IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 October 2022, accepted 23 November 2022, date of publication 28 November 2022,
date of current version 1 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3225000

RESEARCH ARTICLE

Using NFTs and Blockchain for Traceability and Auctioning of Shipping Containers and Cargo in Maritime Industry

FERUZ K. ELMAY[1], KHALED SALAH[1], (Senior Member, IEEE), RAJA JAYARAMAN[2], AND ILHAAM A. OMAR[2]

[1]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
[2]Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, United Arab Emirates

Corresponding author: Raja Jayaraman (raja.jayaraman@ku.ac.ae)

This work was supported by the Khalifa University of Science and Technology under Award CIRA-2019-001.

ABSTRACT The maritime industry is a complex multi-tiered sector that plays a major role in international shipping and trade. Seaborne transportation is responsible for 90% of goods moved globally, and containerized shipments account for more than 50% of those goods. This volume of movement requires efficient tracking and traceability methods that secure the documentation process and ensure the involved stakeholders. However, due to its complexity, the maritime industry suffers from a lack of trust, lack of ownership evidence, protracted documentation procedures, and excessive data aggregation. These shortcomings are reflected in delays and elevated costs in the shipping process. To mitigate these problems, we propose in this paper a blockchain-based solution for the ownership history and traceability of shipping containers in the industry. Our solution utilizes smart contracts and non-fungible tokens (NFTs) on the Ethereum blockchain to accelerate the inefficient shipping process by digitizing ownership transfer and process documentation. Furthermore, the proposed system allows the owners to auction off their cargo in the destination ports, freeing warehouses and containers in the process. We analyze our proposed system in terms of transaction costs and smart contract security. The smart contracts are developed using the Solidity language in the Remix IDE and the code is made publicly available on GitHub.

INDEX TERMS Blockchain, Ethereum, NFTs, supply chain, maritime shipping, auctioning, shipping containers.

I. INTRODUCTION

The shipping industry, accounting for approximately 90% of international trade, plays a major part in the growth of the global economy [1]. The logistics of shipping is a paperwork-intensive multi-tier procedure composed of numerous intricate processes that deal with the transportation and warehousing of goods in the supply chain. A typical export consignment involves approximately up to 100 stakeholders, four contracts, and 37 documents [2]. These numbers are expected to increase with transshipment, port policy differences, and negotiations between the stakeholders. Referring to the data analysis conducted by the authors [3], the industry faces several shortcomings, such as a lack of trust between the stakeholders, lack of ownership proof, lengthy documentation processes, extreme data aggregation, and industry-wide competition, which are ultimately reflected as shipping process drawbacks. Transparency of processes and traceability are hence ongoing requirements in the sector.

Transparency in logistics is the visibility of the goods movement-related processes [4]. This term is often used interchangeably with traceability. On the other hand, traceability refers to accessing the recorded information related to a product in the supply chain [5]. While transparency deals with the downward flow of information, traceability deals with the upward flow [4]. In the supply chain, shipments from heavily regulated industries such as healthcare, military, agricultural, and food sectors require transparency and traceability in their

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

FIGURE 1. A typical process flow a shipment goes through in international maritime shipping.

shipping processes. These shipments are more prone to theft, spoilage, and fraud because of the prolonged shipping procedures and the extended time cargo spends in transit. Transparent, immutable, and secure records of the transactions help both manufacturers and customers by mitigating the aforementioned risks.

The blockchain, a transparent and immutable ledger, offers a huge advantage to the supply chain in satisfying the transparency requirements. It is a distributed ledger that cryptographically links blocks of data and enables peer-to-peer data sharing. Since its introduction with Bitcoin [6] in 2009, blockchain has been gaining attention in many areas, such as AI and healthcare [7], as it disrupts technologies and industries. By allowing the participants to transparently record and broadcast transactions to the network, the technology enables its users to build trust and minimize the number of intermediaries required, resulting in a decentralized and secure network.

