Using Smart Contracts to Automate a Food Hub

(extended abstract of the MSc dissertation)

Rui Mira Gomes Gama da Costa
Instituto Superior Técnico, Universidade de Lisboa

Advisor: Prof. Kevin Gallagher, Prof. Miguel Correia

***Abstract*—Blockchain technology has been used for several purposes in the context of a supply chain, including traceability, transparency, and real-time monitoring systems. A supply chain is a complex process that requires a large number of intermediaries to deal with. Blockchain technology can be used to enable disintermediation, reduce transaction costs, and improve payment efficiency. This work creates a solution that uses smart contracts to automate the role of a food hub. The system automatically pairs multiple farmers to fulfill larger orders, thus integrating the farmers into the supply chain. It does it in a fair way, where we make sure that a farmer will eventually participate in an order. This work deploys three types of smart contracts that the actors, small and mid-sized farmers as well as buyers, can interact with. This system also automates the negotiation process with the buyer for farmers and effectively reduces the number of intermediaries. The system is evaluated using both qualitative and quantitative methods.**

I. Introduction

Blockchain technologies have been used for several purposes in the context of a supply chain. Currently, the main focus has been on traceability, not just for supply chains as a whole [1], but also in the food supply chain [2]. The reason behind this is that blockchain is a distributed ledger that stores information in a distributed way. This, together with cryptographic mechanisms, such as hashes, and replication, makes tampering with information difficult and thus provides greater trust compared to centralized storage systems. Consumers increasingly want more information about the origin of the products they consume and the manufacturing steps [1]. In addition to that, by providing better traceability systems it also increases the level of food safety because it makes it easier for products not safe to consume to be recalled.

Although the use of blockchain technology has increased in recent years, the support of small farmers is still an area where little work has been done [2]. Small and mid-sized farmers don't have enough production capabilities to fulfill larger orders, thus having difficulties entering some supply chains. Besides this, the negotiation process for this kind of transaction is still very inefficient, where farmers and buyers need to trade several messages until an agreement is met. Current solutions for this problem are based on the use of intermediaries [3], [4]. Farmers either sell their products through farm gate sales or food hubs. In a farm gate sale, farmers can sell their products directly to consumers, but they can also sell to an intermediary. Another solution for farmers is to sell their products to food hubs. Food hubs are intermediaries that buy from small and mid-sized farmers and sell to wholesale buyers or consumers. Food hubs also aggregate multiple farmers to be able to sell to their buyers. Using farm gate sales or food hubs, farmers still need to use intermediaries that eat part of the profits that could have been directed to the farmers.

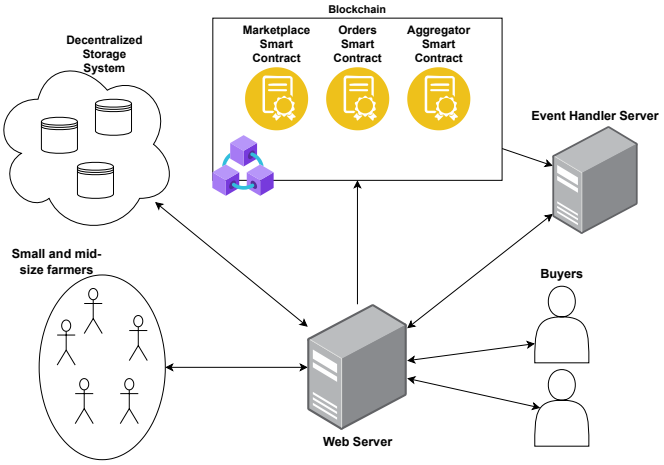

Fig. 1: System architecture

This work creates a solution that uses smart contracts to automate the role of a food hub. Smart contracts pair multiple farmers together to fulfill larger orders from buyers, which can be restaurants, hospitals, etc. Figure 1 shows the proposed general architecture. This work deploys to the blockchain three types of smart contracts that the actors, small and mid-sized farmers as well as buyers, can interact with through the Web Server. The first type is the Marketplace Smart Contract, through which farmers can update their stocks. The second type, the Aggregation Smart Contract, is where buyers are able to place orders. The last type of smart contract acts as an Escrow Smart Contract. Every time an order is placed and the buyer agrees with the proposed offer, this smart contract holds the money of the order. This smart contract is then used to automatically pay the farmers who participate in each order. In addition to smart contracts, the system also makes use of a decentralized storage system to store the order history and other information provided by the farmers. This is done not just to pay fewer gas fees per transaction with the smart contracts but also to allow us to store information in a manner that no single entity controls. This information is used to

help in the aggregation process of the farmers, but it will be explained later in the document. The system also has a Web Server that is responsible for interacting directly with the smart contracts and the decentralized storage system to fulfill farmer and buyer requests, as well as another server, the Event Handler Server, that listens to important Blockchain events emitted by the Aggregator Smart Contract.

A. Objectives

This work seeks to create a smart contract system that **removes the need for intermediaries**, such as food hubs, by aggregating multiple farmers so that they can together fulfill larger orders. Furthermore, this work provides farmers not only with a gas fee consumption system that encourages them to participate in the supply chain through our system but also with a **fairness factor** in the aggregation process. With this work is also expected that farmers will be able to obtain better prices for their crops since the revenue will not have to be shared with an intermediary. Moreover, this work also presents a modular decentralized storage system that stores data separately about each product and only retrieves data about the desired product whenever necessary. The system also contributes with a way that automatically takes care of the negotiation process, provided by the implementation, and automatic payment, provided by both the use of blockchain technologies and the implementation.

B. Overview

The remainder of the document is structured as follows. Section II of this document will talk about blockchain technologies and more specifically smart contracts in supply chains, with more focus on the food supply chain. Section III describes in detail the solution for the presented problem. Section IV-B defines the evaluation methods for the system. Finally, Section V concludes.

