IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 July 2022, accepted 30 July 2022, date of publication 3 August 2022, date of current version 8 August 2022.
Digital Object Identifier 10.1109/ACCESS.2022.3196162

RESEARCH ARTICLE

Blockchain-Based Traceability for the Fishery Supply Chain

PRATYUSH KUMAR PATRO[1], **RAJA JAYARAMAN**[1], **KHALED SALAH**[2], **(Senior Member, IEEE), AND IBRAR YAQOOB**[2]
[1]Department of Industrial and Systems Engineering, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
[2]Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
Corresponding author: Ibrar Yaqoob (ibrar.yaqoob@ku.ac.ae)

This work was supported by the Center for Digital Supply Chain and Operations Management—Khalifa University of Science and Technology under Award RCII-2019-002.

ABSTRACT Efficient traceability management is necessary for managing products in the fishery supply chain. Any negligence in products' traceability can result in food fraud that may adversely affect the consumer's health. Monitoring and tracking of the fishery supply chain operations can assist system stakeholders in identifying the origins and causes of product fraud and malpractice. Most of the existing systems and technologies used to manage the fishery supply chain processes fall short of providing traceability, accountability, transparency, reliability, trust, privacy, and security features. In this paper, we propose a private Ethereum blockchain-based solution to efficiently manage the fishery supply chain operations in a manner that is decentralized, transparent, traceable, secure, private, and trustworthy. We present the solution architecture and propose five smart contracts to automate the processes in fishery supply chain. We present ten algorithms along with their full implementation, testing, and validation details. We conduct a security analysis to show that the proposed solution is both secure and trustworthy. We compare our solution with the existing blockchain and non-blockchain-based solutions to show its effectiveness and novelty. We make the smart contract code publicly available on GitHub.

INDEX TERMS Blockchain, Ethereum, fishery supply chain, smart contract, traceability.

I. INTRODUCTION

Fish is among the most consumed food products globally. It is considered one of the essential sources of all proteins [1]. All fish products are delivered to their final destination via complex supply chain networks involving various stakeholders handling various supply chain operations [2], [3]. Generally, the fishery supply chain can be classified into two categories based on production processes and harvesting methods. One is farmed fish, and the other is wild-caught [2]. The farmed fish production technique involves growing and harvesting fish in a controlled aquaculture ecosystem. On the other hand, the wild-caught fish production technique signifies catching fish from natural ecosystems such as oceans and rivers [2]. Therefore, monitoring the product flow in these fishery supply chains is vital for ensuring food safety and preventing fraudulent activities.

Traceability is one of the significant challenges that persist in the fishery supply chain. In most cases, fish product consumers find it hard to realize what species they are eating [16]. Product traceability is one of the significant challenges that persist in the fishery supply chain. It is often difficult for fish consumers to identify what species they consume. In addition, mislabelling, species substitution; illegal, unreported, unregulated (IUU) fishing; and species adulteration have become major challenges that undermine consumer trust and have a catastrophic effect on consumer health [3]. Therefore, an efficient traceability system to combat food fraud in the fishery supply chain has become an essential requirement [4]. The growing popularity and adoption of digital technologies in various business sectors has improved the scale of efficient product traceability systems. Near Field

The associate editor coordinating the review of this manuscript and approving it for publication was S. K. Hafizul Islam.

Communication (NFC), RFID, and Quick Response (QR) codes are among the most widely adopted product tracking and tracing tools. However, these tools and techniques are ineffective in dealing with fragmented data despite their user-friendliness and ease of implementation [5], [6]. In addition, these techniques use a centralized data management system, which could pose vulnerabilities to data tampering and centralized point failure. Furthermore, as the fishery supply chain handles a huge variety of species globally, the stakeholders in the supply chain face significant problems in managing fragmented product data, recording product data during product transformation, managing data asymmetry, and maintaining data compliance [7], [8]. In addition, when a product gets recalled, the lack of an efficient traceability system can make it challenging to identify all the users who might have been affected [8]. Therefore, developing an efficient traceability system can address these challenges [9].

Blockchain technology can address the aforementioned challenges by bringing together all the supply chain stakeholders under one platform and providing a feasible solution guaranteeing product traceability. Some organizations like the Worldwide Fund for Nature (WWF), Food and Agriculture Organization (FAO), and Fishcoin have already exploited the features of blockchain technology and implemented them successfully in their projects [10]–[12]. As blockchain is tamper-proof, its contribution to any system can effectively bring security, traceability, and trustworthiness. Although the current blockchain-based techniques in the fishing industry maintain product information continuity during the pre-processing and post-processing stages, they fail to track fish data during the fish processing phase when the fish undergoes morphological changes. Also, these approaches rely on RFID and QR code-based tools to track and store information about product movement. Although integrating blockchain with IoT devices ensures better data integrity and management, it falls short of providing efficient traceability management of the aquaculture supply chain [14]–[15]. Such gaps in the supply chain may invite species adulteration and counterfeiting during fish processing due to the removal of RFID tags and QR codes [10], [13]. Thus, fraudsters can exploit this opportunity to commit food fraud by switching high-value species for low-value ones. So, once the skin of a fish species is removed, it's easy for fraudsters to change the fish products [16].

Another possible drawback with the current blockchain-based techniques in the fishery supply chain is the inability to differentiate wild-caught and farmed fishes during fish processing when there is a common fish processing facility. Failure to identify the fillets of each processed fish based on their origin of production may result in species substitution [17], [33]. To overcome such challenges, the PCR-DNA techniques of fish species have become a reliable method for determining the genetic details of the fish species and identifying the origin of fish species [16]–[18], [26]. Studies on applying blockchain technology to human genomics are nascent, and few approaches seem suitable for practical implementation [19]. However, there has not yet been a feasible solution to implement such a technique in the fishery supply chain to prevent fish product fraud and maintain better product traceability.

To address the plausible challenges and improve upon existing fish traceability methods, a programmable blockchain platform such as Ethereum can be used. Users can execute these programs in the Ethereum Virtual Machine environment. Typically, smart contracts are protocols that allow traceable and irreversible transactions to be verified and executed without the intervention of a third party [39]. The objective of our research is to develop an efficient blockchain-based traceability system that can handle both wild-caught and farmed fish in a secure and trustworthy manner.

In this paper, we propose a blockchain-based fish traceability solution that can handle both wild-caught fish and farmed fishery supply chain processes using Ethereum smart contracts in a manner that is decentralized, transparent, private, secure, and trustworthy. Figure 1 depicts an overview of the fishery supply chain process followed across the world and being adopted from the existing studies [1], [3], [6], [11], [19], [31]. The main contributions of this paper are as follows:

- We propose a private Ethereum blockchain-based solution to handle the fishery supply chain operations in a decentralized, traceable, accountable, transparent, private, secure, and trustworthy manner.
- We present system architecture and sequence diagrams to explain the stakeholders and their interactions in the fishery supply chain processes.
- We propose ten algorithms and develop five smart contracts to implement and test the functionalities of processes in the fishery supply chain. We make the smart contract code publicly available on GitHub.[1]
- We present the security analysis to show our solution is secure and trustworthy and compare it with the existing blockchain and non-blockchain-based solutions to highlight its novelty and effectiveness.

The rest of this paper is organized as follows. Section II presents related work. Section III discusses the proposed blockchain-based system with all system participants and demonstrates stakeholder interactions through sequence diagrams. In Section IV, we describe how smart contracts are implemented and the algorithms used. Section V presents the testing and validation details of the proposed solution. In Section VI, we discuss the security analysis, comparison study, and the generalization aspect. We provide concluding remarks in Section VII.

II. RELATED WORK

In this section, we present the existing blockchain and non-blockchain-based solutions proposed for the traceability of fish products in wild-caught and farmed fishery supply chains.

[1]https://github.com/PratyushKumarPatro/Fish-Supply-chain

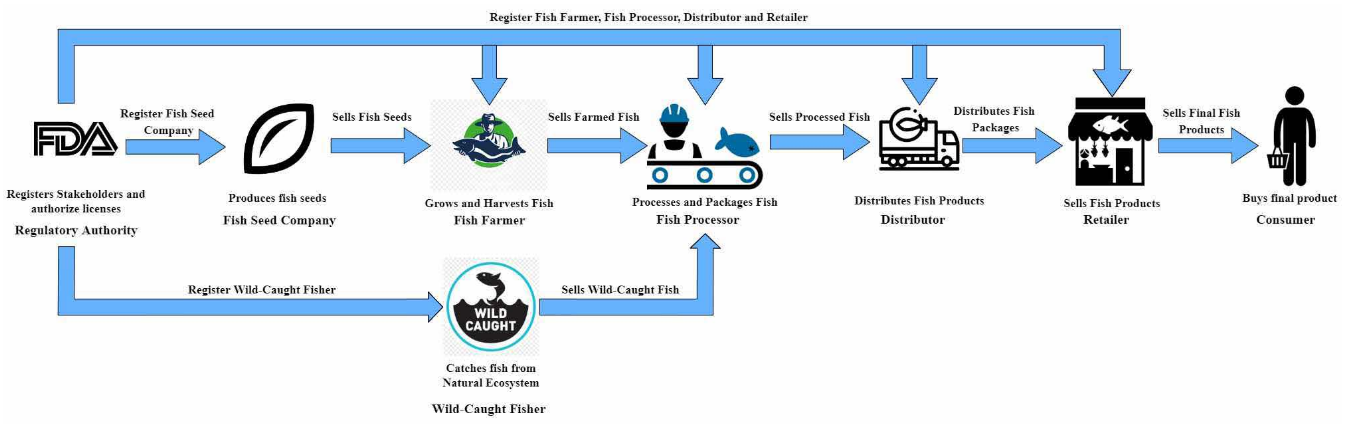

Register Fish Farmer, Fish Processor, Distributor and Retailer
FDA
Register Fish Seed Company
Registers Stakeholders and authorize licenses
Regulatory Authority
Produces fish seeds
Fish Seed Company
Sells Fish Seeds
Grows and Harvests Fish
Fish Farmer
Sells Farmed Fish
Processes and Packages Fish
Fish Processor
Sells Processed Fish
Distributes Fish Products
Distributor
Distributes Fish Packages
Sells Fish Products
Retailer
Sells Final Fish Products
Buys final product
Consumer
Register Wild-Caught Fisher
WILD CAUGHT
Catches fish from Natural Ecosystem
Wild-Caught Fisher
Sells Wild-Caught Fish

FIGURE 1. An overview of the fishery supply chain process [1], [3], [6], [11], [19], [31].

A. NON-BLOCKCHAIN-BASED APPROACHES

In the supply chain process of every product, all stakeholders must interact in a trustworthy manner. Therefore, building an efficient traceability system is necessary to identify the root causes of food fraud throughout the supply chain. The authors in [3] categorized possible food frauds in the seafood supply chain. Among the nine categories of food fraud, the majority were mislabelling, species substitution, IUU, and fishing substitution. The authors recommended a few control point techniques to mitigate food fraud.

In [19] and [37], the authors highlighted the illegal use of food additives and glaze water on fish products as one of the emerging concerns of food fraud in the fishery supply chain. The use of additives and glaze water in fish products severely affects the quality of those products. Furthermore, the findings of the study show that the lack of a commonly understood nomenclature of fish species is a significant challenge. To reduce the risk of food fraud, the authors initiated a project for fish tracing and proposed documenting various key details of fish, such as morphological data of species, DNA barcoding, and geographical origin of fish, for a better understanding of the supply chain stakeholders. However, this approach involves manual activities and is centralized, leading to data tampering and negatively affecting the supply chain processes.

The design of an efficient traceability system significantly relies on adopting vital traceability principles. In [21], the authors proposed six important traceability principles for wild-caught fish products, as most of the time, wild-caught fish harvesting practices include illegal and unregulated fish catch methods. In addition, the authors also recommended five principal roles of a traceability system that can be implemented for both wild-caught and farmed fishery supply chains. Similarly, in [23], the authors recommended four important tools and six principles required for developing a successful traceability system. These six principles signify the following requirements: full-chain traceability; digital information tracking and sharing; transparency and public access to the information; effective tracking post product transformation; verification; and defining essential information. However, these studies only show theoretical approaches; they don't offer a prototype solution for a traceability system that different organizations can easily use.

There are various organizations harnessing cutting-edge techniques like RFID, QR codes, and barcodes to implement them in supply chain processes. In [5], the authors studied the current methodologies and approaches adopted by seafood supply chain companies. In this study, various parameters were identified, such as the geographical origin of fish species, production methods, etc., that are needed to track the origins of food fraud. The study also explains how DNA profiling and spectrometry inspection can be used to analyze the DNA of caught fish products to authenticate their quality, using the results of this study. In this paper, several suggestions are given and recommendations are made for the implementation of a robust traceability system, but the issue of food fraud is not addressed when the fish is cut into fillets in the supply chain. In addition, the study fails to study how to maintain transparency in the sharing of data among all parties in a trusted manner.

In [4], the authors highlighted the parameters for seafood product tracking and proposed tracking of three vital parameters: species identification, method of production, and geographical origin of seafood. In this study, the authors also explain a few techniques that can enhance the existing traceability methods that can help prevent seafood fraud. However, the study does not demonstrate any system architecture and prototype for applying these techniques to combat fish fraud and malpractice.

In [24], the authors proposed a prototype for tracking the mobility of the live fishery supply chain using RFID technology. In this approach, traceability is conducted by recording feed, conducting inspections and testing, managing stock, and receiving the final product. Throughout the supply chain, RFID tags are attached to the live fish, and the movement of the fish is tracked in real-time to monitor all of these activities. While this approach focuses on the tracking of live fish, it does not address the issue of species substitutions and adulteration during processing. Furthermore, this approach is centralized and manual. The data may be lost in the event of a failure of the central point.

In [6], the authors aimed to develop an automated tracking system for the product flow in the farmed fishery supply chain. The advantages of this approach were the automatic collection of data using RFID tags. Moreover, the young fish and eggs were tracked by labeling the fish tank and cages in this approach. However, the method does not provide any solution for preventing species substitution, illegal catches, etc., of farmed fish. One application of RFID technology was to monitor the temperature and humidity of the cold chain part of the fishery supply chain. Using RFID tags to monitor fishes' temperature and environmental conditions, the authors propose a method of maintaining the quality of fish [25]. The approach, however, focuses solely on the cold chain aspect of fish quality and proposes no solution to combat fish fraud and malpractice. Ocean aquaculture is rapidly being depleted due to IUU fishing practices that have been increasing every year. Hence, an effective traceability system should be utilized to track wild-caught fisheries.

Developing effective documentation schemes and tracing systems can prevent the sale of illegally caught seafood products. Furthermore, the authors mention the risk due to the introduction of illegally caught fish products into the farmed fishery supply chain if there is more than one fish processor in different geographic locations [26]. Additionally, the researcher examines three major challenges in establishing a successful traceability system: system compatibility, data standardization, and defining a traceable entity. A prototype or practical implementation is not demonstrated in this study. This study does not create a mechanism to separate wild-caught from farmed fish. Unlike the approaches mentioned previously, various methods for inspecting food quality were developed using cutting-edge techniques such as DNA barcoding, genome sequencing, and so on. For example, in [27], the authors recommended using PCR to identify the DNA of fish species in the salmon fishery supply chain. In the interlaboratory trials, it was observed that utilizing enzyme cut techniques can help identify the exact species type. These results led other labs to adopt this method. Even so, this approach did not come up with a good way for stakeholders to share the results of the PCR test in a way that kept a high level of integrity.

All the aforementioned approaches fail to provide a trusted, reliable, and tamper-proof data management system in the fishery supply chain. Therefore, distributed and decentralized technology such as blockchain can help address the challenges. The following subsection demonstrates the relevant blockchain-based techniques to establish traceability in the fishery supply chain.

B. BLOCKCHAIN-BASED APPROACHES

All stakeholders in any supply chain must be kept up to date on the status of their product flow. Blockchain technology can help create a platform where the data can be visible to all legitimate stakeholders. Furthermore, the data transacted in the blockchain system is tamper-proof and permanently stored in the ledger. Therefore, all the fishery supply chain stakeholders are accountable for their information during their business transactions and cannot deny their actions. In a report published by the Food and Agriculture Organization [11], the authors explain some ways to harness blockchain technology in the seafood supply chain. The study categorized seafood supply chain processes into six categories. The authors also recommended a method to collect key data elements from seafood supply chain operations by categorizing the functions of the supply chain based on the location of the operations into five states. To address the existing challenges in the life cycle of seafood products, the researchers in [28] examined the application of blockchain in seafood supply chain management. According to the authors, using NFC, RFID, and QR code techniques together could help combat traceability challenges. However, these techniques fail to present a prototype and focus only on the supply chains of seafood, not farmed fish. Therefore, there is species substitution as a result of introducing outside-caught fish into the ecosystem. The study also found problems in the seafood supply chain that haven't been fixed, such as damage to fish tags and labels caused by transportation.

