avoid fraud and to ensure trust, anomalies in data must be detected and corrected before adding them to the Blockchain. Additionally, research like (Oh et al., 2019; Shukla et al., 2022) proposes a method for business logic verification and anomaly connections to ensure that the process is not vulnerable to Interference from external sources.

Also, authors like (Günther et al., 2019; Talha et al., 2019; X. Wang et al., 2021) have proposed methods to perform data quality evaluations analyzing the business data structure to highlight the conflicts of data quality management systems. Finally, authors like (Balagolla et al., 2021) propose fraud prevention by detecting anomalies in credit card transactions using Blockchain technology to ensure transparency.

2. **Smart Contract Anomaly Detection**

Luo et al., in (Luo et al., 2022) review common scenarios where false information at any stage of the supply chain may represent a significant

challenge to authorities. They explain how Blockchain, and smart contracts can improve supply chain traceability transparency, security, and data immutability to address these scenarios. In (Kumar et al., 2022) the authors explore Blockchain technology's applicability in logistics and supply chains using smart contracts in Ethereum, comparing their proposed solution with the ones currently available.

Other authors like (Hameed et al., 2022) pursue the detection of security threats in Blockchain-based industrial applications and consider the introduction of the concept of Blockchain 5.0 needs to be able to introduce artificial intelligence as a feature of Blockchain to address the security threats such as data tampering and fraud detection.

3. Blockchain Trust

Within the Blockchain network, transparency and trust in information are emphasized by multiple authors such as (Cao et al. 2021; Guo & Yu, 2022; L. Liu et al., 2022; Marchesi et al., 2022; Patel & Shrimali, 2021), some authors suggest that proposed Blockchain solutions lack trust due to being developed for a specific process and difficult to generalize, while others have proposed the use of ML algorithms to detect abnormal smart contracts to avoid security risk.

Authors like (W. Wu et al., 2022) have proposed using neural networks in a framework of smart reporting systems based on the Blockchain and Internet of Things (IoT) technologies to evaluate the authenticity of the collected data. Authors like (Kravitz, 2018; J. Wang et al., 2022) mentioned that data security issues are becoming increasingly important and must be addressed by improving the storage mechanism in Blockchain technology and implementing reputation traceability to increment the trust of the data stored in a Blockchain scheme.

4. Blockchain Data Reliability

Information sharing is an essential feature of Blockchain, so data reliability must be ensured; authors like (Yang et al., 2020) have proposed a mechanism to ensure data reliability in a big data environment with Blockchain technology using the user's private and public keys. Authors like (Cozzini et al., 2022) have proposed using Blockchain technology to improve the data reliability of COVID-19 genomic sequences from all over the world.

Finally, authors like (Casado-Vara et al., 2018; Deepa et al., 2022; Hussien et al., 2021) present multiple trends and opportunities to

improve data reliability in various areas by implementing different data reliability methods using Blockchain technology and how Blockchain with its features could be the cornerstone in the industry 5.0 development.

5. Blockchain/IoT-based Supply Chain Traceability

Research on this group shows the improvements in traceability when Blockchain technology is used; authors like (C. Y. Chen et al., 2021; Etemadi et al., 2020; Jannat et al., 2021; Subashini & Hemavathi, 2022) has addressed different Blockchain implementations for traceability in supply chains, showing the weaknesses and strengths of the traceable scheme before and after the use of Blockchain. Also, these authors mention the opportunities and challenges to increase the acceptance and reception of Blockchain technology in both traceability and supply chain aspects.

Authors like (Rodríguez et al., 2021) have tried to understand the technology better. They select the best one for supply chain traceability by analyzing multiple implementations and proposing a three-layered architecture for agriculture processes. In (Li et al., 2022), the authors propose a Blockchain framework to improve supply chain resilience so that companies using this technology can achieve higher performance.

There are multiple approaches to supply chain traceability using Blockchain technology proposed by different authors, each focused on improving one or more critical aspects of the process. Despite this, none of them suggest using smart contracts to improve data reliability by enabling them to detect anomalies.

2.2.2 Results and Analysis

The results of steps A, B, C, and D of the literature mapping and systematic review are presented in the previous section. As a result, step E is shown below to highlight the results and their analysis.

E. Extraction and synthesis of information: As indicated in the previous subsection, several researchers have tried to address the trust problem of Blockchain-based supply chain traceability, applying different approaches to make sure that the final users welcome the incorporation of Blockchain. Moreover, the research explores the critical aspects of including Blockchain technology in the non-distant future.

2.2.3 Gaps and contributions

The following investigations are also highlighted from these groups:

- (Pradhan et al., 2021): The authors proposed a Blockchain-based solution for a blood bank. The solution can trace blood-related data, check bloodstock on nearby sites, trace shipments, and more. The authors also include a conditional to prevent storage conditions violation in the proposed smart contract.

- (Marchesi et al., 2022): An agri-food industrial Ethereum smart contract that is easy to customize and reuse to shorten development times of future decentralized applications (dApps). The proposed approach helps design and develop Apps to improve agricultural supply chain processes by creating higher-quality smart contracts.

- (L. Liu et al., 2022): The authors proposed using Heterogeneous Graph Transformer Networks (S_HGTNs) for smart contract anomaly detection, specializing in financial fraud detection on Ethereum. Using this technique, the authors obtained the necessary features to classify the smart contracts on the platform as normal or anomalous.

- (Iyer et al., 2019): A wastewater reuse monitoring system is proposed using Blockchain and anomaly detection. Blockchain technology is used to incentive wastewater reuse with tokens. In contrast, anomaly detection algorithms are used to detect anomalies in the data sent to the network by IoT sensors deployed all over wastewater treatment and industrial plants.

- (Demertzis et al., 2020): Authors introduce the concept of deep learning smart contracts to detect anomalies in a Blockchain environment using Deep Autoencoders. The authors manage to use Cloud ML Services from inside the smart contract in the Wave Blockchain platform to analyze the DataStream received by the smart contract to classify it as normal or abnormal.

- (Tukur et al., 2021): In this research, a Blockchain enable anomaly detection technique in an IoT environment is proposed. Edge computing is used as nodes on the Blockchain network; this ensures the integrity of the data collected and processed using edge computing as close as possible to the Blockchain nodes in the Ethereum platform.

- (Shukla et al., 2022): A Blockchain system model for anomaly detection in smart grids using linear support vector machine for anomaly

detection in a fog computing environment. The fog nodes work as miners on the Blockchain network and perform data validations on the transaction data.

- (Oh et al., 2019): Using business context data, authors proposed a novel distributed ledger system for supply chains that verifies the semantic correctness of transactions on Hyperledger Sawtooth. The proposed method can detect anomaly transactions in the life cycle and velocity in the supply chain.

- (Shao et al., 2020): Authors proposed a framework to auto-update smart contracts in a Blockchain-based log system. The smart contracts are updated based on log anomaly detection using ML so that the recent smart contract version can detect anomalous log entries in the Blockchain.

Research in this area exhibits an affinity with the developments undertaken within our research program. A gap analysis of the selected investigations is presented in table 7.

Table 7. Located gaps.

Research	Gap
(Pradhan et al., 2021)	Although the authors perform data validation to detect violations in the blood bank, the smart contract cannot detect anomalies in the received data.
(Marchesi et al., 2022)	The authors do not consider the possibility of data anomalies being present on the traced data.
(L. Liu et al., 2022)	Even though the author's goal is anomaly detection, their development is centered on detecting abnormal smart contracts.
(Iyer et al., 2019)	The authors only use Blockchain technology to store the transaction data and to give tokens according to the wastewater reuse.
(Demertzis et al., 2020)	Although the authors manage to perform anomaly detection on the data sent on the transaction in the Blockchain network, it depends on an external tool to detect the anomalies with all the security risks it implies.
(Tukur et al., 2021)	Even though the authors use edge computing as nodes in the Blockchain network, the anomaly detection is performed outside the network, and the results uploaded are the corrected data delivery by edge computing.

(Shukla et al., 2022)	The authors only use Blockchain technology to store the transactional data, and all the anomaly detection process is conducted outside the Blockchain network.
(Oh et al., 2019)	Although the authors proposed to perform anomaly detection on the transaction, it only checks if the transaction complies with the business context, leaving aside the correctness of the transaction data.
(Shao et al., 2020)	The proposed framework only considers anomalous logs on the Blockchain network, but abnormal data can be present in normal transaction logs, so the smart contract will not be able to detect these kinds of anomalies.

Detected gap for each highlighted research.

- Most Industry 4.0 Anomaly Detection group research has focused on fraud detection or detecting anomalous transactions based on the information available about the involved users, logs, etc.

- Research on Smart Contract Anomaly Detection, Blockchain Trust, and Blockchain Data Reliability has focused on aspects such as validating data before reaching the Blockchain network or using external tools to detect anomalies in the transaction data.

- Although some investigations have tried to provide smart contracts with data anomaly detection capabilities, most have been using external tools, so the detection is not conducted inside the Blockchain network.

- Some investigations have proposed smart contract auto-update, but only one has updated the contract based on anomaly detection. Nevertheless, the anomaly detections have been leaving aside abnormal data on the Blockchain transaction.

- During the systematic mapping and review no evidence of abnormal data detection on a Blockchain transaction using smart contracts with autonomous update capabilities without the interference of an external tool.

2.3. Summary

This chapter presents the theoretical concepts related to our doctoral research proposals, such as Blockchain, Smart Contracts, Traceability, Data Analysis, and Anomalies.

Subsequently, the most relevant research in the systematic mapping process and subsequent review of the results obtained in the different bibliometric bases consulted were exposed. The systematic mapping was oriented based on the investigation problem described in chapter 1. The localized investigations were separated into groups, allowing us to locate the contributions and research gaps better. Most of the study focused on smart contract anomaly detection has been using external tools to rely on anomaly detection or to detect abnormal smart contracts or transaction logs on the Blockchain network.

Chapter 3

Syntaxis Anomalies Detection Strategy

This project's main objective is to develop an autonomous anomaly detection Blockchain traceability network using smart contracts. To perform this, we review anomalies that may arise in supply chain traceability data, anomaly detection methods, and implementations that may benefit our investigation. The Syntaxis anomaly detection strategy is discussed in section 3.1. Section 3.2 shows the network and smart contract deployment process, including the Blockchain network solution selection and the tool used for smart contract performance evaluation. Section 3.3 shows the results of the performance test. Section 3.4 and 3.5 shows results for the use case of the developed base smart contracts for a simulated case and an actual use case on production sites, respectively. Finally, section 3.6 shows the advantages and disadvantages of the implemented strategy.

3.1. Detection Strategy

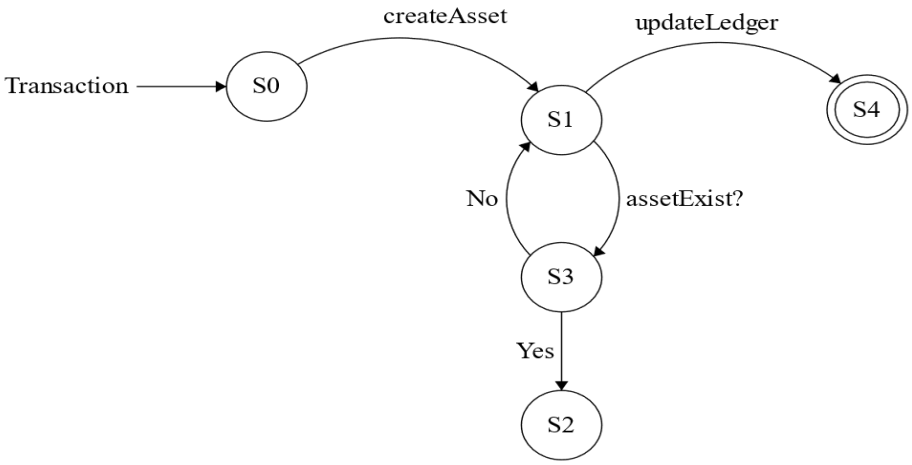
Considering that smart contracts execute inside the Blockchain network, our proposed strategy uses the information stored on the Blockchain ledger to raise alerts about anomalies found in the current transaction data. We developed the following to provide the smart contract with the ability to detect the selected anomalies.

3.1.1 Base Smart Contract

This section will introduce the essential parts of our smart contract, from asset creation and asset update. Algorithm 1 shows the asset creation process. The algorithm uses transactional data related to a traced good using an asset ID. Each time a traced product variable is modified, this algorithm adds the information to the traceability Blockchain network. If the asset ID is new, it will create the asset on the Blockchain ledger. If not, the asset update process (algorithm 2) is executed.

Inputs:
ID: key value of a sensed good
PSI: Traceable sensed information
Outputs:
product = GetState(ID);
if product **==** null **then**
PutState(key, “none”, “none”, 1.0, 1.0);
else
assetUpdate();
end if
Asset created with the traced data.

Algorithm 1. Asset creation algorithm.



LHS		RHS
S0	→	createS1
S1	→	AssetExist?S3
S1	→	updateLedgerS4
S3	→	NoS1
S3	→	YesS2
S4	→	λ

Figure 4. Algorithm 1 Finite State Machine (FSM).

Figure 4 shows the FSM and the transitions between the states available for this stage, where Transaction receive (S0), Asset creation (S1), Asset update (S2), Asset validation (S3), and Ledger update (S4).

Algorithm two shows the asset update process. This algorithm updates the asset states on the ledger; as previously mentioned, each traced good has an ID stored as an asset.ID. Using this ID, the smart contracts update the state of all the

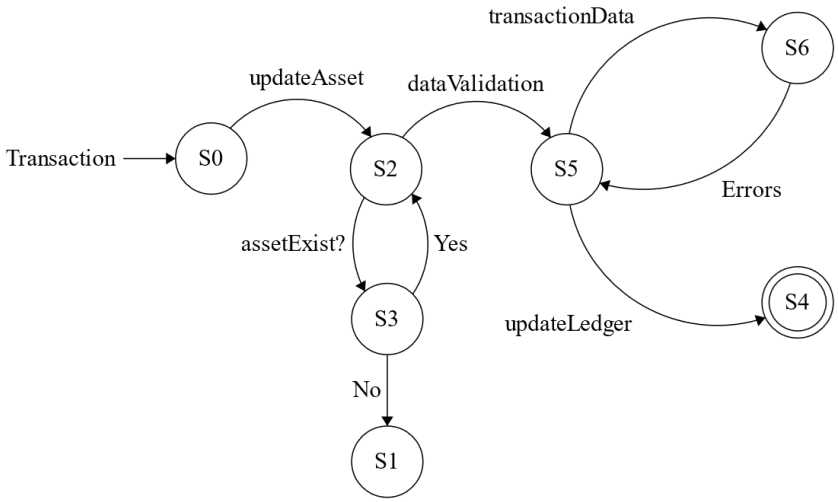
product data on the ledger. The algorithm sets new states on the Blockchain ledger using the asset if the data is valid.ID; if data is invalid, the algorithm will update the conditions of the anomaly data states and update the asset states using the last good information available. If the asset has not been created, the algorithm executes asset creation.

Inputs:
ID: key value of a product
PSI: Product sensed information
Outputs:

```
product = GetState(ID);
if product != null then
        errorsString = syntaxErrCheck(PSI);
        PutState(key, errorsString);
else
        createAsset();
end if
```

Asset updated with the traced data; syntax errors located.

Algorithm 2. Asset update algorithm.



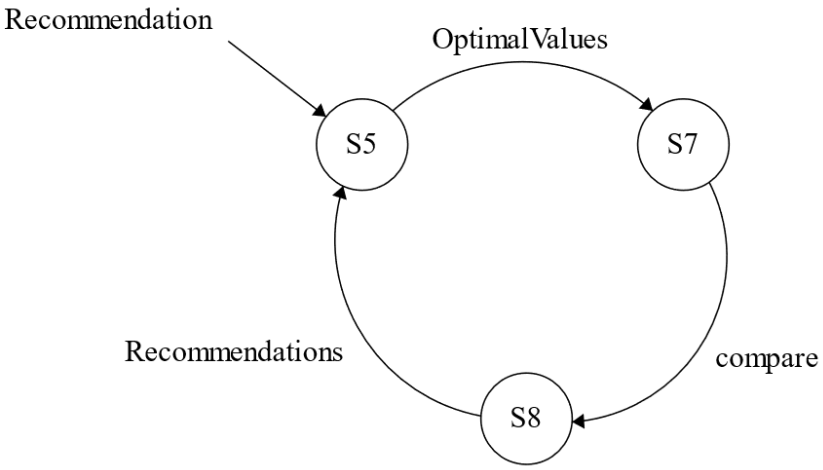
LHS		RHS
S0	→	assetUpdateS2
S2	→	AssetExist?S3
S3	→	NoS1
S3	→	YesS2
S2	→	dataValidationS5
S5	→	transactionDataS6
S6	→	errorsS5
S5	→	updateLedgerS4
S4	→	λ

Figure 5. Algorithm 2 Finite State Machine (FSM).

Figure 5 shows the FSM and the transition between states, where Transaction receive (S0), Asset creation (S1), Asset update (S2), Asset validation (S3), Ledger update (S4), Data validation (S5) and Syntax error check (S6).

```
Inputs:
        PSI: Product sensed information
        ID0: Optimal Product Information ID
Outputs:
        OP = GetState(ID0);
        if OP != null then
                if PSI != OP then
                        reco = “recommendation”;
                else
                        reco = “none”;
                end if
        end if
        return reco;
Management recommendation
```

Algorithm 3. Management Recommendation algorithm.



LHS		RHS
S5	→	optimalValuesS7
S7	→	compareS8
S8	→	recommendationsS5

Figure 6. Algorithm 3 Finite State Machine (FSM).

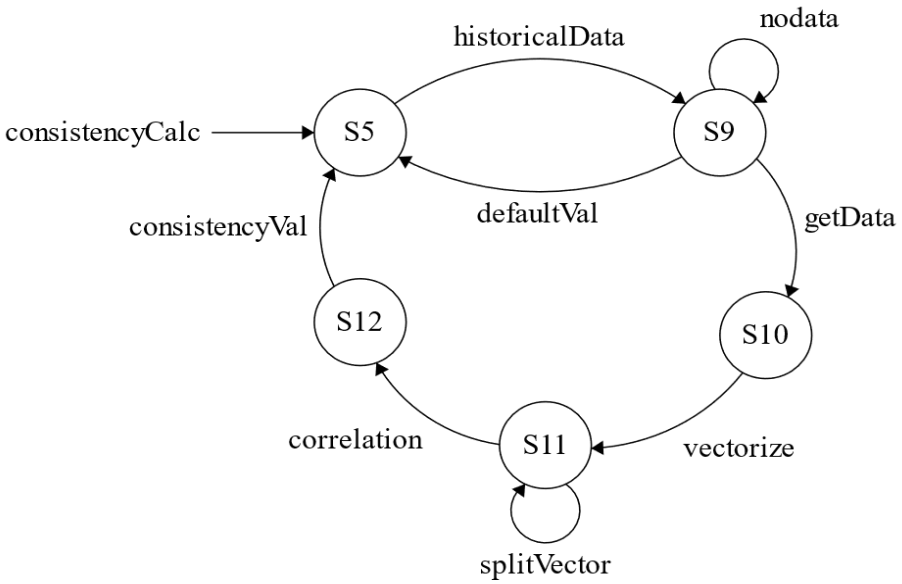
The FSM model presented in Figure 6 shows the transitions between the states available at this stage for the smart contract: Data validation (S5), Optimal values retrieval (S7), and Data comparison (S8). Algorithm 3 compares the traced

product data with the predefined optimal data and generates recommendations for the traced product management based on the collected knowledge.

Algorithm 4 shows how the consistency value is calculated. In this case, all the historical data is used, the historical data is divided into two vectors of equal size, and the correlation between these vectors is then estimated. This value is used as the consistency parameter. Algorithm 5 calculates the reliability of the data; this measure is estimated by dividing the amount of data in good condition by the total amount of data.

```
Inputs:
        PSI: Product sensed information
        ID: key value of the current product
Outputs:
        productHistory = GetHistoricalState(ID);
        totaldata = [];
        if productHistory != null then
                for data in PSI
                        totaldata.append(data);
                end for
                for data in productHistory.PSI
                        totaldata.append(data);
                end for
                data1, data2 = split(totaldata);
                consistency = correlation(data1,data2);
                return consistency;
        end if
Consistency Value
```

Algorithm 4. Consistency calculation pseudocode



LHS		RHS
S5	→	historicalDataS9
S9	→	nondataS9
S9	→	defaultValS5
S9	→	getDataS10
S10	→	vectorizeS11
S11	→	splitVectorS11
S11	→	correlationS12
S12	→	consistencyValS5

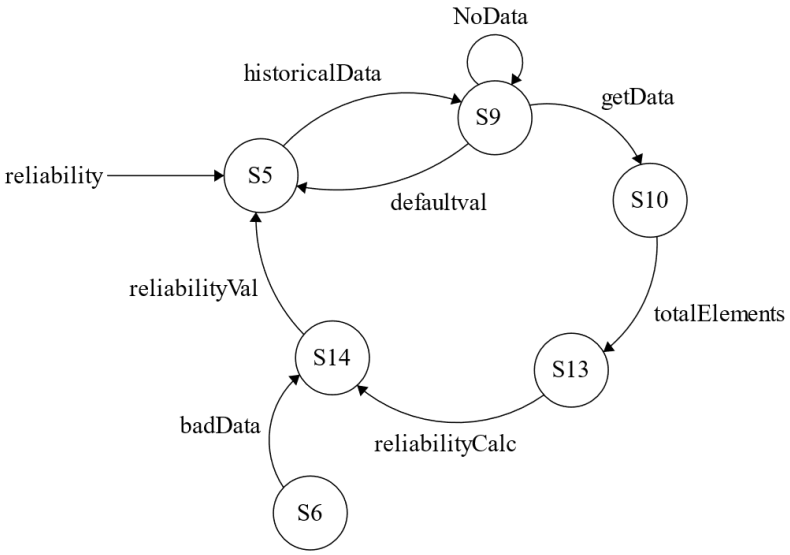
Figure 7. Algorithm 4 Finite State Machine (FSM).