II. Background and Related Work

A blockchain is a distributed ledger that stores records of information and transactions [5]. The nodes inside a blockchain network store a copy of the ledger and can verify its validity. Blockchain works by creating a chain of blocks that are linked together by the hash of the previous block. A hash function is a function that transforms the original data into a string that identifies the original data but that, at the same time, is impossible to obtain the original data having the output string. Using a consensus mechanism, this chain of blocks is then agreed upon by the miners in the Blockchain network. This makes it very hard to alter the data that is stored inside the blockchain because an attacker needs to alter not only the targeted block but also all the blocks that follow. The attacker then needs to convince the majority of nodes in the blockchain network to accept these changes. With the consensus mechanism, the hash of the previous blocks, and digital signatures, the blockchain can provide trust, authentication, integrity, and non-repudiation [6]. Blockchain also enables smart contracts. Smart contracts are executable programs that run on the blockchain and can automatically enforce the agreed terms when the agreed predefined conditions are met [7].

A. Blockchain in Supply Chain

Most of the work done with blockchain technologies with a specific area in mind has been conducted in the agricultural sector [8]. Blockchain technology has been used alongside other technologies such as Internet of Things (IoT), RFID, NFC, QR codes, and third-party sensors to get real-world information to improve agriculture and food supply chains [2], [9], [10]. A recent survey studied the current state and potential applications of blockchain technology in supply chain management [11]. The authors found that the main topics of research, or issues that blockchain research is trying to solve, are traceability and transparency, collaboration, supply chain integration, and digitalization. Other studies also confirm that traceability is the area where most of the work with blockchain technologies has been done [1], [12], [13]. Besides this, blockchain can also be used for payments, via a digital currency [14], and facilitate access to finance services such as obtaining credit and trading [15]. The usage of blockchain technologies also contributes to reducing the need for intermediaries due to direct communication between different supply chain actors [9], [16] and has the potential of increasing profits and customer satisfaction [17]. It also contributes to the reduction of costs and transaction times [16]. Although this area has been researched a lot, few studies have focused on system implementations [1].

B. Smart Contracts in Supply Chain

A blockchain-based solution to efficiently manage the supply chain when shipping goods has been proposed [18]. Using a set of smart contracts, the authors propose a system that together with IoT sensors is capable of monitoring shipment conditions, automating payments, and ensuring that the shipment is received by the legitimate buyer. Their system proves that smart contracts can be used to create trust between suppliers and buyers without the need for a third party. Another system that works on the traceability of shipments has used smart contracts together with decentralized storage solutions to reduce costs [19]. Another traceability system using smart contracts concluded that their usage provided stakeholders with data accessibility and data immutability [20]. Smart contracts have also been used for providing data integrity to a fish farm [21]. A study was conducted where two scenarios were proposed [22]. One scenario was a traditional marketplace and the other scenario was where the supply chain was managed on the blockchain. This study found that by managing the supply chain on the blockchain retailers would order larger quantities and would be able to get higher profits. The authors of the study also argue that the inefficiency of the transactions of a traditional marketplace reduced profits and discouraged sales. Although this is true, smart contracts also come with their challenges [7]. Smart contracts are immutable and once deployed they cannot have their code changed. Also, smart contracts cannot obtain real-time information.

C. Food Hubs

Food Hubs are organizations that act as intermediaries between small farmers and wholesale buyers or consumers and provide a platform to aggregate and distribute small farmers' products [3]. There are two types of Food Hubs [3]. The first type is simple online platforms that connect buyers and producers but don't handle transactions or logistics. The second type acts as an online market and can carry out payments and coordinate logistics. Another study also considers that there is a third type of Food Hub that is a hybrid model between these two [23]. There is also the possibility of a food hub directly linking a farmer to a buyer [24]. Food hubs have five main functions: Logistics, such as aggregation and distribution of products, Marketing, such as searching for new markets, Product services, such as providing storage facilities, Consultancy Services, such as providing guidelines, and "Web of practices", acting as a sharing platform of knowledge and experiences [3]. Most of the research is focused on the Support Services function of Food Hubs, with less focus on other functions [25]. Food Hubs together with digital technologies, such as e-commerce platforms, have the potential to overcome constraints, facilitate communication, aggregation, and organization [23]. The implementation of good practices, such as horizontal collaboration and transportation outsourcing, allows the reduction of costs and the increase in logistics capabilities [26]. Much of the time farmers are not willing to adapt to new and improved practices [27]. This is one of the biggest challenges when working with farmers. The idea of an e-community is not new [28]. In this study, the authors propose a conceptual food hub model that has as one of the focus the cooperation among stakeholders in the supply chain. Another study has shown that short food supply chain are better in terms of price achieved, payment security and gives farmers a greater bargaining power [4]. Although Food Hubs give some advantages they all require the usage of a single warehouse where farmers deliver their food to and that aggregates their products all together [29]. Food Hubs actively serve as an intermediary between farmers and buyers. Inevitably, they end up eating part of the profits that could go to farmers.

D. Online Blockchain Marketplaces

A proof-of-concept market for the energy sector based on blockchain technology has been discussed [5]. In this market, producers publish their energy offers on the blockchain. A consumer reads the offers and accepts the cheapest offer. Although a simple idea, the authors show that blockchain technology can be used to connect suppliers with consumers/retailers. Furthermore, their concept together with smart contracts can be used to make more complex transactions, such as trying to bundle multiple small producer offers together to fulfill bigger consumer/retailer orders.

The system described in this document is not entirely new. A system that aggregates multiple farmers in a group and makes available the farmers' crops to be purchased has already been proposed [30]. This system aggregates farmers into groups based on the expected harvesting date and the location of the farmers and the quantities are then made available to be purchased through a website. A buyer proposes a price and farmers can choose to accept or negotiate the price until an agreement is reached. At this point, their system uses smart contracts to enforce the agreed conditions between farmers and the buyer. Although the system just described is similar to what this work implements, some things can be improved. For example, the aggregation mechanism is not described and the aggregation itself is done outside of the blockchain. By doing the aggregation inside the blockchain, the solution in this document brings trust not only between the farmer and the buyer but also between the farmers themselves.

BeIMP [31] is a platform based on blockchain technology that enables manufacturers to form contracts and perform transactions. Their proposal integrates the participation of producers/sellers, buyers, financial institutions that provide loans if necessary, insurance companies, and regulatory agencies. Their system is implemented using a consortium blockchain where the actors participate by joining with three Ethereum nodes. In their system, blockchain is used as a distributed database.

FarMarketplace [32] is a digital marketplace based on blockchain technologies for agricultural products. That system is composed of three main components. An app, from which farmers can publish their offers, contracts, or bids, the digital platform where the trades happen, and the blockchain itself, together with the smart contracts and an off-chain storage system, only the hash is stored inside the blockchain.