Blockchain technologies are already being implemented in the fishery supply chains of several countries. A sustainable fishery supply chain system is a top priority for fishing industries across the globe. [29] describes a study that used blockchain technology to improve Ghanaian fishery supply chains. As part of the supply chain process, seven stakeholders were involved in this study. As a way to track fish species entering the fishery supply chain, digital profiles of fish species were proposed as a means of verification. The study suggests putting the digital profiles of the species on the blockchain and checking the fish's identity before shipping in case the RFID tags get damaged. As a solution to this issue, researchers recommended verifying fish species based on the digital profile they created for the species as well as developing product tags based on the verified fish species. Similar to the previous approach, in [30], the authors demonstrated a blockchain-based proposal to create a transparent and traceable platform in the fishery supply chain. QR codes, which are created based on critical data elements' inputs, are used to track fish products at every stage of the supply chain. However, there are still problems with preventing species replacement after fish are processed and their shapes change.

Traceability can be further enhanced by integrating cutting-edge technologies. Nonetheless, blockchain technology and IoT are not new to the seafood supply chain. Researchers in [15] presented a review paper highlighting the benefits of the adoption of IoT combined with Blockchain technology. This study also looks at how IoT devices can store data on blockchain for better traceability management. As part of the implementation, the authors of [35] proposed a blockchain-based solution that would reduce the scalability challenges associated with blockchain-based approaches for the fishery supply chain. To record transacted data in the blockchain, hash functions are used to create hash values. With IoT devices, this approach uses public and private Ethereum blockchain platforms to track and monitor events and environmental data. However, these approaches have not discussed the security and vulnerability aspects of blockchain implementation in the fishery supply chain.

The existing blockchain-based technologies are capable of tracking fish product data efficiently, but the input of legitimate and reliable data to the blockchain system is crucial to meeting the goal of an effective data management system. It is only possible if all stakeholder inputs are trusted and valid. A fisherman is one of the essential stakeholders for data input. In [12], the project stakeholders developed an Ethereum blockchain platform by rewarding small fish farmers to share the legitimate fish catch data. This method aims to address the challenges such as highly fragmented stakeholders, information asymmetries, etc. Furthermore, this project used Ethereum smart contracts for the development of a wallet for Fishcoin deposits, recording data from IoT sensors, and implementing a protocol to handle the data elements of business transactions in the seafood industry. This method is fairly new, and it makes it easier for real data to get into the blockchain system. However, it doesn't solve the problems caused by species substitution during processing.

To address the entry of illegal and unethical products into the seafood supply chain, the WWF conducted a traceability pilot project. The supply chain operations in this project were automated using Ethereum smart contracts. The proposed approach encompassed three components: a collection of physical and human resources data; tracking devices; and smart contracts. With this approach, the seafood species is given an RFID tag that transfers data to the database in real-time using Near-Field communication technologies. The RFID tags are removed after the fish arrive at the processing facility, and they are attached to a QR code during packaging. This approach, however, does not show how the fish is processed; rather, the fish is gutted and gilled before packaging. It therefore fails to offer any solution to prevent the substitution of species with fillets or adulteration of species with fillets [10]. Also, there is no discussion of how smart contracts are secured and no explanation of how blockchain is applied to the farmed fishery supply chain.

Fish processing is an essential value-adding activity in the wild-caught fish and farmed fishery supply chains. It is therefore possible for a single fish processor to handle both wild-caught and aquaculture-sourced fish [33]. [34] proposes a blockchain-based traceability system based on Ethereum for both farmed and wild fish. With this approach, the smart contract has been encoded with logic that tracks fish capturing and processing, transportation, selling, and quality assessment events. Yet, this approach fails to explain the responsible entities' activities clearly, and regulatory authority is absent. A further problem with this approach is that it fails to consider species substitution given the morphological changes that occur during processing. Consequently, many organizations utilize DNA barcoding and PCR technologies to test fish products to prevent food fraud. A feasibility study on blockchains in genomics is necessary in order to ensure the tracking of DNA details of fish species and share them with all stakeholders. The authors of [20] conducted a study and explained how blockchain technology can be used to store genetic details of patients. Furthermore, the study points out that data storage is a major challenge for blockchain systems to handle data from millions of patients. Although this approach is only conducted for humans, it can be applied to ensure the quality of fish products.

In summary, none of the above-mentioned blockchain and non-blockchain-based approaches to fishery supply chain traceability provides a full-proof solution to establish an integrated system for handling farmed fish and wild-caught fish products. Additionally, current solutions do not address plausible issues like adulteration of wild and farmed fish at the point of fish processing. Despite their shortcomings, none of these approaches analysed the security aspects of smart contracts. Therefore, in our approach, we present an integrated blockchain-based prototype solution for product traceability in both wild-caught fish and farmed fishery supply chains with few unique product data points.

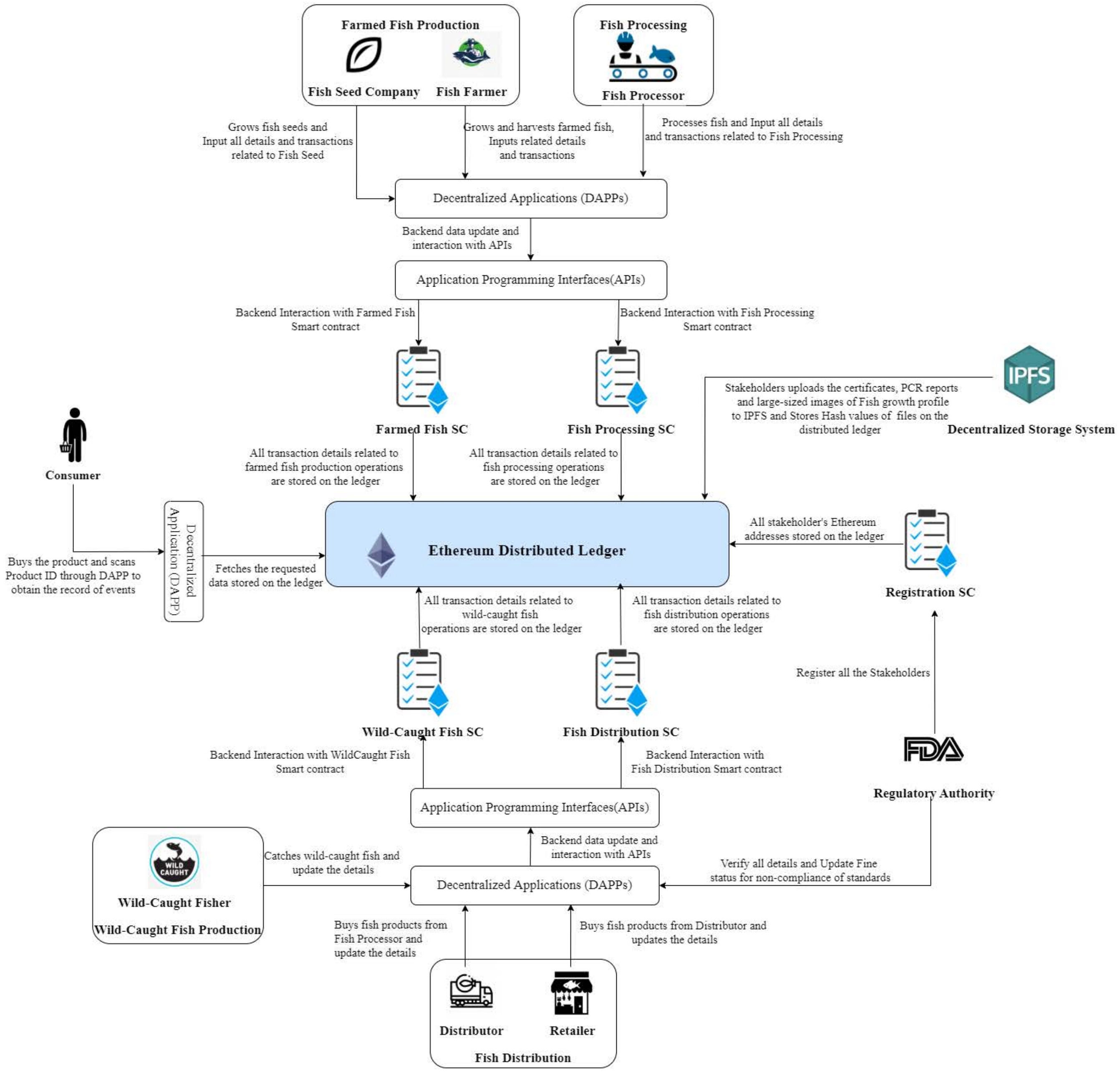
III. PROPOSED BLOCKCHAIN-BASED SOLUTION

In this section, we present our proposed system architecture that combines blockchain technology with Ethereum-based smart contracts to establish traceability both in farmed fishery supply chains as well as wild fishery supply chains. Figure 2 shows the stakeholders, system elements, and responsibilities of each stakeholder within the fishery supply chain process.

A. SYSTEM PARTICIPANTS AND ELEMENTS

Herein, we discuss the key activities of each stakeholder in the fishery supply chain. It also describes the key parts of the system that make it possible for stakeholders and elements to share important data.

- **FDA**: For each business operation, it is the FDA's job to register and make sure that all legitimate stakeholders on the blockchain network are in compliance with the law [31]. It is also in charge of imposing penalties on the responsible stakeholders in the event of noncompliance with standards.
- **Fish Seed Company**: The Fish Seed Company acts as a producer of fish seeds or eggs by breeding the fish only in the farmed fishery supply chain. It also conducts PCR tests to identify the DNA of the fish seeds and stores the details on an IPFS storage system for the verification of the fish species. It is also responsible for selling fish seeds to the fish farmer.
- **Fish Farmer**: The fish farmer buys the fish seeds and grows the fish in an aquaculture ecosystem. The aquaculture ecosystem could be made up of both salt water and fresh water. It is also in charge of monitoring and tracking the stages of fish growth and the aquaculture's



Farmed Fish Production
Fish Seed Company
Fish Farmer
Fish Processing
Fish Processor
Grows fish seeds and Input all details and transactions related to Fish Seed
Grows and harvests farmed fish, Inputs related details and transactions
Processes fish and Input all details and transactions related to Fish Processing
Decentralized Applications (DAPPs)
Backend data update and interaction with APIs
Application Programming Interfaces(APIs)
Backend Interaction with Farmed Fish Smart contract
Backend Interaction with Fish Processing Smart contract
Stakeholders uploads the certificates, PCR reports and large-sized images of Fish growth profile to IPFS and Stores Hash values of files on the distributed ledger
IPFS
Decentralized Storage System
Farmed Fish SC
Fish Processing SC
Consumer
All transaction details related to farmed fish production operations are stored on the ledger
All transaction details related to fish processing operations are stored on the ledger
Decentralized Application (DAPP)
Ethereum Distributed Ledger
All stakeholder's Ethereum addresses stored on the ledger
Buys the product and scans Product ID through DAPP to obtain the record of events
Fetches the requested data stored on the ledger
Registration SC
All transaction details related to wild-caught fish operations are stored on the ledger
All transaction details related to fish distribution operations are stored on the ledger
Register all the Stakeholders
Wild-Caught Fish SC
Fish Distribution SC
FDA
Backend Interaction with WildCaught Fish Smart contract
Backend Interaction with Fish Distribution Smart contract
Regulatory Authority
Application Programming Interfaces(APIs)
Backend data update and interaction with APIs
WILD CAUGHT
Wild-Caught Fisher
Wild-Caught Fish Production
Catches wild-caught fish and update the details
Decentralized Applications (DAPPs)
Verify all details and Update Fine status for non-compliance of standards
Buys fish products from Fish Processor and update the details
Buys fish products from Distributor and updates the details
Distributor
Retailer
Fish Distribution

FIGURE 2. A representation of the proposed blockchain-based system architecture with smart contracts for traceability in the fishery supply chain.

environmental conditions. In addition, the fish farmer uploads the fish growth images and profiles for future auditing of standards.

- **Wild-Caught Fisher**: The wild-caught fisher catches fish from natural ecosystems such as the ocean, ponds, and lakes. It is also responsible for updating details about wild-caught fish and selling fish to the fish processor.
- **Fish Processor**: The fish processor is the most important stakeholder in our study. A fish processor receives various fish species and segregates them based on their geographic origin and method of production. A fish processor processes the fish by removing skin and fins from the fish and cutting them into fillets. Later, it also conducts PCR tests of the fish products to identify the genetic data and stores the details on IPFS for future verification. After processing the fish, it does packaging of the processed fish and sells it to the distributor.
- **Distributor**: The distributor is responsible for getting orders in bulk and delivering the goods to registered retailers.
- **Retailer**: The retailer is responsible for buying processed fish packages from the distributor and selling them to the consumer.

- **Consumer**: The consumer is the final stakeholder in the fish product life cycle. The consumer buys fish products and consumes the food.
- **Smart Contracts**: Smart contracts have programmable logic and can execute functions and trigger events. These contracts enforce the rules agreed by the stakeholders. We have designed a system architecture consisting of five smart contracts; namely, the Registration Smart Contract, Farmed Fish Smart Contract, Wild Caught Fish Smart Contract, Fish Processing Smart Contract, and Fish Distribution Smart Contract. Smart contracts trigger events when their functions are initiated.
- **Distributed Data Storage System**: To overcome the blockchain's storage constraints, distributed storage systems like IPFS are capable of storing large files, images, and records. Our method uses large files, like pictures and videos of fish growth profiles, PCR reports of fish species, and other important documents. The blockchain ledger can be used to store the hash values of large amounts of data. Because of blockchain technology's immutability feature, information can be verified and accessed by stakeholders after it is placed on the IPFS system.
- **Decentralized Application (DApps)**:Supply chain stakeholders interact with the smart contracts through a Decentralised Application (DApp). A DApp is used by stakeholders to scan QR codes, RFID tags, etc., and obtain relevant information about the products they handle. The stakeholders in our proposed solution interact with their respective DApps separately to obtain product information
- **Application Programming Interface (API)**: Application Programing Inteface (API) is utilized to link the software devices which contain the DApp with the smart contracts.

B. SEQUENCE OF OPERATIONS

Four diagrams are presented below to demonstrate all the proposed supply chain operations. A variety of functions and events are used to present all the necessary inputs and outputs to the stakeholders.

The sequence diagram presented in Figure 3 shows the interaction among the Fish Seed Company, Fish Farmer, Fish Processor, and IPFS with the Registration and Farmed Fish smart contracts. These stakeholders are registered on the blockchain system by the FDA, which acts as the regulatory authority. Upon deploying the registration smart contract and successfully registering all the stakeholders, the blockchain system generates a unique identification code, which is used to identify the transactions created and received by the stakeholders. On successful registration of all the stakeholders, the Fish Seed company deploys the Farmed Fish Smart Contract and updates the farmed fish seed details such as species name of the fish seeds, geographic origin, number of fish seeds available, PCR report ID of the fish seeds, aquaculture water type, and hash value of IPFS. During this process, the fish seed company shares the large-sized documents and files with the IPFS, which can be accessed by the stakeholders. After successful update of the fish seed details, the fish farmer calls the function *PlaceSeedsPurchaseOrder* to create a purchase order to buy fish seeds. While executing this function, the fish farmer enters key inputs such as the Ethereum address of the fish seed purchaser, the Ethereum address of the receiver, and the number of fish seeds ordered. This function creates a fish seed purchase order ID containing the details of the fish seed purchase order. The fish farmer calls *ConfirmFishSeedsPurchaseOrder* function in response to the acceptance or rejection of the purchase order. The fish farmer executes the function, which includes the fish seed purchase order ID. Upon successful acceptance of the fish seed purchase order, the fish seed company initiates the fish seed shipment. Later, the fish farmer invokes the function *ReceiveFishSeedsShipment* to notify all stakeholders of the receipt of the fish seed shipment.

Upon receiving the fish seed shipment, the fish farmer grows fish and updates the fish growth details using the function *UpdateFarmedFishGrowthDetails*. The output of this function sends notifications to all the stakeholders. The fish processor must use the function *PlaceFarmedFishPurchaseOrder* to order the farmed fish from the fish farmer. Upon successful execution of this function, a farmed fish purchase order ID is generated that includes details such as the Ethereum address of the farmed fish purchaser, the Ethereum address of the farmed fish seller, the species name, and the number of fish ordered. Upon receiving the farmed fish purchase order ID, the fish farmer accepts or rejects the purchase order by updating the status using the function *ConfirmFarmedFishPurchaseOrder*. The fish farmer initiates the farmed fish shipment after receiving confirmation of the purchase order. Upon successful receipt of the shipment, the fish farmer generates a farmed fish shipment ID and uses it as a key input in the transaction. Upon successful receipt of the shipment, the fish processor calls the function *ReceiveFarmedFishShipment*, which notifies all the stakeholders about the status of the consignment.

