

An Agri-Food Supply Chain Traceability Management System based on Hyperledger Fabric Blockchain

Angelo Marchese and Orazio Tomarchio ^a

Dept. of Electrical Electronic and Computer Engineering, University of Catania, Catania, Italy

Keywords: Blockchain, Hyperledger Fabric, Agri-Food Supply Chain.

Abstract: Consumers are nowadays very interested in food product quality and safety. It is challenging to track the provenance of data and maintain its traceability throughout the whole supply chain network without an integrated information system. For this purpose, Agriculture and Food (Agri-Food) supply chains are becoming complex systems which are responsible, in addition to track and store orders and deliveries, to guarantee transparency and traceability of the food production and transformation process. However, traditional supply chains are centralized systems, mainly depending on a third party for trading and trusting purposes. In this paper we propose a fully distributed approach, based on blockchain technology, to define a supply chain management system able to provide quality, integrity and traceability of the entire supply chain process. A prototype based on Hyperledger Fabric has been designed and developed in order to show the effectiveness of the approach and the coverage of the main use cases needed in a supply chain network.

1 INTRODUCTION


The problem of agri-food supply chain management is gaining more and more importance given the recent attention for food quality and safety (Ray et al., 2017; Li et al., 2014). One of the main problems is to guarantee the traceability of products, providing a complete view of the different phases of their harvesting, processing and distribution (Olsen and Borit, 2018; Dabbene et al., 2014; Bosona and Gebresenbet, 2013). Current supply chain management systems allow to automate the monitoring and collection of information related to the various activities within the supply chain. In this way, the set of traceability information of a product allows the future consumer to know the provenance of that product and the events related to its entire life cycle from harvesting to retail.

However, most of today's supply chains are managed by centralized systems: members of such supply chains rely on a centralized authority, more specifically an information supervision center, to transfer and share their information. These centralized systems are often non-transparent, monopolistic and asymmetric information systems. This can pose a serious threat to the security and reliability of the traceability information and make fraud, corruption and data falsification easier. Furthermore, such cen-

tralized systems have limited scalability and a single point of failure.

To deal with such issues, the usage of blockchain technology in this domain has recently been proposed to support the management of supply chain traceability (Zhao et al., 2019; Antonucci et al., 2019; Galvez et al., 2018). Blockchain technology in particular offers cryptographic primitives to store data within a distributed ledger, guaranteeing their immutability and authenticity. This eliminates the need for supply chain members to trust a single entity to manage their traceability information. Furthermore, being a distributed system, the blockchain can solve the problems of limited scalability and single point of failure.

In this work we propose a complete model of a blockchain based agri-food supply chain traceability system and provide an implementation of a system prototype to show the applicability of blockchain technology in this domain. The proposed system in particular allows supply chain members to store and manage product-related traceability information in a distributed and immutable way. An important component of this system is the *smart contract*, a blockchain primitive that allows to automate some of the management operations related to supply chain activities. This smart contract offers some operations that allow to store and update traceability information and to reconstruct a complete history of the transactions

^a  <https://orcid.org/0000-0003-4653-0480>

related to a product during its life cycle within the supply chain. The implemented system also makes it possible to associate rules with supply chain products, allowing the expression of product specific quality control mechanisms and to verify in an automated way the regulatory compliance of products at runtime. This last aspect has been taken into consideration in our work, because in the context of agri-food supply chains the regulatory aspects are of fundamental importance to ensure food safety and quality, also taking into account that these aspects vary from a case to another one and dynamically evolve over time (Chen et al., 2015).

Our system was implemented using the Hyperledger Fabric¹ blockchain, an emerging open-source technology widely used also in other proposed examples of supply chain management systems (Wang et al., 2018). In addition, the components of our system are deployed in a cloud environment within a Kubernetes² cluster, showing that, although our system is a prototype, it can be easily migrated to a scalable production environment.

The rest of the paper is organized as follows. In Section 2, we provide a background of technologies exploited for this work and present related works. Section 3 presents the overall architecture of our framework and provides some implementation details about system components and the operations offered by the smart contract. Section 4 discusses about an example of usage of our system in a prototype environment. Finally, Section 5 concludes the work.

2 BACKGROUND AND RELATED WORK

This section provides some background information about the technologies exploited in our work, such as the blockchain technology and in particular the Hyperledger Fabric system. Then, after briefly outlining the advantages that would derive from their use to support agri-food supply chain traceability systems, some related work in the literature are discussed.

2.1 Blockchain Technology and Hyperledger Fabric

Blockchain technology represents a particular class of distributed systems and as such was born with the aim of overcoming some of the problems related to centralized systems (Kolb et al., 2020; Gamage et al.,

2020). The application area in which the blockchain was initially introduced is that of transactional systems, in particular electronic payment systems. That is the case, for example, of the Bitcoin blockchain (Nakamoto, 2008). However, today blockchain technology is increasingly being adopted in a lot of different application domains.

In general, operations within a blockchain are carried out by nodes connected to each other through a peer-to-peer network. In public blockchains, like Bitcoin, every node can participate in network operations and can decide to exit at any time. Each node participating in the blockchain maintains a local copy of a distributed ledger which contains a set of append-only logs that encode the status information of the blockchain. More specifically, an ordered sequence of blocks is stored inside the ledger. Each block consists of an header and a body that contains an ordered list of transactions which are validated and executed by the peers of the network. To guarantee the immutability and reliability of the data in the ledger, each block of the sequence contains a cryptographic hash of the previous block within a header field. In this way, a malicious attempt to change the content of a block would require to correspondingly modify the header of all the following blocks in the sequence, which is a computationally expensive task thanks to the non-invertibility property of hash functions.