The complexity of the shipping process also extends to the transfer of ownership, which is represented by a bill of lading (BoL). The BoL is paperwork that serves multiple purposes, such as a document of title, receipt of freight, and proof of a contract of carriage [8]. Despite its significance, the document's processing time adds to the shipping process's delay, which ranges from 5 to 13 days excluding resales, letters of credit, and bank procedures [8]. Furthermore, the BoL is vulnerable to misplacement and fraud, causing impediments in the release of cargo or resulting in successful fraudulent claimers. For more details on the current procedures and shortcomings involving BoL in the shipping process, we would like to direct the reader to [8].

Unclaimed goods in ports, a frequent problem in the shipping process, also contributes to the delay in ports. Such instances are typically caused by exorbitant port fees, out-of-date perishable commodities, or a forgotten or misplaced shipment of cargo. According to the Federation of Freight Forwarders Association (FIATA), before any cargo is declared unclaimed or abandoned, the consignee is given a time window of 20 to 90 days depending on the type of goods as well as the laws enforced by the port of discharge [9]. However, these cargoes return with accumulated demurrage and detention charges directed at the consignee, the freight forwarder, or the shipping line depending on the shipping contract of unclaimed goods. Shipping lines usually auction unclaimed cargo to cover the port and customs due charges, as briefly shown in Figure 1. The auction is further delayed in situations where cargo is forgotten or held up because of paperwork or customs procedures, which raises the cost of port and warehouse management.

In conclusion, the pain points of the shipping process that we address and attempt to solve in this paper are: the transfer of ownership, the flexibility of cargo auction, lengthy documentation, transparency of freight movement, and trust among the stakeholders. We thus propose a blockchain-based approach for cargo management in maritime transportation to mitigate these shortcomings in the shipping process. By representing the cargoes as unique digital assets, we enable the consignor and consignee the ease of ownership transfer and cargo traceability, freedom of earlier auctioning schedules, and a secure shipping process through a transparent and traceable record of the shipping process. In brief, the major contributions of the paper are summarized as follows:

- We propose an Ethereum blockchain-based solution using smart contracts for container management in the shipping industry, enabling transparent real-time tracking of the freight in the shipping process.
- We provide a thorough discussion on how the use of NFTs can be harnessed to replace the lengthy and paper-intensive process of ownership transfer between the consignor and consignee while also allowing flexible cargo auction schedules.
- We present our proposed solution with system architecture, process flow, entity relationships, and stakeholder engagement and communication.
- We evaluate our system using test cases, analyze the execution and performance of the smart contracts, and provide security and cost analysis to highlight the feasibility of the proposed system.

The rest of the paper is structured as follows: Section II briefly discusses the technologies utilized in realizing the proposed solution. Section III summarizes the literature-reported related works. Sections IV and V explain the proposed

solution and the implementation details, respectively. Section VI details the test cases and validation processes. Finally, we conclude and provide our final remarks in Sections VII and VIII.

II. BACKGROUND

This section highlights the key technologies that are employed in the development of the proposed system.

A. ETHEREUM BLOCKCHAIN AND SMART CONTRACTS

The bitcoin blockchain is limited to the financial sector since it is not customizable. The Ethereum blockchain was introduced in 2013 to be able to integrate blockchain into other sectors [10]. The Ethereum blockchain, among others, enables users to interact with the blockchain via smart contracts. Smart contracts are executable programs that reside on the blockchain, interact with the ledger, and enable the development of decentralized applications (DApps). These programs are developed using Solidity, a JavaScript-like programming language, and are invoked when a certain condition is satisfied or through another smart contract. Smart contracts keep evolving and are currently represented by any of the three standards in the Ethereum ecosystem: ERC20, ERC721, and ERC1155 [11]. ERC20 represents the standard for fungible tokens, ERC721 for non-fungible tokens(NFTs) [12], and ERC1155 for semi-fungible tokens, where one smart contract represents multiple NFTs [13].

A fungible token is a divisible and exchangeable token that does not have any unique property. For instance, a Bitcoin token is equivalent and can be exchanged with another Bitcoin token, or divided into multiple Satoshis. Non-fungible tokens, on the other hand, are indivisible and distinct digital assets represented in the blockchain by unique identities [14]. In the blockchain, NFTs represent a physical asset, an art piece, digital material, or other media and serve as a distinct proof of ownership and authenticity [15]. Moreover, NFTs have transformed the world of art, not only in the way we value art but also in the way we view asset ownership [16]. Prior to minting NFTs, the related information, referred to as metadata, is stored in a database or usually decentralized storage systems in a JSON file format.