A system, based on five smart contracts, has been proposed that connects multiple actors in the healthcare supply chain [33]. The proposed system uses several smart contracts to register manufacturers, distributors, and healthcare providers, place orders, assign distributors, rebate settlements, and for loyalty rewards.

Most of the marketplaces described do not address the problem this work tries to solve. Only one work tries to aggregate multiple farmers together, but their work lacks implementation details, does not use blockchain technologies for all the transaction steps, and only performs qualitative analysis [30]. Other works [31], [32] use some system that stores information both on-chain and off-chain. Some works store all the information in a centralized database and the most important information is also stored in the blockchain [31], others store all information in a centralized database and then store the hash of that information in the blockchain [32]. Only the system that is focused on the healthcare supply chain uses a decentralized storage system [33].

III. Solution

A. Description

The main problem that this project solves is for small and medium-sized farmers to be able to fulfill larger client orders that otherwise they could not. Currently, the best solution for these farmers is to sell their products to food hubs that then sell to manufacturers or consumers. Inevitably, this leads to lower prices for farmers due to intermediaries in the food

supply chain. This work develops a smart contract system that dynamically pairs multiple farmers. The system also automates the negotiation process with the buyer and effectively reduces the number of intermediaries, thus achieving better pricing for farmers. To be able to accomplish this, the system makes use of three distinct types of smart contracts: Marketplace Smart Contract, Aggregation Smart Contract, and Orders Smart Contract. The Marketplace Smart Contract stores the Content Identifier (CID)s of the Stocks and the Orders of each product. The Aggregation Smart Contract is the smart contract that checks if an order is possible and performs the aggregation of farmers. The Orders Smart Contract holds the funds and pays farmers when he is told to do so.

The system is also auditable. It implements mechanisms that make it so that someone can look at the aggregation process and reason if the process is accomplishing the objectives or not. If it is not possible, a new Aggregation Smart Contract can be deployed with a new aggregation process that better satisfies the objectives of the system. Check Section III-D for more details about the auditable aspect of the system and the aggregation process that was implemented.

In a Blockchain Network used in this work, a price is paid, called gas fees, for each executed instruction, and the higher the amount of information stored inside this blockchain, the higher the amount paid in gas fees. Because of that the system makes use of an InterPlanetary File System (IPFS). IPFS is a peer-to-peer data storage system that stores data in a distributed manner [12]. This also keeps information stored in a decentralized manner where no single entity controls it. The IPFS stores information on the farmer's stock and order history. When a farmer calls the stock update function, the smart contract stores in the IPFS the information provided by the farmer, setting the state of the stock to available. The smart contract separates stocks of different products, storing them separately. By doing this, upon the initiation of the aggregation process, it is possible to retrieve only the stocks in the IPFS that are from the product the buyer wants. This is an advantage because it also allows for less gas consumed since there are fewer stocks to filter. Section III-D explains this in more detail.

The system also makes use of a Web Server through which farmers and buyers make their requests. If no Web Server existed the farmers and buyers would be the ones interacting with the decentralized storage system, IPFS, and the smart contracts. By being the ones exchanging the information between smart contract and the IPFS, there could be the case where one farmer, for example, could remove other farmers' stocks from the available stocks. This situation is not desired and that is the main reason for the use of the Web Server in this system. Although such a problem could be easily tracked, since it is possible to check the changes made to the CID in the Marketplace Smart Contract and the blockchain address that made them, the decision to create the Web Server was made to have a higher degree of certainty that no Stock was being unfairly removed. Another server, the Event Handler Server, is used in the system. This server is used after an order request is made to the Orders Smart Contract. This Aggregator Smart Contract emits an event that is caught by the Event Handler Server. This server then sends that event to be processed by the Web Server. Events were used in this case because that is the way to retrieve information from the Blockchain after a writing operation, or a transaction. Section III-D explains the writing that takes place in the aggregation process. The Event Handler Server was used so that we could retrieve information about the order request from the Blockchain.

During the making of a new order, there can be a problem with information usage due to a concurrency problem. If nothing is made, it is possible to have two order requests for the same product come approximately at the same time. One of these order requests is picked first and the Aggregator Smart Contract decides which stocks to use for that order and emits the event with those orders. The Event Handler catches this event and sends it to the Web Server to be processed. After that, the Web Server updates the Stocks CID and the Orders CID, for the product of the order, in the Marketplace Smart Contract after updating the IPFS with the new information. Before this information would be stored in the IPFS and in the Marketplace Smart Contract, the Web Server would process the other order request. This request then arrives at the Aggregator Web Server that receives the request to fulfill that new order with stocks that are no longer available. To solve this problem the Web Server implements a locking mechanism. For each supported product exists a lock. Whenever a transaction begins the Web Server acquires the product lock and after that, the lock is released. In functionalities that require two blockchain transactions, such as the case of making a new order, where one transaction requests the order and the other updates the IPFS and the Marketplace Smart Contract CIDs, the first transaction acquires the lock and the second one releases it. This way we can guarantee that there is only one functionality for each product happening at any given time. In this work, we also assume the pre-existence of a Decentralized Autonomous Organization (DAO). A DAO is an organization composed of several individuals, in this case farmers, without a single leader. Instead, they need to vote for a change to happen. This assumption was made to prevent larger farmers from taking over the system and start fulfilling the orders all by themselves. The system only allows members of the DAO to interact with it. For simplification reasons and because it was not the focus of this work, the DAO is implemented as a simple list.