Hyperledger Fabric is an open-source blockchain platform, which falls within the category of permissioned blockchains (Androulaki et al., 2018). In a permissioned blockchain, only authorized peers can participate in blockchain operations. Hyperledger Fabric is a distributed operating system that runs applications written in general purpose programming languages, such as Go, Java, JavaScript, and Python. It introduces the execute-order-validate blockchain model for transaction processing unlike other traditional blockchain systems that use the order-execute model. Like some other blockchains, Hyperledger Fabric offers the smart contract primitive. A smart contract is a combination of data and code that encodes a set of transformations on that data. It exposes a set of operations that can be invoked by the users of the blockchain with the aim of changing the state of the distributed ledger. The concept of smart contract therefore makes this kind of blockchain a distributed execution environment of general purpose programmable logic.

Thanks to the aforementioned properties, blockchain technology is a good candidate to address some of the actual problems related to traditional centralized agri-food supply chain traceability systems. In particular, it can guarantee the transparency,

¹<https://www.hyperledger.org/use/fabric>

²<https://kubernetes.io/>

verifiability and immutability of traceability data, simplifying the information sharing between the supply chain entities often belonging to distinct administrative organization. In this way the traceability of the supply chain products can be guaranteed, allowing the consumer to reconstruct the entire product's life cycle within the supply chain and to verify its origin and authenticity. Finally, smart contracts can be used to automate the supply chain management and product quality control operations.

2.2 Related Work

In the literature there is a variety of works that propose the use of blockchain technology to build agri-food supply chain management systems and in some cases implementations of such systems are also proposed. Some of these works are briefly described below.

In (Malik et al., 2018) a permissioned blockchain system, called ProductChain, is proposed. The system is administered by a consortium of entities participating in a generic food supply chain, including governmental and regulatory entities. It stores product traceability information made accessible to consumers. The authors propose the use of a three-tier sharded architecture that ensures reliability and availability of data for consumers and scalability with respect to transaction execution throughput. They also propose the use of a transaction vocabulary and the implementation of access control mechanisms to manage read and write privileges on the blockchain.

(Wang et al., 2019) propose a product traceability system based on the Ethereum blockchain and the smart contract primitive. The system stores information related to the products life cycle and also provides for the implementation of event-response mechanisms to verify the identities of both parties of all transactions at the time of their submission, so that their validity is guaranteed. All the events are kept in the system permanently, so that any disputes can be managed and the responsible for certain actions can be traced.

In (Caro et al., 2018) the AgriBlockIoT is proposed, a totally distributed and blockchain-based supply chain management system, able to integrate multiple IoT devices that collect and produce digital data along the supply chain. To efficiently evaluate Agri-BlockIoT, the authors defined a use case based on the from-farm-to-fork model. This use case was then implemented using two different blockchain systems, namely Ethereum and Hyperledger Sawtooth.

(Casino et al., 2019) propose a distributed functional model based on blockchain to create distributed

and automated traceability mechanisms for a generic agri-food supply chain. To evaluate the feasibility of the proposed model, a use case is presented. The applicability of the model is also illustrated through the development of a fully functional smart contract and a private blockchain.

(Feng Tian, 2017) propose a food supply chain traceability system for real-time food tracing based on HACCP (Hazard Analysis and Critical Control Points), blockchain and Internet of Things, which provides a platform that ensures openness, transparency, neutrality, reliability and security for traceability information. The proposed system uses BigchainDB, which combines the key benefits of distributed databases and blockchain.

(Biswas et al., 2017) propose a blockchain-based system to achieve the traceability of the activities that occur within the supply chain related to wine production. The proposed traceability system uses Multi-Chain to implement a private blockchain.

Like the aforementioned research works, in our work we propose a complete solution of a blockchain-based agri-food traceability system, providing in particular a description of the architectural components, the information model and the business logic of this system. A distinctive contribution of our work, is the capability to allow the specification of custom regulations for supply chain products at runtime and to automate the validation of these regulations. Our framework has addressed this aspect considering the heterogeneity of product regulations among supply chains and the fact that these regulations change over time.

3 SYSTEM ARCHITECTURE

This section first provides a high-level description of our blockchain-based system for agri-food supply chain traceability, the objectives that guided its design and its general architecture. Then the business logic of the system is described, focusing on the smart contract operations. Finally some implementation details of the developed prototype are provided.

3.1 General Architecture

The proposed system is designed to manage the traceability information of products and activities related to one or more agri-food supply chains. The main objective is to allow to reconstruct the entire flow of activities and transactions related to a certain product from origin to the end consumer. The system has to automate all those operations related to product quality control and regulatory compliance. It has to be

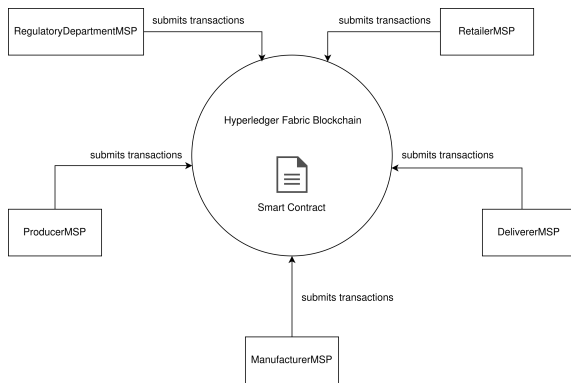


Figure 1: Typical agri-food supply chain scenario.

able to dynamically adapt to changes in laws and regulations. It should also be scalable, able to handle an ever-increasing amount of information. Finally, the system has to guarantee reliability and availability, especially when dealing with environments characterized by continuous flows of transactions.