B. DECENTRALIZED STORAGE SYSTEMS

Due to the storage limitations of the blockchain ledger, decentralized storage systems have been introduced to securely store files off-chain. Some examples of such systems are the InterPlanetary File System (IPFS), Filcoin, and Storj, to name a few. In this paper, we use IPFS, a content-based storage system that distributes chunks of the stored data to peer-to-peer nodes for persistent storage. It generates a unique content identifier(CID) that acts as a fingerprint of the stored file. In the NFT ecosystem, these persistent and secure storage systems are used to store the metadata of the token.

III. RELATED WORK

In this section, we discuss the use of blockchain and non-fungible tokens(NFTs) for traceability and ownership data provenance in the supply chain in general and shipping containers in particular.

A. SUPPLY CHAIN TRACEABILITY WITH NFTs AND BLOCKCHAIN

Several state-of-the-art papers address the use of blockchain technology to record and broadcast the process stages onto the blockchain for all the involved stakeholders to observe. Hirata et al. [17] introduce PiChain, a descriptive conceptual framework that utilizes blockchain and PI technology for smart ports. They outline in their paper that blockchain is used for information and financial flow where communications and payments occur through smart contracts and function events, while the scheduling of an optimized route for the container is taken care of through AI algorithms. Hasan et al. [18], present a blockchain-based approach for IoT-equipped smart containers to effectively and safely deliver goods in the pharmaceutical supply chain. They discuss how blockchain and smart contracts strengthen supply chain management and enhance communication between the sender and the recipient. Smart contracts and IoT devices are used to monitor shipping conditions and trigger and record actions.

In order to support connected ships in the maritime industry, Komathy [19] discusses a distributed blockchain shipping system. The framework integrates and connects all the business processes in the context of cargo shipping, including finance, banks, supply chain, and insurance. The suppliers, shipping companies, banks, insurers, and other associated parties are represented as nodes and are able to transparently view and share all the transaction data and significantly decrease information sharing and handling delays. Toyoda et al. [20] describe an ownership solution that prevents counterfeiting in the post-supply chain by utilizing the idea of ''proof of possession of balance''. They emphasize the use of bitcoin balance as ownership verification after the delivery of goods. The ownership record is propagated through the blockchain in terms of balance using immutable and secure transaction records on the blockchain ledger, with the manufacturer acting as the first owner. By tracing back the records, buyers can authenticate products as genuine or counterfeit on the basis of the RFID tags.

Expanding the integration of blockchain, researchers also use NFTs for ownership and information traceability in the supply chain. An exemplary implementation of a semi-fungible framework is presented by Khun et al. [13]. The authors underline the significance of traceability systems in recognizing self-contained goods as well as multiple instances of these goods without overwhelming the scalability of the system. In the system, the assembly structures are represented using the ERC1155 smart contract standard, which allows both fungible and non-fungible token representation in a consortium Ethereum blockchain. This expands the system to represent structures as fungible tokens while their instances are represented with non-fungible tokens of unique addresses. Finally, for better data visualization and ease of interaction, a decentralized application referred to as

TokenTrail is implemented to represent the function events as manufacturing steps in the multi-echelon assembly structures of differing scales and orders.

Meyer et al. [21] explain a concept that integrates blockchain technology with the physical internet. They highlight the barriers preventing the complete implementation of the physical internet in the current logistics industry and propose the advantages of blockchain in overcoming these hurdles. The framework is constructed based on the equal participation authority of the nodes and is a resilient and robust network with a rewards system that incentivizes companies to join the network. In their implementation, they represent every container in the network as a token with a smart contract based on the ERC721 standard. The creation, approval, and transfer of tokens are monitored via a smart contract. Finally, they concluded that blockchain provides network trust and decentralized administration.