Figure 7 shows the transitions between the states available at this stage for the smart contract: Data validation (S5), Historical data retrieval (S9), Data selection (S10), Vectorization (S11), and Correlation (S12).

```
Inputs:
        PSI: Product sensed information
        ID: key value of the current product
Outputs:
        productHistory = GetHistoricalState(ID);
        totaldata = [];
        if productHistory != null then
                for data in PSI
                        totaldata.append(data);
                end for
                for data in productHistory.PSI
                        totaldata.append(data);
                end for
```

```
            goodData = totaldata.size() - badData;
            reliability = goodData/totaldata.size();
            return reliability;
        end if
Reliability Value
```

Algorithm 5. Reliability calculation pseudocode



LHS		RHS
S5	→	historicalDataS9
S9	→	nondataS9
S9	→	defaultValS5
S9	→	getDataS10
S10	→	totalElementsS13
S13	→	reliabilityCalcS14
S6	→	badDataS14
S14	→	reliabilityValS5

Figure 8. Algorithm 5 Finite State Machine (FSM).

The FSM model presented in figure 8 shows the transitions between the states available at this stage for the smart contract: Data validation (S5), Syntax error check (S6), Historical data retrieval (S9), Data selection (S10), Total elements (S13), Reliability Calculation (S14).

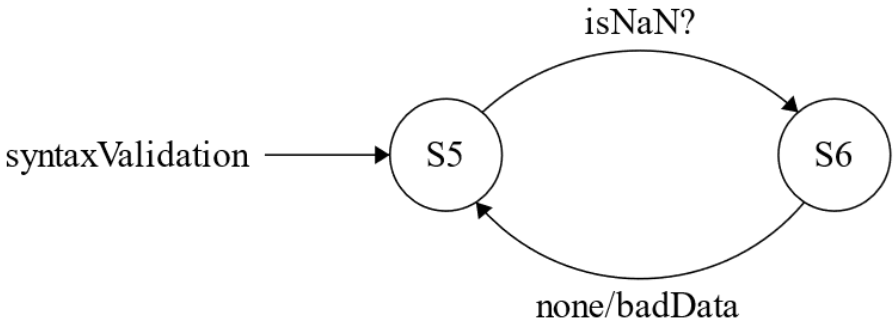
Consistency and Reliability as measurements used in conjunctions with the syntaxis anomaly detection, the anomalies detected affects the values of these two metrics.

3.1.2 Syntaxis Anomaly Detection

To give the smart contract the ability to detect syntaxis anomalies, we developed a Go and JavaScript algorithm that looks through the data annexed in the current transaction data. This algorithm, based on the knowledge given checks is the transactions data received correspond with the expected data types for each transactional component; if one or several of these components do not correspond with the desired data types, the smart contract stores them on the ledger as anomalies to be addressed by the networks operators and keep the last correct transaction data value as the current data value.

```
Inputs:
        PSI: Traceable sensed information
Outputs:
        for data in PSI
                if Type(data) then
                        return “none”;
                else
                        badData += 1;
                        baddataString = baddataString + data;
                end if
        end for
        return string(baddataString);
Syntax error detected on the traceable sensed information
```

Algorithm 6. Syntaxis anomaly detection



LHS		RHS
S5	→	isNaN?S6
S6	→	none/badDataS5

Figure 9. Algorithm 6 Finite State Machine (FSM).

Algorithm 6 shows the pseudo-code of the developed syntaxis anomaly detection tool. In this case, the smart contract checks if the data can be converted from string to the expected data type. If the data cannot be, the algorithm returns the data value stored on the ledger as corrupted data to be analyzed later. If the data can be converted, the algorithm returns the string none. Figure 9 shows the FSM and the transition between states, where Data validation (S5) and Syntax error check (S6).

3.2. Network & Smart Contract Deployment

Blockchain technology is currently accessible through multiple platforms that allow the deployment of applications based on this technology. Next, we will review some of these platforms. In this case, we do not include Bitcoin since this platform is purely monetary and does not allow the execution of smart contracts.

Stellar

It is a platform that facilitates the transfer of value between assets, cryptocurrency exchanges, and currencies based on fiat currency (*Stellar*, n.d.). On this platform, it is possible to develop banking tools and wallets for mobile and smart devices.

Hyperledger Fabric

It is a platform for creating solutions or applications based on the Blockchain using a modular architecture (*Hyperledger*, n.d.). This allows network developers to connect specific components, such as membership or consensus services, to the network. Hyperledger Fabric is designed for permissioned networks, allowing known identities to participate within a system. Participants within this network must be authorized and have the credibility to participate in the Blockchain .

Hyperledger Sawtooth

This platform, like Fabric, is modular and enterprise-grade but focuses on creating, deploying, and running distributed ledgers (*Hyperledger*, n.d.). This platform uses the PoET (Proof of Elapsed Time) consensus mechanism, which allows it to integrate with trusted execution environments. Its most recent version includes web assembly smart contracts and Python transaction processors.

Hyperledger Iroha

The system is based on a rapid and highly safe consensus algorithm called Yet Another Consensus (YAC) (*Hyperledger*, n.d.)., which protects Iroha networks from adversary nodes. The platform is highly applicable to supply chain and IoT use cases since it is portable and compatible with macOS and Linux environments. However, it is primarily designed for small data and/or specific

situations. The platform is currently under development, but it holds great potential for the future.

Corda

It is a Blockchain platform that allows institutions to transact directly with smart contracts. It does not have a built-in cryptocurrency or token and is a permission Blockchain platform that only allows authorized participants to access data, not the entire network (*Corda*, n.d.).

EOS

Is a Blockchain platform founded by Block.one. It is designed to develop Dapps (decentralized applications) (*EOS*, n.d.). The platform aims to offer decentralized application hosting, decentralized enterprise solution storage, and smart contract capability, solving the scalability issues of Ethereum and Bitcoin. Also, there is no need to pay to reap the benefits of an EOS-based Dapp. It used the POS (Proof of Stake) consensus protocol

Ethereum

Ethereum is an open-source, Blockchain-based, distributed computing platform that is public (permissionless) and built for restricted access versus mass consumption (*Ethereum*, n.d.). It is known for running smart contracts on a custom Blockchain. Ethereum Virtual Machine (EVM) provides the runtime environment for smart contracts on Ethereum using Proof of Work (POW) as a consensus protocol.

Table 8 shows a summary comparison of the reviewed Blockchain platforms. It is considering its consensus protocol, focus, and smart contract support, among others.

Table 8. Blockchain platform comparison

Platform	Consensus protocol	Focus	Governance	Network Type	Smart Contract Support
Stellar	SCP	Financial	Stellar Development Foundation	Authorized & Permissionless	**Yes**
Fabric	Multiple Plug & Play	Cross Industry	Linux Foundation	Authorized	**Yes**
Sawtooth	PoET	Cross Industry	Linux Foundation	Authorized	**Yes**
Iroha	YAC	Cross Industry	Linux Foundation	Authorized	**Yes**
Corda	Multiple Plug & Paly	Cross Industry	R3 consortium	Authorized	**Yes**
EOS	POS	Cross Industry	EOSIO Central Arbitration Forum (ECAF)	Authorized	**Yes**
Ethereum	POW	Cross Industry	Ethereum Developers	Permissionless	**Yes**

After reviewing multiple platforms, we used Hyperledger Fabric to implement our proposed solution. Because the network does not necessarily need a consensus algorithm, it requires fewer resources. If it is determined that a consensus algorithm is necessary, it can be added without affecting the network deployment currently underway. A cross-industry approach and not focusing on specific applications allows the deployment of multiple applications over a network.

To deploy our test network, we use the Minifabric tool (*Minifabric*, 2020); this tool's main goal is to help Hyperledger Fabric users to deploy in an easier way test network with the desired configurations without the need for the creation of multiple configuration files needed. Minifabric can stand up a Fabric network on a small machine like a VirtualBox VM but can also deploy Fabric networks across multiple production-grade serves, and support Hyperledger Fabric releases 1.4.4 and up.

Hyperledger Fabric supports multiple channels and does not require cryptocurrency for its operation. It can use Plug and Play consensus models to adopt the most efficient consensus model for each use case. The permission allows determining who can read or write on the Blockchain; all digital assets are stored in a key-value database, allowing easy retrieval of information from the network. By default, it uses RATF (RAFT, 2016) consensus protocol

Additionally, it allows the execution of smart contracts called chain codes; these can be implemented in Go, Java, JavaScript, and Node.js. In the same way, it will enable the management of Plug and Play identities using managers such as LDAP or OpenID. Since all business network participants have known identities within an organization, PKI with X509 certificates is used to issue cryptographic certificates to organizations, participants, or application users. To deploy the Hyperledger Fabric network, the following requirements must be fulfilled:

- ✓ Docker 17.03 or higher
- ✓ Docker-compose 1.8 or higher
- ✓ Node.js 8.9 or higher
- ✓ npm 5.x
- ✓ git 2.9.x or higher
- ✓ python 2.7.x or higher

In our case, we have a laptop PC with an Intel Core i7 processor, 16 Gb of RAM, 512 Gb of internal storage, and a desktop PC with an Intel Core i7 processor and 32 Gb of Ram, and 1 Tb of internal storage. We developed a python tool to create the network channels and install the selected smart contracts on the channel. Follow the step-by-step guide (*Minifabric*, 2020) docs' page to add another device.

Once the Blockchain network is up, the developed module looks up the selected smart contract and performs the deployment process. The first step is to install the smart contract on the network channel that we choose; the installing method comprises the copying of the contract code and dependencies to a folder on the Blockchain directory; after that, the code is built; if no error is encountered during this stage, the contract will be correctly installed.

The next step is discovering the installed smart contract. This process tells all the peers and organizations connected to the channel where the smart contract was established that a new contract had been found. Finally, the smart contract is initialized; in this step, the Blockchain network calls the initial function on the smart contract to define all the necessary assets for the smart contract execution. Although, assets can be created later if there are no initial values or the contract's initial conditions have not been agreed upon.

A set of flags in the deployed smart contracts will assist in the autonomous update process. Our test scenarios make use of a set of techniques to generate a measurement for data reliability, including penalties and recommendations based on the number of anomalies found on the data. Consistency is one of the criteria for data reliability; this measurement is based on the correlation between data stored on the Blockchain ledger; if the data is consistent, the correlation will be high; otherwise, it will be low. As for another reliability measurement, the Kivenson method is used (Agmon & Ahituv, 1987). The values are based on a scale of 0 to 1, with 1 being the most reliable and 0 being the least reliable.

To evaluate the feasibility of our base smart contracts and Blockchain network, we decided to use Hyperledger Caliper; this is a Blockchain benchmark tool that allows us to measure the performance of our Blockchain with a set of predefined use cases (*Hyperledger Caliper*, 2018). Caliper provides a unified Blockchain benchmark framework that can be used with multiple Blockchain solutions, not only those developed by the Hyperledger Foundation. To install and run Caliper, follow the instructions found (*Installing and Running Caliper*, 2018).

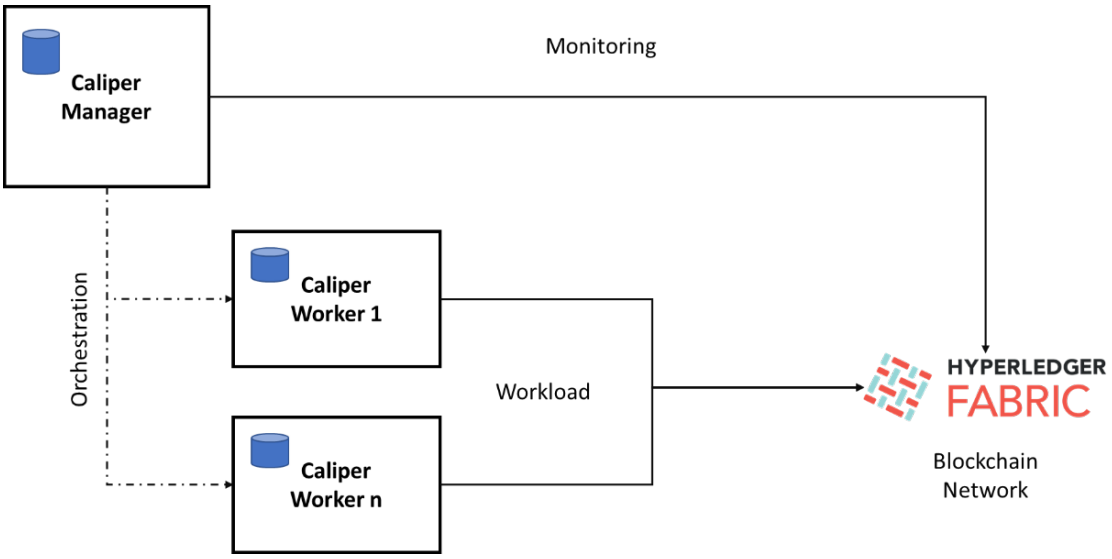


Figure 10. Hyperledger caliper monitoring/orchestration scheme

Figure 10 show the caliper monitoring and orchestration scheme, the tool stablishes a connection with the deployed Blockchain network and begin to send transaction to the network while monitors it behavior. Caliper can give us metrics about Transaction/read throughput, Transaction/read latency (minimum, maximum, average, percentile), and more, depending on the type of network and configuration. We will measure throughput and latency metrics based on (Treiblmaier, 2019); we decided to focus only on these Blockchain challenges to ensure that the proposed smart contract will not threaten the network.

Caliper throughput metric is the rate at which the Blockchain commits valid transactions in the defined time window. Latency metric is defined as a network-wide view of the amount of time taken for a transaction's effect to be usable across the network and the amount of successful and failed transactions. Next, the Caliper configuration file is shown.

```
test:
 name: Base-Contract-Test
 description: Base smart contract benchmark
 workers:
  type: local
  number: 6
 rounds:
  - label: readAsset
   description: read asset benchmark
   txDuration: 60
   rateControl:
    type: fixed-load
    opts:
     transactionLoad: 2
   workload:
    module: workload/readAsset.js
```

```
    arguments:
      assets: 10
      contractId: BaseSC

  - label: updateAsset
    description: Update asset benchmark
    txDuration: 60
    rateControl:
      type: fixed-load
      opts:
        transactionLoad: 2
    workload:
      module: workload/updateAsset.js
      arguments:
        assets: 10
        contractId: BaseSC
```

The configuration most important parameters are workers and rounds. Under workers options we set the number of Caliper workers or parallel process to be active during the test. Under rounds you can configure the different test to be made, in this specific case we will be doing a read and an update test with a 60 seconds duration each, as rate control we use Fixed Load. This controller will maintain a defined backlog of transactions within the system by modifying the driven transaction per second (TPS). The result is the maximum possible TPS for the system while retaining the pending transaction load, as pending transaction load the configuration files is at 2, so the system will keep a maximum of 2 transaction waiting while the current one finish.

Under workload configuration, we can see the module to be used on each round, the number of assets to be created, and the id of the smart contract to be evaluated. The first round will assess the read transaction to create make ten assets for each worker; after that, the tool will send a read transaction consulting the information of the built asset; the transaction could come from any worker.

Once again, the second round, or the update test, will create ten holdings for each worker; after that, the tool will send update transactions. These transactions will try to change the information about the asset on the network; in this case, the base contract will perform the syntaxis anomaly detection and measurement for data reliability using kivenson and correlation reliability. Finally, all created assets will be deleted at the end of the process. We can also configure the necessary parameters to use other tools to get additional metrics from our network if needed. Nevertheless, we could not get the data from the different tools due to compatibility issues.

After the Caliper configuration file, we prepare the network configuration file. This file has all the necessary information about the Blockchain network that the Caliper tool needs to connect and send transactions on the channel where the

smart contract is installed. In this case, we got the test name, Hyperledger Fabric major version, 2.0.0. The Blockchain type, the channels used, and the smart contract id. Finally, for all the organizations to be used on the test, we will set all the parameters needed to send transactions on behalf of one of the members of said organization.

Finally, we create using JavaScript the workload modules for the Caliper tools. These files will have the code to generate the read/update transactions on the Blockchain network. After all is done, we can run the Caliper tool and evaluate our network with the developed smart contracts.

3.3. Smart Contract with Syntaxis Anomaly Detection Test

Next, we show the results obtained for our smart base contracts. The first test was performed using ten as transaction load, six workers and test duration from 60 to 600 seconds using the base smart contract 1 (***BaseSC1***). This contract was developed for food traceability research.

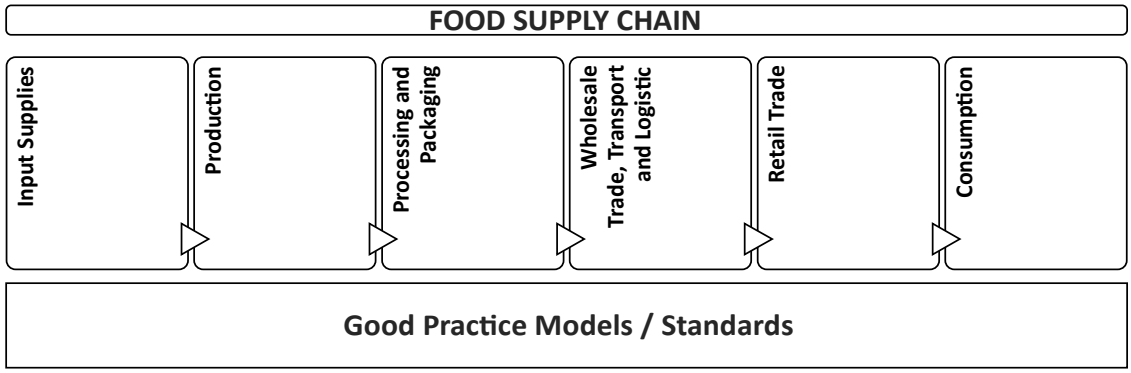


Figure 11. Food supply chain, based on (Aung & Chang, 2014).

In Figure 11, a general food supply chain is shown; each block is divided into several blocks depending on how many members are involved in the chain. Depending on how the process is conducted and how many processes are involved, each block may have a long or short internal supply chain. Depending on how each member of the supply chain performs their job, the supply chain becomes more complex; this could reduce traceability and increase vulnerability (Katsikouli et al., 2021). Due to its general approach, BaseSC1 can be used to trace any kind of food product in the production block on the food supply chain.

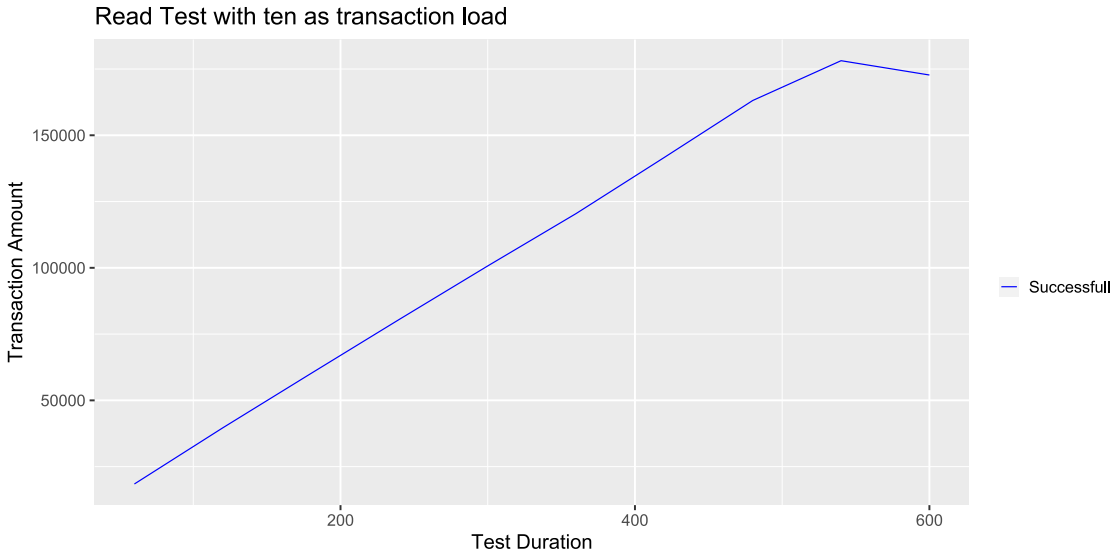


Figure 12. Read test BaseSC1

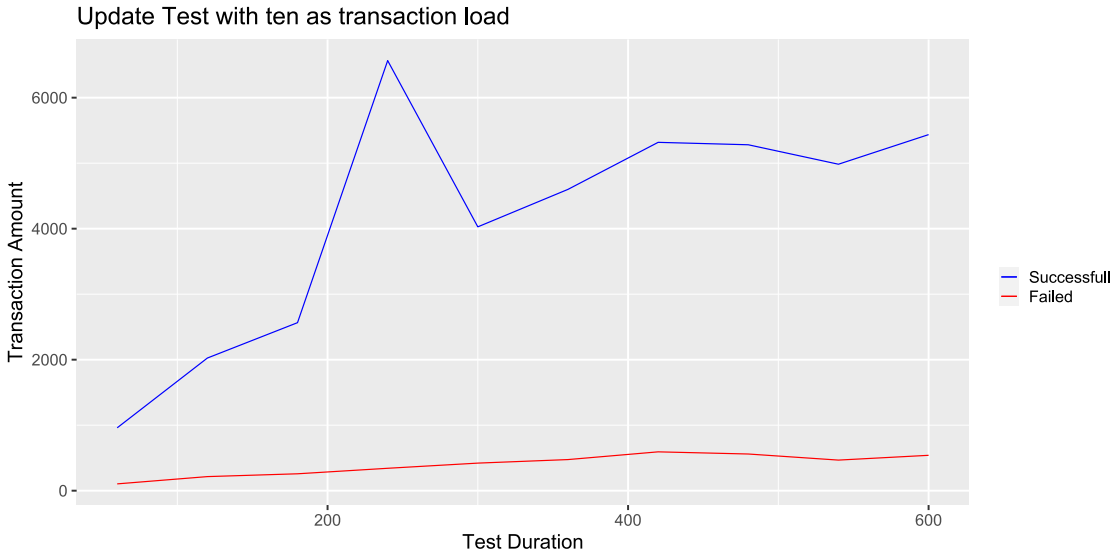


Figure 13. Update test BaseSC1

Figure 12 shows that the number of successful transactions in the reading stage creates the incrassates for the variables associated with the asset. We can see a decrease in the number of transactions, but no failed transactions were logged.