The fundamental part of the framework consists of a permissioned blockchain, implemented through the Hyperledger Fabric framework. In this blockchain, the core of the system’s business logic is executed in the form of a *smart contract*. The smart contract offers several operations that allow users of the system to add and modify information in the blockchain in a secure and traceable way. Users of the system are the supply chain members and the regulatory departments. The former add and modify information related to their products, while the latter deal with the management and regulation aspects of supply chains. More specifically, the entities participating in the system operations are user organizations, where each user is identified by a certificate issued by a certification authority associated with the organization to which the user belongs. Since the blockchain is permissioned, only a well-defined set of organizations can participate in the system operations. The interaction between users and the blockchain takes place through a client application that runs within an application server and the interaction with the latter takes place through a frontend application that is hosted by a web server. Each organization has its own application server and web server.

Each organization has its own role. This role defines the interactions of this organization with the system and the operations it can perform. According to common models of agri-food supply chain described in literature (Wang et al., 2019; Feng Tian, 2017) we consider the following roles:

- *Producer*: organization that requires the registration of one or more primary products (i.e. prod-

ucts whose batches do not derive from any other batch). If a registration request is accepted, this organization can register batches associated with the registered product or products in the system.

- *Manufacturer*: organization that requires the registration of one or more derived products (i.e. products whose batches derive from batches of other primary or derived products). If a registration request is accepted this organization can register batches associated with the registered product or products in the system, specifying a list of batches from which the registered batch derives.
- *Deliverer*: organization that buys batches from organizations and resells them to other organizations.
- *Retailer*: organization that sells products to consumers.
- *Regulatory Department*: organization that manages and monitors the activities within the various supply chains. More specifically, an organization with the role of *Regulatory Department* adds product types to the system, associating them with rules and assigning roles to the various organizations.

In the following, while describing the behaviour of our framework, we refer to a scenario involving five organizations, one for each of the roles listed above, which is depicted in Figure 1.

3.2 Information Model

Figure 2 shows a domain model of the system’s business logic. In this context, when we discuss about the system’s business logic we refer to the business logic layer performed by the smart contract that is located within the blockchain. This business logic is exposed to the users of the system through operations that can be requested and executed. Client and frontend applications are nothing more than interfaces through which users interact with the smart contract. Consequently, from a functional point of view, it is possible to represent the interaction between users and smart contract as a direct interaction and assume that the user directly invokes a smart contract operation.

Each *Organization* of the system is identified by a unique identifier and can be associated with a *RoleSet*, which represents a list of roles. A *RoleSet* associated with an organization defines the set of smart contract operations the organization can invoke. An organization playing the role of *Regulatory Department* can register one or more product types in the system. A *ProductType* is uniquely identified by a name and can be either primary or derived. In case the product

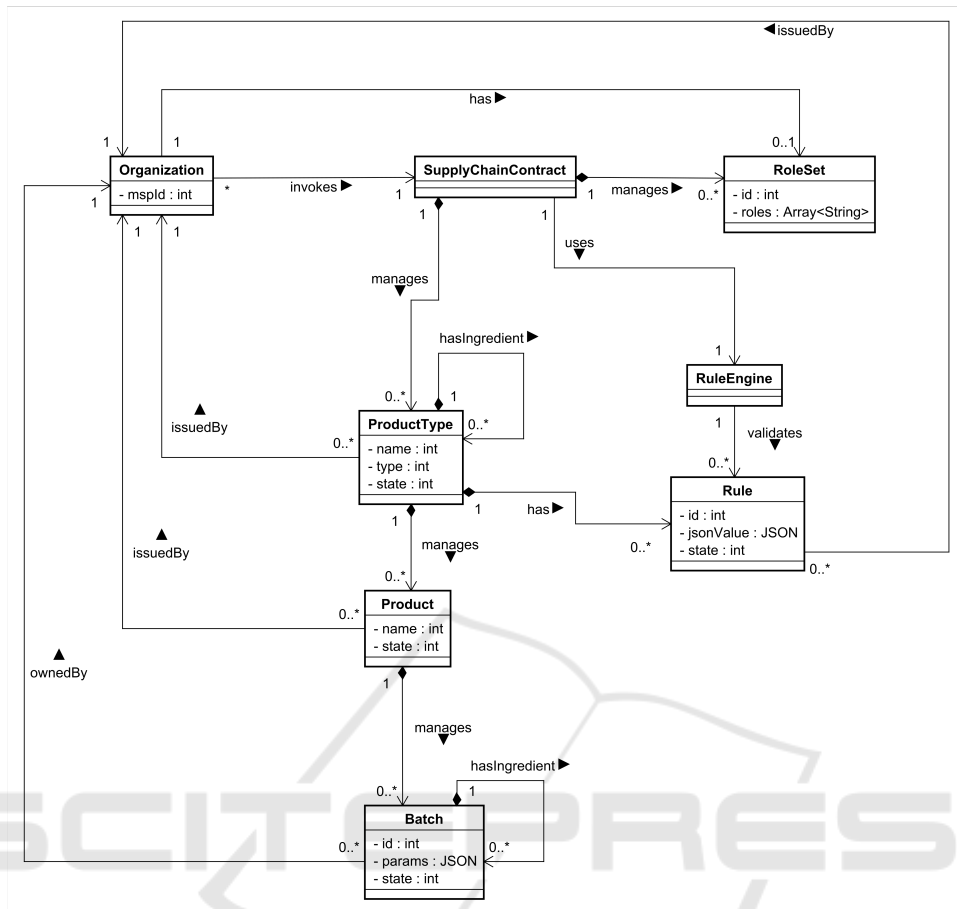


Figure 2: Domain model of the business logic.

type is derived, it has a list of product types ingredients which it is derived from. This means that any *Batch* associated with this product type must have a list of batches ingredients whose respective product types are in the list of product types ingredients. It is possible to associate one or more rules to a product type, where each *Rule* represents a set of conditions that have to be respected when registering batches associated with that product type. At the moment of a batch registration these rules are validated using a *RuleEngine*.