Similar to the farmed fishery supply chain processes, the wild-caught fishery supply chain processes are also executed by deploying the Wild-Caught Fish Smart contract. Figure 4 demonstrates the supply chain process interaction among the wild-caught fisher, the fish processor, and smart contracts. The Wild-Caught Fisher deploys the Wild-Caught Fish Smart Contract and updates the Wild-Caught Fish details such as species name, geographic origin, type of ecosystem, number of wild-caught fish available, water type, and date of catch. Upon successful update of the wild-caught fish key details, the fish processor calls the function *PlaceWildCaughtFishPurchaseOrder* to buy the wild-caught fish. Upon receiving notification of a wild-caught fish purchase order, the wild-caught fisher accepts or rejects the purchase order by calling the function *ConfirmWilCaughtFishPurchaseOrder*. Upon successful acceptance of the purchase order, the wild-caught fisher initiates the shipment of wild-caught

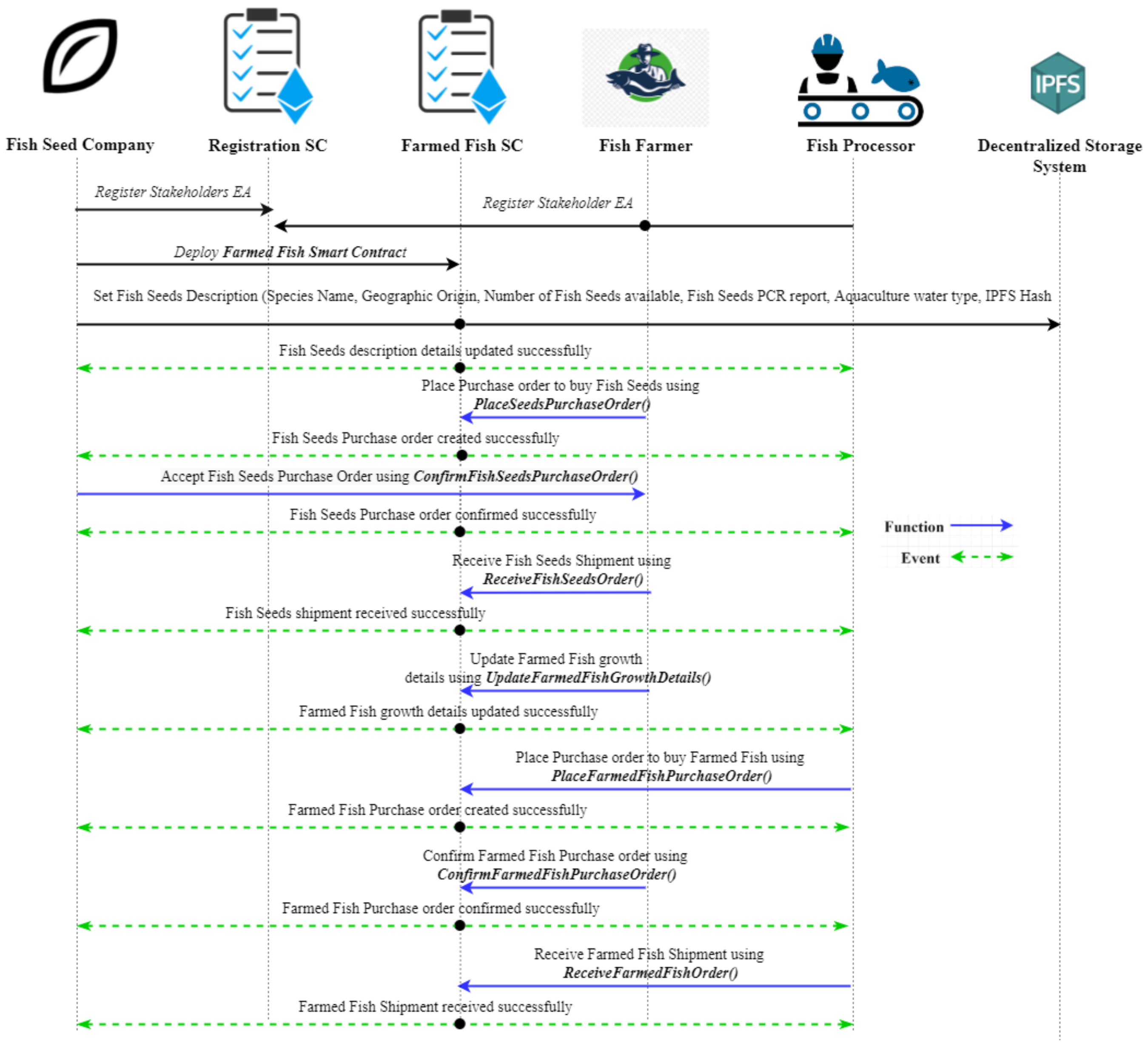

Fish Seed Company
Registration SC
Farmed Fish SC
Fish Farmer
Fish Processor
Decentralized Storage System
IPFS
Register Stakeholders EA
Register Stakeholder EA
Deploy Farmed Fish Smart Contract
Set Fish Seeds Description (Species Name, Geographic Origin, Number of Fish Seeds available, Fish Seeds PCR report, Aquaculture water type, IPFS Hash
Fish Seeds description details updated successfully
Place Purchase order to buy Fish Seeds using PlaceSeedsPurchaseOrder()
Fish Seeds Purchase order created successfully
Accept Fish Seeds Purchase Order using ConfirmFishSeedsPurchaseOrder()
Fish Seeds Purchase order confirmed successfully
Function
Event
Receive Fish Seeds Shipment using ReceiveFishSeedsOrder()
Fish Seeds shipment received successfully
Update Farmed Fish growth details using UpdateFarmedFishGrowthDetails()
Farmed Fish growth details updated successfully
Place Purchase order to buy Farmed Fish using PlaceFarmedFishPurchaseOrder()
Farmed Fish Purchase order created successfully
Confirm Farmed Fish Purchase order using ConfirmFarmedFishPurchaseOrder()
Farmed Fish Purchase order confirmed successfully
Receive Farmed Fish Shipment using ReceiveFarmedFishOrder()
Farmed Fish Shipment received successfully

FIGURE 3. **Sequence diagram presenting interactions among fish seed company, smart contracts, fish farmer, fish processor and smart contracts.**

fish. By calling *ReceiveWildCaughtFishShipment* function, the fish processor informs stakeholders about the status of receiving the wild-caught fish shipments.

After the successful receipt of both farmed fish and wild-caught shipments, the fish processor starts processing the fish. After completion of fish processing, the fish processor deploys the fish processing smart contract and updates the processed fish details to the Fish Processing smart contract. In this process, the fish processor calls the function *CreateProcessedFishPackageID*, which creates a unique ID. Figure 5 illustrates how the fish processor and the smart contract interact using a sequence diagram. The fish processor uploads large files and PCR result reports to the IPFS and stores the hash of the reports and documents in the blockchain.

The sequence diagram presented in Figure 6 explains the mutual interaction among the FDA, Fish Processor, Distributor, Retailer, Consumer, and smart contracts. To purchase the processed fish and distribute to the retailer, the distributor calls the function *PlaceProcessedFishPurchaseOrder*. This function triggers notification to all the stakeholders and fish processor responds to this purchase order by calling the function *ConfirmProcessedFishPurchaseOrder*. The fish processor initiates processed fish shipment after successful acceptance of the processed fish purchase order. After the successful delivery of the processed fish consignment

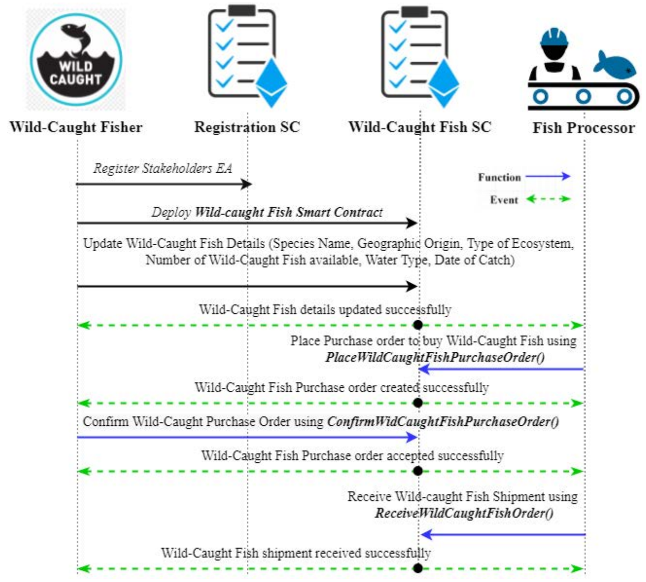

Wild-Caught Fisher
Registration SC
Wild-Caught Fish SC
Fish Processor
Function
Event
Register Stakeholders EA
Deploy Wild-caught Fish Smart Contract
Update Wild-Caught Fish Details (Species Name, Geographic Origin, Type of Ecosystem, Number of Wild-Caught Fish available, Water Type, Date of Catch)
Wild-Caught Fish details updated successfully
Place Purchase order to buy Wild-Caught Fish using PlaceWildCaughtFishPurchaseOrder()
Wild-Caught Fish Purchase order created successfully
Confirm Wild-Caught Purchase Order using ConfirmWidCaughtFishPurchaseOrder()
Wild-Caught Fish Purchase order accepted successfully
Receive Wild-caught Fish Shipment using ReceiveWildCaughtFishOrder()
Wild-Caught Fish shipment received successfully

FIGURE 4. Sequence diagram presenting interactions among wild-caught fisher, smart contracts, fish processor.

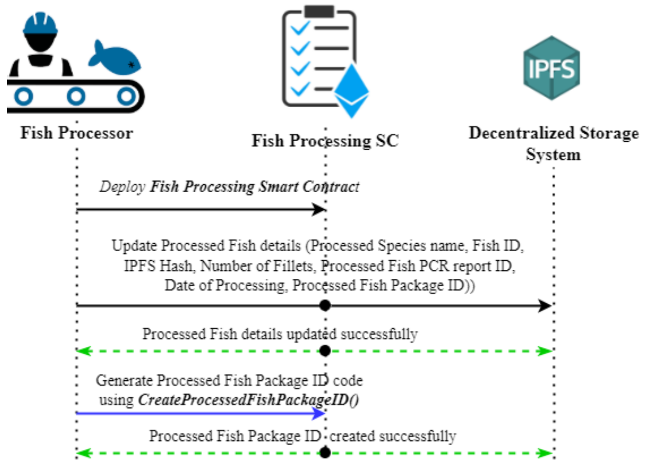

Fish Processor
Fish Processing SC
Decentralized Storage System
IPFS
Deploy Fish Processing Smart Contract
Update Processed Fish details (Processed Species name, Fish ID, IPFS Hash, Number of Fillets, Processed Fish PCR report ID, Date of Processing, Processed Fish Package ID))
Processed Fish details updated successfully
Generate Processed Fish Package ID code using CreateProcessedFishPackageID()
Processed Fish Package ID created successfully

FIGURE 5. Sequence diagram presenting interactions between fish processor and smart contract.

the retailer notifies to the stakeholders about the status of the shipment by calling the function *ReceiveProcessedFishShipment*.The retailer creates a purchase order for the processed fish packages by calling *CreaterRetailerPurchaseOrder* function to transport them from the distribution centre to the retail shop. Afterwards, the distributor accepts or rejects the purchase order by calling the function *ConfirmRetailerPurchaseOrder*. The successful acceptance of the retailer purchase order, the distributor initiates the shipment. Once the shipment gets received, the retailer calls the function *ReceiveRetailerShipment* to confirm receiving of the retailer shipment. After the product is made available to the market, the consumer buys the final product from the retailer. In our study, the consumer is not part of the private blockchain as the consumer may not be comfortable in sharing his/her

Algorithm 1 Updating Fish Seeds Description Details

```
1  Input: Registration smart contract EA, Species Name
   Geographic Origin, Number of fish seeds available, fish
   seeds PCR result report ID, Aquaculture water type,
   IPFS Hash
2  Output: Generate FishSeedsDescriptionsSet Event
3  if FunctionCaller ≠ FishSeedCompany then
4      The Ethereum address of Fish Seed Company is
       invalid.
5  end
6  else
7      if Function Caller EA = Fish Seed Company EA
        then
8          Notify the stakeholders by triggering an event
           about the fish seed details: Ethereum address of
           Registration smart contract, Species Name
           Geographic Origin, Number of fish seeds
           available, fish seeds PCR result report ID,
           Aquaculture water type, IPFS Hash, and
           Ethereum address of Fish Seed Company.
9      end
10     else
11         Bring the contract back to its original state by
            highlighting an error message.
12     end
13 end
```

product purchase data to a public platform, which could invade his/her privacy. In order to obtain the record of events, the consumer can only view the transaction details by scanning the product QR code through a decentralized application. The decentralized application is a FDA authorized interface system, which obtains the transnational data which is already available in the blockchain. In our approach, the role of the FDA is not only restricted to registering the stakeholders, but also ensuring compliance of standards in supply chain operations followed by other stakeholders. The FDA calls the function *CreateFineStatus* to update the Fine status result in case of any violations.

IV. IMPLEMENTATION DETAILS

In this section, we present ten algorithms along with their implementation details. We discussed our five developed smart contracts that incorporate business logic for automated transaction execution. For the development and execution of smart contracts, we use the Remix IDE tool.

Algorithm 1 explains the process of updating the fish seed description details such as species name, geographic origin, number of fish seeds available, fish seed PCR result report ID, aquaculture water type, IPFS Hash, and Ethereum address of the fish seed company. According to the proposed system's architecture, the fish seed company is the stakeholder that provides all of these inputs to the system and updates all of this information to the Farmed Fish smart contract to notify

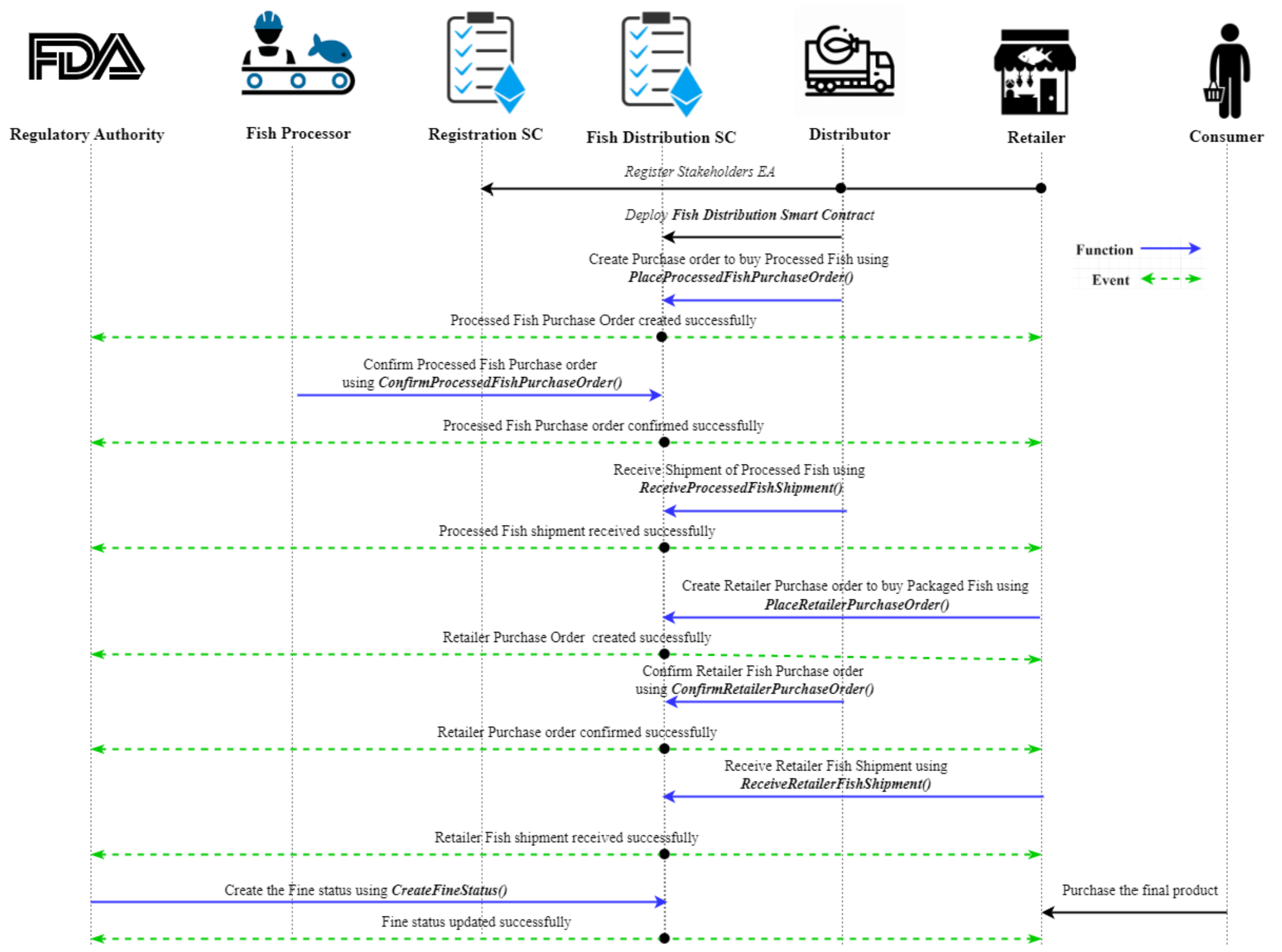

Regulatory Authority
Fish Processor
Registration SC
Fish Distribution SC
Distributor
Retailer
Consumer
Register Stakeholders EA
Deploy Fish Distribution Smart Contract
Create Purchase order to buy Processed Fish using PlaceProcessedFishPurchaseOrder()
Function
Event
Processed Fish Purchase Order created successfully
Confirm Processed Fish Purchase order using ConfirmProcessedFishPurchaseOrder()
Processed Fish Purchase order confirmed successfully
Receive Shipment of Processed Fish using ReceiveProcessedFishShipment()
Processed Fish shipment received successfully
Create Retailer Purchase order to buy Packaged Fish using PlaceRetailerPurchaseOrder()
Retailer Purchase Order created successfully
Confirm Retailer Fish Purchase order using ConfirmRetailerPurchaseOrder()
Retailer Purchase order confirmed successfully
Receive Retailer Fish Shipment using ReceiveRetailerFishShipment()
Retailer Fish shipment received successfully
Create the Fine status using CreateFineStatus()
Fine status updated successfully
Purchase the final product