B. Farmer Interactions

Figure 2 shows the different interactions that a farmer can trigger in the system. In interaction 1, farmers provide the Web Server with information on the expected harvest period, the type of product, the quality of the product, the quantity of the product, the location of the farm, the price that he pretends to get, a threshold for the price he pretends to get and their Ethereum public address. The Web Server retrieves the current CID for the product specified by the farmer from the Marketplace Smart Contract, interaction 2. The Web Server retrieves it from the IPFS, appends the new information to

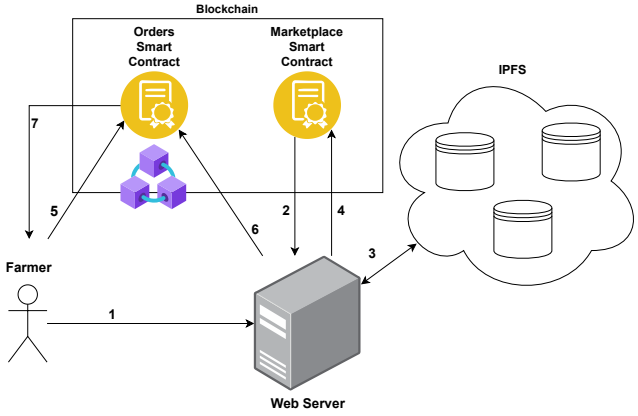

Fig. 2: Farmer Interactions

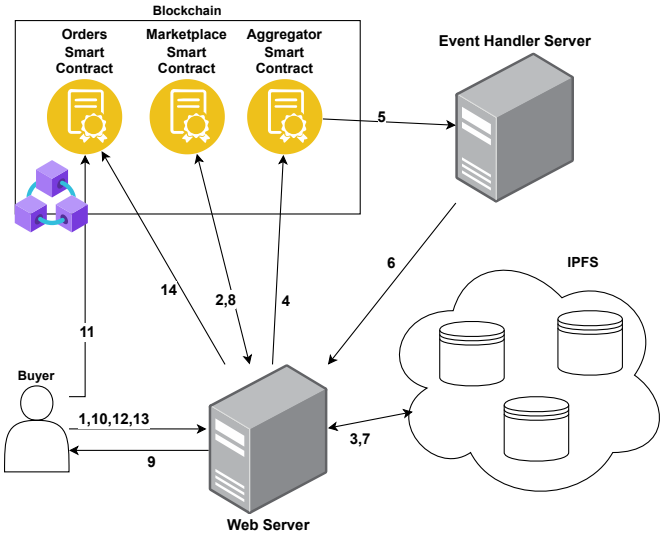

Fig. 3: Buyer interactions

the existing one, and stores it back in the IPFS, interaction 3, retrieving the new product stock CID in the process. After that, the Web Server updates the Marketplace Smart Contract with the new CID, interaction 4. Once an order with the stocks of a farmer has been accepted, the farmer is notified of how much he needs to transfer to the Orders Smart Contract. Farmers are then required to transfer a deposit, a percentage of the amount they will receive from that order, interaction 5. In this case, this deposit is set to 10% of the amount the buyer pays for that farmer's stock. The system does this to guarantee that the farmers comply with its obligations. After that, farmers send their products to the buyer. This part, which can include the tracking of shipments, is considered out of the scope of this work, but, according to the literature review, a lot of the work done with blockchain in the agricultural sector was done for traceability purposes. When a buyer informs the Web Server that the farmer shipment has arrived, the Web Server demands the Orders Smart Contract to pay the farmer and returns part of the deposit made by the farmer, interactions 6 and 7. A small percentage of the deposit is kept in the Orders Smart Contract. This percentage that is kept is used to finance the system. If for some reason there is the need to increase or decrease this percentage there is no need to deploy a new smart contract. The percentage is a parameter that is stored in the smart contract and is changeable. This was done in order to save on gas consumption. Changing the value of the parameter is cheaper than deploying a new smart contract.

C. Buyer Interactions

Figure 3 shows the different interactions that a buyer can trigger in the system. Buyers place orders by providing the Web Server, interaction 1, with information about their location, a radius from that location, this defines an area where the buyer wants products to come from and the aggregation is only done with farmers from inside that radius, the date at which they want the crops, the product they desire with the amount wanted and their Ethereum public address. After this, the Web Server interacts with the Marketplace Smart Contract, interaction 2, to fetch the CID of the desired product and gets the product stocks from the IPFS, interaction 3. After that the Web Server requests the Aggregation Smart Contract, interaction 4, to perform the aggregation process; explained in Section III-D. When the aggregation process finishes, the Aggregation Smart Contract emits an event, interaction 5, that is caught by the Event Handler Server. This event contains information on whether it is possible or not to fulfill the order. The Event Handler Server interacts with the Web Server, interaction 6, which processes the information and saves the new order in the IPFS, the CID in the Marketplace Smart Contract, and then informs the buyer with the proposed offer, interaction 7, 8 and 9 respectively. The buyer can accept the proposed price, decline it, or propose a different price, interaction 10. If the buyer chooses to propose a new price, the system can quickly check if it is possible to fulfill the order at that price based on the price threshold provided by the farmers. The buyer can only propose a new price one time. If the price is possible to fulfil then the order is automatically accepted. If the price is too low and farmers are not able to fulfill it then the order is automatically rejected. This is implemented so that buyers cannot abuse the system to consistently get the lowest possible price. If the number of new proposes was not capped, the buyer could perform something like a binary search, where he proposes a new price every time, checking if the order is possible to fulfill at that price or not, and always getting the lowest possible price that the farmers picked for that order are willing to also accept. When the buyer accepts the order the server notifies all the farmers and the buyer. The buyer then needs to transfer the total amount to the smart contract, interaction 11. The farmer's obligations at this stage are explained in Section III-D. Shipments arrive individually to the buyer, one from each farmer, which then interacts with the system, interaction 12, to update the status of a specific farmer shipment. This involves reading the CID from the Marketplace Smart Contract, retrieving the information from the IPFS,

storing the new one in the IPFS, and storing the new CID in the Marketplace Smart Contract. When the buyer notifies that a shipment arrives, interaction 13, the Web Server requests the Orders Smart Contract, interaction 14, to pay the farmer and to return the down payment made by the farmer. All these three interactions are done through the Web Server, which interacts with the Orders Smart Contract to automate the payment and with the IPFS and Marketplace Smart Contract to update the state of each farmer's stock. The stock of each farmer has several states: available, negotiating, shipping, and used. When a farmer updates his stock, it is stored in the IPFS as available. After the buyer places an order and before accepting it, the stock is in the negotiating state. When the buyer accepts an order, the state changes to shipping. If the order is rejected the stock comes back to the available state. The stock is kept in this state until the buyer says that a shipment has arrived, at which point the state is updated to "used".