We can also see a drastic drop in the number of successful transactions on the network on the update transactions (figure 13) due to the additional processes to which each transaction is subjected, in this case, the syntaxis anomaly validation and the recommendation processes.

Failed transactions occur when two or more workers simultaneously try to update the same asset. So, the system only lets one of the workers perform the update, and the other transactions are rejected. In a typical environment, the probability

of this happening will be close to zero because each asset will be assigned to its manager at each stage. There will not be multiple users updating the same asset information simultaneously.

We can also see that after 260 seconds of test duration, there was a decrease in the transaction rate; this could be due to a server-side problem or the smart contracts resource consumption being heavy on the system. However, after debugging the smart contracts and performing multiple tests, we determined that the system had problems with the deployed tools. After heavy use of the Hyperledger Fabric solutions, the performance decreases.

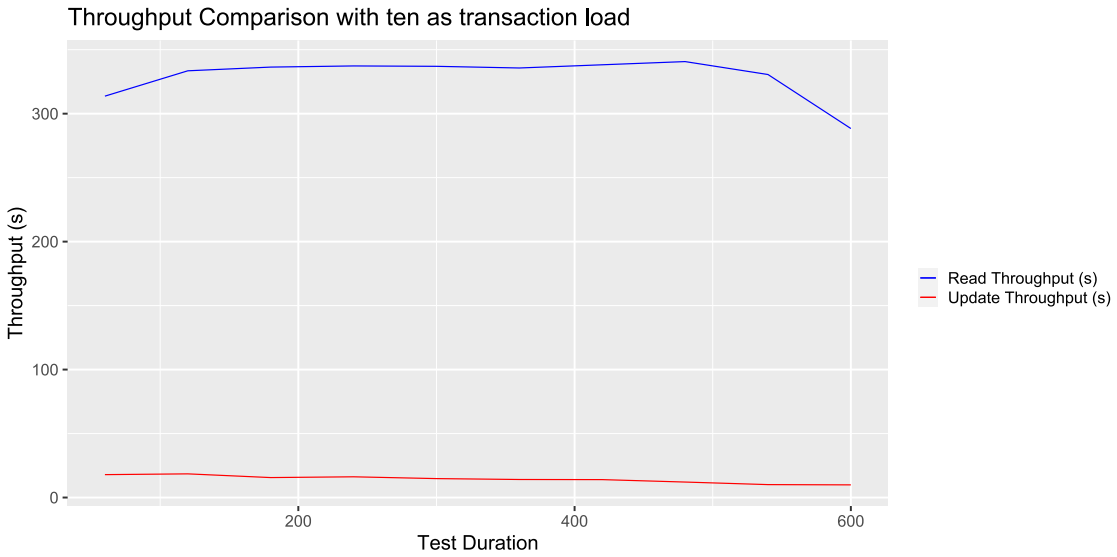


Figure 14. Throughput comparison BaseSC1

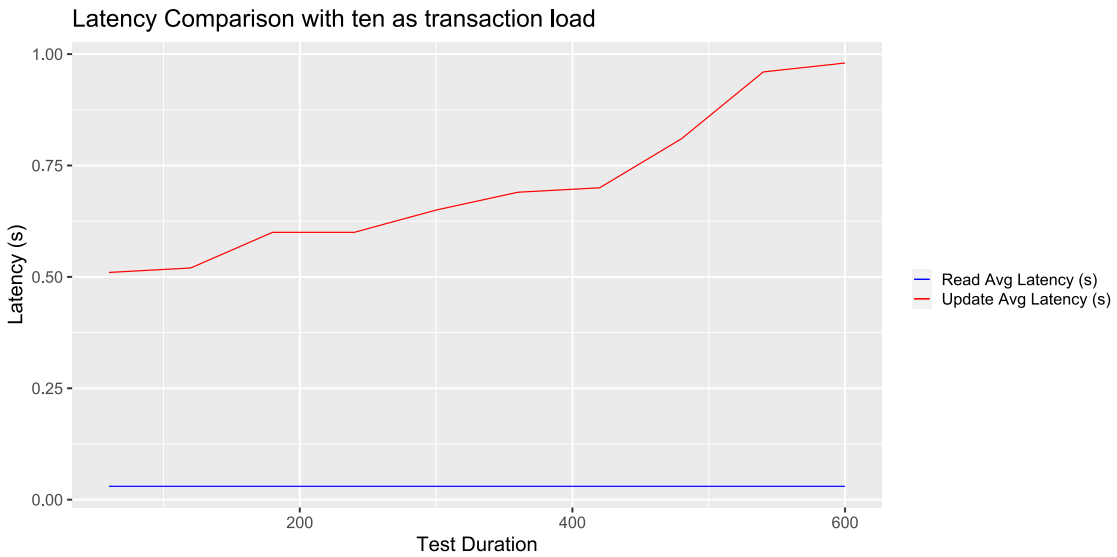


Figure 15. Latency comparison BaseSC1

Figures 14 and 15 show that throughput and latency differ significantly between the two types of transactions performed on the network. We believe that in a natural environment, the smart contracts would be executed without any inconveniences on the proposed platform owing to the use of multiple channels to trace various assets, even though the update transactions require more processing time due to the different processes needed to be conducted. On average, it would take only 0.4 seconds for the updated information to appear on the network for other nodes after updating information.

Next, we show the results for the base smart contract 2 (***BaseSC2***); this contract was developed for coffee distribution and storage research.

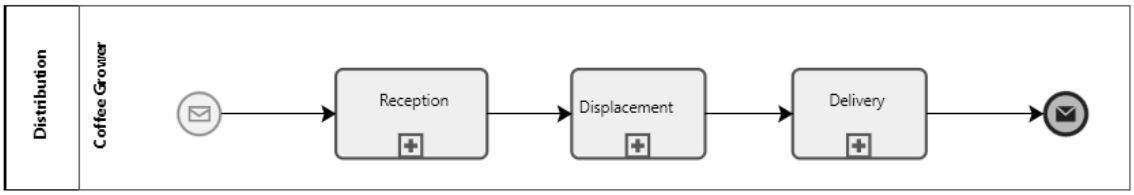


Figure 16. Distribution processes on the coffee supply chain.

Figure 16 shows how the transportation process begins with the arrival of the coffee bags. Following this, the bags are transported to the destination point, where they are delivered to the corresponding stage. Thus, our smart contract proposal focuses on displacement. To ensure that the coffee is in an appropriate condition upon delivery, it must be monitored at this stage.

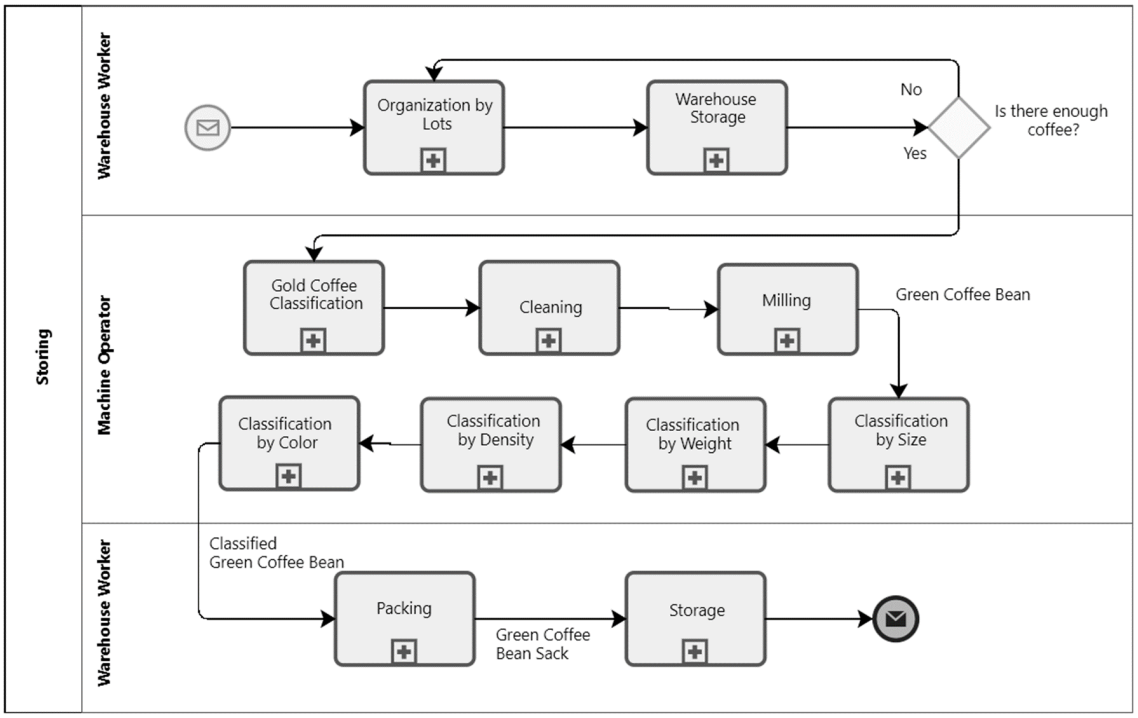


Figure 17. Storage Processes on the coffee supply chain

Figure 17 shows the coffee bags storage processes. It begin by being organized by lots and stored until enough are collected. Following classification, cleaning, and milling, the coffee is sorted by size, weight, density, and color. These classifications will set the coffee bean quality and price. As a final step, it is packaged and stored for sale or distribution.

Considering that; quality of coffee beans are susceptible to variables such as bean moisture, temperature, and water activity, these stages are crucial. In the best case scenario, 100 kilograms of cherry coffee will yield 13 kilograms of commercial coffee. BaseSC2 has a built in function to generate price and management recommendations based on expert knowledge and the Colombian National Federation of Coffee Growers (FNC by its acronym in Spanish) management and price regulations (Federación Nacional de Cafeteros, 2016).

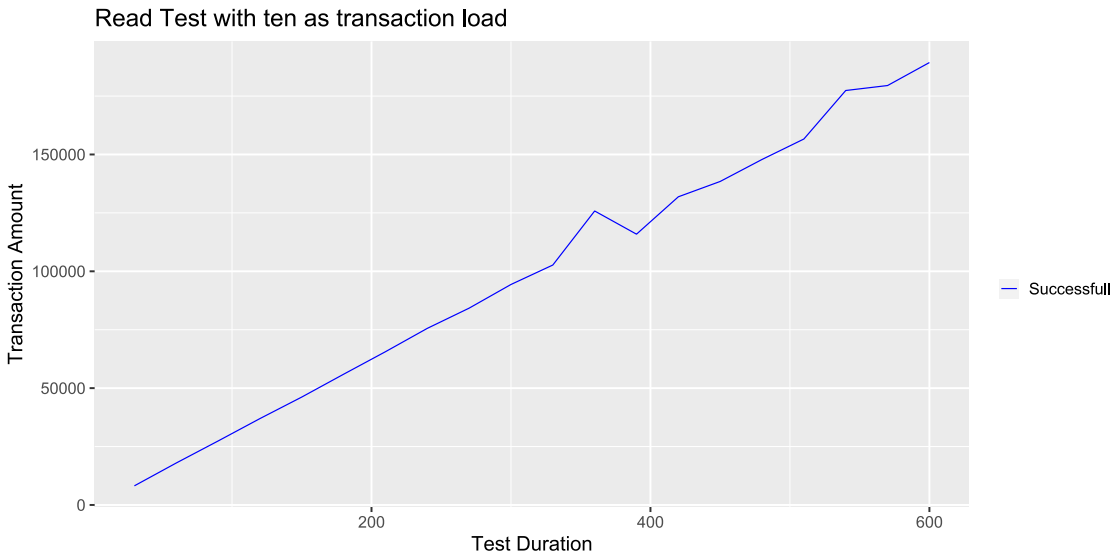


Figure 18. Read test BaseSC2

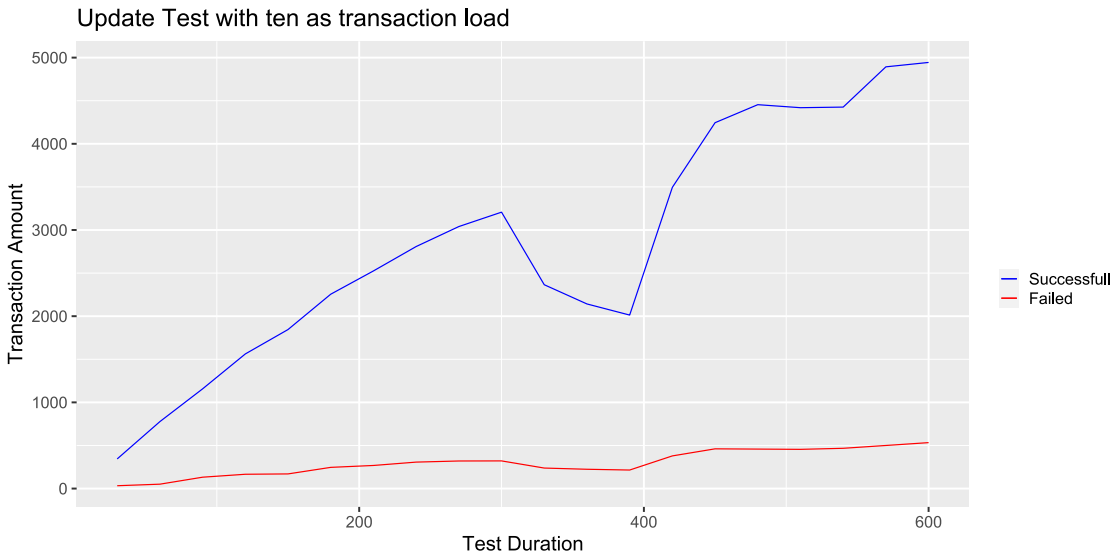


Figure 19. Update test BaseSC2

As shown in figure 18, the task of reading information about the asset resulted in many successful transactions. The test did not result in any failed transactions. Nevertheless, we can see in Figure 19 that the number of successful transactions decreased dramatically on each run while the number of failed transactions increased. As a result, we debugged the smart contract; after that, we confirmed that the decreased number of successful transactions and the increased number of failed transactions were not caused by bugs in the smart contract. As a result, we concluded that the system performed a background task during the tests; however, we decided not to repeat the tests since the results were still good enough for the smart contract research for which we were aiming.

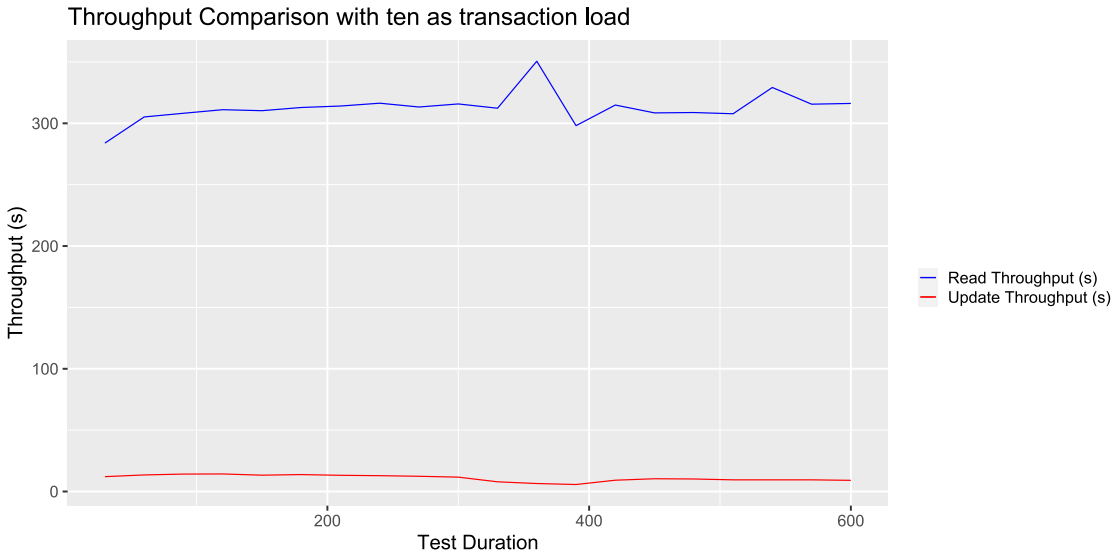


Figure 20. Throughput comparison BaseSC2

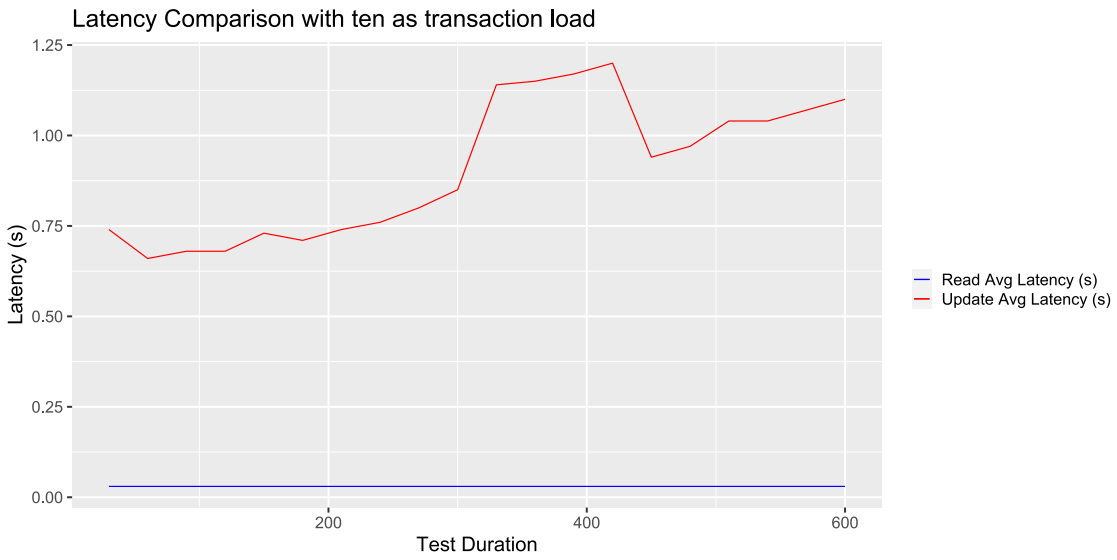


Figure 21. Latency comparison BaseSC2

Figures 20 and 21 show the Throughput and Average latency differences while sending the read and update transaction to the Blockchain network. While sending read transactions, the throughput is high, with an average of 300 transactions per second. It is fast enough to respond to the environment where the smart contract would run.

However, to update transactions while executing the smart contract, the throughput falls to an average of approximately ten transactions per second. In this case, considering the update window used on the research sites, the transaction rate per second remains sufficient to carry control of the assets. In the case of average latency, we can see in Figure 18 that the latency is constant (0.02 seconds) for the reading stage. At the same time, in the case of an update, this value goes up and remains at an average of 0.85 seconds. Still, good enough for research purposes.

Results at ***A Smart Contract for Coffee Transport and Storage with Data Validation*** published in the journal IEEE Access, vol. 10, pp. 37857-37869, 2022. Also, ***Blockchain for Supply Chain Traceability with Data Validation*** was presented at the 17th International Conference on Soft Computing Models in Industrial and Environmental Applications SOCO'22 and published as part of the Lecture Notes in Networks and Systems, vol 531, pp 156–165, 2023.

Considering the previous results, we decided to perform a test for the developed smart contracts. First, a test using simulated information for the BaseSC2 and BaseSC1 was evaluated in a real environment.

3.4. Simulated test BaseSC2

After testing BaseSC2 using the Caliper tool, we did a simulated test. In this test, we simulated coffee control data. All the simulated data was based on expert knowledge. This test was performed to evaluate the metrics implemented for data reliability.

To simulate the data sent by various sensors, we use the *simstudy* library from R. We develop an application using R Studio. Subsequently, using the *Node-Red* tool to simulate the sensor package data sending to a Hub that would later send this data to the deployed Blockchain network, two-minute time intervals between data are defined. This sequence can be seen in Figure 22.