FIGURE 6. **Sequence diagram presenting interactions among FDA, fish processor, smart contracts, distributor, retailer and consumer.**

the stakeholders. All the stakeholders must be aware of the description and status of the fish seed details to act further. In this approach, the input species name is required to identify the category of the fish species that will be farmed. The input geographic origin signifies the country of origin of the fish seeds. The fish seed PCR result report ID contains the DNA report of the fish species, and the aquaculture water type indicates whether the fish species are grown in salt water or fresh water. The IPFS storage system is used to store large files of data for stakeholder reference. The IPFS hash value and address of the IPFS storage system are then stored on the blockchain. Using these details, all the stakeholders become aware of the fish species that they will harvest and process in the future. This also ensures only legitimate stakeholders can access the information.

The logic written Algorithm 2 explains the logic of creating a purchase order for the fish seeds, confirming the purchase order, and receiving the shipment. The fish farmer acts as the purchaser of the fish seeds and the initiator of the algorithm. The inputs given by the fish farmer include the Ethereum address of the fish seed company, the number of fish seeds requested, and the status of the purchase order request. The code written in the algorithm deploys Keccak256 and abi.encodePacked cryptographic hash function [38] to encrypt the input details to create a unique hash code value, used as the fish seeds purchase order ID. Upon creation, this ID remains collision-free and irreversible. With this unique ID, stakeholders can track the status of the purchase order. A notification is sent to the stakeholders following the successful creation of the purchase order. In return, the fish seed company accepts the purchase order and sends the fish seed shipment to the fish farmer. The fish farmer receives the fish seeds and grows the fish.

According to Algorithm 3, fish growth details are updated from the juvenile to adult stages. The fish farmer periodically uploads the data about fish growth, which includes the weight of each fish, the total number of fish in the farm, the fish farmers' Ethereum address, and the IPFS Hash value. The parameters are essential for ensuring transparency in the entire process of fish growth and catch. Providing these details helps stakeholders avoid conflicts in product size throughout the entire supply chain. Fish farmers upload

Algorithm 2 Creating a Purchase Order for Fish Seeds and Receiving the Order

```
1  Input: Fish Farmer EA, Fish Seed Company EA, and
   NumberOfFishSeedsRequested
2  Output: Create FishSeedsPurchaseOrderPlaced,
   Create FishSeedsPurchaseOrderConfirmed,
   Create FishSeedsOrderReceived Events
3  if FunctionCaller ≠ FishFarmer then
4      This is an incorrect address for Fish Farmer.
5  end
6  else
7      if Function Caller EA = Fish Farmer EA then
8          Use Keccak256 to create
             FishSeedsPurchaseOrderID.
9          Change the status of the Fish Seeds purchase
             order to "Pending".
10         Trigger an event to inform stakeholders of the
             creation of the Fish Seed purchase order using
             the FishSeedsPurchaseOrderID,
             FishSeedsPurchaserEA,
             FishSeedsPurchaseOrderReceiverEA, and
             NumberOfFishSeedsRequested.
11     end
12     If
13     Function Caller EA = Fish Seed Company EA
       Accept the Fish Seeds Purchase Order request and
       change the status to Accepted
14     Else
15     Change the status of the purchase order to
       Rejected
16 end
17 if Function Caller EA= Fish Farmer EA then
18     FishSeedsOrderStatus==Accepted; "The Order is
         not received yet."
19     FishSeedsOrderStatus==Received; "The Order has
       been received."
20 end
21 else
22     Bring the contract back to its original state by
       highlighting an error message.
23 end
```

images and videos of the fish growth life cycle to IPFS, and the IPFS hash value is put on the blockchain for other stakeholders to examine.

Algorithm 4 highlights the process of creating a purchase order and receiving the shipment of harvested farm-fish. In this algorithm, the fish processor places the purchase order by calling the relevant function, and the fish farmer receives the purchase order. The fish processor provides the following data as inputs: the Ethereum address of the fish farmer; the Ethereum address of the fish processor; the number of fish ordered, and the species name. Following that, the hash function Keccak256 and abi.encodepacked command are used to

Algorithm 3 Updating Farmed Fish Growth Details

```
1  Input: Fish Farmer EA, WeightOfEachFish,
   TotalNumberOfFish, IPFS hash value
2  Output: Generate FarmedFishGrowthDetailsUpdated
   Event
3  if FunctionCaller ≠ FishFarmer then
4      The address of Fish Farmer is incorrect.
5  end
6  else
7      if Function Caller EA = Fish Farmer EA then
8          Notify the stakeholders by triggering an event
             about the farmed fish growth details:
             Ethereumaddressoffishfarmer,
             weightofeachfish, Totalnumberoffish,
             speciesname, and IPFSHashvalue.
9      end
10     else
11         Bring the contract back to its original state by
             highlighting an error message.
12     end
13 end
```

create the unique farmed fish purchase order ID [38]. The system then generates the farmed fish purchase order ID and notifies the stakeholders regarding the purchase order. Later, the fish farmer accepts the order as the responding entity to this request. Finally, the fish farmer ships farmed fish to the fish processor, who receives the shipment.

As explained in the previous sections, wild-caught fish harvesting is another method of fish production. In Algorithm 5, we demonstrate the process of capturing and updating information regarding wild-caught fish. In this algorithm, the wild-caught fisher catches fish from natural ecosystems such as oceans, lakes, and ponds, and inputs the critical information to the blockchain system. The input details are the Ethereum address of the wild-caught fisher, species name, geographic origin, type of ecosystem, number of wild-caught fish, water-type, and date of catch. When these details are updated, the people who need to know about wild-caught fish find out and act accordingly.

The fish processor places a purchase order to buy the wild-caught fish at the wild-caught fisher's discretion, and the wild-caught fisher accepts the purchase order. Upon confirmation, the wild-caught fisher initiates shipment of the wild-caught fish. Algorithm 6 explains the complete process of ordering and receiving the wild-caught fish shipment. The receiving status of the shipment gets recorded in the blockchain system when the fish processor accepts the shipment and all the stakeholders get notified about the operation.

Algorithm 7 explains updating the processed fish details and creating a processed fish package ID. The fish processor processes both wild-caught and farmed fish. During processing, the fish processor sends input data to the blockchain such as processed species name, unprocessed fish ID, IPFS

Algorithm 4 Creating Purchase Order for Farmed Fish and Receiving the Order

```
1  Input:Fish Processor EA, Fish Farmer EA, Number of
   Fish ordered, and Species name
2  Output:
   Create$FarmedFishPurchaseOrderPlaced$,
   Create$FarmedFishPurchaseOrderConfirmed$,
   Create$FarmedFishOrderReceived$ Events
3  if $FunctionCaller \neq FishProcessor$ then
4      The Ethereum address of Fish Processor is not valid
       and farmed fish purchase order request is rejected.
5  end
6  else
7      if $Function\ Caller\ EA = Fish\ Processor\ EA$ then
8          Create FarmedFishPurchaseOrderID using
           Keccak 256 function containing the details:
           Number of fish ordered, species name.
9          Change the status of the Farmed fish Purchase
             order to "Pending".
10         Send a notification to the stakeholders regarding
             the farmed fish purchase order with the Farmed
             Fish Purchase Order ID.
11     end
12     If
13     Function Caller EA = Fish Farmer EA
14     Accept the Farmed Fish Purchase Order request and
       change the status to *Accepted*
15     Else
16     Change the status of the purchase order to *Rejected*
17 end
18 if $Function\ Caller\ EA = Fish\ Processor\ EA$ then
19     FarmedFishOrderStatus=Accepted; "The Order is
       not received yet."
20     FarmedFishOrderStatus=Received; "The Order has
       been received."
21 end
22 else
23     Bring the contract back to its original state by
         highlighting an error message.
24 end
```

Algorithm 5 Updating Wild-Caught Fish Details

```
1  Input: Wild-Caught Fisher EA, Species name,
   Geographic Origin, Type of Ecosystem, Number of
   Wild-Caught Fish, Water Type, and Date of Catch
2  Output: Create $WildCaughtFishDetailsUpdated$ Event
3  if $FunctionCaller \neq Wild - CaughtFisher$ then
4      Updating of Wild-Caught Fish details update is
       rejected.
5  end
6  else
7      if $FunctionCallerEA = Wild - CaughtFisher$ then
8          Update Wild-Caught fish details.
9          Notify the stakeholders regarding the
             wild-caught fish details using an event.
10     end
11 end
12 else
13     Bring the contract back to its original state by
       highlighting an error message.
14 end
```

hash, Date of processing, Ethereum addresses of registration smart contracts, wild-caught fish smart contracts, and farmed fish smart contracts. The input parameter "Processed Species Name" depicts the species type which gets processed. All the stakeholders must know what fish species they deal with. The parameter "Unprocessed fish ID" is the product ID that gets created and attached by both the wild-caught fisher and the fish farmer to the fish after catching. This detail is necessary to establish the link between unprocessed fish and processed fish fillets. The IPFS hash is another data input that the stakeholders use to access the files and PCR reports of the fish products. Processed fish PCR report ID is another parameter stored on the blockchain to access the large-sized reports and files uploaded to IPFS. The date of processing is another input data that can help the stakeholders identify the exact date of processing. Our approach established the link between the fish processing smart contract and the registration, wild-caught, and farmed fish smart contracts. This will ensure complete tracking of all the operations that happen in different smart contracts, and the stakeholders can view and access necessary information. After updating the processed fish details, the fish processor calls the required functions to create the processed fish package ID with the inputs using the Keccak256 function [38].

Upon successful information update on the processed fish details, the distributor gets notified about the availability of processed fish. The distributor sends a purchase order to the fish processor, which gets created using Keccak 256 and abi.encodePacked functions [38]. The output of this command creates a unique ID referred to as the ProcessedFishPurchaseOrderID. The input details for creating the ProcessedFishPurchaseOrderID are as follows: The Ethereum address of the fish processor, the Ethereum address of the distributor, and the number of processed fish lots ordered. This function notifies the ProcessedFishPurchaseOrderID to all the stakeholders, and upon receiving the purchase order, the fish processor accepts it. Algorithm 8 demonstrates the creation of ProcessedFishShipmentID using Keccak 256 and abi.encodePacked functions [38], which contains the PackagedFishPurchaseOrderID as an input parameter. After successfully creating the ProcessedFishShipmentID, the fish processor ships the processed fish to the distributor, and the distributor accepts the shipment.

The retailer buys fish products from the distributor after receiving the processed and packaged fish shipment. Algorithm 9 explains the logic in which the Retailer calls

Algorithm 6 Creating a Purchase Order for Wild-Caught Fish and Receiving the Order

```
1  Input: Wild-Caught Fisher EA, Fish Processor EA, and
   Number Of Wild-Caught Fish Ordered
2  Output:
   Create WildCaughtFishPurchaseOrderPlaced,
   Create WildCaughtFishPurchaseOrderConfirmed,
   Create WildCaughtFishOrderReceived Events
3  if FunctionCaller ≠ FishProcessor then
4      The address of Fish Processor is incorrect.
5  end
6  else
7      if Function Caller EA = Fish Processor EA then
8          Create WildCaughtFishPurchaseOrderID using
           Keccak256 function.
9          Update Wild-Caught Fish Purchase Order Status
             to Pending.
10         Notify the stakeholders by triggering an event
             regarding the creation of Fish Seed purchase
             order using the WildCaughtFishPurchaserEA,
             WildCaughtFishSellerEA, and
             NumberOfWildCaughtFishOrdered.
11     end
12     If
13     Function Caller EA = Wild-Caught Fisher EA
       Accept the Wild-Caught Fish Purchase Order
       request and change the status to Accepted
14     Else Change the status of the purchase order to
       Rejected
15 end
16 if Function Caller EA= Fish Farmer EA then
17     WildCaughtFishOrderStatus=Accepted; "The
       Shipment is not received yet."
18     WildCaughtFishOrderStatus=Received; "The
         Shipment has been received."
19 end
20 else
21     Bring the contract back to its original state by
       highlighting an error message.
22 end
```

Algorithm 7 Updating Fish Processing Details and Creating ProcessedFishPackageID

```
1  Input: Processed Species name, Unprocessed Fish ID,
   IPFS Hash, Processed Fish PCR Report ID, Date Of
   Processing, Fillets in Packets, Catch Method,
   Registration Contract EA, Wild-Caught Fish Smart
   Contract EA or Farmed Fish Smart Contract EA
2  Output: Create ProcessedFishDetailsUpdated, Create
   ProcessedFishPackageIDCreated Event
3  if FunctionCaller ≠ FishProcessor then
4      Updating of Processed Fish details is not authorized.
5  end
6  else
7      if FunctionCallerEA=FishProcessor then
8          Update processed fish details:
           ProcessedSpeciesName, UnprocessedFishID,
           IPFSHash, DateOfProcessing
9      end
10     Initiate an event to notify the stakeholders regarding
         the processed fish details.
11 end
12 else
13     Bring the contract back to its original state by
       highlighting an error message.
14 end
15 if FunctionCallerEA=FishProcessor then
16     Update ProcessedFishPCRreportId, SpeciesName,
       CatchMethod, FilletsInPacket details
17     Create ProcessedFishPackageID using Keccak256
         function.
18     Update Processed Fish Package ID Status to
         Created.
19 end
20 Initiate an event to notify the stakeholders regarding the
   creation of processed fish ID.
21 else
22     Bring the contract back to its original state by
       highlighting an error message.
23 end
```

the function which creates the RetailerPurchaseOrderID upon deployment of the Keccak 256 [38] function containing the inputs such as Ethereum address of the Retailer, Ethereum address of the Distributor, ProcessedFishPackageID, and NumberOfFishPackageOrdered. The encoded padded byte string creates RetailerPurchaseOrderID as a unique identifier to track the purchase order requested by the retailer. Later, the distributor accepts the purchase order and ships the processed fish packages to the retailer. The retailer accepts the shipment and notifies all the stakeholders about the shipment status.

Upon successful availability of the fish products at the retailer, the consumer buys the product. In our approach, the consumer is an external entity which is not part of our private Ethereum blockchain, but rather there is an off-chain communication between the retailer and consumer. The consumer requests the retailer to share the history of events, and the retailer obtains the records, which are stored in the ledger. The retailer uses a Dapp, which interacts with the smart contracts using application programming interfaces (API) to obtain the information stored in the distributed ledger.