A *ProductType* may be associated with one or more *Products*, for each of which an owner organization requires the registration in the system. A request for the registration of a product can be accepted by an organization playing the role of *Regulatory Department* and from that moment the organization that owns the product can register batches of that product in the system. A product is uniquely identified by a name.

A *Product* may be associated with one or more *Batches* that are registered by the organization that

owns that product. A batch is uniquely identified by an ID and a set of parameters as specified at registration time. When registering a batch associated with a derived product type, it is necessary to specify a list of batches ingredients from which this batch derives. This list must be consistent with the list of product types ingredients associated with the product type of the registered batch. A batch can be transferred from one organization to another one and an organization that owns a batch can use that batch as an ingredient when registering a new batch. The domain model depicted in Figure 2 also shows that each resource in the system has a state. It provides information on the current conditions of that resource, determines the operations that can be performed on it and the subsequent states in which it can transit.

3.2.1 Smart Contract Operations

In this Section we describe more in detail the operations provided by the smart contract, which are summarized in Figure 3, together with the state transitions

SupplyChainContract
addRoleSet() addProductType() addRule() enableRule() disableRule() blockProductType() unblockProductType() requestProductRegistration() acceptProductRegistration() refuseProductRegistration() blockProduct() unblockProduct() registerBatch() blockBatch() unblockBatch() requestBatchTransfer() acceptBatchTransfer() refuseBatchTransfer() getBatchHistory()

Figure 3: Smart contract operations.

of the different resource types caused by the execution of these operations, shown respectively in Figures 4, 5, 6 and 7.

A new product type can be registered by an organization with role of *Regulatory Department* with the operation *addProductType()* and initially it starts from the state *Blocked*. In this state all the products related to this product type are also blocked and no organization can request the registration of a new product for this product type. From the state *Blocked* a product type can be unblocked by an organization with role of *Regulatory Department* with the operation *unblockProductType()*, causing it to pass to the state *Unblocked*.

A new rule, associated with a product type, can be registered by an organization with role of *Regulatory Department* with the operation *addRule()* and initially it starts from the state *Disabled*. In this state, at the moment of registration of a new batch of the product type with which this rule is associated, the rule is not validated. From the state *Disabled* a rule can be enabled by an organization with role of *Regulatory Department* with the operation *enableRule()*, causing it to pass to the state *Enabled*. The ability to add, enable and disable custom rules for a product type and to do it at runtime allows to implement product-specific quality control mechanisms that can change over time. This aspect is of fundamental importance due to the requirement of today's agri-food supply chains to establish products specific regulations that can frequently evolve over time.

Organizations with role of *Producer* and those with role of *Manufacturer* can request, calling the operation *requestProductRegistration()*, the registration of a new product associated with a primary and derived product type respectively. An organization with

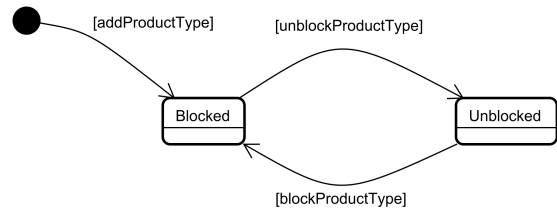


Figure 4: State diagram of resource type ProductType.

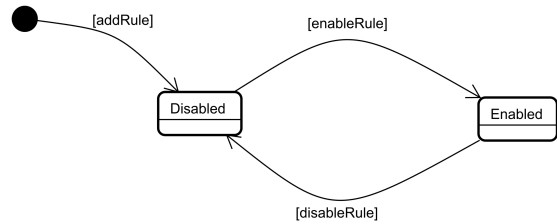


Figure 5: State diagram of resource type Rule.

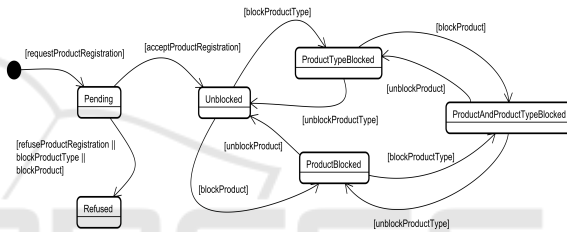


Figure 6: State diagram of resource type Product.

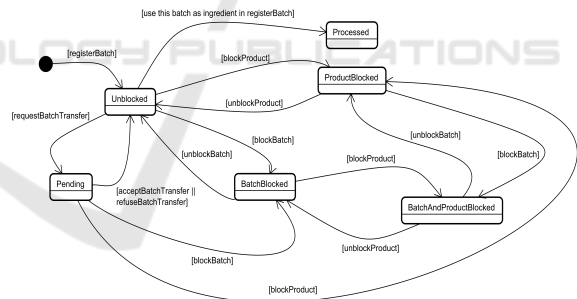


Figure 7: State diagram of resource type Batch.

role of *Regulatory Department* can accept the registration request for a new product with the operation *acceptProductRegistration()*, causing the product to switch to the state *Unblocked*, or it can reject the request with the operation *refuseProductRegistration()*, causing the product to switch to the state *Refused*. A product can be directly blocked by the owner organization or by an organization with role of *Regulatory Department* with the operation *blockProduct()*. While a product is blocked all the batches related to it are also blocked and no new batch for this product can be registered.