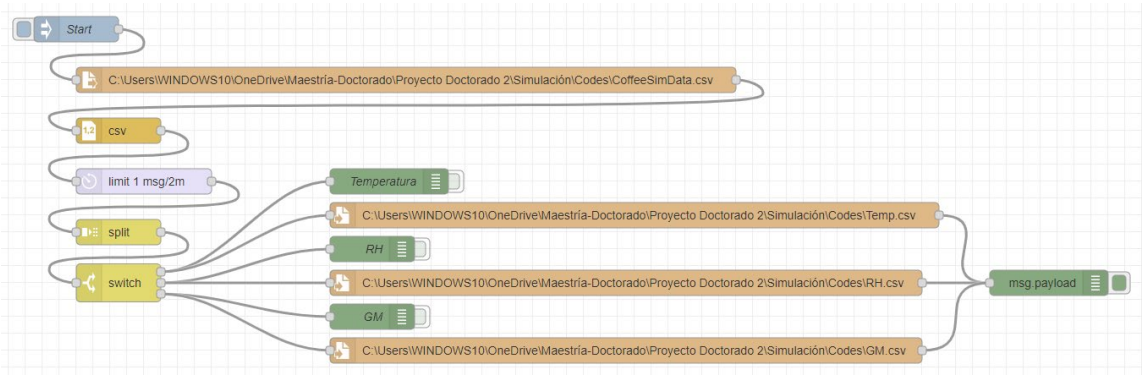


Figure 22. Sequence developed in Node-RED.

Once the simulated data is sent, a python tool developed to detect the arrival of these data packages generates the transactions that will be sent to the Blockchain network. Figure 23 summarizes the sequence described.

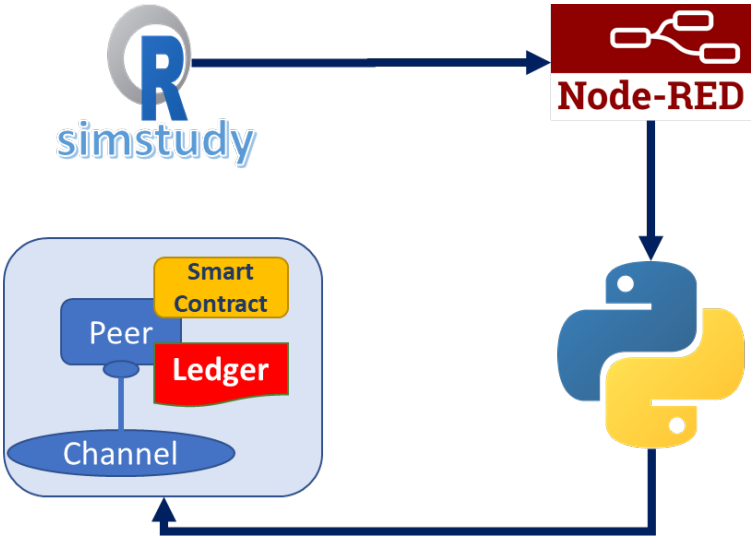


Figure 23. Tests sequence

The results obtained in this process using our developed smart contracts are the following.

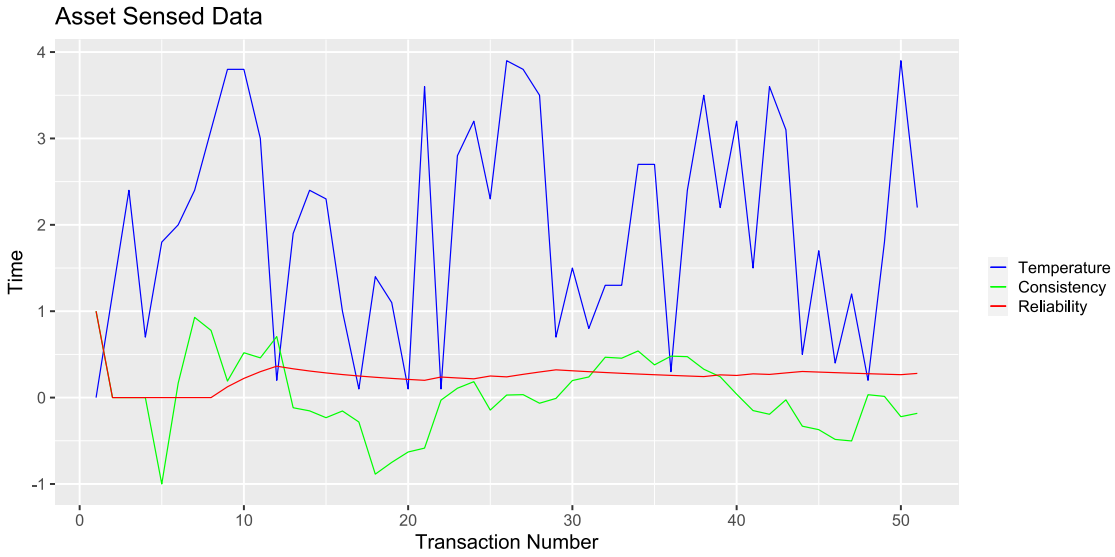


Figure 24. First test: Simulated Temperature Data

A simulation of 60 temperature data points from 0 to 4 degrees Celsius is shown in Figure 24. Although temperature changes are not drastic in most cases, we can see that the consistency (based on correlation) shows that the data does not have any. Values close to 1 will indicate that the data has consistency or is reliable, while values close to zero or below will indicate the opposite.

Nevertheless, the response from the metrics gave us a clue that they might be suitable for data with a tendency. As for reliability (based on kivenson), in this simulation, we include some random transactions with syntax anomalies; these errors affect the metric value, so for the 60 data points, the reliability was low, so the data has no reliability.

Considering this, we decided to modify the smart contract to send transactional data with a clear tendency and no errors.

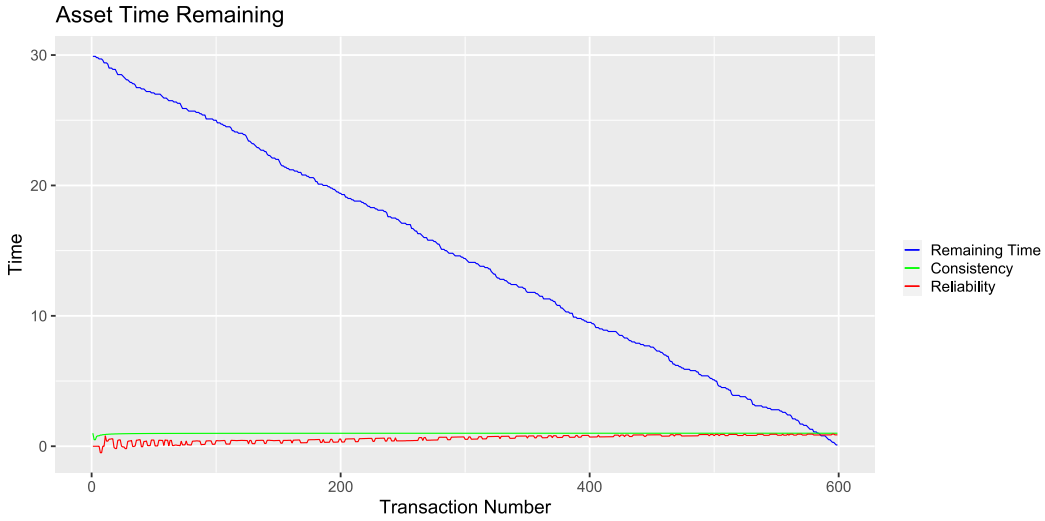


Figure 25. Second test: Simulated data with a tendency

As shown in figure 25, the two metrics tend to go up until they reach values close to 1. This means that the data is reliable because no syntax anomalies were found, and due to the precise data tendency, it has consistency. As a result, we decided to keep the metrics but consider their results based on the type of data being sensed. For now, the reliability variable seems to be the most useful.

3.5. Advantages and disadvantages

- **Advantages**

The smart contract can detect syntaxis anomalies in the traced data sent at each network transaction with the implemented strategy. It generates two metrics related to the anomalies detected. We also doted the smart contract on the use cases with the capacity to create recommendations, in this case, based on one expert's knowledge but can be extended to use more knowledge to generate better recommendations.

Smart Contracts not only detect syntaxis anomalies but also perform a correction step, keeping the last correct value on the traced variable and storing the incorrect data alongside all the necessary information so that an operator can see the detected problem and take the steps needed to avoid it in the future. Traceable partner ratings could be generated based on a combined consistency and reliability metric.

- **Disadvantages**

Not all syntaxis anomalies were considered in the implementation due to the diverse kinds of data that can be traced. I.E., date data could have other formats, and a specific design could be required for some processes. Also, different alphanumeric characters or string data structures could be used to name traceable variables, but the system might be working with alphabet names only; on these cases, syntaxis anomaly detection would be needed, but due to the multiple considerations necessary to embrace all possible syntaxis cases only numeric syntaxis anomalies were considered.

3.6. Summary

This chapter presents the development of our proposed strategy for syntaxis anomaly detection, the action of the smart base contract, and the selection of the Blockchain solutions to deploy and evaluate our development.

We introduce the smart base contract with the syntaxis anomaly detection and two metrics to evaluate data. After reviewing multiple Blockchain solutions, we

decided to use Hyperledger Fabric because there is no need to use tokens to perform a transaction, so no mining process is required. Also, Hyperledger Fabric has a Plug & Play feature that lets the users add a consensus protocol at any time without impacting the network deployment. To evaluate the developed Smart Contracts, we use Hyperledger Caliper.

Next, we deploy and evaluate the developed smart contract using Caliper to assess the stress over the Blockchain network, a simulated environment to evaluate the proposed data reliability metrics. Subsequently, a short supply chain traceability test at two production sites was launched, gathering accurate data and assessing the behavior of our smart base contracts and the deployed Blockchain network. We were able to determine the network's stability and make queries of the stored data on the Blockchain ledger to analyze the collected data and some measurements automatically performed by the smart contract related to the reliability and consistency of the data. Finally, some advantages and disadvantages of our developed syntaxis anomaly detection strategy are presented.

Chapter 4

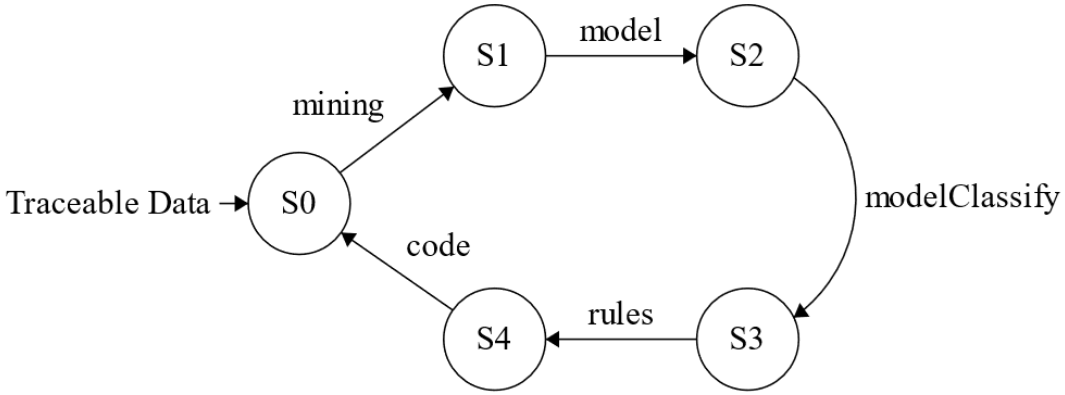
Semantic Anomaly Detection Strategy

This project's main objective is to develop an autonomous anomaly detection Blockchain traceability network using smart contracts. To perform this, we review anomalies that may arise in supply chain traceability data, anomaly detection methods, and implementations that may benefit our investigation. The semantic anomaly detection strategy is discussed in section 4.1. Section 4.2 shows the ML algorithm evaluated and the evaluation metrics used to select the best models for our test case. Section 4.3 shows the autonomous anomaly detection approaches assessed. Section 4.4 shows the test results using the modified version of the BaseSC2, adapted for artisan sweets supply chain traceability. This smart contract has rules for semantic anomaly detection. Section 4.5 shows the results of the autonomous smart contract update for semantic anomaly detections. Finally, section 4.6 offers the advantages and disadvantages of the implemented strategy.

4.1. Detection Strategy

We decided to evaluate two approaches to enable the smart contract to detect semantic anomalies. The developed approaches use the Point Anomaly Detection - Classification Method with all the data stored on the Blockchain ledger to detect an anomaly in the current transaction data. We use Z-Score (Brase & Brase, 2016) to generate the initial labels of the semantic anomalies.

The first approach trains multiple ML algorithms; after that, the data of the anomalies are used to generate rules that can detect anomalies without the need for ML models. The second approach trains a decision tree model and then exports its rules to be utilized later.



LHS		RHS
S0	→	miningS1
S1	→	modelS2
S2	→	modelClassifyS3
S3	→	rulesS4
S4	→	codeS0

Figure 26. Semantic Anomaly Detection FSM.

The general flow process of the semantic anomaly detection is shown in Figure 26 FSM where Transaction data mining (S0), Model process (S0), Modeling classification (S2), Rules generation (S3), Code generation (S4). As mentioned, the transaction data is processed with all the data stored on the Blockchain ledger. Then a mining process is applied to adjust the data to the model specifications; after that, the different models are trained and evaluated to generate the rules that can be used to give the smart contract the ability to detect the anomalies by itself.

4.2. Production Site Traceability Test BaseSC1

For our following test scenario, we deployed a set of 10 IoT Sensors on two production sites. The case scenario called **Sweet BIoT** has three components, one focused on IoT sensors and the developed interconnection app, another on the deployment of the Blockchain network, and the third component focused on the smart contracts that allow traceability processes to be conducted autonomously (see Figure 27).

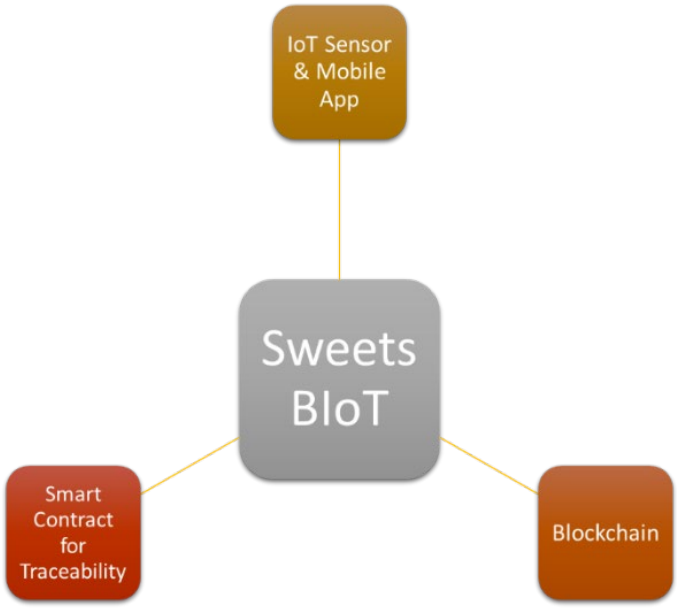


Figure 27. Sweets BIoT Components

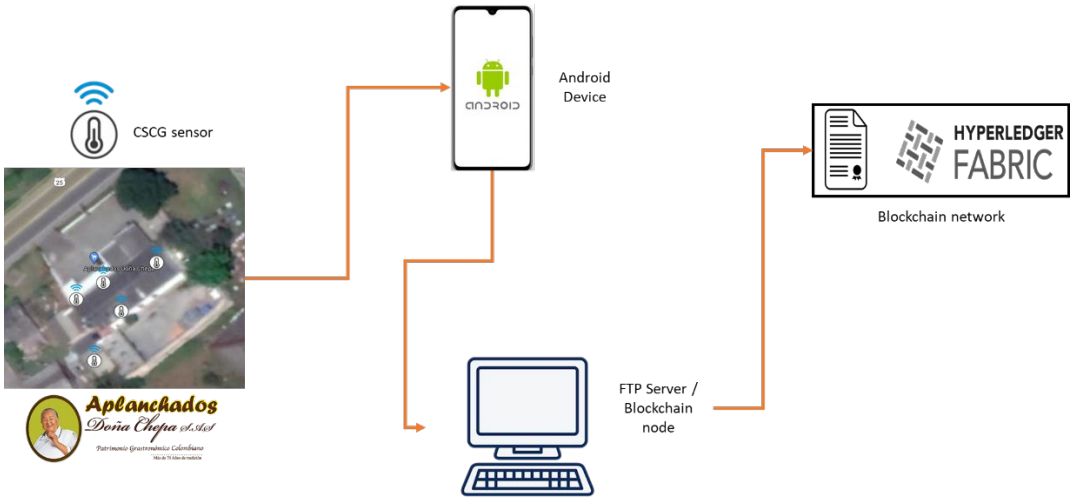


Figure 28. Sweets BIoT Architecture

Figure 28 shows the architecture of the deployed solution. At each production site a set of 5 IoT sensors were deployed, and we use an android device to download data and uploaded it to a server to be send to the Blockchain network.

A lack of appreciation and recognition may lead to the disappearance of the artisan sweets tradition in Cauca, Colombia, especially in Popayán. This prompted a study to identify the artisan sweet supply chain in the region. In figure 26, you can see the supply chain levels.

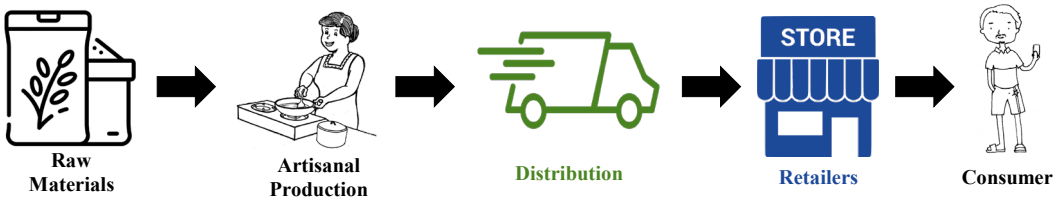


Figure 29. Artisan Sweet level 1-3 supply chain.

Figure 29 shows the three-level supply chain for artisan sweets. In level 1 of the supply chain, the producer acquires the raw materials needed to produce the sweet and then sells the sweet directly to the final consumer (black-colored stages). In level 2, the producer offers his products through retailers (blue-colored stage), either his own or third parties. At level 3, sweets are distributed to retailers (green-colored stages).

Based on the framework proposed in (Treiblmaier, 2019), users are the sweet artisan producers; they will send data related to the storage of raw materials, production processes, the storage of finished products, and their transportation. This case involves irreversible data.

Two devices will be used to create the Blockchain network's peers; each device will have one or more peers connected to a channel, depending on the number of producers. A certificate authority on the network assigns digital certificates to peers as identifiers. A blockchain channel determines a peer's rights based on its identity whenever it connects to the channel. Assets can be retrieved over the network but cannot be created or updated independently by peers. It is necessary for all peers on the network to reach a consensus; once the consensus is reached, a new block is created. Transaction fees are not required on the deployed network because of the consensus algorithm used.

To evaluate our proposal in a natural environment, we deploy a set of IoT sensors on the production site of “Aplanchados Doña Chepa,” one of the most famous producers of Artisan Sweets in the region. Kike's Kitchen was the other well-known production site selected. In our tracking system, we use 10 Elitegroup Computer Systems (ECS) tags (*CSCG Tag | ECS Global*, n.d.); these tags are equipped with ambient light, temperature, humidity, and shock/tilt sensors.

To download the sensed data from the deployed ECS tags, we develop an Android app that connects to each tag to download the data and upload the data to a File Transfer Protocol (FTP) server on the machine used to deploy the Blockchain network. The ECS tags were configured to record temperature and humidity measurements only. Once the information is uploaded to the server, it proceeds to conduct transactions on the Blockchain network. To do this, it loads into memory each file received and generates a transaction with each line of data in these files. When the transaction is sent to the network, the deployed smart

contract is executed, and it reviews the data being tried to upload to the network and the transaction's validity. Once the transaction is validated with the data, it becomes part of a block in the Blockchain network.

Next, the deployed sensors' description, location, and minimum capacities required for using the Android App are related on table 9.

Table 9. Deployed equipment.

Description	Quantity	Location / Minimum software requirements
IoT temperature and humidity sensors. Models: GWS-CSCG Logistic Monitoring Tag	10	**Doña Chepa**: Raw Material Storage, Finished Product Storage, Production Zone 1, Production Zone 2, Transport. **Kike's Kitchen:** Sales, Refrigerator 1, Refrigerator 2, Production Area, Raw Material
Android Mobile Device	1	Must have Android version 9.0 and at least 2Gb of RAM.
Desktop Server	1	Configure as FTP.
PC	1	To run the developed tools to deploy the network and sent data transactions

Equipment deployed during the production site pilot test.

Table 9 show the equipment used for the pilot test to evaluate our proposal in a natural environment. The tags were located on key locations inside the production zones and used for the tracking system; each tag has ambient light, temperature, humidity, and shock/tilt sensors. We use a python developed code to autonomously send transactions to the Blockchain network only for temperature and humidity recollected data on the FTP server using the smart contracts.

Figure 30 show the structure used to develop our strategy to the autonomous update of the smart contract, in order to use one of the most important features of the Blockchain network, its decentralization. Each module will have a task to perform during all the network execution time as we can see resume on the figure.