To validate the compliance of standards followed by the stakeholders, the FDA initiates a fine against the responsible stakeholder. In this study, we allocate the fish processor as the fine bearer in case of any violations. Algorithm 10 explains the logic behind the initiation of fine status based on the number of violations. In the event that the number of violations becomes more than the threshold number of violations, then

Algorithm 8 Creating Purchase Order for Processed Fish and Receiving the Order

```
1  Input:Fish Processor EA, Distributor EA,
     ProcessedFishPackageID, and
     Quantityoffishpackageordered
2  Output:
     Create ProcessedFishPuchaseOrderPlaced,
     ProcessedFishPurchaseOrderConfirmed,
     ProcessedFishOrderReceived Events
3  if FunctionCaller≠ Distributor then
4      Creation of ProcessedFishPurchaseOrderID is
       rejected.
5  end
6  else
7      if FunctionCallerEA=Distributor then
8          Create ProcessedFishPurchaseOrderID using
           Keccak 256 function containing the details:
           ProcessedFishPackageId, Ethereum address of
           Fish Processor, Ethereum address of Distributor,
           Quantityoffishpackageordered
9      end
10     Send out a notification to stakeholders with details
       of the Processed Fish Purchase Order.
11 end
12 If
13 Function Caller EA = Fish Processor EA Accept the
     Processed Fish Purchase Order request and change the
     status to Accepted
14 Else
15 Change the status of the purchase order to Rejected
16 if Function Caller EA= Distributor EA then
17     ProcessedFishOrderStatus=Accepted; “The
       Shipment is not received yet.”
18     ProcessedFishOrderStatus=Received; “The
       Shipment has been received.”
19 end
20 else
21     Bring the contract back to its original state by
       highlighting an error message.
22 end
```

Algorithm 9 Creating Retailer Purchase Order and Receiving the Shipment

```
1  Input:Retailer EA, Distributor EA,
     ProcessedFishPurchaseOrderID, and
     NumberOfFishPackagesOrdered
2  Output: Create RetailerPuchaseOrderPlaced,
     Create RetailerPurchaseOrderConfirmed,
     Create RetailerOrderReceived Event
3  if FunctionCaller≠Retailer then
4      Creation of RetailerPurchaseOrderID is rejected.
5  end
6  else
7      if FunctionCallerEA=Retailer then
8          Create RetailerPurchaseOrderID using Keccak
           256 function containing the details:
           ProcessedFishPackageID, Ethereum address of
           Retailer, NumberOfFishPackageOrdered,
           Ethereum address of Distributor
9      end
10     Send out a notification to stakeholders with details
       of the Retailer Fish Purchase Order.
11 end
12 If
13 Function Caller EA = Distributor EA Accept the Retailer
     Fish Purchase Order request and change the status to
     Accepted
14 Else
15 Change the status of the purchase order to Rejected
16 if Function Caller EA= Retailer EA then
17     RetailerFishOrderStatus=Accepted; “The Shipment
       is not received yet.”
18     RetailerFishOrderStatus=Received; “The Shipment
       has been received.”
19 end
20 else
21     Bring the contract back to its original state by
       highlighting an error message.
22 end
```

the smart contract automatically issues a fine and notifies all stakeholders.

V. SYSTEM TESTING AND VALIDATION

Following the implementation of our proposed blockchain-based approach, this section demonstrates the output results of functions and events. Each smart contract is deployed and executed in the Remix IDE environment. We prevent unauthorized function callers by using modifier functions for restricted functions. Whenever a stakeholder initiates a function call, an error notification is triggered and the code is reverted to its original state. Our smart contract code includes events, which helps maintain data provenance and traceability of information. All actions performed by the respective stakeholders can be easily tracked, and the Ethereum address of each stakeholder can be found in the event output. The proposed blockchain-based framework can improve the traceability of the fish product throughout the most important part of its life cycle. In our solution, the Ethereum addresses for each stakeholder are assigned in Table 1.

A. FISH PRODUCTION AND HARVESTING

Herein, we explain the functions and events of fish production and fish harvesting in farmed fish and wild-caught fishery supply chains.

Figure 7 demonstrates the input data for deploying the farmed fish smart contract and output data of the executed functions of the farmed fish smart contract. The constructor function *SetFishSeedsDescriptions* enables the fish seed company to update the initial details of the fish seed in a

Algorithm 10 Updating Fine Status

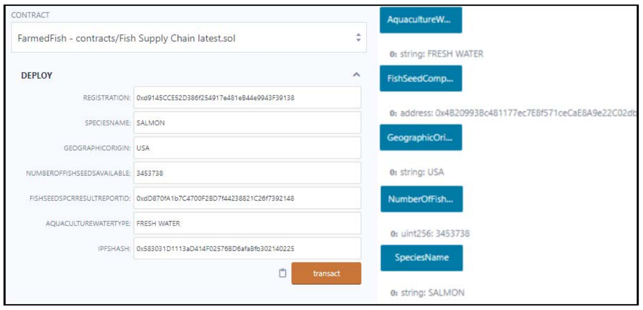
```
1  Input:FDA EA, Fish Processor EA, Number of
   violations, and Acceptable number of violations
2  Output: Create FineStatusCreated Event
3  if FunctionCaller≠FDA then
4  |   Issuing Fine is not authorized.
5  end
6  else
7  |   if FunctionCallerEA=FDAEA then
8  |   |   Check number of violations.
9  |   |   if Numberofviolations≥
   |   |     Acceptablenumberofviolations then
10 |   |   |   Set FineStatus=True.
11 |   |   end
12 |   |   else
13 |   |   |   Set FineStatus to False.
14 |   |   end
15 |   |   Initiate an event to impose fine to Fish Processor
   |   |     and notify the stakeholders.
16 |   end
17 end
18 else
19 |   Bring the contract back to its original state by
   |   highlighting an error message.
20 end
```

TABLE 1. Stakeholder's Ethereum address.

Name of Stakeholder	Ethereum Address
FDA	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
FishSeedCompany	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
FishFarmer	0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
Wild-CaughtFisher	0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7
FishProcessor	0x617F2E2fD72FD9D5503197092aC168c91465E7f2
Distributor	0x17F6AD8Ef982297579C203069C1DbfFE4348c372
Retailer	0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C

farmed fishery supply chain. In addition, the logic encoded in the FarmedFish smart contract verifies the Ethereum address of the fish seed company and validates the legitimate user authorized to input the details. After successfully updating the details, the smart contract notifies the stakeholders by triggering an event named FishSeedsDescriptionsSet. This event also displays the key information: species name, geographic origin, number of fish seeds available, fish seeds PCR result reportID, aquaculture water type, and IPFS Hash.

After the successful deployment of the farmer's fish smart contract, the fish farmer calls the function *PlaceFishSeedsPurchaseOrder*. Figure 8 presents the log of the successful creation of FishSeedsPurchaseOrderID. After the execution of this function, the status of the purchase order changes to Pending. The receiver of the fish seeds purchase order verifies the details and notifies the purchase orderer about the status of the purchase order by calling the function *ConfirmFishSeedsPurchaseOrder*. The fish seed company can only call this function. Figure 9 depicts the function execution result and describes the acceptance or rejection status of the fish


CONTRACT

FarmedFish - contracts/Fish Supply Chain latest.sol

DEPLOY

REGISTRATION: 0xd9145CCE52D386f254917e481eB44e9943F39138

SPECIESNAME: SALMON

GEOGRAPHICORIGIN: USA

NUMBEROFFISHSEEDSAVAILABLE: 3453738

FISHSEEDSPCRRESULTREPORTID: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148

AQUACULTUREWATERTYPE: FRESH WATER

IPFSHASH: 0x583031D1113aD414F02576BD6afaBfb302140225

0: string: FRESH WATER

0: address: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

0: string: USA

0: uint256: 3453738

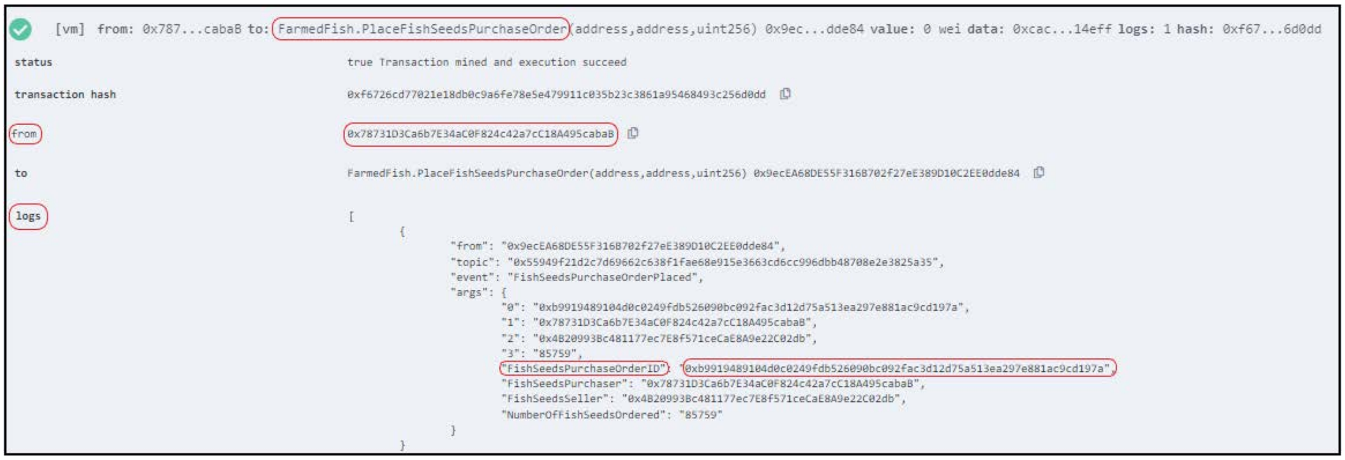
0: string: SALMON

FIGURE 7. Highlighting input and output details of farmed fish smart contract deployment.

seed purchase order. After accepting the fish seed purchase order, the fish seed company initiates the shipment. In order to inform all stakeholders about the receipt of the fish seed order, the fish farmer calls the function *ReceiveFishSeedsOrder* upon receiving a shipment of fish seeds. The log detailing the receipt of the fish seed shipment is shown in Figure 10.

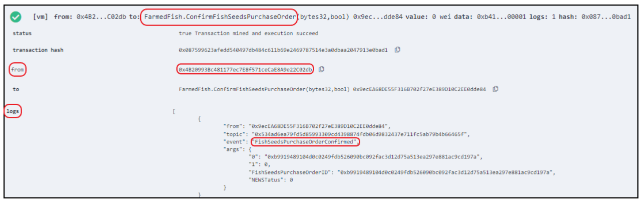
After receiving the fish seeds, the fish farmer grows them and updates the fish growth details by calling the function *UpdateFarmedFishGrowthDetails*. The successful execution of this function triggers an event named FarmedFishGrowthDetailsUpdated, containing the details depicted in Figure 11. After successfully updating the farmed fish growth details, the fish processor calls the function *PlaceFarmedFishPurchaseOrder* function. As a response, the farmed fish smart contract creates a purchase order for buying farmed fish and modifies the order status to Pending. Figure 12 depicts the detailed log of the execution of this function. The change in the status of the purchase order gets notified to all the stakeholders. Following this, the fish farmer calls the *[*ConfirmFarmedFishPurchaseOrder] function to report whether it has confirmed or rejected the order. When the *[*ConfirmFarmedFishPurchaseOrder] function is successfully executed, the fish farmer's response gets added to the blockchain. An event "FarmedFishPurchaseOrderConfirmed" is emitted to notify about the current status of the purchase order. In this event, a record of the updated status of the purchase order of farmed fish is maintained. During this event, the farmed fish purchase order is updated with its latest status. The fish farmer initiates the farmed fish shipment after the farmed fish purchase order is confirmed. Upon successful receiving of the farmed fish shipment, the fish farmer calls the function *ReceiveFarmedFishOrder*, which triggers an event named FarmedFishOrderReceived to notify the stakeholders about the receipt of the shipment. Figure 13 displays the output details of unique identity codes generated after executing a few functions in the farmed fish smart contract.

After the wild-caught fisher catches fish from the natural ecosystems, it calls the constructor function to deploy the wild-caught fish smart contract to update the fish data on the blockchain. Figure 14 demonstrates the input and output information created after the successful wild-caught fish smart contract deployment. The successful deployment



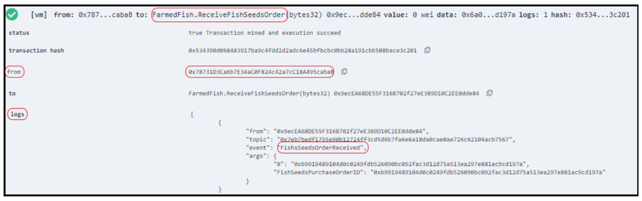
```
[vm] from: 0x787...cabaB to: FarmedFish.PlaceFishSeedsPurchaseOrder(address,address,uint256) 0x9ec...dde84 value: 0 wei data: 0xcac...14eff logs: 1 hash: 0xf67...6d0dd
status                     true Transaction mined and execution succeed
transaction hash           0xf6726cd77021e18db0c9a6fe78e5e479911c035b23c3861a95468493c256d0dd
from                       0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
to                         FarmedFish.PlaceFishSeedsPurchaseOrder(address,address,uint256) 0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84
logs                       [
                               {
                                   "from": "0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84",
                                   "topic": "0x55949f21d2c7d69662c638f1fae68e915e3663cd6cc996dbb48708e2e3825a35",
                                   "event": "FishSeedsPurchaseOrderPlaced",
                                   "args": {
                                       "0": "0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea297e881ac9cd197a",
                                       "1": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                                       "2": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                                       "3": "85759",
                                       "FishSeedsPurchaseOrderID": "0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea297e881ac9cd197a",
                                       "FishSeedsPurchaser": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                                       "FishSeedsSeller": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                                       "NumberOfFishSeedsOrdered": "85759"
                                   }
                               }
```

FIGURE 8. Log presenting the successful creation of FishSeedsPurchaseOrderID.



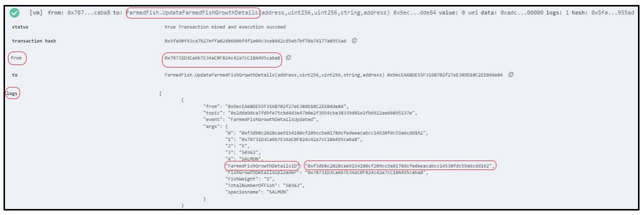
```
[vm] from: 0x4B2...C02db to: FarmedFish.ConfirmFishSeedsPurchaseOrder(bytes32,bool) 0x9ec...dde84 value: 0 wei data: 0xb41...00001 logs: 1 hash: 0x087...0bad1
status                     true Transaction mined and execution succeed
transaction hash           0x087599623afedd540497db484c611b69e2469787514e3a0dbaa2047913e0bad1
from                       0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
to                         FarmedFish.ConfirmFishSeedsPurchaseOrder(bytes32,bool) 0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84
logs                       [
                               {
                                   "from": "0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84",
                                   "topic": "0x534ad6ea79fd5d85993309cd4398874fdb06d9832437e711fc5ab79b4b66465f",
                                   "event": "FishSeedsPurchaseOrderConfirmed",
                                   "args": {
                                       "0": "0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea297e881ac9cd197a",
                                       "1": 0,
                                       "FishSeedsPurchaseOrderID": "0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea297e881ac9cd197a",
                                       "NEWSTatus": 0
                                   }
                               }
```

FIGURE 9. Log presenting the successful confirmation of FishSeedsPurchaseOrderID.