The owner organization of a product can register,

with the operation *registerBatch()*, new batches associated with it. A new registered batch starts from the state *Unblocked*. An organization that wants to purchase a batch can submit a transfer request for this batch with the operation *requestBatchTransfer()*, causing it to pass to the state *Pending*. The owner organization can reject the transfer request with the operation *refuseBatchTransfer()* or it can accept the request with the operation *acceptBatchTransfer()*, causing the batch to change owner. A batch can be used as an ingredient for another batch and in this case it passes to the state *Processed*. A batch can be directly blocked by the owner organization or by an organization with role of *Regulatory Department* with the operation *blockBatch()*. While a batch is blocked it cannot be transferred to other organizations and cannot be used as an ingredient for another batch.

Finally, the operation *getBatchHistory()* allows to obtain a complete history of the state transitions related to a batch. In this way any organization can view the entire batch life cycle and the chain of its owners.

3.3 Implementation Details

The designed framework has been implemented and a prototype has been deployed within a Kubernetes cluster in order to emulate the distributed nature of the whole system, and to increase its portability and interoperability with existing organization IT systems. Figure 8 shows the system architecture in terms of the main components composing our framework: in particular, it shows the components for each organization of the scenario presented in Section 3.1, plus a set of components making up the Hyperledger Fabric blockchain.

Each of the participating organizations runs a peer node that maintains information about its local copy of the distributed ledger in a dedicated CouchDB database node. The management of blockchain transactions ordering is handled by an Orderer node. Each organization runs its own certificate authority that issues certificates for that organization's users and peer nodes. In addition, each organization runs an application server which executes the client application logic to submit transactions to the blockchain, a MongoDB database where the application server keeps user data and a web server that hosts a frontend application that allows users to interact with the application server. The system also runs a certificate authority that issues TLS certificates. These certificates are used by users and system nodes to secure communications. Each component of the system runs on a Docker container inside a Kubernetes Pod that is managed by a Kubernetes Deployment. Each Pod is exposed to the re-

maining components of the cluster through a specific Kubernetes Service.

The core of the system's business logic is represented by a smart contract. This smart contract was implemented using the Node.js Fabric SDK. Figure 9 shows a simplified class diagram of the smart contract. The *SupplyChainContract* class extends the *Contract* class, which is part of the SDK, and represents a controller class for the smart contract itself. Indeed, this class implements methods that, except for the *init()* method, represent the smart contract operations that have been illustrated previously in Section 3.2.1. These operations allow users to create and modify resources in the blockchain ledger. The *init()* method is the first method of this class that is invoked as soon as the smart contract is deployed and allows to initialize it. The *SupplyChainContract* class has a reference to an object of the *SupplyChainContext* class, which extends the SDK *Context* class. This object allows to read and modify the ledger state and to retrieve information about a transaction, such as the identity of the user who submitted that transaction. More specifically, it has a reference to the *RoleSetList*, *ProductTypeList*, *RuleList*, *ProductList* and *BatchList* classes. These classes extend the *StateList* class and represent repositories that allow to create, modify and retrieve objects of the *RoleSet*, *ProductType*, *Rule*, *Product* and *Batch* classes respectively. These latter classes extend the *State* class and represent an abstraction layer to interact with the corresponding resources in the ledger. Finally the *SupplyChainContract* class has a reference to the *RuleEngine* class which implements the *getJSONRuleFromString()* and *verifyJsonRule()* methods. The first is called during the execution of the *addRule()* method of the *SupplyChainContract* class and starting from the string representation of a rule, validates the rule string format and returns the corresponding JSON object of that rule which then is stored in the ledger. The latter is called during the execution of the *registerBatch()* method of the *SupplyChainContract* class and validates a rule on the parameters of a batch at the time of its registration.

4 USE CASE

This section illustrates an example of usage of our system in the context of the scenario presented in Section 3.1 and depicted in Figure 1 where five different organization (one for each of the defined role) are present.

The use case demonstrates the system's ability to automate supply chain operations, maintain traceability information and provide a complete life cycle his-