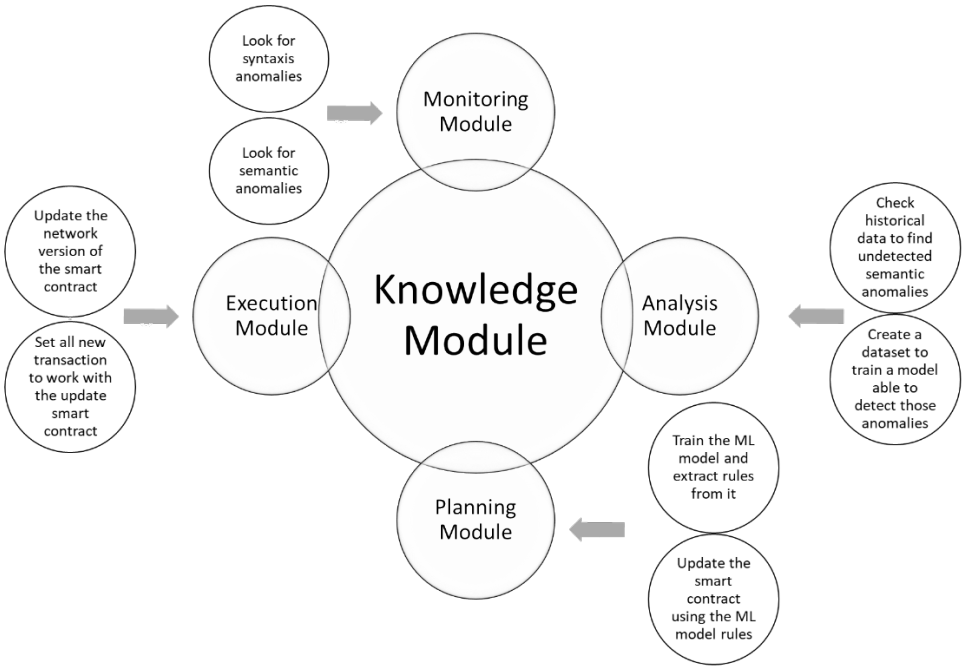


Figure 30. MAPE-K framework applied to our proposed solution

The smart contracts have five critical components based on a MAPE-K (Arcaini et al., 2015). As mentioned before, MAPE-K framework is ideal to use on a Blockchain network since it can be decentralized. We will have a knowledge module that will store all the relevant information collected and generated by the other four modules. Errors, anomalies, or risks represent the Knowledge module that the ECS tags sensed data could show at any transaction. The Knowledge module will be each block in terms of the Blockchain network.

The Monitoring & Analyze module will check each transaction made on the Blockchain network. These modules will be part of the Smart Contract development and will look for anomalies in the data stored in the private storage of the Smart Contract and on every new transaction made using the specific Smart Contract. The module output will be all the anomalies and threats detected, and as part of the Smart Contract, all this information will be stored on the Blockchain network. The Monitoring module is represented by the functions that collect all the values sent at any transaction on the Blockchain network. After that, the Analyze module functions tries to find the previously defined error patterns; if an error is detected, update the error entity on the network using the detected error for later analysis and the target entity will be is updated using the previous state. The target entity is updated using the current transaction value if no error is found.

The Planning module will create autonomously new Smart Contracts to validate the data anomalies based on the anomalies found by the Monitoring & Analyze Module. This module will be outside the Blockchain network, but it will consult the anomalies directly using the Smart Contract. To do this, the Smart Contract will store all the anomalies found at each transaction in the Blockchain blocks. On

the other hand, the planning module will also define management recommendations, and risk alerts based on the potential risks found. This part of the module will be part of the Smart Contract. Finally, the Execution module will update the Smart Contract on the Blockchain network with the new one generated in the Planning module.

All the information on the Blockchain is exported to plain text files for the Planning and Execution modules, to be used as the latest information in the knowledge module and to generate new rules for the detection of anomalies.

Next, the test results for the adjusted BaseSC1 are presented. This contract was adjusted to work with artisan sweets data. The contract can self-check the transactional data looking for syntax errors in all cases. Also, it compares the newly entered data with the optimal values agreed by members to check if any recommendation must be issued.

As mentioned before our focus is on Throughput and Latency mainly because the smart contract must validate the growing amount of data to be stored in the network. This proposal does not consider other challenges because it requires more development and a broader deployment for assessment.

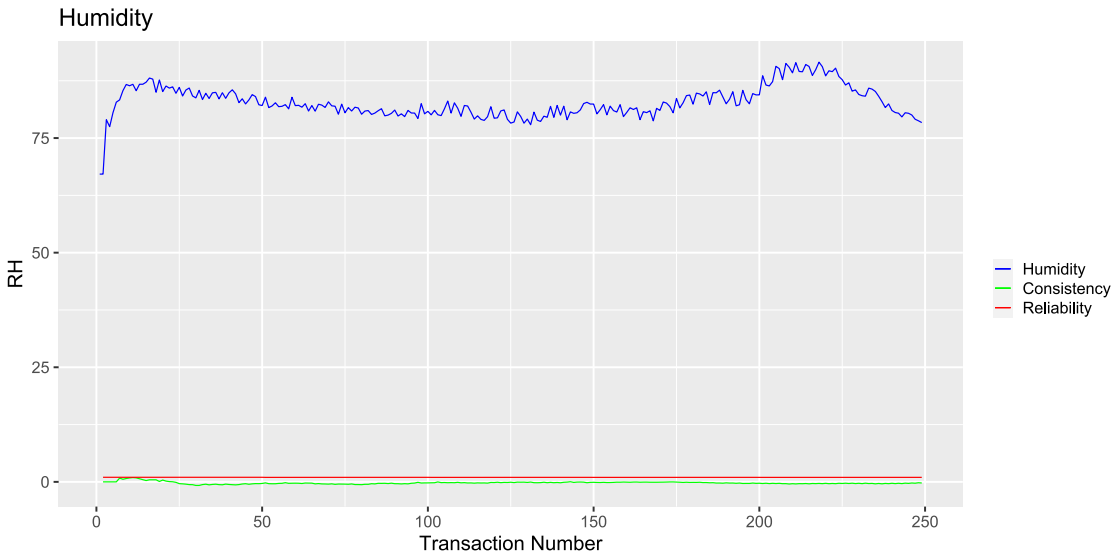


Figure 31. Kike's Kitchen humidity sensed data.

In figure 31, we can see the humidity, consistency, and reliability values during the test. The Consistency and Reliability values calculations are made inside the smart contracts using the new data available on the recent transactions and the previously stored data from the old transactions. These transactions will also include the ones made by previous versions of the smart contracts. Humidity begins at the initialized value, goes up to current levels, and varies between 75 and 80%, occasionally reaching this maximum value. Reliability always remains at one throughout the test duration. This means that the humidity transactions did

not present anomalous values in terms of syntax or values below the limits established by the producer.

Nevertheless, the values that exceed 80% RH could be considered semantic anomalies on the data, this type of anomaly will be detected once the new versions of the smart contracts with semantic anomaly detection are deployed. In the case of consistency, we can see that it varies between 0 and -1, finally remaining at a value close to zero; that is, the data does not have consistency. As mentioned before, consistency correlates with the data sent to the network. It can be seen how the humidity variations affect the consistency measure, which presents significant variations throughout the execution time.

Figure 32 shows the same results for the temperature data. In this case, we decided to put a limit at 16 degrees Celsius to evaluate the reliability measurement using only the data of the ECS tag located on the final product storage unit. So, the temperature data should be between 16 and 25 degrees Celsius to be considered a suitable storage temperature; it should be clarified that this limit does not represent the reality of the storage temperature of the production site, current temperatures at the region would not endanger the product.

In this case, the reliability goes up from zero initially, but after the temperature drops below the 16 degrees limit, it rapidly goes to 0. Also, we see that the consistency stays close to 1 when the temperature data continuously drop. Still, at the beginning, when the temperature data drop and then goes up, and at the end, when the temperature starts to go up again, the consistency drops to zero. Considering the previous results, we perform the same process using the transport tag. The results are shown in Figures 33-35.

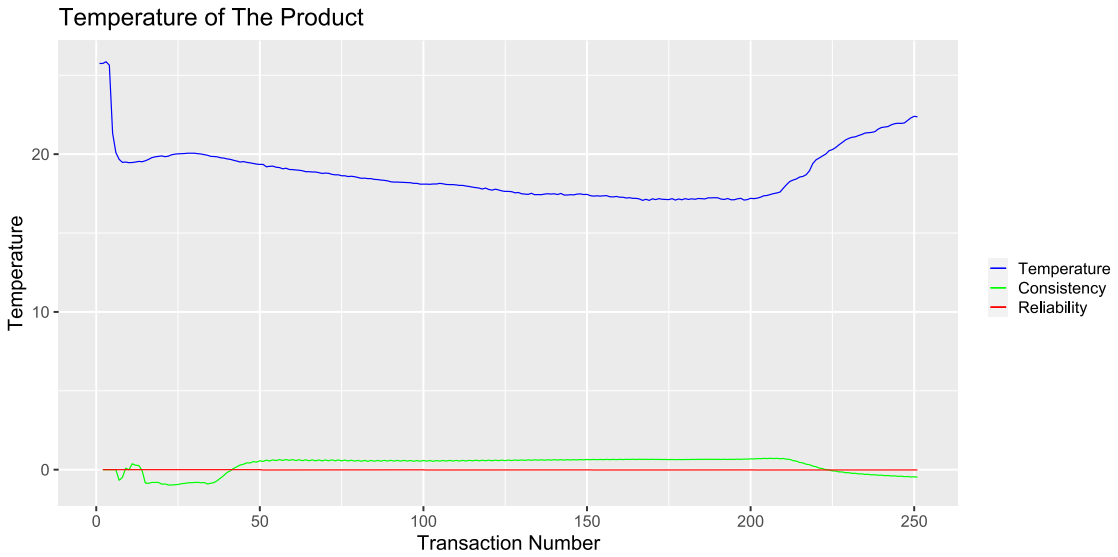


Figure 32. Kike's Kitchen temperature sensed data.

Figure 33 and 34 shows the humidity and temperature values during the transport stage. In this case, the behavior is like the results found during the storage stage in consistency and reliability measurement, although the humidity values and behavior are different. Also, figure 34 shows that in the case of transport where no temperature control has been stipulated, the reliability value stays at 1. The consistency ones again go from 0 to 1 once the temperature values start to drop and decrease after the temperature increases.

In these cases, we see that the reliability values depend on the agreed terms because no syntax errors were found in the data collected from the sensors.

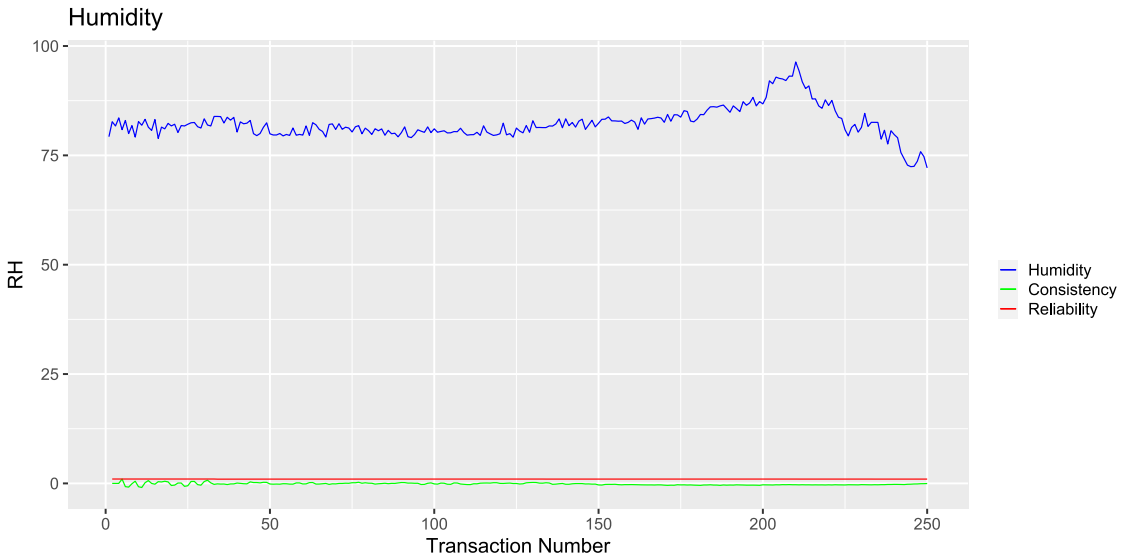


Figure 33. Doña Chepa transport unit sensed humidity.

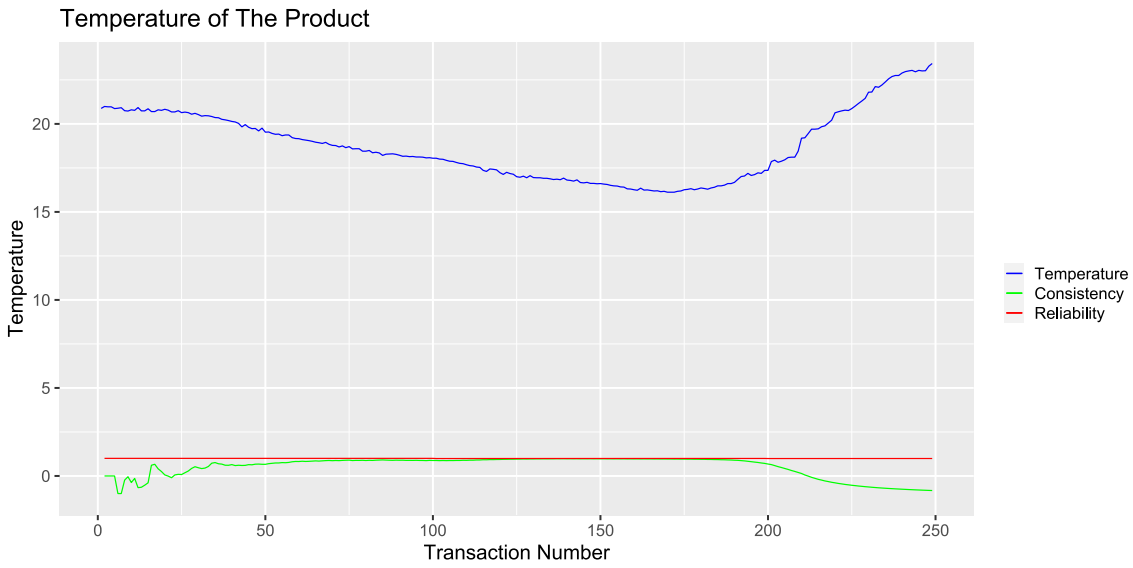


Figure 34. Doña Chepa transport unit sensed temperature

Figure 35 shows the results for the shock-sensor data, only available for the transport tag. In this case, we can see that the sensor detects a shock at the final stage of the transport process, going up from 0 to 1. Due to this, the consistency measurement stays at zero, and the reliability value remains at one throughout the test. For this case, a semantic error will be anything below 0, above 1, or between those two. Figure 35 shows that the product shook firmly to activate the sensor response.

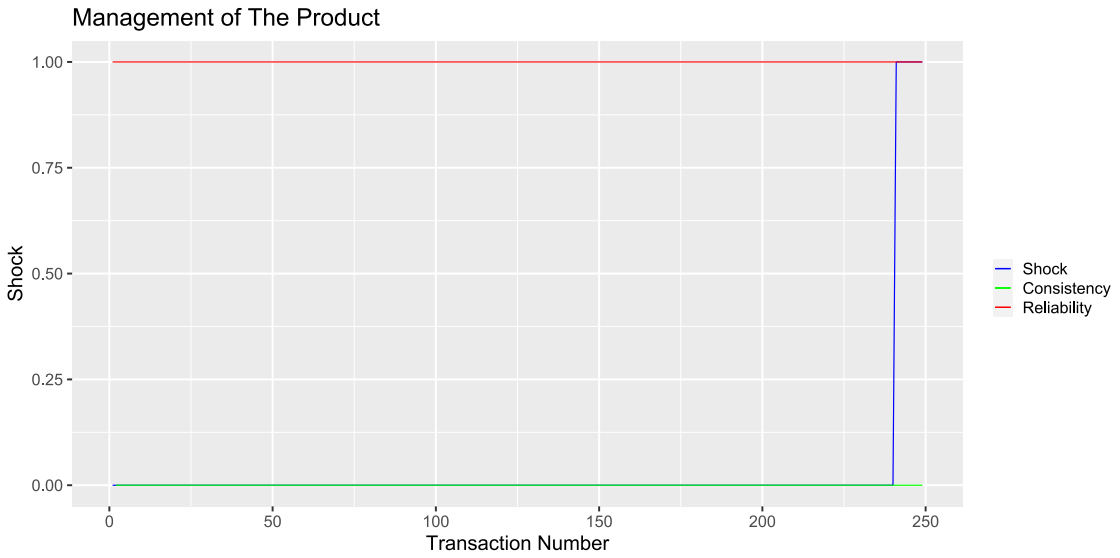


Figure 35. Doña Chepa sensed shock data at transport unit.

Results at ***Smart Contract to Traceability of Food Social Selling*** to be published in the journal CMC-Computers, Materials & Continua ISSN:1546-2226.

4.3. ML Anomaly Detection Training/Evaluation

To give the smart contract the ability to detect semantic anomalies, we developed on Python a module that ask the ledger for all the data related to a traced good and perform a data mining process on the store data. This mining process is focused on the detection of outliers in the data stream.

Using the scikit learn python module (Pedregosa et al., 2011) we build a set of functions that transform the stored data from the Blockchain ledger to data that can be used in an ML process. During this process, we evaluate several ML algorithms, always considering that the selected ones should not consume many computational resources even when large data arrays are being processed.

A. K-Nearest Neighbors (KNN)

Since it relies on the estimated distance, typically normalizing the data can improve the result accuracy, but it will depend on the processed data (Piryonesi & El-Diraby, 2020). KNN for outlier detection uses the distance from the current point to its kth nearest neighbor and use it as an outlier score.

B. Local Outlier Factor (LOF)

It is an algorithm for finding anomalies by measuring the local deviation of a data point according to its neighbors (Breunig et al., 2000). The anomaly score of each sample is called the Local Outlier Factor; it depends on how isolated the data point is concerning the surrounding, using the locality given by the k-nearest neighbors.

C. Principal Component Analysis (PCA)

PCA is a linear dimensionality reduction method that uses Singular Value Decomposition to project the current data to a lower dimensional space. (Tipping & Bishop, 1999). Therefore, a new low dimensional space can have most of the data variance, so outlier scores are estimated by the sum of the projected distance of a sample on all eigenvectors

D. Isolation Forest

By randomly selecting a feature and then randomly selecting a split value between its maximum and minimum values, the Isolation Forest isolates observations (F. T. Liu et al., 2012). Using a tree structure to represent recursive partitioning, the number of splitting required to isolate a sample equals the path length from the root node to the terminating node, the path length is a measure of normality.

E. Rotation-based Outlier Detector (ROD)

Using the Rodrigues rotation formula, rotation-based Outlier Detection (ROD) works intuitively in three-dimensional space since it requires no statistical distribution assumptions (Almardeny et al., 2020). Three-dimensional vectors representing data points are rotated two times counterclockwise around the geometric median. By analyzing their volumes as cost functions, the outlying scores are calculated using the median absolute deviations of the parallelepiped that results from the rotation. By averaging the overall 3D-subspace scores that result from decomposing the original data space, the overall score for high dimensions (more than 3) is calculated.

F. Random Forest

Each tree in the forest is based on the values of a random independent vector and has the same distribution (Breiman, 2001). The response with the most votes is usually the one submitted by the forest since each tree produces only one vote in response to input X.

G. AdaBoost

Adaptive Boosting is an algorithm that fits a classifier on a dataset and then replicates it on the same dataset, but adjusts weights based on wrong classifications so that subsequent classifiers focus more on difficult cases (Freund & Schapire, 1997). AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified.

H. Bagging

An assembly technique known as bootstrapping or bagging implies that each model trained in the assembly is given the same weight when trying to solve a classification or regression problem. In contrast to stacking, which allows the use of diverse types of algorithms, Bagging uses multiple versions of the same algorithm (Breiman, 1996). For regression, it returns the average value of the prediction for each version of the regressor, whereas, for classification, it is decided by a vote.

I. Decision Tree (DT)

Is a non-parametric supervised learning method used for classification and regression task (Alpaydin, 2010). A key feature of DT is that they represent rules, which are easy to understand and can be applied to languages such as SQL to access and classify databases.

To evaluate the ML algorithm, we use the following metrics:

Accuracy: This is the fraction of predictions our model got right. Defined as follows:

$$Accuracy = \frac{Number\ of\ correct\ preductions}{Total\ number\ of\ predictions}$$

Equation 5. Accuracy definition

Also, for binary classification problems:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 6. Accuracy definition for binary problems

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Recall: This metric measures the proportion of actual positives correctly classified and defined as follows.

$$Recall = \frac{TP}{TP + FN}$$

Equation 7. Recall definition

Precision: This is a metric that measures the proportion of identification that was correct. It is defined as follows.

$$Precision = \frac{TP}{TP + FP}$$

Equation 8. Precision definition

4.4. Autonomous Anomaly Detection

Finally, after selecting the best algorithm, we try two rules generation methods, one based on the classification made by multiple ML algorithms and the other translating the decision rules generated by the decision tree algorithm to Go and JavaScript. Next, we show to talk about these two methods.

Both versions use the standard score to label anomalies if needed. Z-Scores are standard deviations by which the value of data points differs from the mean value of what is being observed.