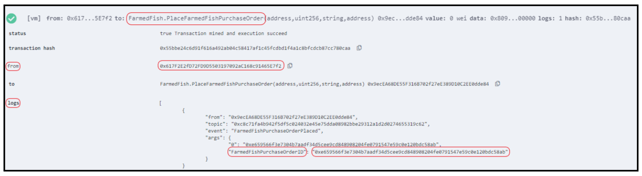
```
[vm] from: 0x787...cabaB to: FarmedFish.ReceiveFishSeedsOrder(bytes32) 0x9ec...dde84 value: 0 wei data: 0x6a0...d197a logs: 1 hash: 0x534...3c201
status                     true Transaction mined and execution succeed
transaction hash           0x534390d068483917ba9c4fdd2d2adc6e45bfbcbc0bb28a191cbb508bace3c201
from                       0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
to                         FarmedFish.ReceiveFishSeedsOrder(bytes32) 0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84
logs                       [
                               {
                                   "from": "0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84",
                                   "topic": "0x7eb7bedf1795e90b12724ff3cd5d6b7fa6e6a18da0cae0ae726c62104acb7567",
                                   "event": "FishsSeedsOrderReceived",
                                   "args": {
                                       "0": "0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea297e881ac9cd197a",
                                       "FishSeedsPurchaseOrderID": "0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea297e881ac9cd197a"
                                   }
                               }
```

FIGURE 10. Log presenting the successful receiving of fish seeds order.



```
[vm] from: 0x787...cabaB to: FarmedFish.UpdateFarmedFishGrowthDetails(address,uint256,uint256,string,address) 0x9ec...dde84 value: 0 wei data: 0xadc...00000 logs: 1 hash: 0x5fa...955ad
status                     true Transaction mined and execution succeed
transaction hash           0x5fa90f43ca7b27effa02d86606f4f1e04c3ce84d2cd5eb7bf70a74177a0955ad
from                       0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
to                         FarmedFish.UpdateFarmedFishGrowthDetails(address,uint256,uint256,string,address) 0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84
logs                       [
                               {
                                   "from": "0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84",
                                   "topic": "0x2dda9dca7fd9fe75cbd4d3e47b0e2f3954cba38339d01e1fb6922ae6b895137e",
                                   "event": "FarmedFishGrowthDetailsUpdated",
                                   "args": {
                                       "0": "0xf3db8c2028cae9154180cf209cc9a8178dcfedeeacabcc14530fdc59a6cdd162",
                                       "1": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                                       "2": "5",
                                       "3": "50363",
                                       "4": "SALMON",
                                       "FarmedFishGrowthDetailsID": "0xf3db8c2028cae9154180cf209cc9a8178dcfedeeacabcc14530fdc59a6cdd162",
                                       "FishGrowthDetailsUploader": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                                       "FishWeight": "5",
                                       "TotalNumberOfFish": "50363",
                                       "speciesname": "SALMON"
                                   }
                               }
```

FIGURE 11. Log presenting the successful update of farmed fish growth details.



```
[vm] from: 0x617...5E7f2 to: FarmedFish.PlaceFarmedFishPurchaseOrder(address,uint256,string,address) 0x9ec...dde84 value: 0 wei data: 0x809...00000 logs: 1 hash: 0x55b...80caa
status                     true Transaction mined and execution succeed
transaction hash           0x55bbe24c6d91f616a492ab04c58417af1c45fcdbd1f4a1c8bfcdcb87cc780caa
from                       0x617F2E2fD72FD9D5503197092aC168c91465E7f2
to                         FarmedFish.PlaceFarmedFishPurchaseOrder(address,uint256,string,address) 0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84
logs                       [
                               {
                                   "from": "0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84",
                                   "topic": "0xc8c71fa4b942f5df5c024032e45e75dda08982bbe29312a1d2d0274655319c62",
                                   "event": "FarmedFishPurchaseOrderPlaced",
                                   "args": {
                                       "0": "0xe659566f3e7304b7aadf34d5cee9cd848908204fe0791547e59c0e120bdc58ab",
                                       "FarmedFishPurchaseOrderID": "0xe659566f3e7304b7aadf34d5cee9cd848908204fe0791547e59c0e120bdc58ab"
                                   }
                               }
```

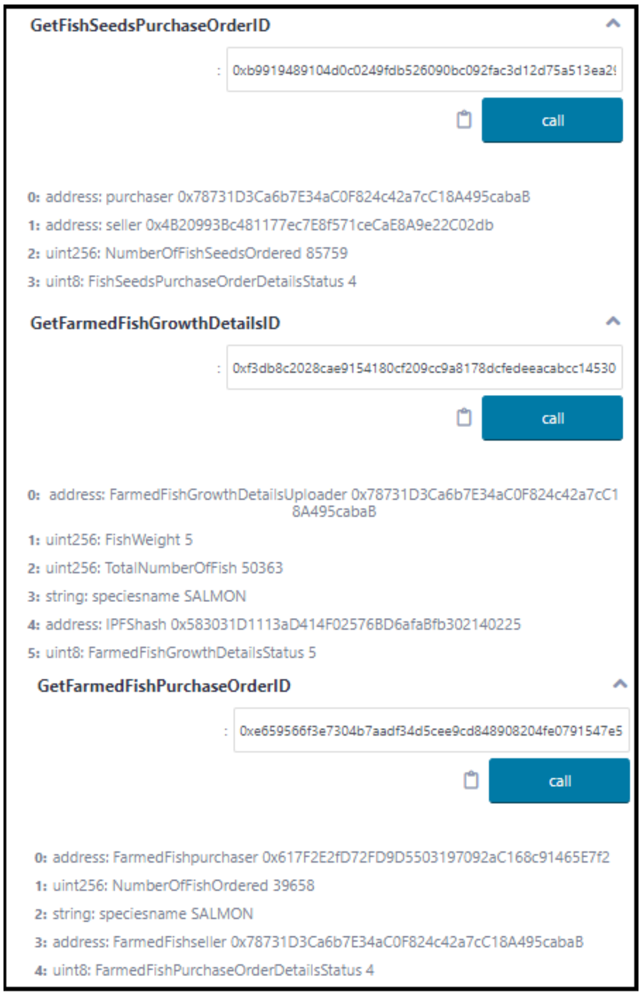
FIGURE 12. Log presenting the successful creation of FarmedFishPurchaseOrderID.

of this smart contract sends notification to all the stakeholders about the status of the availability of wild-caught fish. Later, the fish processor enables the creation of a purchase order request using the function [PlaceWildCaughtFishPurchaseOrder], as shown in Figure 15. Using this function, the smart contract generates a unique ID named ''WildCaughtFishPurchaseOrderID'' and triggers an event called ''WildCaughtFishPurchaseOrderPlaced''. In addition, the status of the purchase order changes to ''Pending.'' After getting notified about the wild-caught fish purchase order, the wild-caught fisher calls the function *ConfirmWildCaughtFishPurchaseOrder* and responds to the purchase order by accepting or rejecting the request. Responding to the purchase order using the function *ConfirmWildCaughtFishPurchaseOrder* triggers an event named ''WildCaughtFishPurchaseOrderConfirmed'' and notifies the stakeholders about the acceptance of the order. Later, the wild-caught fisher initiates shipment of the requested order and ships the order to the fish processor. Upon successful receipt of the shipment, the fish processor calls the function *ReceiveWildCaughtFishOrder*. As a result, the function initiates the following event: ''WildCaughtFishOrderReceived''. Stakeholders are notified by this event when the shipment of wild-caught fish has been delivered. Figure 16 highlights all the wild-caught fish purchase order details stored on the blockchain after successfully receiving the shipment.

B. FISH PROCESSING AND DISTRIBUTION

In this subsection, we present the execution output details of fish processing and distribution processes. The functions and events presented in this subsection are handled by the Fish Processing and Fish Distribution smart contracts.

After successfully receiving the wild-caught fish and farmed fish, the fish processor starts processing the fish by removing the fins and other non-consumable body parts. In this process, the fish processor calls the constructor



GetFishSeedsPurchaseOrderID

: 0xb9919489104d0c0249fdb526090bc092fac3d12d75a513ea2!

call

```
0: address: purchaser 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
1: address: seller 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
2: uint256: NumberOfFishSeedsOrdered 85759
3: uint8: FishSeedsPurchaseOrderDetailsStatus 4
```

GetFarmedFishGrowthDetailsID

: 0xf3db8c2028cae9154180cf209cc9a8178dcfedeeacabcc14530

call

```
0: address: FarmedFishGrowthDetailsUploader 0x78731D3Ca6b7E34aC0F824c42a7cC1
                                8A495cabaB
1: uint256: FishWeight 5
2: uint256: TotalNumberOfFish 50363
3: string: speciesname SALMON
4: address: IPFShash 0x583031D1113aD414F02576BD6afaBfb302140225
5: uint8: FarmedFishGrowthDetailsStatus 5
```

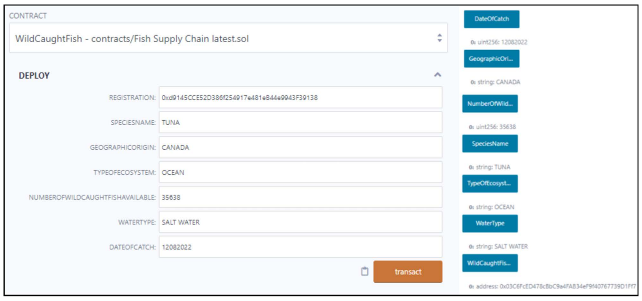
GetFarmedFishPurchaseOrderID

: 0xe659566f3e7304b7aadf34d5cee9cd848908204fe0791547e5

call

```
0: address: FarmedFishpurchaser 0x617F2E2fD72FD9D5503197092aC168c91465E7f2
1: uint256: NumberOfFishOrdered 39658
2: string: speciesname SALMON
3: address: FarmedFishseller 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
4: uint8: FarmedFishPurchaseOrderDetailsStatus 4
```

FIGURE 13. Highlighting the output details of farmed fish SC functions.



CONTRACT

WildCaughtFish - contracts/Fish Supply Chain latest.sol

DEPLOY

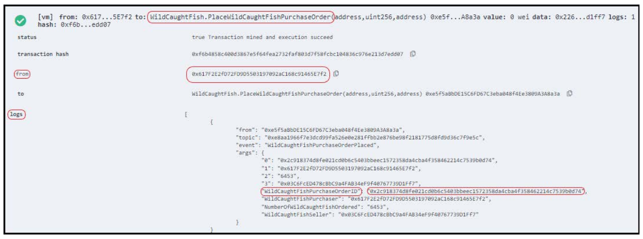
REGISTRATION:	0xd9145CCE52D386f254917e481eB44e9943F39138
SPECIESNAME:	TUNA
GEOGRAPHICORIGIN:	CANADA
TYPEOFECOSYSTEM:	OCEAN
NUMBEROFWILDCAUGHTFISHAVAILABLE:	35638
WATERTYPE:	SALT WATER
DATEOFCATCH:	12082022

transact

```
DateOfCatch
0: uint256: 12082022
GeographicOri...
0: string: CANADA
NumberOfWild...
0: uint256: 35638
SpeciesName
0: string: TUNA
TypeOfEcosyst...
0: string: OCEAN
WaterType
0: string: SALT WATER
WildCaughtFis...
0: address: 0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7
```

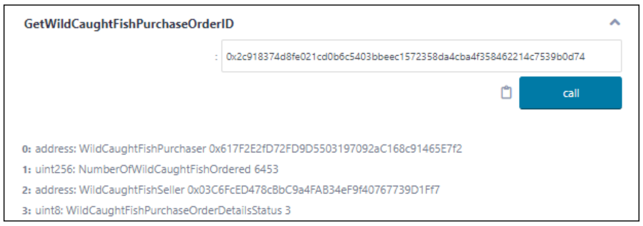
FIGURE 14. Highlighting input and output details of wild-caught fish smart contract deployment.

function to deploy the fish processing smart contract and inputs relevant details such as UnprocessedFishID, ProcessedSpeciesName, IPFSHash, and Date of processing. When these details are entered correctly and the constructor function is called, the Fish processing smart contract is put into action, and an event called ''ProcessedFishDetailsUpdated'' is triggered to let the stakeholders know.



```
[vm] from: 0x617...5E7f2 to: WildCaughtFish.PlaceWildCaughtFishPurchaseOrder(address,uint256,address) 0xe5f...A8a3a value: 0 wei data: 0x226...d1ff7 logs: 1
hash: 0xf6b...edd07
status                 true Transaction mined and execution succeed
transaction hash       0xf6b4858c400d3867e5f64fea2732faf803d7f58fcbc104836c976e213d7edd07
from                   0x617F2E2fD72FD9D5503197092aC168c91465E7f2
to                     WildCaughtFish.PlaceWildCaughtFishPurchaseOrder(address,uint256,address) 0xe5f5aBbDE15C6FD67C3eba048f4Ee3809A3A8a3a
logs                   [
                            {
                                "from": "0xe5f5aBbDE15C6FD67C3eba048f4Ee3809A3A8a3a",
                                "topic": "0xe8aa1966f7e3dcd99fa526e0e281ffbb2e876be98f2181775d8fd9d36c7f9e5c",
                                "event": "WildCaughtFishPurchaseOrderPlaced",
                                "args": {
                                    "0": "0x2c918374d8fe021cd0b6c5403bbeec1572358da4cba4f358462214c7539b0d74",
                                    "1": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                    "2": "6453",
                                    "3": "0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7",
                                    "WildCaughtFishPurchaseOrderID": "0x2c918374d8fe021cd0b6c5403bbeec1572358da4cba4f358462214c7539b0d74",
                                    "WildCaughtFishPurchaser": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                    "NumberOfWildCaughtFishOrdered": "6453",
                                    "WildCaughtFishSeller": "0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7"
                                }
                            }
```

FIGURE 15. Log presenting the successful creation of WildCaughtFishPurchaseOrderID.



GetWildCaughtFishPurchaseOrderID

: 0x2c918374d8fe021cd0b6c5403bbeec1572358da4cba4f358462214c7539b0d74

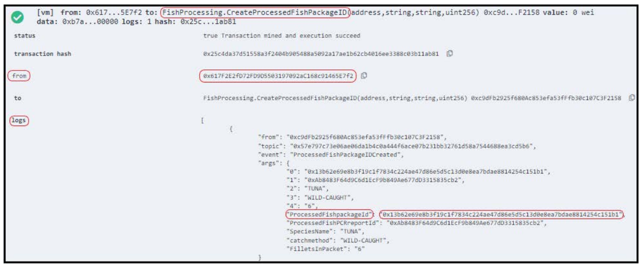
call

```
0: address: WildCaughtFishPurchaser 0x617F2E2fD72FD9D5503197092aC168c91465E7f2
1: uint256: NumberOfWildCaughtFishOrdered 6453
2: address: WildCaughtFishSeller 0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7
3: uint8: WildCaughtFishPurchaseOrderDetailsStatus 3
```

FIGURE 16. Highlighting the output details of wild-caught fish SC functions.

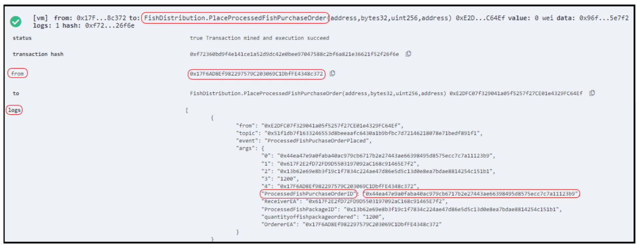
While processing the fish, the fish processor also conducts a PCR test of the processed fish fillets and stores the PCR report on IPFS. It also generates the ProcessedFishPCRReportID code and stores it in the blockchain. In addition to this, the fish processor also ensures the packaging of processed fish fillets. In the packaging process, the fish processor assigns a unique identifier code: ''ProcessedFishPackageID'' that gets created when the fish processor calls the function *CreateProcessedFishPackageID*. Figure 17 depicts the creation of ProcessedFishPackageID, and Figure 18 depicts the output data of the functions execution of the fish processing smart contract. The ProcessedFishPackageID code gets generated when the fish processor inputs the following details: Processed Fish PCR report Id, Species Name, Catch Method, and Number of fillets in the packet. Using these details, the smart contract creates ProcessedFishPackageID and prints it on the processed fish packet in the form of either a QR code or bar code. The ProcessedFishPackageID can be used to track the processed fish until it is consumed.