D. Aggregation Process

The Aggregation Process is the core component of the system. The aggregation of farmers is done by the following variables/properties:

- Expected harvesting period provided by the farmers - Only stocks that have a harvesting period that contains the data provided by the buyer are considered. This is done for obvious reasons. Only these Stocks are available at the time the buyer desires them;
- Geographical radius provided by the buyer - The aggregation process only considers farmers that indicate to the system that their farm is within the buyer's desired geographical radius;
- Product Quality provided by the farmers - The buyer indicates to the system both the product and the desired quality. The aggregation process only takes into account crops that the farmers indicated as having that quality;
- Fairness - In this work, fairness has several meanings. First, this work has the objective of not leaving farmers behind, in the sense that it does not want to fulfill orders always with the same farmers. Second, it is not desirable that a large farmer, capable of fulfilling large orders, can overtake the system and start fulfilling all the orders by itself. This is ensured by the existence of the DAO. This fairness factor is a mandatory requirement for all the farmers who want to participate in the supply chain via this system.

In the Aggregation process stocks are picked randomly. There are two reasons behind this choice. First, picking stocks randomly consumes less gas compared to other systems, such as having a scoring system calculated for each order and picking the stocks with the highest score. Second, by picking stocks that can be used for an order request randomly we can be sure that eventually every farmer participates in an order. True randomness is hard to achieve inside a blockchain, which is a deterministic system. To implement a function that is as random as possible, the Aggregator Smart Contract picks the next stock to be used based on the previous block randao [34], the block timestamp, and the array of available Stocks. The randao is a pseudorandom generated value [34]. For any given block, the randao is defined by performing a XOR operation between the previous randao value and a randomly generated value by the block validator. In the case of the Blockchain used in this system, this randomly generated number is the validator signature of the current epoch number. If there is the need to pick more than one stock for a certain order using only the previous randao and the block timestamp would give always the same value. That is why the Aggregator Smart Contract also uses the array of available stocks. This array changes every time a new stock is picked. The Aggregator Smart Contract computes the hash of these values together, interprets it as an integer, and uses it as a random value.

function MakeNewOrder (S, O)
Input : An array S of the product Stocks,
A structure O with the order request details
$OrderStocks \longleftarrow BeginAggregationProcess(S, O)$
$generateMerkleTree()$
$emitNewOrderEvent()$
function BeginAggregationProcess (S, O)
Input : An array S of the product Stocks,
A structure O with the order request details
Output: An Empty Array if an Order is not possible or an Array with the Product Stocks used in the Order
Initialize Array to add Available Stocks
$totalQuantity \longleftarrow 0$
foreach $s \in S$ **do** `// Filter Stocks`
  if *Stock s meets requirements* **then**
    $totalQuantity \longleftarrow$
    $totalQuantity + s.quantity$
    Add Stock s to Array
  end
end
if $totalQuantity < O.quantity$ **then** `// Order Not Possible`
  return *Empty Stock Array*
end
return $Aggregate(Available\ Stock\ Array)$ `// Order Possible`

function Aggregate (A, q)
Input : An array S of the available product Stocks,
The quantity q for the order
Output: An Array with the Product Stocks used in the Order
Initialize Array to add the Order Stocks
$quantity \longleftarrow 0$
while $quantity < q$ **do**
  $s \longleftarrow pickRandomStock()$
  Add s to Order Stocks Array
  $quantity \longleftarrow quantity + s.quantity$
end
return *Order Stocks Array*

Algorithm 1: Aggregation Process Pseudo code

Algorithm 1 shows the pseudocode of the Aggregation Process. This work aims to encourage farmers to participate in the supply chain with the proposed system and not use an intermediary that requires a large commission or consumes part of the profits. To do that, the aggregation process must choose farmers who are probably not the best fit in terms of harvest date or location, but who have not participated in the aggregation process for some time. For the aggregation itself to happen, the Aggregation Smart Contract starts by filtering the stocks by the variables mentioned above and whether they are in the available state or not. After filtering, the stocks to use in the order are randomly picked. After the aggregation process is finished, the Aggregation Smart Contract saves in a Merkle Tree the farmer stocks input array, the buyer's request, and the aggregation decision. A Merkle Tree is a structure that stores hashs and where each node is the hash of the two child nodes. A Merkle Tree was used in this case because it allows us to store the same amount of information via the hashes of the information while using less space. Because space inside a smart contract is something that can be expensive, using a Merkle Tree allows for less gas consumption. After this, an event is emitted. The event is caught by the Event Handler Server that sends it for processing to the Web Server. The Web Server appends the new information to the existing one, stores it in the IPFS, and updates its CID in the Marketplace Smart Contract. The Web Server also stores logs of each new farmer stock, buyer request, and order output after the aggregation process has occurred. The reason behind this is that this way the system is fully auditable. A person or company can backtrack from any point and can understand the reasons that led to the aggregation decision of a specific order. It is then possible to take conclusions, understand if the fairness process is being fair and how fair, and then the DAO is able to vote for an adjustment to the aggregation process if the members desire to do so. If, for some reason, it is not possible to trust the server logs, the person or company auditing the system can ask farmers and buyers what their inputs were to the system or use the information that is stored in the IPFS obtained with the Marketplace Smart Contract CIDs.

IV. Implementation and Evaluation

A. Implementation Details

The system is developed using the Ethereum Network and using Remix Integrated Development Environment (IDE). Ethereum is a public blockchain, which means that anyone can interact with or join the network at any given time. Besides that, Ethereum provides us with a Turing-complete language, Solidity. Remix is a Solidity IDE used for Web3 Development that allows developers to test their Smart Contracts without the need to first deploy them to the Ethereum Main Network or a Test Network. The deployment of the smart contracts was made to the Ethereum Sepolia Test Network, which, at the time of implementing the system and writing this document, is the recommended testing network by Ethereum. The deployment was also made using Remix IDE since it is also able to connect to a Network through a wallet. For that purpose, MetaMask was used. The Web Server is implemented in Python using the Flask framework to deploy an Application Programming Interface (API). The Event Handler Server is also implemented in Python.

B. Methodology

Based on similar works [31], [32], [33] the system is evaluated based on qualitative and quantitative methods. Regarding the quantitative analysis, the evaluation of the system is done in terms of gas fee consumption for each transaction and functionality as well as the average execution time. Regarding qualitative analysis, the system is evaluated by the security properties it provides to all the stakeholders. In [33] the authors evaluate their system concerning confidentiality, data integrity, availability, authorization, non-repudiation, and vulnerability to cyberattacks. This work is evaluated for the same properties. The randomness in the aggregation process also is evaluated since it provides an important objective of this work, the fairness factor.