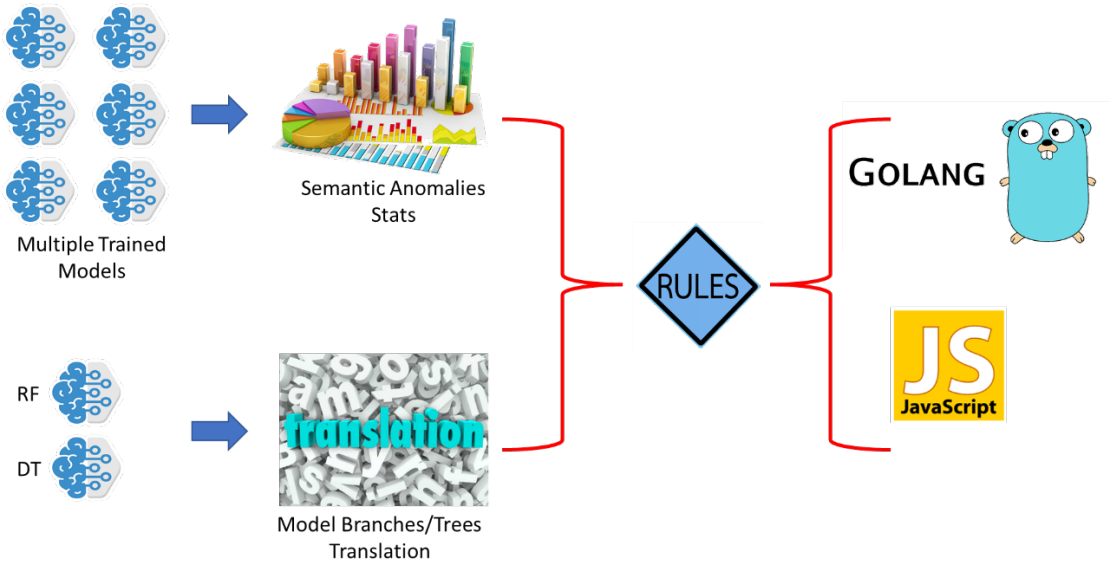


Figure 36. Autonomous semantic anomaly detection approximations.

Figure 36 show the two proposed approximations to obtain rules from the trained machine learning models; one using multiple trained models and the second one using a RF or a DT model. Each approximation has different methods, but the main goal is to obtain rules that can be expressed in Golang or JavaScript code.

A. Multi-Algorithm rules generation approach

To generate rules using the multi-algorithm approach, we constructed a set of rules based on the instances classified as anomalies by the trained models with the best overall response. Using the function to acquire statistics values from all the cases, this module creates conditionals rules to separate anomalies from average data points.

Nevertheless, this approach was discarded after reviewing the classification results of the algorithms. Some rules that could be generated after checking the result of one algorithm were contrary to the rules that could be developed with the results of another of the trained models.

In December 2021 *golearn* package was published, this package tries to bring machine learning into the Go language, but it is still on v0.0. On the other hand, JavaScript has an interesting library to use *TensorFlow* models. Still, it supports web browsers and Node.js execution, so it cannot be called directly on the smart contract. *Ml5js* is another machine-learning library for JavaScript developed for web browser usage. Considering this and because the supported languages on Hyperledger Fabric, Go, and JavaScript does not have a library for ML model generation and execution supported by the Hyperledger Fabric environment, we decided to evaluate a second approach using Random Forest and Decision Tree.

B. Random Forest / Decision Tree rules translation approach

In this approach we use the readability feature of decision trees to translate the tree rules into code rules. To do this, we write a python function that receives a previously trained model from memory or trains a new decision tree if needed. After that, the function exports the tree rules to a pseudo-code like the one used in both versions of the deployed smart contracts.

Once we got the pseudo code, we converted it to Go and JavaScript code. Finally, using previously defined flags on the smart base contracts, the tools know where to place the new code that will give the smart contract the ability to detect anomalies. With this, the smart contract can autonomously update.

To begin with the autonomous update, a predefined number of transactions must be made on the Blockchain network to ensure enough data to perform the anomaly detection training. So, this tool periodically sends transactions to the network consulting the current number of traceability transactions made. When the traceability transactions reach the predefined number, the agency sent a transaction asking for the historical data on the Blockchain ledger. Once the information is downloaded, the module starts its execution, and the smart contract update process begins.

4.5. Anomaly Detection Update Test

Using the acquired data from the production sites, we decided to train multiple ML models to be used as input in the anomaly detection test. But as we mentioned before, some rules generated by analyzing the results of said models were some rules generated using a model were contrary to another set of rules developed using another model. Nevertheless, we decided to show the complete results of this process and the final implementation for the autonomous update of the deployed smart contract.

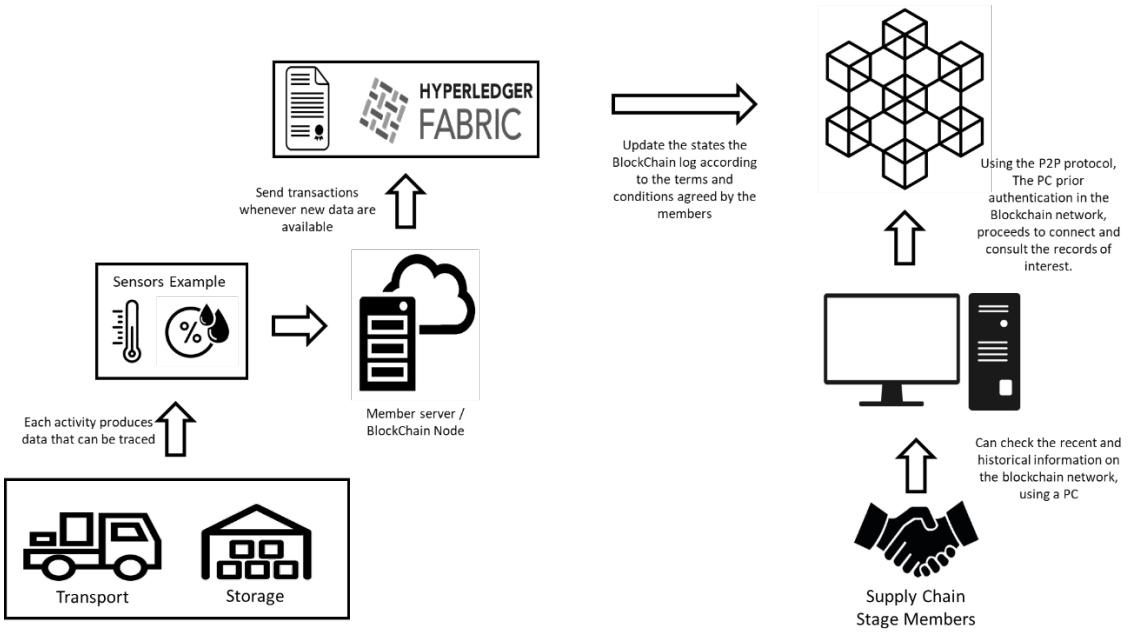


Figure 37. Interaction with the Blockchain network.

Figure 37 show the interaction with the network, we got our tags on the productions site measuring the selected variables, all the data is stored on the FTP server and the blockchain ledger and using another pc connected to the network we check the data to perform the semantic anomaly detection test with the ML algorithms mentioned above, we perform a short data mining process to adapt the data stored on the Blockchain ledger based on the characteristics of the tested algorithms.

The developed tool is programmed to send a transaction requesting the historical information of the traced asset(s); the Blockchain network response will be a message with all the available information. Once the information is downloaded to a plain text file, the tool loads that file and processes the data to generate a dataset that will be stored on a CSV file. Due to the low amount of information stored, we use the date and hour as input variables; the tool generates the dataset with the current independent attributes: Temperature, Humidity, Year, Month, Day, Hour, Minutes, and SensorId.

As a dependent attribute, the tool creates a new binary variable called Outlier, which will label all instances that can be considered semantic anomalies in the downloaded data. The dependent attribute Outlier is created using the Z-Score value, any data point with a score of 3 and above will be considered an outlier and labeled as 1, and any other case will be labeled as 0.

Table 10. Dataset example

Temperature	Humidity	Year	Month	Day	Hour	Minutes	SensorId	Outlier
26.500	88.120	2021	12	14	19	34	6	0
25.170	77.710	2022	1	26	5	45	89	0
24.250	83.270	2021	12	14	19	44	6	0
27.570	70.620	2022	1	26	8	37	89	1
24.250	83.270	2021	12	14	19	44	18	0

Example of the constructed dataset instances.

Table 10 show example instances of the constructed dataset using temperature, humidity and date data. Once the dataset is completed, we trained all the ML models using the default parameters and evaluate them to select the best models to generate the rules.

Table 11. ML models Evaluation

Model	Accuracy	Recall	Precision
K-Nearest Neighbors	0.95	0.52	0.51
Local Outlier Factor	0.92	0.17	0.16
Local Outlier Factor 2	0.95	0.04	0.60
Principal Component Analysis	0.98	0.77	0.75
Isolation Forest	0.61	0.97	0.10
Isolation Forest 2	0.95	0.52	0.50
Rotation-based Outlier Detector	0.92	0.15	0.15
Random Forest	0.99	0.99	0.99
AdaBoost	0.99	0.95	0.98
Bagging	0.99	0.91	0.88
Decision Tree	0.99	0.99	0.99

Example of the constructed dataset instances.

Table 11 shows the evaluation metrics results for the selected models. Most of the models chosen have high accuracy but fail on recall and precision values; a good model must have a high value on these three values to ensure that its response will be accurate in most cases.

Local Outlier Factor 2 and Isolation Forest 2 are implementations of these algorithms PyOD an outlier detection specialized module for Python. We decide to test these two algorithms implementations given the different parameter configuration offered hoping to get better results. Nevertheless, according to the evaluation metrics, the best models are Random Forest and Decision Trees.

Table 12 shows the training and classification time for the best ML-trained models based on this result.

Table 12. Selected ML Training/Classification time

Model	Training time (s)	Classification time (s)
PCA	2.2	2.4
Random Forest	33.2	4.4
AdaBoost	65.2	14.8
Bagging	4.9	2.8
Decision Tree	3.5	2.2

Training and classification time measurements.

Table 12 shows that the training time using 70% of the total data collected on the production sites (700k instances approx.) will not take too much time, although this will depend on the traceability process; for our specific case, it will be fast enough. The classification time is measured using 30% of the total data (around 300k instances approx.).

We initially wanted to use all the selected models to generate the rules for the smart contract, but some rules were not clear enough to cause them automatically. Depending on the model, an anomaly could be over or under a specific value in temperature or humidity, but another model may produce a rule that contradicts this. Based on this finding, we convert Random Forest trees and Decision Tree "trees" to Go and JavaScript code for use in smart contracts. Since the smart contract must be auditable, only the Decision Tree was used in this test because the semantic anomaly detection using the Random Forest rules would be too extensive due to its 100 trees in the default configuration. Nevertheless, the development is complete and can be used at any moment before the tool's deployment. The semantic anomalies dataset construction tool can be found on the annexes, the rules generation tool, and all the development for this project.

We decided to deploy and send transactions on the Blockchain network using the production site sensed data to evaluate the autonomous smart contract update process. After a predefined number of transactions, the tool will generate the first smart contract update, we will use the number of successful transactions, and the number of semantic anomalies detected as indicators for this test.

To run the test, the following configuration files must be edited:

```
- Details:
  transactionNumber: 200
  KeepUp: "keepup"
  SmartContract: "Sweets"
  SCFolder: "D:\\HyperLedger\\Go\\ChainCodes"
  SmartContractGO: "go\\main.go"
  SmartContractJS: "js\\lib\\smartSweets.js"
```

```
Version: 1.0
FTPdir: "93.XXX.XXX.110"
FTPName: "XXXXX"
FTPPass: "XXXXX"
```

The parameter **transactionNumber** sets the number of successful transactions needed to start the autonomous smart contract update process. The **KeepUp** parameter sets the tool behavior when the test ends, "*Keepup*" to keep the Blockchain network running; any other values will shut down the network and delete all files. **SmartContract** option tells the tool the name of the smart contract to be installed once the network is deployed. **SCFolder** is the path where the chain code is to be installed in the store, **SmartContractGO** and **SmartContractJS** are the relative paths to the files that must be modified to update the smart contract autonomously. The **Version** option sets the initial version of the smart contract to be deployed. Finally, **FTPdir**, **FTPName,** and **FTPPass** are the necessary credentials to connect to the FTP server to access the sensed data.

Once the configuration file has been edited, the test can run on a command console **python main.py**. The tool will deploy the network if it is not up and install the smart contract. After the number of successful transactions is reached it will generate a set of files to analyze the simulation results up to that point. Next, we will show the results obtained for 20 iterations of the **transactionNumber.**

Table 13. Simulation results

Number of transactions	Confirmed Anomalies	Detected Anomalies	Accuracy (%)	Smart Contract Version
100	4	0	0	1.0
200	11	11	100	2.0
300	18	18	100	3.0
400	22	22	100	4.0
500	25	25	100	5.0
600	28	27	96	6.0
700	31	30	97	7.0
800	37	36	97	8.0
900	43	42	98	9.0
1000	47	43	91	10.0
1100	55	51	93	11.0
1200	62	58	94	12.0
1300	65	61	94	13.0
1400	74	72	97	14.0
1500	82	77	94	15.0
1600	86	83	97	16.0
1700	93	90	97	17.0

1800	98	93	95	18.0
1900	101	96	96	19.0
2000	106	101	96	20.0

Results obtained after performing 2000 transactions on the network

The semantic anomalies detected by the smart contract with each autonomous update are shown in Table 13. After the update process, the smart contract could see most anomalies in the next batch of data. Even though the previous test indicates that the Decision Tree has a 0.99 accuracy, those results were obtained using all the data collected at the production sites. In this case, however, the tool only uses the data stored in the Blockchain ledger, so anomaly detection could improve with each iteration if no new anomaly types are sent. Still, even so, the accuracy of the smart contract when detecting anomalies with each data batch is high. Also, with each iteration, the number of rules used to detect anomalies will increase, making the semantic anomaly function more robust and trustworthy. The mean accuracy considering the smart base contract, is 92%. Still, considering only after the V2.0 or the first anomaly detection capable smart contract, the accuracy is 96%, which is an excellent accuracy level.

Data collection and dataset generation using the 10 sensors took up to 45 minutes during this test, while rules generation and smart contract update transactions took around 1 second. We limit the data collection from the Blockchain ledger to only those relevant for dataset generation and anomaly detection confirmation, which reduces the time to a mean of 20 minutes per iteration. It does not mean that the node running the update tool will not send more transactions during this period or that the blockchain ledger will be paused until the smart contract update transaction begins. Nevertheless, an improved version of the smart contract could potentially reduce the data collection time for each variable.

Considering the above, an online training model could better replace the original Decision Tree model. To do so, the system will have to execute a historical data request transaction for training the model and an update transaction for updating the smart contract for each traced data transaction. As a result of such a process, the network will be overloaded, and the nodes will be able to make fewer transactions.

The autonomous smart contract update results show that our approach is valid and can be used in real environments. Although some detected anomalies after the update of the smart contract could be average data points after a change or update in the traceability process, and an updated version of the same contract will stop detecting those data as anomalies, it will be essential to define the optimal transaction limit to autonomously update the smart contract by performing

a comprehensive study of the traceability process to apply the developed strategy.

4.6. Modified BaseSC1 (MBSC1) Test

To perform this test, we started with the BaseSC1 and adapted it for semantic anomaly detection. We hope to find semantic anomalies naturally using real production sensed data and perform the best possible test for our development.

First, we compare BaseSC1 syntaxis anomaly detection only with MBSC1, with syntaxis and semantic anomaly detection using six workers, 10 and 20, as transaction loads with a fixed load driver and test duration of 60 to 600 seconds.

Hyperledger Caliper MBSC1

Figure 39 and 39 show the results of the reading test on both smart contracts versions. The results are similar because the asset creation and consult parts are the same. Other tasks running in the background on the host OS can cause slight differences.

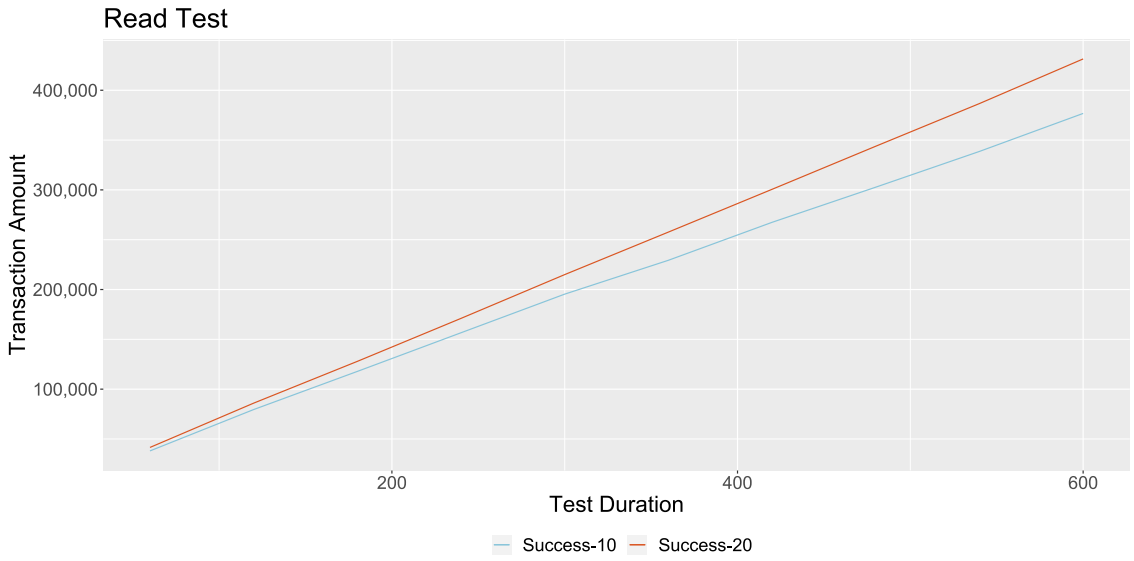


Figure 38. Read results without the semantic anomaly detection.

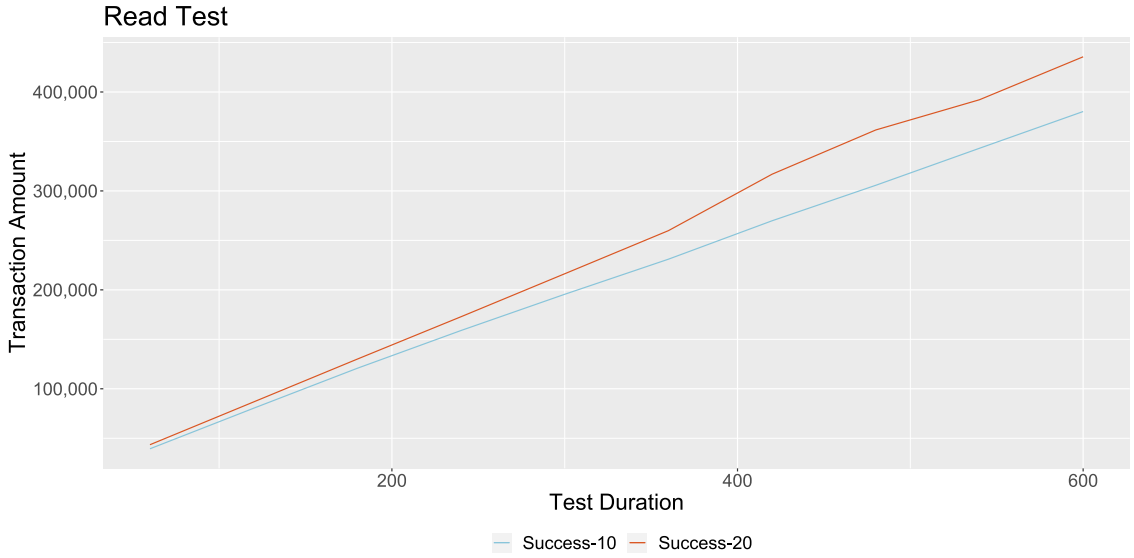


Figure 39. Read results with the semantic anomaly detection.

Figure 40 and 41 show the results for the update test on this case with the syntaxis anomaly detection only and the next one with the syntaxis and semantic anomaly detection. As we can see, the results are similar to what we expected based on the previous Caliper results; there is an increase in the number of failed transactions over time in both cases, but the results are similar, so the addition of the semantic detection function does not seem to impact the Blockchain network negatively.

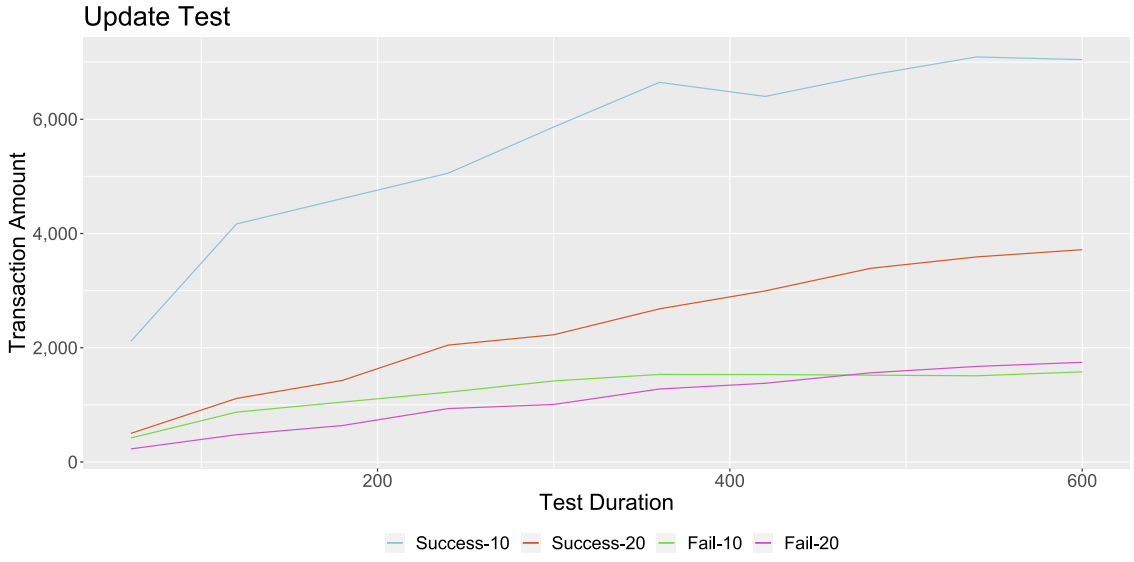


Figure 40. Update results without the semantic anomaly detection.