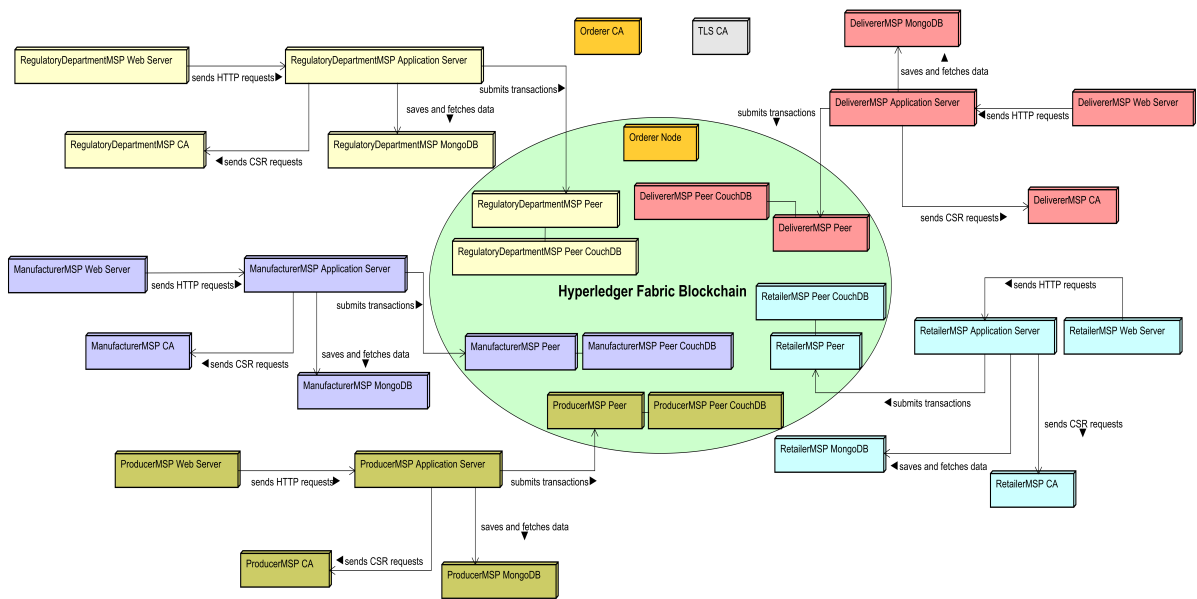


Figure 8: System components architecture.

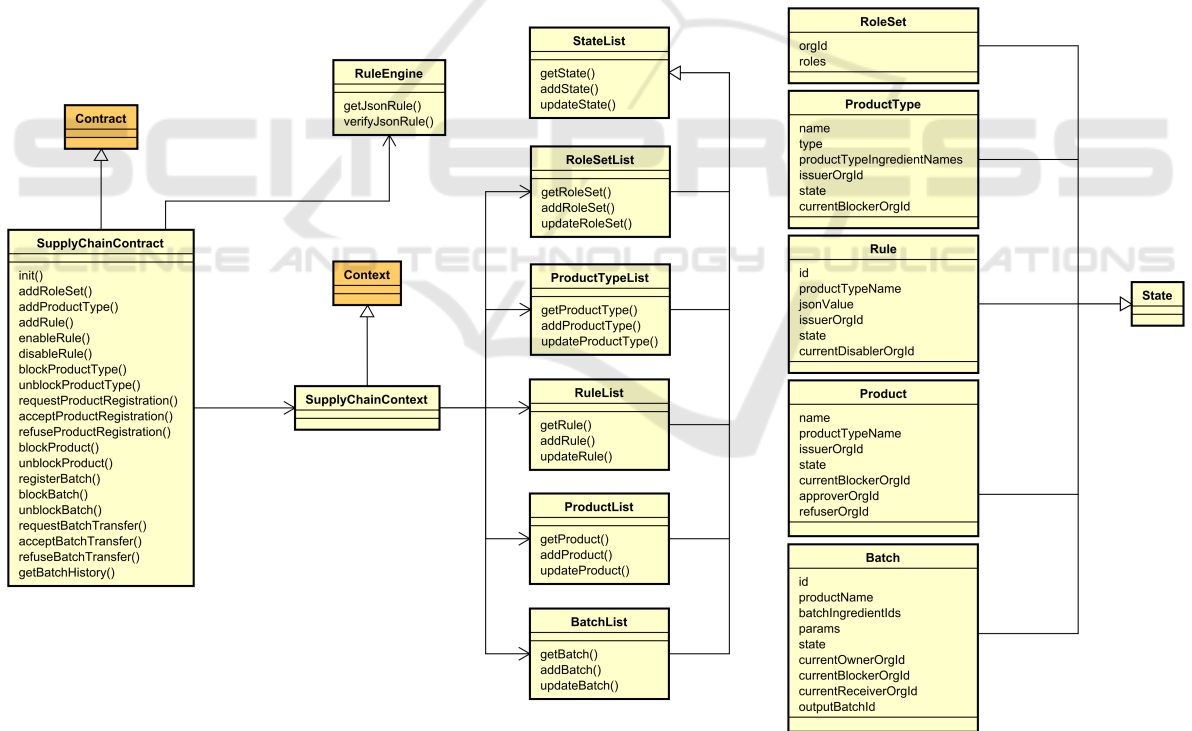


Figure 9: Smart contract class diagram.

tory of each batch. Figures 10 and 11 summarize all the steps of this use case and show how resources are created and updated within the blockchain ledger. For each resource type, different resources are shown (each one in a different row). For each resource, represented by a box in the Figures, the main attributes

and their values are shown. Each step is associated with a number and a colour and the boxes with that colour represent the state of the corresponding resources after the execution of that step.

First of all, when the smart contract is initialized, the role of *Regulatory Department* is associated with

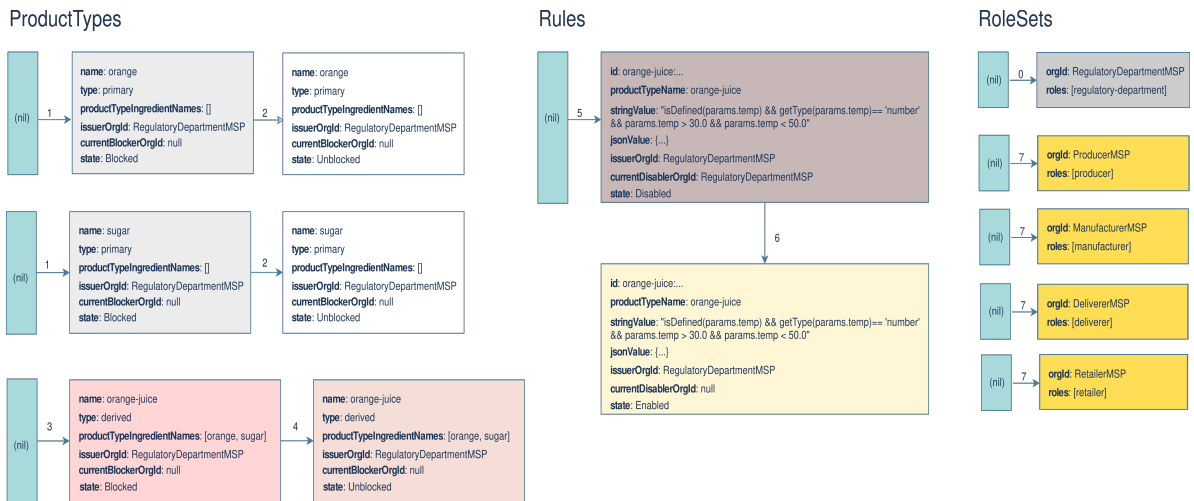


Figure 10: Use case steps from 0 to 7.