To send a purchase order request for buying processed fish packages, the distributor calls the function *PlaceProcessedFishPurchaseOrder*. Figure 19 demonstrates the log and details of the function's execution. In response, the smart contract triggers an event called ''ProcessedFishPurchaseOrderPlaced'' and notifies the stakeholders. After getting notified about the purchase order, the fish processor accepts or rejects the purchase order by calling the function *ConfirmProcessedFishPurchaseOrder*. As a result, the smart contract enables an event named ''ProcessedFishPurchaseOrderConfirmed'' that notifies all stakeholders about accepting or rejecting the purchase order. Later, the fish processor initiates shipment of processed fish packages, and upon successful receiving of the processed fish package order, the distributor calls the function *ReceiveProcessedFishOrder*. With this function, the



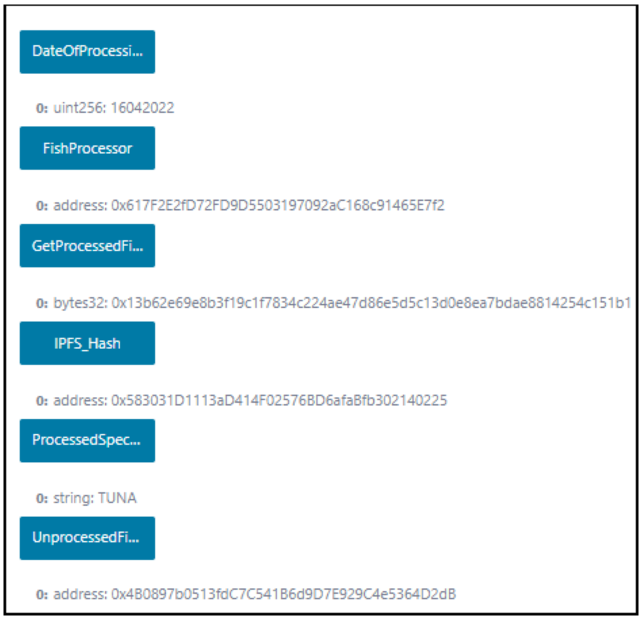
```
[vm] from: 0x617...5E7f2 to: FishProcessing.CreateProcessedFishPackageID(address,string,string,uint256) 0xc9d...F2158 value: 0 wei
data: 0xb7a...00000 logs: 1 hash: 0x25c...1ab81
status               true Transaction mined and execution succeed
transaction hash     0x25c4da37d51558a3f2404b905488a5092a17ae1b62cb4016ee3388c03b11ab81
from                 0x617F2E2fD72FD9D5503197092aC168c91465E7f2
to                   FishProcessing.CreateProcessedFishPackageID(address,string,string,uint256) 0xc9dFb2925f680Ac853efa53fFfb30c107C3F2158
logs                 [
                         {
                             "from": "0xc9dFb2925f680Ac853efa53fFfb30c107C3F2158",
                             "topic": "0x57e797c73e06ae06da1b4c0a444f6ace07b231bb32761d58a7544688ea3cd5b6",
                             "event": "ProcessedFishPackageIDCreated",
                             "args": {
                                 "0": "0x13b62e69e8b3f19c1f7834c224ae47d86e5d5c13d0e8ea7bdae8814254c151b1",
                                 "1": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                 "2": "TUNA",
                                 "3": "WILD-CAUGHT",
                                 "4": "6",
                                 "ProcessedFishpackageId": "0x13b62e69e8b3f19c1f7834c224ae47d86e5d5c13d0e8ea7bdae8814254c151b1",
                                 "ProcessedFishPCRreportId": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                 "SpeciesName": "TUNA",
                                 "catchmethod": "WILD-CAUGHT",
                                 "FilletsInPacket": "6"
                             }
```

FIGURE 17. Log presenting the creation of ProcessedFishPackageID.



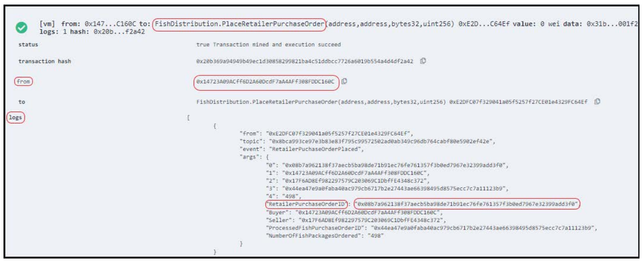
```
[vm] from: 0x17F...8c372 to: FishDistribution.PlaceProcessedFishPurchaseOrder(address,bytes32,uint256,address) 0xE2D...C64Ef value: 0 wei data: 0x96f...5e7f2
logs: 1 hash: 0xf72...26f6e
status               true Transaction mined and execution succeed
transaction hash     0xf72360bd9f4e141ce1a52d9dc42e0bee97047588c2bf6a821e36621f52f26f6e
from                 0x17F6AD8Ef982297579C203069C1DbfFE4348c372
to                   FishDistribution.PlaceProcessedFishPurchaseOrder(address,bytes32,uint256,address) 0xE2DFC07f329041a05f5257f27CE01e4329FC64Ef
logs                 [
                         {
                             "from": "0xE2DFC07f329041a05f5257f27CE01e4329FC64Ef",
                             "topic": "0x51f1db7f1633246553d8beeaafc6430a1b9bfbc7d72146218078e71bedf891f1",
                             "event": "ProcessedFishPuchaseOrderPlaced",
                             "args": {
                                 "0": "0x44ea47e9a0faba40ac979cb6717b2e27443ae66398495d8575ecc7c7a11123b9",
                                 "1": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                 "2": "0x13b62e69e8b3f19c1f7834c224ae47d86e5d5c13d0e8ea7bdae8814254c151b1",
                                 "3": "1200",
                                 "4": "0x17F6AD8Ef982297579C203069C1DbfFE4348c372",
                                 "ProcessedFishPurchaseOrderID": "0x44ea47e9a0faba40ac979cb6717b2e27443ae66398495d8575ecc7c7a11123b9"
                                 "ReceiverEA": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                 "ProcessedFishPackageID": "0x13b62e69e8b3f19c1f7834c224ae47d86e5d5c13d0e8ea7bdae8814254c151b1",
                                 "quantityoffishpackageordered": "1200",
                                 "OrdererEA": "0x17F6AD8Ef982297579C203069C1DbfFE4348c372"
                             }
                         }
```

FIGURE 19. Log presenting the creation of ProcessedFishPurchaseOrderID.



```
DateOfProcessi...
0: uint256: 16042022
FishProcessor
0: address: 0x617F2E2fD72FD9D5503197092aC168c91465E7f2
GetProcessedFi...
0: bytes32: 0x13b62e69e8b3f19c1f7834c224ae47d86e5d5c13d0e8ea7bdae8814254c151b1
IPFS_Hash
0: address: 0x583031D1113aD414F02576BD6afaBfb302140225
ProcessedSpec...
0: string: TUNA
UnprocessedFi...
0: address: 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB
```

FIGURE 18. Highlighting details of Fish Processing data.



```
[vm] from: 0x147...C160C to: FishDistribution.PlaceRetailerPurchaseOrder(address,address,bytes32,uint256) 0xE2D...C64Ef value: 0 wei data: 0x31b...001f2
logs: 1 hash: 0x20b...f2a42
status               true Transaction mined and execution succeed
transaction hash     0x20b369a94949b49ec1d30858299821ba4c51ddbcc7726a6019b554a4d4df2a42
from                 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C
to                   FishDistribution.PlaceRetailerPurchaseOrder(address,address,bytes32,uint256) 0xE2DFC07f329041a05f5257f27CE01e4329FC64Ef
logs                 [
                         {
                             "from": "0xE2DFC07f329041a05f5257f27CE01e4329FC64Ef",
                             "topic": "0x8bca993ce97e3b83e83f795c99572502ad0ab349c96db764cabf80e5902ef42e",
                             "event": "RetailerPuchaseOrderPlaced",
                             "args": {
                                 "0": "0x08b7a962138f37aecb5ba98de71b91ec76fe761357f3b0ed7967e32399add3f0",
                                 "1": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
                                 "2": "0x17F6AD8Ef982297579C203069C1DbfFE4348c372",
                                 "3": "0x44ea47e9a0faba40ac979cb6717b2e27443ae66398495d8575ecc7c7a11123b9",
                                 "4": "498",
                                 "RetailerPurchaseOrderID": "0x08b7a962138f37aecb5ba98de71b91ec76fe761357f3b0ed7967e32399add3f0"
                                 "Buyer": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
                                 "Seller": "0x17F6AD8Ef982297579C203069C1DbfFE4348c372",
                                 "ProcessedFishPurchaseOrderID": "0x44ea47e9a0faba40ac979cb6717b2e27443ae66398495d8575ecc7c7a11123b9",
                                 "NumberOfFishPackagesOrdered": "498"
                             }
                         }
```

FIGURE 20. Log presenting the creation of RetailerPurchaseOrderID.

smart contract creates an event called ''ProcessedFishOrderReceived'' to let people know that processed fish orders have been received.

After the distributor receives the processed fish order, the retailer calls the function *PlaceRetailerPurchaseOrder* to purchase the processed fish from the distributor. An example of the log details of successful execution of the function *PlaceRetailerPurchaseOrder* is shown in figure 20. Upon executing this function, RetailerPurchaseOrderID is created as well as the event ''RetailerPurchaseOrderPlaced''. All stakeholders are notified of the status of the purchase order through this event. The smart contract also changes the status of the RetailerPurchaseOrder to Pending. Later, the distributor calls the function *ConfirmRetailerPurchaseOrder* to accept or reject the purchase order request. As a result, the smart contract changes the status of the purchase order to either accepted or rejected. After updating the status, the distributor ships the fish products to the retailer. After receiving the retail order, the retailer calls *ReceiveRetailerOrder* function to receive it. As a result of this function being executed, the smart contract triggers an event ''RetailerOrderReceived'' that is notified to stakeholders. The output results of the execution of the functions are displayed in figure 21.

After the successful receipt of the packages of fish products by the retailer, the consumer buys the final product. Later, the FDA verifies all the necessary details and calls the function *CreateFineStatus*. As a result, the smart contract notifies the stakeholders about the status of Fine by sending an event named ''FineStatusCreated''. In our study, we selected the fish processor as the penalty bearer for any non-compliance with processes, as the fish processor is the responsible entity for processing fish obtained from legitimate and registered stakeholders. Figure 22 depicts the execution result of the function *CreateFineStatus*.

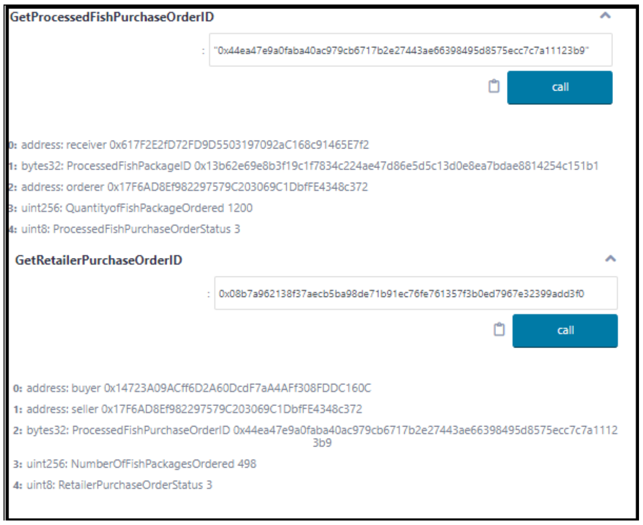
VI. DISCUSSION

In this section, we present the security analysis of our solution. Also, we compare our solution with the existing solutions and discuss the generalization aspect.

A. SECURITY ANALYSIS

Our solution offers better security, robustness, and strength. We outline below the fundamental security aspects and the details of our proposal.

- **Data Integrity and Tamper-proof**
  Data exploitation is a significant risk to data integrity. The blockchain protects the integrity of data using cryptography. Since blockchain transactions are immutable, data integrity is preserved. Our approach requires stakeholders to digitally sign transactions before updating them on the blockchain. Our proposed solution is data-integrity-preserving due to the immutability of transactions. Since transactions are immutable, there can be no modifications or deletions to our proposed solution.



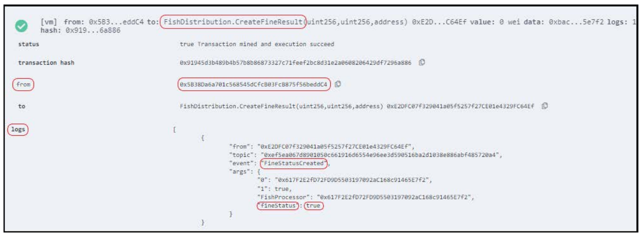
```
GetProcessedFishPurchaseOrderID
: "0x44ea47e9a0faba40ac979cb6717b2e27443ae66398495d8575ecc7c7a11123b9"
call

0: address: receiver 0x617F2E2fD72FD9D5503197092aC168c91465E7f2
1: bytes32: ProcessedFishPackageID 0x13b62e69e8b3f19c1f7834c224ae47d86e5d5c13d0e8ea7bdae8814254c151b1
2: address: orderer 0x17F6AD8Ef982297579C203069C1DbfFE4348c372
3: uint256: QuantityofFishPackageOrdered 1200
4: uint8: ProcessedFishPurchaseOrderStatus 3

GetRetailerPurchaseOrderID
: 0x08b7a962138f37aecb5ba98de71b91ec76fe761357f3b0ed7967e32399add3f0
call

0: address: buyer 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C
1: address: seller 0x17F6AD8Ef982297579C203069C1DbfFE4348c372
2: bytes32: ProcessedFishPurchaseOrderID 0x44ea47e9a0faba40ac979cb6717b2e27443ae66398495d8575ecc7c7a1112
                                         3b9
3: uint256: NumberOfFishPackagesOrdered 498
4: uint8: RetailerPurchaseOrderStatus 3
```

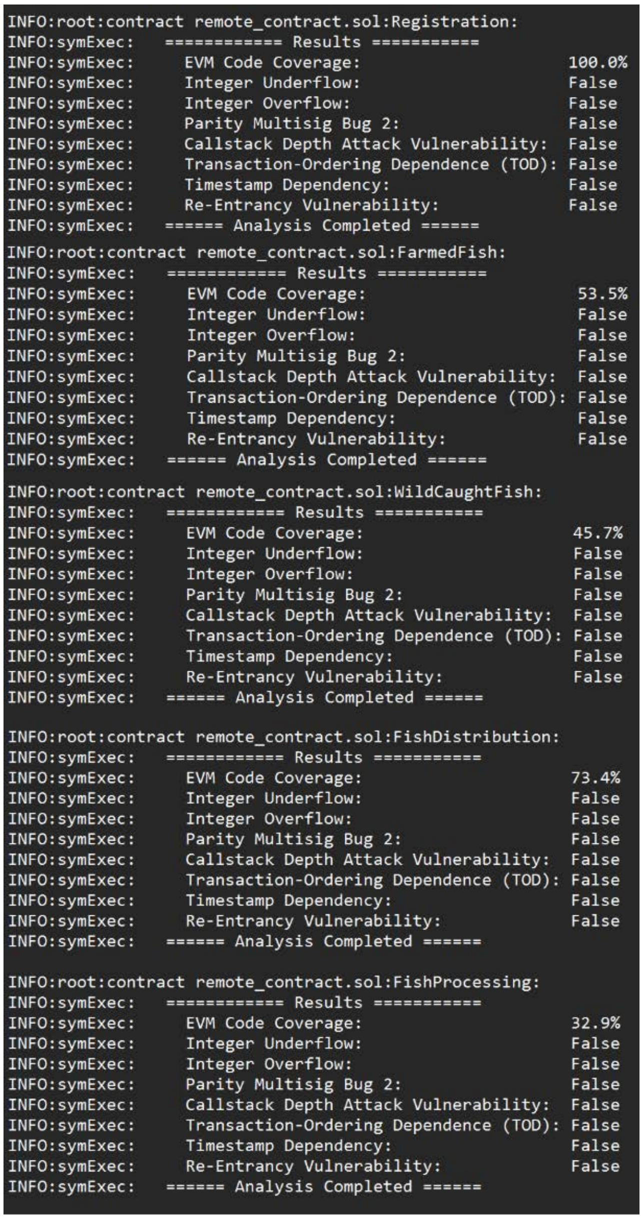
FIGURE 21. Highlighting details of fish distribution output data.



```
[vm] from: 0x5B3...eddC4 to: FishDistribution.CreateFineResult(uint256,uint256,address) 0xE2D...C64Ef value: 0 wei data: 0xbac...5e7f2 logs: 1
hash: 0x919...6a886
status              true Transaction mined and execution succeed
transaction hash    0x91945d3b489b4b57b8b86873327c71feef2bc8d31e2a0608206429df7296a886
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  FishDistribution.CreateFineResult(uint256,uint256,address) 0xE2DFC07f329041a05f5257f27CE01e4329FC64Ef
logs                [
                        {
                            "from": "0xE2DFC07f329041a05f5257f27CE01e4329FC64Ef",
                            "topic": "0xef5ea067d8901050c661916d6554e96ee3d590516ba2d1038e886abf485720a4",
                            "event": "FineStatusCreated",
                            "args": {
                                "0": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                "1": true,
                                "FishProcessor": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                                "fineStatus": true
                            }
                        }
```

FIGURE 22. Highlighting details of fine status in case of non-compliance of standards.