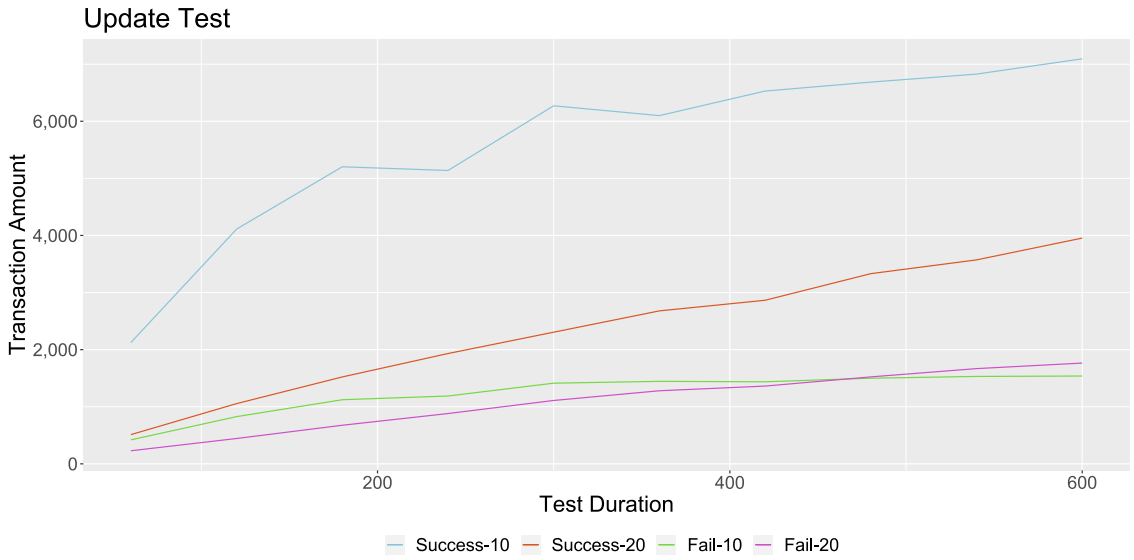


Figure 41. Update results with the semantic anomaly detection.

There is a clear difference between 10 and 20 as transaction loads in this case. During the reading test, the results were close between the two metrics. However, due to anomaly detection procedures, the transaction process took a little longer. Hence, the driver reduced the number of transactions to maintain the selected transaction load on the Blockchain network. However, the behavior of both tests using the smart contract with and without the semantic anomaly detection function is remarkably similar, strengthening the conclusion that this function does not negatively impact the network.

Figures 42- 45 show the latency and throughput results for both smart contract versions. The results confirm what we see on the Read and Update test results. The read-throughput average for both cases is around 350 TPS with an average latency of 0.01 seconds. The updated throughput average in both cases is around 18 TPS with an average latency of 0.8 seconds. Although there is a significant difference in the average values in both cases is enough for the proposed traceability task.

Figure 42. Latency results without the semantic anomaly detection.

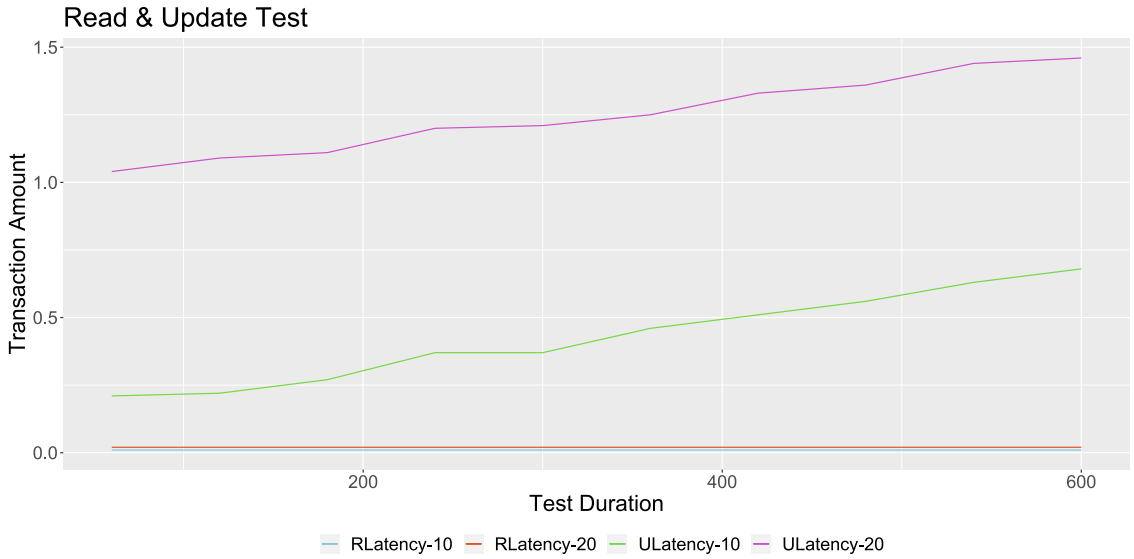


Figure 43. Latency results with the semantic anomaly detection.

It is evident from the latency and throughput results that they have a similar pattern throughout all tests, increasing over time. This illustrates the fact that the amount of stored data will consistently put more work into the semantic anomaly detection functions, the reliability estimation functions, and the recommendation functions built into the contract. Given that for most of this process, the smart contract must consult all the past data of variables. The contract must include limitations on how far back to check for semantic anomaly detection updates, management and price recommendations, and reliability measurements in order to ensure an accurate estimation of these functions, and that the load on the Blockchain network will remain optimal after a long operation period.

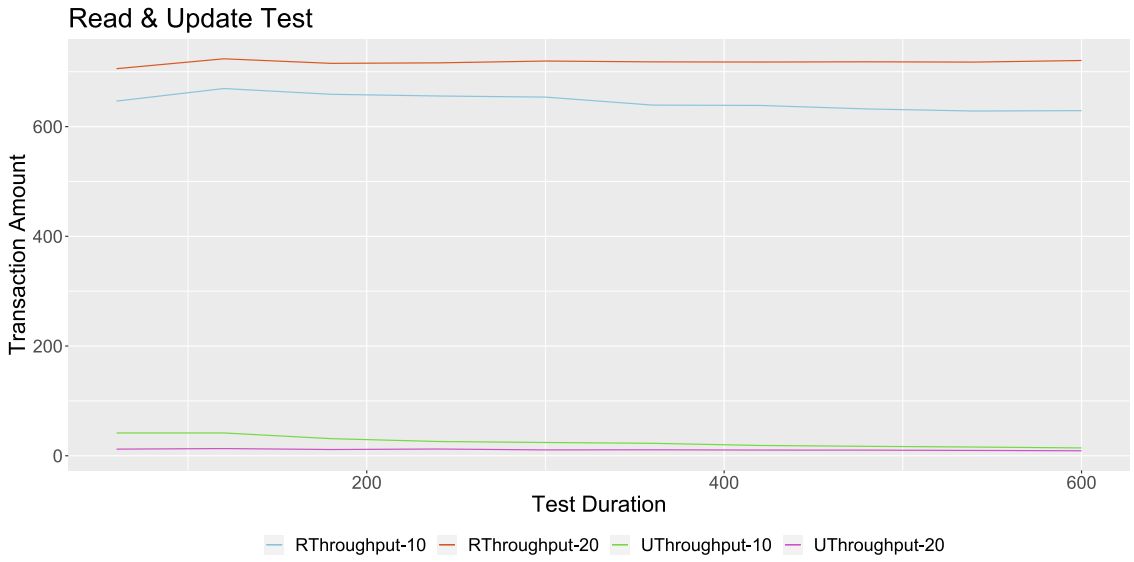


Figure 44. Throughput results without the semantic anomaly detection.

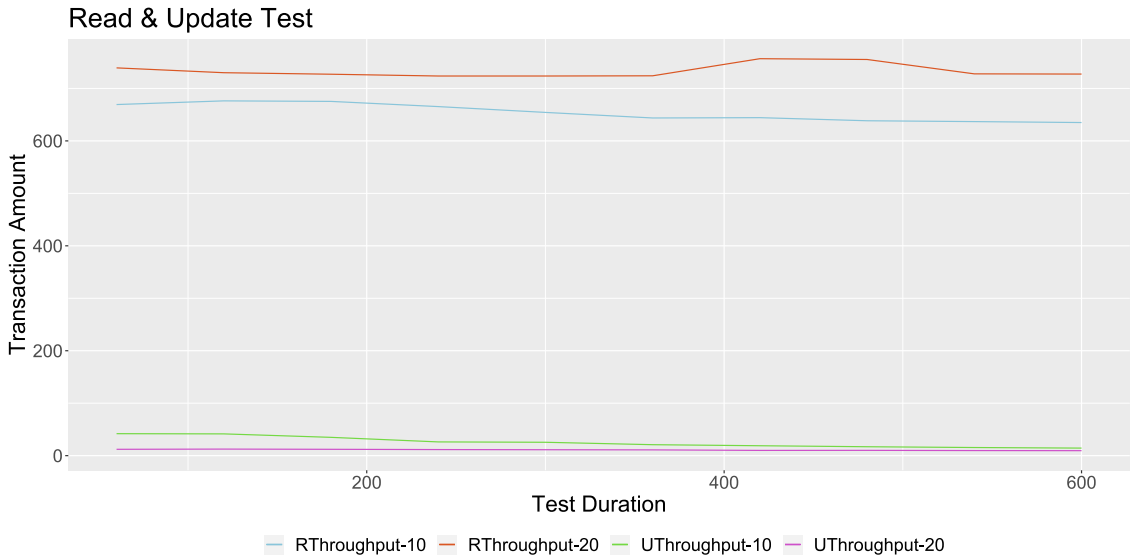


Figure 45. Throughput results with the semantic anomaly detection.

These values were obtained in a simulated environment; in a real environment, traced assets will oversee diverse Blockchain workers, so the number of failed transactions will be expected to be lower or close to zero. Due to the possibility of mistakes being made in the transactions sent to the network, we do not expect any failed transactions. Finally, we can conclude that using the anomaly detection function will not put the Blockchain network at risk and the proposed strategy's feasibility to detect anomalies in transactional data. Although, it may affect the latency and throughput during long executions periods.

4.7. Advantages and disadvantages

- **Advantages**

With the strategy, the smart contract will be able to keep up with new anomalies that may arise in the data stream in the future. In traceable processes, this is important due to the ever-changing environment and relationships between supply chain members.

Also, the data gain a new level of trust, not only for being stored on the Blockchain ledger but also because the smart contract validates the data for each transaction. Supply chain members can consult all the information marked as anomalies at any time and adjust their processes to prevent these problems from occurring in the future.

This information could also qualify supply chain members based on the number of anomalies in their data stream stored on the ledger. Finally, with the development of new methods, smart contracts could propose the proper handling for a traced product based on expert knowledge and the anomaly detection strategy offered.

- **Disadvantages**

As anomaly detection requires a minimum set of successful transactions, if the transaction data is filled with anomalies from the start, it will be classified as standard data. ML algorithms will require more computational resources as the number of stored information increases, resulting in longer training times. To update and deploy smart contracts autonomously and detect semantic anomalies, Python tools must continuously run on all nodes with permission to deploy smart contracts on the Blockchain network.

As training and rules generation will take place outside the Blockchain network, security measures will be needed to ensure that the smart contract update was not tampered with.

4.8. Summary

This chapter presents the test of our developed strategy for semantic anomaly detection from the developed smart contract to the system deployment with real data from two artisan sweets production sites. We use the Hyperledger Caliper tool for performance tests.

Subsequently, we evaluated the semantic anomaly development and the autonomous semantic anomaly detection rules update using the short supply chain traceability data. We assess multiple ML models and select decision trees

and random forest models to develop a module that translates these models' generated rules to Go and JavaScript programming languages.

Finally, we run the final test of our developed strategy using the collected data at the production sites. This part focuses solely on semantic anomaly detection and autonomous, innovative contract update process. The results show that our strategy works correctly, although some improvements can be made based on the results.

Chapter 5

Conclusions and Future Work

This chapter presents the conclusions from this doctoral research project, followed by recommendations, and proposes a future work plan to improve or expand this study.

5.1. Conclusions

Next, the main conclusions obtained with the development of this anomaly's detection strategy are presented.

- We manage to develop a strategy to give a smart contract the ability to detect syntaxis and semantics anomalies in the data of Blockchain transactions in a traceability scheme. This strategy can translate Random Forest and Decision Tree rules into Go and JavaScript code for smart contracts supported by Hyperledger Fabric. This strategy proves helpful in traceability schemes in supply chains where most of the traced data is generated autonomously. Some data points might be corrupted during transmission from the sensors to the database or during manual data conversion processes.

- Although the strategy shows good behavior during the test, more real environment deployment will be required to complete the adjustment that is not necessary for a research field. A Blockchain network with more nodes connected and sending transactions will be an ideal environment to do the closest "real world" test scenario. It will help to define configurations related to which nodes will be performing the smart contract updates process and when will be the best time to perform such updates in terms of network usage and Blockchain ledger size.

- We found that, although some libraries use ML algorithms with the selected programming languages to develop the smart contract, they are still limited to some environments or in an early development stage. Nevertheless, currently, there are community efforts to bring this type of solution to these languages due to its applicability not only to Blockchain technology.

- With our Blockchain network deployment, we found that giving the smart contract the ability to detect syntaxis and semantic anomaly detection does not negatively impact the network behavior and that throughput and latency are good enough for most traceability processes. However, more

measurements like the ones presented by (Treiblmaier, 2019) will be needed to determine if this approach is helpful for most Blockchain network configurations or applicable to other Blockchain solutions.

- The data sent on each transaction to the Blockchain network can be trusted if these anomaly detection methods are implemented in data centers or similar. Still, the use of IoT sensors, particularly ones that can commit blockchain transactions, is expected to increase in the coming days. It is unlikely that devices connected directly to the Blockchain network will be able to perform anomaly detection, so smart contracts will come in handy to detect anomalies in the not-too-distant future.

- We aimed to determine if data validations can be made using smart contracts; having this in mind, we decided to implement syntaxis and semantic error validations in the sensed data and two variables related reliability; consistency and kivenson reliability, allowing us to get some metrics that enable us to decide whether the information in the network is reliable. However, more metrics and tests are needed.

- We were able to observe the behavior of the Blockchain network by stress testing and a pilot assembly of two production sites. This allowed us to determine that using the implemented smart contracts functions does not represent a significant load on the network, and traceability activities can be conducted without saturating the network

- Finally, as we saw, supply chain traceability processes can encounter multiple types of anomalies. We are focusing solely on numeric syntaxis and semantic anomalies in this doctoral dissertation, so there is still more work to do to create smart contracts for traceability purposes, which can be used to validate data in real time and inform supply chain members of anomalies detected and viable solutions.

5.2. Future Work

This study's main objective was to develop an autonomous anomaly detection on a Blockchain traceability network transaction through smart contracts. The primary approach was to create a strategy to update deployed smart contracts based on rules generated by a Decision Tree ML algorithm and a tool to deploy the autonomously modified version of the smart contract.

Considering the above, and to improve and/or complement the developments, the following future works are proposed:

- To develop or apply a security framework to ensure that the autonomous smart contract deployment tool will be secure and unmodified in all nodes where it will be deployed. This will help to keep a safe environment like the one on the Blockchain network and give more trust to the developed tools.

- Implement the proposed strategy utilizing Blockchain solutions such as Hyperledger Sawtooth, which includes a smart contract or transaction processor. The developers have named it that can be written in Python. Smart contracts could now run multiple data mining processes to improve data validation, but the network usage and stress will need to be evaluated.

- Improve the rules generation process by using other rules-based ML models and more databases to see if problems that do not arise in this research will affect the rule generation, like categorical data.

- Improve the anomaly detection function by allowing the smart contract to reprocess past data to detect anomalies not considered by the previous version of the semantic anomaly detection function. Nevertheless, a network load test will be needed, depending on the type of supply chain traceability.

- To evaluate the current development in a wide Blockchain deployment to gather measurements that allow the validation of the proposed solutions in a more robust environment to find security risks that may not arise in a controlled environment.

- To evaluate is Complex Event Processing (CEP) will be adequate to generate rules valid during the cold start of the anomaly detection tool.

Bibliography

Agmon, N., & Ahituv, N. (1987). Assessing Data Reliability in an Information System. *Journal of Management Information Systems*, *4*(2). https://doi.org/10.1080/07421222.1987.11517792

Alharby, M., & Moorsel, A. van. (2017). *Blockchain Based Smart Contracts : A Systematic Mapping Study*. 125–140. https://doi.org/10.5121/csit.2017.71011

Almardeny, Y., Boujnah, N., & Cleary, F. (2020). A Novel Outlier Detection Method for Multivariate Data. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2020.3036524

Alpaydin, E. (2010). Decision Trees. *Introduction to Machine Learning*, *x*, 185–208. https://doi.org/10.1007/SpringerReference_63657

Arcaini, P., Riccobene, E., & Scandurra, P. (2015). Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. *Proceedings - 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015*, 13–23. https://doi.org/10.1109/SEAMS.2015.10

Aung, M. M., & Chang, Y. S. (2014). Traceability in a food supply chain: Safety and quality perspectives. In *Food Control* (Vol. 39, Issue 1, pp. 172–184). Elsevier BV. https://doi.org/10.1016/j.foodcont.2013.11.007

Azzi, R., Chamoun, R. K., & Sokhn, M. (2019). The power of a blockchain-based supply chain. *Computers and Industrial Engineering*, *135*, 582–592. https://doi.org/10.1016/j.cie.2019.06.042

Balagolla, E. M. S. W., Fernando, W. P. C., Rathnayake, R. M. N. S., Wijesekera, M. J. M. R. P., Senarathne, A. N., & Abeywardhana, K. Y. (2021). Credit Card Fraud Prevention Using Blockchain. *2021 6th International Conference for Convergence in Technology, I2CT 2021*. https://doi.org/10.1109/I2CT51068.2021.9418192

Baumgartner Data, P., Australia PeterBaumgartner, C., Alexander Krumpholz, csiroau, Baumgartner, P., & Krumpholz, A. (2021). Anomaly Detection in a Boxed Beef Supply Chain. *2021 The 13th International Conference on Computer Modeling and Simulation*. https://doi.org/10.1145/3474963

Ben-Daya, M., Hassini, E., & Bahroun, Z. (2019). Internet of things and supply chain management: a literature review. In *International Journal of Production Research* (Vol. 57, Issues 15–16, pp. 4719–4742). Taylor and Francis Ltd. https://doi.org/10.1080/00207543.2017.1402140

Beteto, A., Melo, V., Lin, J., Alsultan, M., Dias, E. M., Korte, E., Johnson, D. A., Moghadasi, N., Polmateer, T. L., & Lambert, J. H. (2022). Anomaly and cyber fraud detection in pipelines and supply chains for liquid fuels. *Environment Systems and Decisions*, *42*(2), 306–324. https://doi.org/10.1007/s10669-022-09843-5

Bhardwaj, S., & Kaushik, M. (2018). *Blockchain—Technology to Drive the Future* (pp. 263–271). Springer, Singapore. https://doi.org/10.1007/978-981-10-5547-8_28

Brase, C. H., & Brase, C. P. (2016). *Understanding Basic Statistics* (7th ed.). Cengage Learning. https://books.google.com/books/about/Understanding_Basic_Statistics_Enhanced.html?id=HzFTCwAAQBAJ

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). *LOF: Identifying Density-Based Local Outliers*.