To create a database that was the most realistic, we searched for what is considered a small farmer by the European Union in terms of farm hectares and the average production in kg per hectare in Portugal. To generate the order requests, we randomly generated orders that would request 1.5 to 5 times the average production of the farmers from the generated farmers' stock database. This is done to guarantee that during our testing the aggregation process would be triggered and would aggregate multiple farmers some of the times. All of this was generated for a specific product. For location, we considered only randomly picked locations in Portugal. The quality field on each entry in the database was also picked randomly and followed the classification standards given by the United Nations Economic Commission for Europe [35]. All the dates were also picked randomly and between two dates that were 90 days apart.

The system is tested in the Sepolia Test Network. When using a Test Network the amount of gas consumed is the same but the final price of one transaction or to deploy a smart contract is different because of the price that is paid per gas. For that reason, the results consider an Ethereum price of 1534.725 EUR, collected at 12:45:59 on 16 September 2023, given by the Coinbase API. When testing all functionalities, we also recorded the Proposed Gas Price on the Main Ethereum Network, given by the Etherscan API. During the testing, the average Gas Price given by the API was 9 GWEI, or 0.000000009 ETH, and that is the value used to calculate the final price presented on Tables II, III and I.

C. Quantitative Analysis

In this section, we present the evaluation results of the system in terms of quantitative analysis. We present the cost of deploying smart contracts and the cost of transactions and functionalities. In this section, we also discuss the randomness of the Aggregation Process that we use to pick the stocks to be used in the order.

1) Smart Contracts: Table I shows the cost of deploying the Smart Contract that composes the system. The Final Price is calculated with the conditions referenced in Section IV-B. The cheapest Smart Contract to deploy is the Orders Smart Contract with a cost of 697650 gas. Next is the Marketplace Smart Contract with a cost of 835070 gas. The most expensive, and also most complex, smart contract to deploy is the Aggregator Smart Contract with a cost of 1858694 gas. This higher cost is because this smart contract has more instructions compared to the other two and uses more storage space inside the blockchain.

TABLE I: Smart Contract Deployment

Smart Contract	Gas Used	Price (EUR)
Marketplace	835070	11.534425
Aggregator	1858694	25.673257
Orders	697650	9.636308066

2) System Functionalities: Any transaction in the Ethereum network must pay gas fees. A call to a Smart Contract Function is considered a transaction if there is any sort of write in the blockchain. In this work, every buyer or farmer interaction can be made only by calling four different functions. The results for each of these transactions can be seen in Table II. Results for each system functionality can be seen in Table III. Some functionalities, Informing a Shipment has arrived and making a new order, require two transactions to happen.

In the Marketplace Smart Contract two functions are called, one to update a product stock CID and another to update both a product stock CID and the product orders CID. In the Aggregator Smart Contract, the only function called is to try to make an order. Lastly, when a buyer informs that a shipment has arrived, he triggers a call to the Orders Smart Contract to pay the respective farmer. Two of these transactions, update stock CID and pay address, had a constant Gas amount consumed because they either store a string always with the same length or make a transfer, respectively. They consume 41703 gas and 46173 gas respectively. The transaction that updates the Order CID and the Stock CID has two possible Gas consumption because it is possible that only one of the CIDs is updated. For example, when making an order it is possible that after filtering the stocks we conclude that there is not enough quantity to fulfill the order. In this case, the system records that there was a new order and that it was impossible to fulfill. The system writes this change to the IPFS, getting a new CID for the orders, and the Stocks CID remains the same because no Stock needs the state updated. This means that only one of the CIDs is changed in the Marketplace Smart Contract. When both CID are updated the function consumes 59580 gas. When only one is updated the function consumes 53580 gas. The transaction to make a new order also does not have a constant gas consumed as others do. This is because the transaction has a step in which one stock is picked randomly until the desired quantity is met. This can lead to a new stock being picked any number of times. Besides that, the transaction begins by filtering the stocks. In this filtering process, there is an if statement with 5 conditions. The smart contract checks these conditions one by one and stops checking the stock conditions when it finds a false condition. This means that even the filtering process does not consume a constant amount of gas for the same input size. If an order is impossible to fulfill our testing shows that the function has an average consumption of 1815165 gas. Picking a new Stock to be used in the order added an average of 456112 gas per stock picked to the function cost. Besides the cost of each system functionality, Table III also shows the transactions that are involved in every one of them. Results for gas used, total price, and average execution time in the case of functionality that requires two transactions are the addition of the results of both transactions.

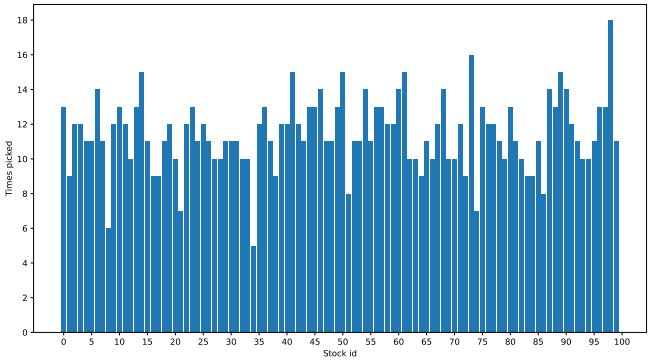

Fig. 4: Histogram of the Randomness in the Aggregation Process

3) Aggregation Process Randomness: To test how random our stock pick method was we performed a different test. For this test, we generate a new database consisting of 100 stocks of a product and 10 order requests. We generated these stocks and orders in a way that it would always be possible to fulfill the order, meaning that the quality was always considered the same, the order dates were always in the middle of every stock harvesting period, and the radius provided by the buyer was always set so that it would always be higher than the maximum possible distance. We then proceeded by performing the 10 order requests. This process was repeated several times until at least 1000 random stocks were picked. This way we could have a good sense of how random the aggregation process actually was. In total, 1145 stocks were randomly picked by the Aggregator Smart Contract. With this test, we expect to see that no stock has a higher probability of being picked than the others. In Figure 4 we can see how many times each Stock was picked in this process. In this figure, we can observe that every stock is picked and no Stock was picked several times which is much higher than the others. In order to see how random it is, and subsequently how fair the system is, a lot more stocks would have to be randomly picked. Nonetheless, this way we can have a good sense of how random and fair the Aggregation process is.