- **Availability**
  The decentralization feature of blockchain technology has made it popular. In a decentralized system, all stakeholders have access to the same information. Thus, there is almost no risk of DoS attacks. Our system is built on the Ethereum blockchain, which gives all stakeholders safe access to how it works.
- **Accountability** New technologies such as blockchain require accountability so that they can be trusted and used in a secure and reliable manner. With blockchain technology, each participant's actions can be tracked and traced with non-repudiation. The non-repudiation agreement confirms the legitimacy of all transactions initiated by stakeholders. Any transaction initiated by a stakeholder cannot be denied in our blockchain approach.
- **Vulnerability analysis of smart contracts**
  The blockchain system can be affected by smart contracts that have vulnerabilities and bugs. Thus, we have performed a security analysis of the smart contract codes using the SmartCheck and Oyente tools to ensure they do not contain any vulnerabilities. We have no significant security vulnerabilities identified in our smart contract codes after analysis by the SmartCheck tool. Similarly, we used Oyente to examine our smart contract codes. As an example, the tool can report integer underflows and overflows, stack depth attacks, Transaction-Ordering Dependency (TOD), timestamp dependency,



```
INFO:root:contract remote_contract.sol:Registration:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                          100.0%
INFO:symExec:     Integer Underflow:                          False
INFO:symExec:     Integer Overflow:                           False
INFO:symExec:     Parity Multisig Bug 2:                      False
INFO:symExec:     Callstack Depth Attack Vulnerability:       False
INFO:symExec:     Transaction-Ordering Dependence (TOD):      False
INFO:symExec:     Timestamp Dependency:                       False
INFO:symExec:     Re-Entrancy Vulnerability:                  False
INFO:symExec:   ====== Analysis Completed ======
INFO:root:contract remote_contract.sol:FarmedFish:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                          53.5%
INFO:symExec:     Integer Underflow:                          False
INFO:symExec:     Integer Overflow:                           False
INFO:symExec:     Parity Multisig Bug 2:                      False
INFO:symExec:     Callstack Depth Attack Vulnerability:       False
INFO:symExec:     Transaction-Ordering Dependence (TOD):      False
INFO:symExec:     Timestamp Dependency:                       False
INFO:symExec:     Re-Entrancy Vulnerability:                  False
INFO:symExec:   ====== Analysis Completed ======

INFO:root:contract remote_contract.sol:WildCaughtFish:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                          45.7%
INFO:symExec:     Integer Underflow:                          False
INFO:symExec:     Integer Overflow:                           False
INFO:symExec:     Parity Multisig Bug 2:                      False
INFO:symExec:     Callstack Depth Attack Vulnerability:       False
INFO:symExec:     Transaction-Ordering Dependence (TOD):      False
INFO:symExec:     Timestamp Dependency:                       False
INFO:symExec:     Re-Entrancy Vulnerability:                  False
INFO:symExec:   ====== Analysis Completed ======

INFO:root:contract remote_contract.sol:FishDistribution:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                          73.4%
INFO:symExec:     Integer Underflow:                          False
INFO:symExec:     Integer Overflow:                           False
INFO:symExec:     Parity Multisig Bug 2:                      False
INFO:symExec:     Callstack Depth Attack Vulnerability:       False
INFO:symExec:     Transaction-Ordering Dependence (TOD):      False
INFO:symExec:     Timestamp Dependency:                       False
INFO:symExec:     Re-Entrancy Vulnerability:                  False
INFO:symExec:   ====== Analysis Completed ======

INFO:root:contract remote_contract.sol:FishProcessing:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                          32.9%
INFO:symExec:     Integer Underflow:                          False
INFO:symExec:     Integer Overflow:                           False
INFO:symExec:     Parity Multisig Bug 2:                      False
INFO:symExec:     Callstack Depth Attack Vulnerability:       False
INFO:symExec:     Transaction-Ordering Dependence (TOD):      False
INFO:symExec:     Timestamp Dependency:                       False
INFO:symExec:     Re-Entrancy Vulnerability:                  False
INFO:symExec:   ====== Analysis Completed ======
```

FIGURE 23. Vulnerability analysis of smart contracts.

TABLE 2. Comparative analysis of existing non-blockchain-based solutions.

Features	[6]	[23]	[24]	[26]	[27]	Our Solution
Prototype	Yes	No	Yes	No	Yes	Yes
Transparency	No	No	No	No	No	Yes
Decentralized	No	No	No	No	No	Yes
Traceability	Yes	No	Yes	Yes	Yes	Yes
Integrated	No	No	No	No	No	Yes

and re-entry attacks. Figure 23 summarizes the result for our system's smart contracts generated by Oyente. The Oyente tool identified all the vulnerabilities as "false". This implies that our system's smart contract is quite secure against the vulnerabilities.

TABLE 3. Comparison between our and the existing non-blockchain-based solutions.

Features	[10]	[11]	[12]	[28]	[34]	Our Solution
Prototype	Yes	No	Yes	Yes	Yes	Yes
Platform	Public	Public	Public	Public	Public	Private
Decentralized	Yes	Yes	Yes	Yes	Yes	Yes
Traceability	Yes	Yes	Yes	Yes	Yes	Yes
Integrated	No	No	No	No	Yes	Yes
Security Analysis	No	No	No	No	No	Yes

B. COMPARISON WITH THE EXISTING APPROACHES

In Table 2 and Table 3, we compare our proposed solution with the existing non-blockchain and blockchain-based solutions. In Table 2, none of the presented solutions provide an integrated traceability system in wild-caught and farmed fishery supply chains. Similarly, none of the approaches is decentralized and provides end-to-end transparency. In addition, these approaches are centralized and vulnerable to the single point of failure problem. However, the methods establish efficient product traceability except [23].

On the other hand, the proposed blockchain-based approaches provide solutions for tracking and tracing fish products efficiently. Except [11], all approaches present prototype solutions using a decentralized public blockchain platform and efficiently establish product traceability in various fishery supply chains. However, all the approaches except [34] predominantly address the challenges of one type of fishery supply chain rather than focusing on both wild-caught fish and farmed fishery supply chains. Moreover, none of the solutions present the security analysis of their blockchain-based system architecture and smart contracts.

In contrast to the studies cited above, our solution is an integrated one, which integrates farmed fish and wild-caught fishery supply chains with a feasible prototype solution. In addition, the study proposes a blockchain-enabled approach to improve fishery supply chain operations. Using Remix IDE, we also implemented and tested our proposed framework and conducted the necessary analysis to confirm our solution is viable.

C. GENERALIZATION

We design, test, and validate our proposed system on the private Ethereum platform to achieve traceability, transparency, and security requirements in the fishery supply chain. Our proposed approach offers a secure and efficient way to record business transactions using blockchain platforms, as these platforms can encrypt data efficiently. The smart contracts proposed can be used by any industry that deals with product traceability.

The proposed system could also be implemented by other industries that deal with product traceability, such as pharmaceuticals, automobiles, logistical industries, etc. The proposed system is capable of effectively tracking and tracing all components and activities in the proposed fishery supply chain. With this approach, it is possible to track stakeholders and their functions in the fishery supply chain, thus making the system applicable to other industries. As an example, automobile manufacturers can use this method if there is a recall requiring tracing and tracking of their vehicles, spare parts, customers, dealers, and suppliers.

We implemented all five smart contracts using Solidity. Our solution is broad and covers a lot of different parts of the supply chain for both farmed fish and fish caught in the wild. With our solution, we enable the existing fish industry product traceability management system to be implemented on different blockchain platforms, thereby enhancing its existing features. Hence, our solution can be adopted as a measure to improve the current practices in the fish industry.

VII. CONCLUSION

In this paper, we have proposed a blockchain-based solution to trace and track fish products in the fishery supply chain in a manner that is transparent, accountable, secure, private, and trustworthy. Our proposed solution prevents different types of fish fraud and malpractice by ensuring transparent interactions among all stakeholders. We developed five smart contracts to automate operations and track events in the fishery supply chain. We presented the system architecture, sequence diagrams, and ten algorithms along with their implementation, testing, and validation details. We present the security analysis to show that our solution is secure enough against security threats and attacks. We compared our solution with the existing blockchain and non-blockchain-based solutions to show its novelty and effectiveness. Our proposed solution is generic enough to be applied to various supply chain processes. In the future, we plan to deploy and test our solution on the real Ethereum network and build DApps for various stakeholders.

REFERENCES

[1] *The State of World Fisheries and Aquaculture 2020*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.fao.org/state-of-fisheries-aquaculture/tr/

[2] S. B. Islam and M. M. Habib, "Supply chain management in fishing industry: A case study," *Int. J. Supply Chain Manage.*, vol. 2, no. 2, pp. 1–11, Jun. 2013.

[3] M. Fox, M. Mitchell, M. Dean, C. Elliott, and K. Campbell, "The seafood supply chain from a fraudulent perspective," *Food Secur.*, vol. 10, no. 4, pp. 939–963, Aug. 2018.

[4] U. Schröder, "Challenges in the traceability of seafood," *J. für Verbraucherschutz und Lebensmittelsicherheit*, vol. 3, no. 1, pp. 45–48, Feb. 2008.

[5] K. Gopi, D. Mazumder, J. Sammut, and N. Saintilan, "Determining the provenance and authenticity of seafood: A review of current methodologies," *Trends Food Sci. Technol.*, vol. 91, pp. 294–304, Sep. 2019.

[6] M. Trebar, A. Grah, A. A. Melcon, and A. Parreno, "Towards RFID traceability systems of farmed fishery supply chain," in *Proc. 19th Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, 2011, pp. 1–6.

[7] (Feb. 16, 2022). *Why Seafood Companies Diving Into Seafood Traceability*. Blockchain for Food Safety. Traceability and Supplychain Transparency. Accessed: Mar. 7, 2022. [Online]. Available: https://tracextech.com/seafoodcompanies-embracing-seafood-traceability/::text=Food20traceability20allows20suppliers20and, and20will20not20be20rejected

[8] N. Tsolakis, D. Niedenzu, M. Simonetto, M. Dora, and M. Kumar, "Supply network design to address united nations sustainable development goals: A case study of blockchain implementation in Thai fish industry," *J. Bus. Res.*, vol. 131, pp. 495–519, Jul. 2021.

[9] E. Mahan. Jun. 26, 2020. Seafood is off the Chain! How do we Integrate Blockchain Technology for Seafood Traceability?. eScholarship. University of California, Accessed: Mar. 7, 2022. [Online]. Available: https://escholarship.org/uc/item/4385m32n

[10] WWF International. *Blockchain: Transforming Seafood Supply Chain Traceability*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.wwf.org.nz/?159612FBlockchain-Transforming-Seafood-Supply-Chain-Traceability

[11] *Blockchain Application in Seafood Value Chains*. Blockchain Application in Seafood Value Chains | E-Agriculture. Accessed: Mar. 7, 2022. [Online]. Available: https://www.fao.org/e-agriculture/blog/blockchain-application-seafood-value-chains-0

[12] Fishcoinblockchain. Apr. 4, 2018. Fishcoin Blockchain Traceability. Fishcoin Blockchain Traceability. Accessed: Mar. 7, 2022. [Online]. Available: https://fishcoinblockchaintraceability.wordpress.com/

[13] *Processing Scantrust Secure Code Workorders—ScanTrust*. Accessed: Mar. 7, 2022. [Online]. Available: https://help.scantrust.com/hc/en-us/articles/115003770354-Processing-Scantrust-Secure-Code-Workorders

[14] L. Hang, I. Ullah, and D.-H. Kim, "A secure fish farm platform based on blockchain for agriculture data integrity," *Comput. Electron. Agricult.*, vol. 170, Mar. 2020, Art. no. 105251.

[15] S. Aich, S. Chakraborty, M. Sain, H.-I. Lee, and H.-C. Kim, "A review on benefits of IoT integrated blockchain based supply chain management implementations across different sectors with case study," in *Proc. 21st Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2019, pp. 1–4.

[16] A. Hassoun. Apr. 30, 2020. Food Fraud: Do you Really Know What Fish Species you are Eating?. Sciencenorway. Accessed: Mar. 7, 2022. [Online]. Available: https://sciencenorway.no/blog-blog-taste-of-the-sea-fish/food-fraud-do-you-really-know-what-fish-species-you-are-eating/1676883

[17] Global Seafood Alliance. Feb. 2, 2022. *What Happens at a Seafood Processing Plant?*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.globalseafood.org/blog/seafood-processing-plant/

[18] Manufacturing.net. Dec. 17, 2015. *Seafood Processing Involves Unique Challenges*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.manufacturing.net/home/article/13149645/seafood-processing-involves-unique-challenges

[19] L. Wan. Feb. 13, 2018. Fish Food Fraud Rife in China With More Than Half of Fillets Mislabelled: Study. Foodnavigator. Accessed: Mar. 7, 2022. [Online]. Available: https://www.foodnavigator-asia.com/Article/2018/02/13/Fish-food-fraud-rife-in-China-with-more-than-half-of-fillets-mislabelled-Study

[20] K. Shuaib, H. Saleous, N. Zaki, and F. Dankar, "A layered blockchain framework for healthcare and genomics," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Sep. 2020, pp. 156–163, doi: 10.1109/SMARTCOMP50058.2020.00040.

[21] FAO. *Document Cardnbsp;: FAO: Food and Agriculture Organization of the United Nations*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.fao.org/documents/card/en/c/I8791EN/

[22] *Taking the First Steps Towards Seafood Traceability—Fishwise*. Accessed: Mar. 7, 2022. [Online]. Available: https://fishwise.org/wp-content/uploads/2018/03/OSMI-Trace-CollabTaking-the-First-Steps-Towards-Seafood-Traceability.pdf

[23] WWF. Apr. 21, 2015. *Traceability Principles for Wild-Caught Fish Products*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.worldwildlife.org/publications/traceability-principles-for-wild-caught-fish-products

[24] Y.-C. Hsu, A.-P. Chen, and C.-H. Wang, "A RFID-enabled traceability system for the supply chain of live fish," in *Proc. IEEE Int. Conf. Autom. Logistics*, Sep. 2008, pp. 81–86, doi: 10.1109/ICAL.2008.4636124.

[25] M. Trebar, M. Lotrič, I. Fonda, A. Pleteršek, and K. Kovačič, "RFID data loggers in fish supply chain traceability," *Int. J. Antennas Propag.*, vol. 2013, pp. 1–9, Aug. 2013.

[26] *Seafood Supply Chain Management: Methods to Prevent*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.researchgate.net/publication/255580534SeafoodSupplyChainManagementMethodstoPreventIllegally-CaughtProductEntryintotheMarketplace

[27] G. L. Hold, V. J. Russell, S. E. Pryde, H. Rehbein, J. Quinteiro, M. Rey-Mendez, C. G. Sotelo, R. I. Pérez-Martin, A. T. Santos, and C. Rosa, "Validation of a PCR-RFLP based method for the identification of salmon species in food products," *Eur. Food Res. Technol.*, vol. 212, no. 3, pp. 385–389, Feb. 2001.

[28] N. N. Ahamed, P. Karthikeyan, S. P. Anandaraj, and R. Vignesh, "Sea food supply chain management using blockchain," in *Proc. 6th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Mar. 2020, pp. 473–476.

[29] A. Rejeb, "Blockchain potential in tilapia supply chain in Ghana," *Acta Technica Jaurinensis*, vol. 11, no. 2, pp. 104–118, Jul. 2018.

[30] S. Larissa and J. Parung, "Designing supply chain models with blockchain technology in the fishing industry in Indonesia," in *Proc. IOP Conf. Mater. Sci. Eng.*, vol. 1072, no. 1, 2021, Art. no. 012020.

[31] WWF. Apr. 21, 2015. *Traceability Principles for Wild-Caught Fish Products*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.worldwildlife.org/publications/traceability-principles-for-wild-caught-fish-products

[32] *Questions and Answers Regarding Food Facility Registration*. Accessed: Mar. 7, 2022. [Online]. Available: https://www.fda.gov/files/food/published/Questions-and-Answers-Regarding-Food-Facility-Registration-28Seventh-Edition29.pdf

[33] E. Cruz and A. M. R. Cruz, "A food value chain integrated business process and domain models for product traceability and quality monitoring: Pattern models for food traceability platforms," in *Proc. 21st Int. Conf. Enterprise Inf. Syst.*, 2019, pp. 285–294.

[34] E. Cruz and A. R. da Cruz, "Using blockchain to implement traceability on fishery value chain," in *Proc. 15th Int. Conf. Softw. Technol.*, 2020, pp. 501–508.

[35] A. E. C. Mondragon, C. E. C. Mondragon, and E. S. Coronado, "Feasibility of Internet of Things and agnostic blockchain technology solutions: A case in the fisheries supply chain," in *Proc. IEEE 7th Int. Conf. Ind. Eng. Appl. (ICIEA)*, Apr. 2020, pp. 504–508, doi: 10.1109/ICIEA49774.2020.9102080.

[36] (Mar. 2022). *SmartCheck*. Accessed: Mar. 07, 2022. [Online]. Available: https://tool.smartdec.net/

[37] FAO. *Document Cardnbsp: FAO: Food and Agriculture Organization of the United Nations*. Accessed: Mar. 8, 2022. [Online]. Available: https://www.fao.org/documents/card/en/c/I8791EN/::text=Abstract3A, food20market20is20largely20unknown

[38] *Hashing With KECCAK256: Solidity by Example: 0.8.10*. Accessed: Mar. 13, 2022. [Online]. Available: https://solidity-by-example.org/hashing/

[39] K. Toyod, P. T. Mathiopoulo, I. Sasase, and T. Ohtsuk, "A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain," *IEEE Access*, vol. 5, pp. 17465–17477, 2017, doi: 10.1109/access.2017.2720760.