Cao, S., Powell, W., Foth, M., Natanelov, V., Miller, T., & Dulleck, U. (2021). Strengthening consumer trust in beef supply chain traceability with a blockchain-based human-machine reconcile mechanism. *Computers and Electronics in Agriculture*, *180*(September 2020), 105886. https://doi.org/10.1016/j.compag.2020.105886

Casado-Vara, R., Prieto, J., la Prieta, F. de, & Corchado, J. M. (2018). How blockchain improves the supply chain: Case study alimentary supply chain. *Procedia Computer Science*, *134*, 393–398. https://doi.org/10.1016/j.procs.2018.07.193

Chandola, V., Banerjee, A., & Kumar, V. (2016). Anomaly Detection. *Encyclopedia of Machine Learning and Data Mining*, 1–15. https://doi.org/10.1007/978-1-4899-7502-7_912-1

Chen, C. Y., Chan, Y. W., Chang, C. H., Kang, T. C., Huang, C. H., & Tsai, Y. te. (2021). The Design and Implementation of Blockchain-Based Supply Chain System with Traceability. *Lecture Notes in Electrical Engineering*, *747*, 197–206. https://doi.org/10.1007/978-981-16-0115-6_18/FIGURES/8

Chen, M., Liu, J., Chen, S., Qiao, Y., & Zheng, Y. (2017). DBF: A general framework for anomaly detection in RFID systems. *Proceedings - IEEE INFOCOM*. https://doi.org/10.1109/INFOCOM.2017.8056986

Corda. (n.d.). Retrieved August 21, 2022, from https://www.corda.net/

Cozzini, P., Agosta, F., Dolcetti, G., & Righi, G. (2022). How a Blockchain Approach Can Improve Data Reliability in the COVID-19 Pandemic. *ACS Medicinal Chemistry Letters*, *13*(4), 517–519. https://doi.org/10.1021/ACSMEDCHEMLETT.2C00077

CSCG Tag | ECS Global. (n.d.). Retrieved February 1, 2022, from https://www.ecs.com.tw/en/Product/IoT/CSCG_Tag/overview

D'Amato, I., & Papadimitriou, T. (2013). Legitimate vs illegitimate: the luxury supply chain and its doppelganger. *International Journal of Retail & Distribution Management*, *41*(11/12), 986–1007. https://doi.org/10.1108/IJRDM-01-2013-0015

Data analysis. (2013). https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/datopic.html

Deepa, N., Pham, Q. V., Nguyen, D. C., Bhattacharya, S., Prabadevi, B., Gadekallu, T. R., Maddikunta, P. K. R., Fang, F., & Pathirana, P. N. (2022). A survey on blockchain for big data: Approaches, opportunities, and future directions. *Future Generation Computer Systems*, *131*, 209–226. https://doi.org/10.1016/j.future.2022.01.017

Demertzis, K., Iliadis, L., Tziritas, N., & Kikiras, P. (2020). Anomaly detection via blockchained deep learning smart contracts in industry 4.0. *Neural Computing and Applications*, *32*(23), 17361–17378. https://doi.org/10.1007/s00521-020-05189-8

EIP-190: Ethereum Smart Contract Packaging Standard. (n.d.). Retrieved October 24, 2021, from https://eips.ethereum.org/EIPS/eip-190

EOS. (n.d.). Retrieved August 21, 2022, from https://eos.io/

Etemadi, N., Borbon-Galvez, Y., & Strozzi, F. (2020). Blockchain technology for cybersecurity applications in the food supply chain: A systematic literature review. *Proceedings of the Summer School Francesco Turco*.

Ethereum. (n.d.). Retrieved August 21, 2022, from https://ethereum.org/en/

Federación Nacional de Cafeteros, F. (2016). *Resolución 02 de 2016*. https://federaciondecafeteros.org/app/uploads/2019/11/6.-Normas-de-calidad-para-café-verde-en-almendra-para-exportación-Resolución-2-2016.pdf

Florkowski, W., Shewfelt, R., Brueckner, B., & Prussia, S. E. (2009). Postharvest Handling. In *Postharvest Handling*. Elsevier. https://doi.org/10.1016/B978-0-12-374112-7.X0001-7

Foorthuis, R. (2021). On the nature and types of anomalies: a review of deviations in data. *International Journal of Data Science and Analytics*, *12*(4), 297–331. https://doi.org/10.1007/S41060-021-00265-1/FIGURES/3

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/JCSS.1997.1504

Griggs, K. N., Ossipova, O., Kohlios, C. P., Baccarini, A. N., Howson, E. A., & Hayajneh, T. (2018). Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring. *Journal of Medical Systems*, *42*(7), 130. https://doi.org/10.1007/s10916-018-0982-x

Grover, J., & Sharma, S. (2016). Security issues in Wireless Sensor Network-A review. *2016 5th International Conference on Reliability, Infocom Technologies and Optimization, ICRITO 2016: Trends and Future Directions*, 397–404. https://doi.org/10.1109/ICRITO.2016.7784988

GS1. (n.d.). *Traceability And Blockchain WP.pdf*. https://www.gs1.org/sites/default/files/gs1_traceability_and_blockchain_wp.pdf

GS1. (2012). *Global Traceability Standard*.

GS1. (2020). *GS1 - The global language of Business*. https://www.gs1.org/

Günther, L. C., Colangelo, E., Wiendahl, H. H., & Bauer, C. (2019). Data quality assessment for improved decision-making: A methodology for small and medium-sized enterprises. *Procedia Manufacturing*, *29*, 583–591. https://doi.org/10.1016/j.promfg.2019.02.114

Guo, H., & Yu, X. (2022). A survey on blockchain technology and its security. *Blockchain: Research and Applications*, *3*(2), 100067. https://doi.org/10.1016/j.bcra.2022.100067

Haleem, A., Khan, S., & Khan, M. I. (2019). Traceability implementation in food supply chain: A grey-DEMATEL approach. *Information Processing in Agriculture*, *6*(3), 335–348. https://doi.org/10.1016/j.inpa.2019.01.003

Hameed, K., Barika, M., Garg, S., Amin, M. B., & Kang, B. (2022). A taxonomy study on securing Blockchain-based Industrial applications: An overview, application perspectives, requirements, attacks, countermeasures, and open issues. *Journal of Industrial Information Integration*, *26*(December 2021), 100312. https://doi.org/10.1016/j.jii.2021.100312

Hu, B., Zhang, Z., Liu, J., Liu, Y., Yin, J., Lu, R., & Lin, X. (2020). A comprehensive survey on smart contract construction and execution: Paradigms, Tools and Systems. In *arXiv*. arXiv. https://doi.org/10.1016/j.patter.2020.100179

Hussien, H. M., Yasin, S. M., Udzir, N. I., Ninggal, M. I. H., & Salman, S. (2021). Blockchain technology in the healthcare industry: Trends and opportunities. *Journal of Industrial Information Integration*, *22*(March), 100217. https://doi.org/10.1016/j.jii.2021.100217

Hyperledger. (n.d.). Retrieved August 21, 2022, from https://www.hyperledger.org/

Hyperledger Caliper. (2018). https://www.hyperledger.org/use/caliper

Installing and Running Caliper. (2018). https://hyperledger.github.io/caliper/v0.5.0/installing-caliper/

ISO 9000 Sistemas de gestión de la calidad — Fundamentos y vocabulario, 2000 Info 35 (2015).

Iyer, S., Thakur, S., DIxit, M., Katkam, R., Agrawal, A., & Kazi, F. (2019). Blockchain and Anomaly Detection based Monitoring System for Enforcing Wastewater Reuse. *2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019*. https://doi.org/10.1109/ICCCNT45670.2019.8944586

Jan, S. U., Ahmed, S., Shakhov, V., & Koo, I. (2019). Toward a Lightweight Intrusion Detection System for the Internet of Things. *IEEE Access*, *7*, 42450–42471. https://doi.org/10.1109/ACCESS.2019.2907965

Jannat, M. U., Ahamed, R., Mamun, A., Ferdaus, J., Costa, R., & Biswas, M. (2021). Organic Food Supply Chain Traceability using Blockchain Technology. *2021 International Conference on Science and Contemporary Technologies, ICSCT 2021*. https://doi.org/10.1109/ICSCT53883.2021.9642543

Jindal, A., Gerndt, M., Bauch, M., & Haddouti, H. (2020). Scalable Infrastructure and Workflow for Anomaly Detection in an Automotive Industry. *2020 International Conference on Innovative Trends in Information Technology, ICITIIT 2020*. https://doi.org/10.1109/ICITIIT49094.2020.9071555

Jordan, M., & Bishop, C. (2004). Neural Networks. In *Computer Science Handbook, Second Edition* (pp. 1–16).

Karlsen, K. M., & Olsen, P. (2016). Problems and Implementation Hurdles in Food Traceability. In *Advances in Food Traceability Techniques and Technologies: Improving Quality Throughout the Food Chain* (pp. 35–46). Woodhead Publishing. https://doi.org/10.1016/B978-0-08-100310-7.00003-X

Katsikouli, P., Wilde, A. S., Dragoni, N., & Høgh-Jensen, H. (2021). On the benefits and challenges of blockchains for managing food supply chains. *Journal of the Science of Food and Agriculture*, *101*(6), 2175–2181. https://doi.org/10.1002/jsfa.10883

Khalfaoui, M., Molva, R., & Gomez, L. (2013). Secure Alert Tracking In Supply Chain. *2013 International Conference on Security and Cryptography (SECRYPT)*, 1–11.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. *Engineering*, *2*, 1051. https://doi.org/10.1145/1134285.1134500

Koberg, E., & Longoni, A. (2019). A systematic review of sustainable supply chain management in global supply chains. In *Journal of Cleaner Production* (Vol. 207, pp. 1084–1098). Elsevier Ltd. https://doi.org/10.1016/j.jclepro.2018.10.033

Konovalenko, I., & Ludwig, A. (2022). Generating decision support for alarm processing in cold supply chains using a hybrid k-NN algorithm. *Expert Systems with Applications*, *190*, 116208. https://doi.org/10.1016/J.ESWA.2021.116208

Kravitz, D. W. (2018). Transaction immutability and reputation traceability: Blockchain as a platform for access controlled iot and human interactivity. *Proceedings - 2017 15th Annual Conference on Privacy, Security and Trust, PST 2017*, 3–12. https://doi.org/10.1109/PST.2017.00012

Kumar, A., Abhishek, K., Nerurkar, P., Ghalib, M. R., Shankar, A., & Cheng, X. (2022). Secure smart contracts for cloud-based manufacturing using Ethereum blockchain. *Transactions on Emerging Telecommunications Technologies*, *33*(4). https://doi.org/10.1002/ETT.4129

Li, G., Xue, J., Li, N., & Ivanov, D. (2022). Blockchain-supported business model design, supply chain resilience, and firm performance. *Transportation Research Part E: Logistics and Transportation Review*, *163*, 102773. https://doi.org/10.1016/J.TRE.2022.102773

Liu, D., Alahmadi, A., Ni, J., Lin, X., & Shen, X. (2019). Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain. *IEEE Transactions on Industrial Informatics*, *15*(6), 3527–3537. https://doi.org/10.1109/TII.2019.2898900

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2012). Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(1). https://doi.org/10.1145/2133360.2133363

Liu, L., Tsai, W. T., Bhuiyan, M. Z. A., Peng, H., & Liu, M. (2022). Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Generation Computer Systems*, *128*, 158–166. https://doi.org/10.1016/J.FUTURE.2021.08.023

Liu, N., Bouzembrak, Y., van den Bulk, L. M., Gavai, A., van den Heuvel, L. J., & Marvin, H. J. P. (2022). Automated food safety early warning system in the dairy supply chain using machine learning. *Food Control*, *136*, 108872. https://doi.org/10.1016/J.FOODCONT.2022.108872

Liu, X., Jiang, F., & Zhang, R. (2020). A New Social User Anomaly Behavior Detection System Based on Blockchain and Smart Contract. *2020 IEEE International Conference on Networking, Sensing and Control, ICNSC 2020*. https://doi.org/10.1109/ICNSC48988.2020.9238118

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 14–23. https://doi.org/10.1002/widm.8

Luo, Q., Liao, R., Li, J., Ye, X., & Chen, S. (2022). Blockchain Enabled Credibility Applications: Extant Issues, Frameworks and Cases. *IEEE Access*, *10*, 45759–45771. https://doi.org/10.1109/ACCESS.2022.3150306

Marchesi, L., Mannaro, K., Marchesi, M., & Tonelli, R. (2022). Automatic Generation of Ethereum-Based Smart Contracts for Agri-Food Traceability System. *IEEE Access*, *10*, 50363–50383. https://doi.org/10.1109/ACCESS.2022.3171045

Masciari, E. (2012). SMART: Stream Monitoring enterprise Activities by RFID Tags. *Information Sciences*, *195*, 25–44. https://doi.org/10.1016/j.ins.2012.01.041

Minifabric. (2020). https://github.com/hyperledger-labs/minifabric

Oh, B., Jun, T. J., Yoon, W., Lee, Y., Kim, S., & Kim, D. (2019). Enhancing trust of supply chain using blockchain platform with robust data model and verification mechanisms. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, *2019-Octob*, 3504–3511. https://doi.org/10.1109/SMC.2019.8913871

Pal, O., Alam, B., Thakur, V., & Singh, S. (2019). Key management for blockchain technology. *ICT Express*. https://doi.org/10.1016/j.icte.2019.08.002

Patel, H., & Shrimali, B. (2021). AgriOnBlock: Secured data harvesting for agriculture sector using blockchain technology. *ICT Express*, *xxxx*. https://doi.org/10.1016/j.icte.2021.07.003

Pedregosa, F., Michel, V., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. *Ease*, 68–77.

Piryonesi, S. M., & El-Diraby, T. E. (2020). Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems. *Undefined*, *146*(2), 04020022. https://doi.org/10.1061/JPEODX.0000175

Pradhan, N. R., Singh, A. P., & Kumar, V. (2021). Blockchain-Enabled Traceable, Transparent Transportation System for Blood Bank. *Lecture Notes in Electrical Engineering*, *683*, 313–324. https://doi.org/10.1007/978-981-15-6840-4_25

RAFT. (2016). *Raft Consensus Algorithm*. https://raft.github.io/

Reyna, A., Martín, C., Chen, J., Soler, E., & Díaz, M. (2018). On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, *88*, 173–190. https://doi.org/10.1016/J.FUTURE.2018.05.046

Rodríguez, J. P., Montoya-Munoz, A. I., Rodriguez-Pabon, C., Hoyos, J., & Corrales, J. C. (2021). IoT-Agro: A smart farming system to Colombian

coffee farms. *Computers and Electronics in Agriculture*, *190*(July), 106442. https://doi.org/10.1016/j.compag.2021.106442
Romeu, J. L. (2003). Practical Reliability Engineering. In *Technometrics* (Vol. 45, Issue 2). https://doi.org/10.1198/tech.2003.s133
Sánchez, C., Schneider, G., & Leucker, M. (2018). Reliable smart contracts: State-of-the-art, applications, challenges and future directions. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11247 LNCS*, 275–279. https://doi.org/10.1007/978-3-030-03427-6_21
Scott, B. (2016). How Can Cryptocurrency and Blockchain Technology Play a Role in Building Social and Solidarity Finance? *United Nations Research Institute for Social Development*, *1*, 17. https://doi.org/10.1007/s10273-011-1262-2
Seines, M., Babuska, R., & Verbruggen, H. B. (1998). Rule-based modeling: Precision and transparency. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *28*(1), 165–169. https://doi.org/10.1109/5326.661100
Setboonsarng, S., Sakai, J., & Vancura, L. (2009). *Food Safety and ICT Traceability Systems: Lessons from Japan for Developing Countries*.
Seuring, S., & Müller, M. (2008). From a literature review to a conceptual framework for sustainable supply chain management. *Journal of Cleaner Production*, *16*(15), 1699–1710. https://doi.org/10.1016/J.JCLEPRO.2008.04.020
Shao, W., Wang, Z., Wang, X., Qiu, K., Jia, C., & Jiang, C. (2020). LSC: Online auto-update smart contracts for fortifying blockchain-based log systems. *Information Sciences*, *512*, 506–517. https://doi.org/10.1016/j.ins.2019.09.073
Sheldon, M. D. (2021). Auditing the blockchain oracle problem. *Journal of Information Systems*, *35*(1), 121–133. https://doi.org/10.2308/ISYS-19-049
Shukla, S., Thakur, S., & Breslin, J. G. (2022). Anomaly Detection in Smart Grid Network Using FC-Based Blockchain Model and Linear SVM. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13163 LNCS*, 157–171. https://doi.org/10.1007/978-3-030-95467-3_13/FIGURES/6
Skurichina, M., & Duin, R. P. W. (2002). Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis and Applications*, *5*(2), 121–135. https://doi.org/10.1007/s100440200011
Sodhi, M. M. S., & Tang, C. S. (2019). Research Opportunities in Supply Chain Transparency. *Production and Operations Management*, *28*(12), 2946–2959. https://doi.org/10.1111/POMS.13115
Stellar. (n.d.). Retrieved August 21, 2022, from https://www.stellar.org/?locale=en
Stergiou, C., & Siganos, D. (n.d.). *Neural Networks*. Retrieved March 29, 2018, from https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
Subashini, B., & Hemavathi, D. (2022). Detecting the Traceability Issues in Supply chain Industries using Blockchain Technology. *Proceedings -*

IEEE International Conference on Advances in Computing, Communication and Applied Informatics, ACCAI 2022. https://doi.org/10.1109/ACCAI53970.2022.9752478

Szabo, N. (1996). *Smart Contracts : Building Blocks for Digital Markets*. 1–11.

Talha, M., el Kalam, A. A., & Elmarzouqi, N. (2019). Big data: Trade-off between data quality and data security. *Procedia Computer Science*, *151*, 916–922. https://doi.org/10.1016/j.procs.2019.04.127

Tian, F. (2016). An agri-food supply chain traceability system for China based on RFID & blockchain technology. *2016 13th International Conference on Service Systems and Service Management, ICSSSM 2016*, 1–6. https://doi.org/10.1109/ICSSSM.2016.7538424

Tipping, M. E., & Bishop, C. M. (1999). *Probabilistic principal component analysis*.

Tomić, I., & McCann, J. A. (2017). A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols. *IEEE Internet of Things Journal*, *4*(6), 1910–1923. https://doi.org/10.1109/JIOT.2017.2749883

Treiblmaier, H. (2019). Toward More Rigorous Blockchain Research: Recommendations for Writing Blockchain Case Studies. *Frontiers in Blockchain*, *0*, 3. https://doi.org/10.3389/FBLOC.2019.00003

Tsolaki, K., Vafeiadis, T., Nizamis, A., Ioannidis, D., & Tzovaras, D. (2022). Utilizing machine learning on freight transportation and logistics applications: A review. *ICT Express*. https://doi.org/10.1016/J.ICTE.2022.02.001

Tukur, Y. M., Thakker, D., & Awan, I. U. (2021). Edge-based blockchain enabled anomaly detection for insider attack prevention in Internet of Things. *Transactions on Emerging Telecommunications Technologies*, *32*(6), e4158. https://doi.org/10.1002/ett.4158

United Nations Global Compact, & Business for Social Responsibility. (2014). *A Guide to traceability: A Practical Approach to Advance Sustainability in Global Supply Chains About the united Nations Global compact*.

U.S. Government Accountability Office. (2019). *ASSESSING DATA RELIABILITY Applied Research and Methods*. *December*.

Wang, J., Chen, J., Ren, Y., Sharma, P. K., Alfarraj, O., & Tolba, A. (2022). Data security storage mechanism based on blockchain industrial Internet of Things. *Computers and Industrial Engineering*, *164*(December 2021), 107903. https://doi.org/10.1016/j.cie.2021.107903

Wang, J., Chen, W., Wang, L., Simon Sherratt, R., Alfarraj, O., & Tolba, A. (2020). Data secure storage mechanism of sensor networks based on blockchain. *Computers, Materials and Continua*, *65*(3), 2365–2384. https://doi.org/10.32604/cmc.2020.011567

Wang, M., Wu, Q., Qin, B., Wang, Q., Liu, J., & Guan, Z. (2018). Lightweight and Manageable Digital Evidence Preservation System on Bitcoin. *Journal of Computer Science and Technology*, *33*(3), 568–586. https://doi.org/10.1007/s11390-018-1841-4

Wang, X., Pan, Z., Zhang, J., & Huang, J. (2021). Detection and elimination of project engineering security risks from the perspective of cloud

computing. *International Journal of Systems Assurance Engineering and Management*. https://doi.org/10.1007/s13198-021-01405-3

Wu, H. S. (2017). A survey of research on anomaly detection for time series. *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2017*, 426–431. https://doi.org/10.1109/ICCWAMTIP.2016.8079887

Wu, W., Chen, W., Fu, Y., Jiang, Y., & Huang, G. Q. (2022). Unsupervised neural network-enabled spatial-temporal analytics for data authenticity under environmental smart reporting system. *Computers in Industry*, *141*, 103700. https://doi.org/10.1016/j.compind.2022.103700

Xiang, F., Huaimin, W., & Peichang, S. (2019). Proof of Previous Transactions (PoPT): An Efficient Approach to Consensus for JCLedger. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–10. https://doi.org/10.1109/tsmc.2019.2913007

Xie, X., Lam, K.-M., Dai, Q., Peng, X., Yang, J., Yang, J., Geng, X., Smith-Miles, K., Choi, S., Dass, S. C., Pankanti, S., Prabhakar, S., Zhu, Y., Uchida, K., Grother, P., Rakvic, R., Broussard, R., Kennell, L., Ives, R., … Neri, A. (2009). Incremental Learning. In *Encyclopedia of Biometrics* (pp. 731–735). Springer US. https://doi.org/10.1007/978-0-387-73003-5_304

Yang, J., Wen, J., Jiang, B., & Wang, H. (2020). Blockchain-based sharing and tamper-proof framework of big data networking. *IEEE Network*, *34*(4), 62–67. https://doi.org/10.1109/MNET.011.1900374

Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, *14*(4), 352–375. https://doi.org/10.1504/IJWGS.2018.095647

Zunic, E., Tucakovic, Z., Delalic, S., Hasic, H., & Hodzic, K. (2020). Innovative Multi-Step Anomaly Detection Algorithm with Real-World Implementation: Case Study in Supply Chain Management. *2020 IEEE / ITU International Conference on Artificial Intelligence for Good, AI4G 2020*, 9–16. https://doi.org/10.1109/AI4G50087.2020.9311045