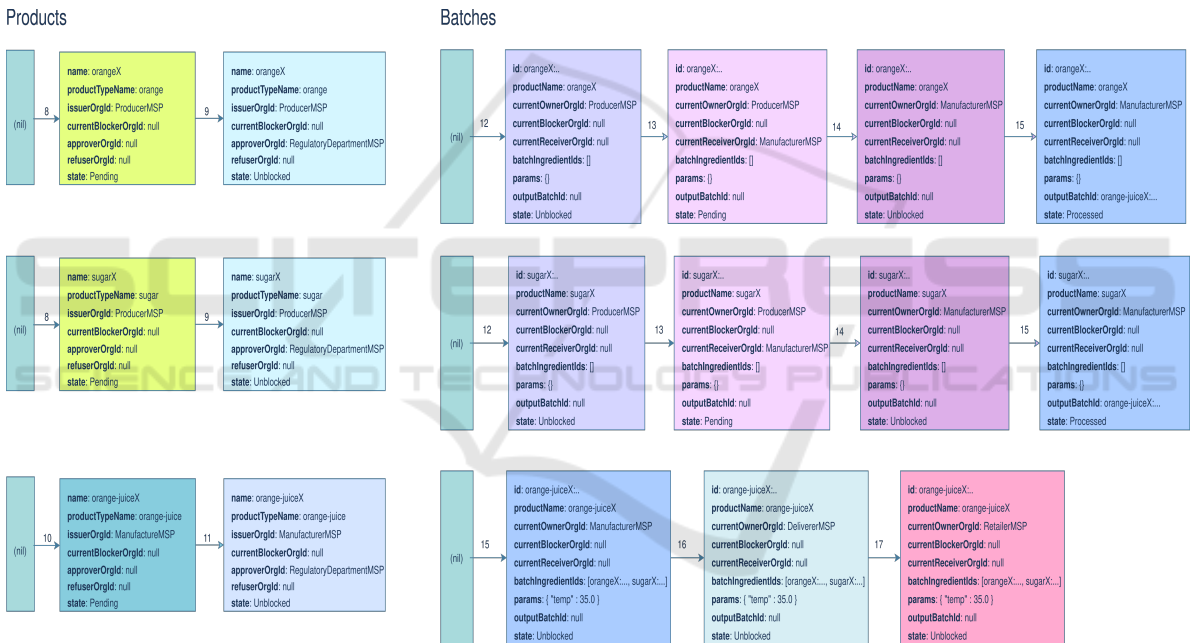


Figure 11: Use case steps from 8 to 17.

the organization RegulatoryDepartmentMSP (*step 0*). With this role the organization RegulatoryDepartmentMSP can perform administrative operations.

The RegulatoryDepartmentMSP organization registers the primary product types *orange* and *sugar* (*step 1*). These product types are initially in the state *Blocked*. So in this state no organization can request the registration for a product related to these product types. Then the RegulatoryDepartmentMSP organization unblocks the two product types causing them to pass to the state *Unblocked* (*step 2*). After registering the product types *orange* and *sugar* the RegulatoryDepartmentMSP organization can register the derived

product type *orange-juice*, specifying the two primary product types as ingredients (*step 3*). Also in this case the product type *orange-juice* starts from the state *Blocked* and after the RegulatoryDepartmentMSP unblocks it, this product type passes to the state *Unblocked* (*step 4*).

The RegulatoryDepartmentMSP organization then registers a new rule associated with the product type *orange-juice* (*step 5*). This rule requires that batches related to this product type contain, among the parameters, a temperature parameter with a value that must fall within a specific range. The rule is initially in the state *Disabled* and in this state

the rule is not activated for batch validation. The RegulatoryDepartmentMSP organization enables the rule causing it to pass to the state *Enabled* (step 6).

The RegulatoryDepartmentMSP organization then associates the ProducerMSP, ManufacturerMSP, DelivererMSP and RetailerMSP organizations with the roles of *Producer*, *Manufacturer*, *Deliverer* and *Retailer* respectively (step 7). In this way the ProducerMSP and ManufacturerMSP organizations can request the registration for a primary and derived product respectively. Moreover the DelivererMSP organization can buy and resell batches from and to other organizations and the RetailerMSP organization can only buy batches.

After having gained the role of *Producer*, the ProducerMSP organization requests the registration of the products *orangeX* and *sugarX* associated with the product types *orange* and *sugar* respectively (step 8). This products are initially in the state *Pending* and after the RegulatoryDepartmentMSP organization accepts the registration requests they pass to the state *Unblocked* (step 9). In the same way the ManufacturerMSP organization, after having gained the role of *Manufacturer*, requests the registration of the product *orange-juiceX* associated with the product *orange-juice* (step 10). This product is initially in the state *Pending* and after the RegulatoryDepartmentMSP organization accepts the registration request it pass to the state *Unblocked* (step 11).