V. Conclusion

Small and medium-sized farmers have difficulty trying to integrate the supply chain. Current solutions require the use

TABLE II: Results Table by Transaction

Transaction		Gas Used	Price (EUR)	Avg. Execution Time (s)
Update Stock CID		41703	0.576024	18.18
Update Order CID and Stock CID		53980 or 59580	0.745600 or 0.823075	13.50
Pay Address		46173	0.637766	14.67
Make New Order	Impossible to fulfil Order (Avg.):	1815165	25.072012	18.08
	Pick New Stock (Avg.):	456112	6.300058	

TABLE III: Results Table by Functionality

Functionality	Transaction(s) triggered	Total Gas Used	Total Price (EUR)	Avg. Execution Time (s)
Update Stock	Update Stock CID	41703	0.576024	18.18
Money Sent (Farmer)	Update Stock CID	41703	0.576024	18.18
Accept/Reject/Negotiate Order	Update Order CID and Stock CID	59580	0.823075	13.50
Money Sent (Buyer)	Update Order CID and Stock CID	59580	0.823075	13.50
Inform Shipment has Arrived	Pay Address + Update Stock CID	87876	1.213789	32.85
Make Order	Make New Order + Update Order CID and Stock CID	Starts at 1869145	Starts at 25.82	31.58

of an intermediary that eats part of the profits that can be diverted to the farmers. This work implements a system that automatically pairs multiple farmers to fulfill larger orders, thus integrating the farmers into the supply chain. Currently, no existing system is able to do that and be fair at the same time. The system described in this document pairs farmers randomly until the buyers' request is met. In addition to that, the system described in this document also reduces the inefficiency of the negotiation process by automatically taking care of it. In this system, payments are also automatic and efficient. The system is important for farmers because it allows them to stop being price takers and allows them to define the price they feel is fair for them. It encourages farmers to participate in the supply chain through the system because the aggregation process is made in a way that is certain that sooner or later the farmer will participate in an Order. For buyers, this system allows them to communicate with only one entity instead of multiple to fulfill their desired quantities. Using smart contracts and blockchain technologies to implement such a system provides farmers with a platform that brings trust to the whole process without the need to know each other

A. Future Work and System Limitations

An important thing that was left out of the scope of this work is to find a way to verify that what farmers and buyers say to the system is true. As it is, there is the possibility that a malicious buyer makes several order requests without the intention of actually accepting one. Doing so would have the potential to spend all the funds that the system has. It is also important to find a way to verify that farmers also have the product that they inform the system they have. Achieving this can be done through the integration of this work system with a traceability system that requires photos to be taken, for example. This way it would not just be possible to verify that farmers are telling the truth but also trace the shipments during the shipment process. Some solutions try to create true and verifiable randomness, such as Chainlink Verifiable Random Function (VRF), but this solution comes with the drawback that it would cost an extra 0.25 LINK for each request, increasing the price of the transaction, and is not synchronous.

REFERENCES

[1] M. K. Lim, Y. Li, C. Wang, and M.-L. Tseng, "A literature review of blockchain technology applications in supply chains: A comprehensive analysis of themes, methodologies and industries," *Computers & Industrial Engineering*, vol. 154, p. 107133, Apr. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360835221000371

[2] A. Kamilaris, A. Fonts, and F. X. Prenafeta-Bold, "The rise of blockchain technology in agriculture and food supply chains," *Trends in Food Science & Technology*, vol. 91, pp. 640–652, Sep. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0924224418303686

[3] G. Berti and C. Mulligan, "Competitiveness of Small Farms and Innovative Food Supply Chains: The Role of Food Hubs in Creating Sustainable Regional and Local Food Systems," *Sustainability*, vol. 8, no. 7, p. 616, Jul. 2016. [Online]. Available: http://www.mdpi.com/2071-1050/8/7/616

[4] A. Malak-Rawlikowska, E. Majewski, A. Was, S. O. Borgen, P. Csillag, M. Donati, R. Freeman, V. Hoàng, J.-L. Lecoeur, M. C. Mancini, A. Nguyen, M. Saïdi, B. Tocco, Török, M. Veneziani, G. Vittersø, and P. Wavresky, "Measuring the Economic, Environmental, and Social Sustainability of Short Food Supply Chains," *Sustainability*, vol. 11, no. 15, p. 4004, Jul. 2019. [Online]. Available: https://www.mdpi.com/2071-1050/11/15/4004

[5] J. J. Sikorski, J. Haughton, and M. Kraft, "Blockchain technology in the chemical industry: Machine-to-machine electricity market,"

Applied Energy, vol. 195, pp. 234–246, Jun. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306261917302672

[6] M. Di Pierro, "What Is the Blockchain?" *Computing in Science & Engineering*, vol. 19, no. 5, pp. 92–95, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8024092/

[7] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, Apr. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X19316280

[8] P. Gonczol, P. Katsikouli, L. Herskind, and N. Dragoni, "Blockchain Implementations and Use Cases for Supply Chains-A Survey," *IEEE Access*, vol. 8, pp. 11 856–11 871, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8952728/

[9] S. Köhler and M. Pizzol, "Technology assessment of blockchain-based technologies in the food supply chain," *Journal of Cleaner Production*, vol. 269, p. 122193, Oct. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S095965262032240X

[10] P. Burgess, F. Sunmola, and S. Wertheim-Heck, "Blockchain Enabled Quality Management in Short Food Supply Chains," *Procedia Computer Science*, vol. 200, pp. 904–913, 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050922002976

[11] S. E. Chang and Y. Chen, "When Blockchain Meets Supply Chain: A Systematic Literature Review on Current Development and Potential Applications," *IEEE Access*, vol. 8, pp. 62 478–62 494, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9047881/

[12] W. Lin, X. Huang, H. Fang, V. Wang, Y. Hua, J. Wang, H. Yin, D. Yi, and L. Yau, "Blockchain Technology in Current Agricultural Systems: From Techniques to Applications," *IEEE Access*, vol. 8, pp. 143 920–143 937, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9159588/