The ProducerMSP organization then registers two batches associated with the products *orangeX* and *sugarX* respectively (step 12). These batches are initially in the state *Unblocked*. The ManufacturerMSP organization submits a transfer request for the two batches causing them to pass to the state *Pending* (step 13). After the ProducerMSP organization accepts the transfer requests, the ManufacturerMSP organization becomes the new owner of the two batches and these return to the state *Unblocked* (step 14). The ManufacturerMSP organization can then register a batch associated with the product *orange-juiceX*, using the new acquired batches which pass to the state *Processed* (step 15). In the registration operation the ManufacturerMSP organization specify a value for the temperature parameter compatible with the range specified in the rule associated with the product type *orange-juice*.

The DelivererMSP organization then submits a transfer request for the new registered batch and the ManufacturerMSP organization accepts the request causing the DelivererMSP organization to become the new owner (step 16). Finally in the same way the RetailerMSP organization submits a transfer request for the same batch and the DelivererMSP organization accepts the request causing the RetailerMSP or-

ganization to become the new owner (step 17).

The realization of the use case, demonstrates that the proposed system supports all the basic lifecycle of an agri-food product, from its origin to the end consumer. The simple rules implemented, also show the flexibility of the framework to add any kind of rule at runtime in order to cope with specific quality control strategies needed by any of the organization involved in the supply chain.

5 CONCLUSIONS

In this work we proposed a complete model of a blockchain-based agri-food supply chain traceability system, also providing a prototype implementation. The use of blockchain technology eliminates the need for supply chain members to trust a single entity to manage supply chain activities and store traceability information. Furthermore, this fully distributed approach solves the problems of limited scalability and single point of failure. The proposed system allows to automate supply chain management operations and maintain traceability information in a secure and immutable way. Moreover, the possibility to add rules at runtime allows the implementation of product-specific quality control mechanisms in a flexible way. Finally, the system provides a complete view of the different phases of harvesting, processing and distribution to which batches of product are subject allowing to reconstruct the entire life cycle of each batch and obtain provenance information.

REFERENCES

- Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S. W., and Yellick, J. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18.
- Antonucci, F., Figorilli, S., Costa, C., Pallottino, F., Raso, L., and Menesatti, P. (2019). A review on blockchain applications in the agri-food sector. *Journal of the Science of Food and Agriculture*, 99(14):6129–6138.
- Biswas, K., Muthukkumarasamy, V., and Lum, W. (2017). Blockchain based wine supply chain traceability system. In *Future Technologies Conference (FTC 2017)*, pages 56–62.
- Bosona, T. and Gebresenbet, G. (2013). Food traceability as an integral part of logistics management in food and

- agricultural supply chain. *Food Control*, 33(1):32 – 48.
- Caro, M. P., Ali, M. S., Vecchio, M., and Giaffreda, R. (2018). Blockchain-based traceability in agri-food supply chain management: A practical implementation. In *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany)*, pages 1–4.
- Casino, F., Kanakaris, V., Dasaklis, T., Moschuris, S., and Rachaniotis, N. (2019). Modeling food supply chain traceability based on blockchain technology. *IFAC-PapersOnLine*, 52:2728–2733.
- Chen, K., xin WANG, X., and ying SONG, H. (2015). Food safety regulatory systems in europe and china: A study of how co-regulation can improve regulatory effectiveness. *Journal of Integrative Agriculture*, 14(11):2203 – 2217.
- Dabbene, F., Gay, P., and Tortia, C. (2014). Traceability issues in food supply chain management: A review. *Biosystems Engineering*, 120:65 – 80.
- Feng Tian (2017). A supply chain traceability system for food safety based on haccp, blockchain internet of things. In *2017 International Conference on Service Systems and Service Management*, pages 1–6.
- Galvez, J. F., Mejuto, J., and Simal-Gandara, J. (2018). Future challenges on the use of blockchain for food traceability analysis. *TrAC Trends in Analytical Chemistry*, 107:222 – 232.
- Gamage, H. T. M., Weerasinghe, H. D., and Dias, N. G. J. (2020). A survey on blockchain technology concepts, applications, and issues. *SN Computer Science*, 1(2):114.
- Kolb, J., AbdelBaky, M., Katz, R. H., and Culler, D. E. (2020). Core concepts, challenges, and future directions in blockchain: A centralized tutorial. *ACM Comput. Surv.*, 53(1).
- Li, D., Wang, X., Chan, H. K., and Manzini, R. (2014). Sustainable food supply chain management. *International Journal of Production Economics*, 152:1 – 8. Sustainable Food Supply Chain Management.
- Malik, S., Kanhere, S. S., and Jurdak, R. (2018). ProductChain: Scalable blockchain framework to support provenance in supply chains. In *NCA 2018 - 2018 IEEE 17th International Symposium on Network Computing and Applications*.
- Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.
- Olsen, P. and Borit, M. (2018). The components of a food traceability system. *Trends in Food Science and Technology*, 77:143 – 149.
- Ray, Z., Xun, X., and Lihui, W. (2017). Food supply chain management: systems, implementations, and future research. *Industrial Management & Data Systems*, 117(9):2085–2114.
- Wang, S., Li, D., Zhang, Y., and Chen, J. (2019). Smart contract-based product traceability system in the supply chain scenario. *IEEE Access*, 7:115122–115133.
- Wang, Y., Han, J. H., and Beynon-Davies, P. (2018). Understanding blockchain technology for future supply chains: a systematic literature review and research agenda. *Supply Chain Management: An International Journal*, 24.
- Zhao, G., Liu, S., Lopez, C., Lu, H., Elgueta, S., Chen, H., and Boshkoska, B. M. (2019). Blockchain technology in agri-food value chain management: A synthesis of applications, challenges and future research directions. *Computers in Industry*, 109:83 – 99.