[13] M. Pournader, Y. Shi, S. Seuring, and S. L. Koh, "Blockchain applications in supply chains, transport and logistics: a systematic review of the literature," *International Journal of Production Research*, vol. 58, no. 7, pp. 2063–2081, Apr. 2020. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207543.2019.1650976

[14] S. E. Chang, Y.-C. Chen, and M.-F. Lu, "Supply chain re-engineering using blockchain technology: A case of smart contract based tracking process," *Technological Forecasting and Social Change*, vol. 144, pp. 1–11, Jul. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0040162518305547

[15] D. Kos and S. Kloppenburg, "Digital technologies, hyper-transparency and smallholder farmer inclusion in global value chains," *Current Opinion in Environmental Sustainability*, vol. 41, pp. 56–63, Dec. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877343519300557

[16] Y. Kayikci, N. Subramanian, M. Dora, and M. S. Bhatia, "Food supply chain in the era of Industry 4.0: blockchain technology implementation opportunities and impediments from the perspective of people, process, performance, and technology," *Production Planning & Control*, vol. 33, no. 2-3, pp. 301–321, Feb. 2022. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09537287.2020.1810757

[17] S. Stranieri, F. Riccardi, M. P. Meuwissen, and C. Soregaroli, "Exploring the impact of blockchain on the performance of agri-food supply chains," *Food Control*, vol. 119, p. 107495, Jan. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0956713520304114

[18] H. Hasan, E. AlHadhrami, A. AlDhaheri, K. Salah, and R. Jayaraman, "Smart contract-based approach for efficient shipment management," *Computers & Industrial Engineering*, vol. 136, pp. 149–159, Oct. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360835219304140

[19] L. Wang, L. Xu, Z. Zheng, S. Liu, X. Li, L. Cao, J. Li, and C. Sun, "Smart Contract-Based Agricultural Food Supply Chain Traceability," *IEEE Access*, vol. 9, pp. 9296–9307, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9317793/

[20] S. Wang, D. Li, Y. Zhang, and J. Chen, "Smart Contract-Based Product Traceability System in the Supply Chain Scenario," *IEEE Access*, vol. 7, pp. 115 122–115 133, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8804170/

[21] L. Hang, I. Ullah, and D.-H. Kim, "A secure fish farm platform based on blockchain for agriculture data integrity," *Computers and Electronics in Agriculture*, vol. 170, p. 105251, Mar. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016816991932006X

[22] P. De Giovanni, "Blockchain and smart contracts in supply chain management: A game theoretic model," *International Journal of Production Economics*, vol. 228, p. 107855, Oct. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925527320302188

[23] F. Sgroi and G. Marino, "Environmental and digital innovation in food: The role of digital food hubs in the creation of sustainable local agri-food systems," *Science of The Total Environment*, vol. 810, p. 152257, Mar. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0048969721073332

[24] D. A. Cleveland, N. M. Müller, A. C. Tranovich, D. N. Mazaroli, and K. Hinson, "Local food hubs for alternative food systems: A case study from Santa Barbara County, California," *Journal of Rural Studies*, vol. 35, pp. 26–36, Jul. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0743016714000436

[25] F. R. Hermiatin, Y. Handayati, T. Perdana, and D. Wardhana, "Creating Food Value Chain Transformations through Regional Food Hubs: A Review Article," *Sustainability*, vol. 14, no. 13, p. 8196, Jul. 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/13/8196

[26] A. Marusak, N. Sadeghiamirshahidi, C. C. Krejci, A. Mittal, S. Beckwith, J. Cantu, M. Morris, and J. Grimm, "Resilient regional food supply chains and rethinking the way forward: Key takeaways from the COVID-19 pandemic," *Agricultural Systems*, vol. 190, p. 103101, May 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0308521X21000548

[27] A. Mittal, C. Krejci, and T. Craven, "Logistics Best Practices for Regional Food Systems: A Review," *Sustainability*, vol. 10, no. 2, p. 168, Jan. 2018. [Online]. Available: http://www.mdpi.com/2071-1050/10/1/168

[28] M. Ioannis, M. George, and M. Socrates, "A Community-Based Agro-Food Hub Model for Sustainable Farming," *Sustainability*, vol. 11, no. 4, p. 1017, Feb. 2019. [Online]. Available: http://www.mdpi.com/2071-1050/11/4/1017

[29] A. Mittal and C. C. Krejci, "A hybrid simulation modeling framework for regional food hubs," *Journal of Simulation*, vol. 13, no. 1, pp. 28–43, Jan. 2019. [Online]. Available: https://www.tandfonline.com/doi/full/10.1057/s41273-017-0063-z

[30] M. Kumarathunga, R. Calheiros, and A. Ginige, "Towards Trust Enabled Commodity Market for Farmers with Blockchain Smart Contracts," in *Proceedings of the 2020 Asia Service Sciences and Software Engineering Conference*. Nagoya Japan: ACM, May 2020, pp. 75–82. [Online]. Available: https://dl.acm.org/doi/10.1145/3399871.3399891

[31] C.-H. Liao, H.-E. Lin, and S.-M. Yuan, "Blockchain-Enabled Integrated Market Platform for Contract Production," *IEEE Access*, vol. 8, pp. 211 007–211 027, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9265212/

[32] G. Leduc, S. Kubler, and J.-P. Georges, "Innovative blockchain-based farming marketplace and smart contract performance evaluation," *Journal of Cleaner Production*, vol. 306, p. 127055, Jul. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0959652621012749

[33] I. A. Omar, R. Jayaraman, M. S. Debe, K. Salah, I. Yaqoob, and M. Omar, "Automating Procurement Contracts in the Healthcare Supply Chain Using Blockchain Smart Contracts," *IEEE Access*, vol. 9, pp. 37 397–37 409, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9363880/

[34] M. Kalinin and D. Ryan, "EIP-4399: Supplant DIFFICULTY opcode with PREVRANDAO," *Ethereum Improvement Proposals*, no. 4399, Oct. 2021, [Online; accessed 23-October-2023]. [Online]. Available: https://eips.ethereum.org/EIPS/eip-4399

[35] U. N. E. C. for Europe, "Fresh Fruit and Vegetables - Standards," 2022, [Online; accessed 23-October-2023]. [Online]. Available: https://unece.org/trade/wp7/FFV-Standards