Westerkamp et al. [11] propose a management solution for models and recipes with traceability in the manufacturing processes. A smart contract and a non-fungible token of the ERC721 standard are used to represent any good in the process. A specific good may consist of a single item or batches of items and have measurement data associated with the token, such as the size, weight, count, or items. For the system to accommodate both single products and products with multiple batches, the size of the token is set to be flexible. To successfully represent the manufacturing process, the authors propose the deployment of a new token as a combination of multiple input tokens.

In the pharmaceutical supply chain, Chiacchio et al. [22] describe an NFT-based solution that enables stakeholders to track and trace drugs from the production stage through delivery. The implementation extends a serialization process to which drug manufacturing companies adhere in order to be able to sell products on the market. This process, which is fully described in the 2011/62/EU directive, requires every prescription drug manufactured to possess a unique identifier referred to as a GTIN (global trade item number), which serves as the NFT identifier in the proposed framework. The solution also addresses the hierarchical packaging of products by grouping the NFTs of the individual products. The implementation is carried out on the VeChain Thor blockchain test environment with a proof of authority (PoA) consensus algorithm.

Lim et al. [23] attempt to map business processes to the use of blockchain by using smart contracts. They leverage blockchain technology in the supply chain by tokenizing the BoL and invoices using ERC721 smart contracts. This makes the transfer of ownership between the participating parties possible while also penalizing the responsible party when and if the shipping process does not go as planned. They discuss the security challenges and cross-chain compatibility challenges and how to overcome them by suggesting alternatives. Arcenegui et al. [24] utilize blockchain technology and smart contracts to securely manage IoT devices and transparently store their data and operations. Each IoT device is represented by an NFT and, therefore, is enabled with a blockchain address to sign transactions in the network while the token ownership is managed by the application manager. The blockchain address is secured by storing the seed in a physical unclonable function (PUF) placed in the physical device. Various fields of study in supply chain have also seen research with NFTs such as car-sharing [25], producing third-party certification [26], and event reselling [27].

B. AUCTION AND BIDDING WITH NFTs AND BLOCKCHAIN

The use of blockchain smart contracts also expands beyond traceability into auctions and bidding. Mezquita et al. [28] propose an NFT-based bidding framework integrated with multi-agent systems in the food supply chain. By using ERC721 tokens to represent agricultural products, they enable buyers to bid and promote efficient communication among the participants without the necessity of an intermediary. The framework also utilizes blockchain traceability and RFID to track shipments. Similarly, Wang et al. [29] introduce a platform referred to as ''ArtChain'' for art ownership, provenance, and trading. The platform includes a user front-end, a trading layer, and ArtChain, which is based on smart contracts to enable users to register, bid, order, and view the ownership history of art assets in the art market.

Another auction and bidding framework are discussed by Guerar et al. [30]. The authors suggest an invoice financing solution for small and medium enterprises (SME) using smart contracts and a decentralized storage system, IPFS. To prevent double financing fraud, the invoice is stored in IPFS and its hash is broadcast into the blockchain, thereby enabling the investors to view the status, place a bid, and make an investment. Martins et al. [31] also introduce a decentralized marketplace for business-to-business and business-to-customer transactions where customers suggest services through auction and suppliers fulfill these services through bidding. The orders are placed individually or aggregated, inviting suppliers to submit bids. The winning supplier is then assigned a tracking ID, and the service quality is evaluated using a reputation contract.

IV. PROPOSED BLOCKCHAIN-BASED SYSTEM

In this section, we discuss the details of the proposed blockchain-based framework for shipment traceability and ownership transfer records. We use blockchain technology's properties to provide a secure, trusted, and orderly shipping process flow. This section details the proposed system's architecture, entity relationship diagram, and the order of interactions between the stakeholders and the smart contracts.

A. SYSTEM ARCHITECTURE

The proposed solution consists of the Ethereum blockchain, smart contracts, and a decentralized storage system, as illustrated in Figure 2. We deploy smart contracts that enable stakeholders to communicate with the blockchain ledger transparently. Using the ERC721 standard of smart contracts, we enable the participants to create NFTs representing

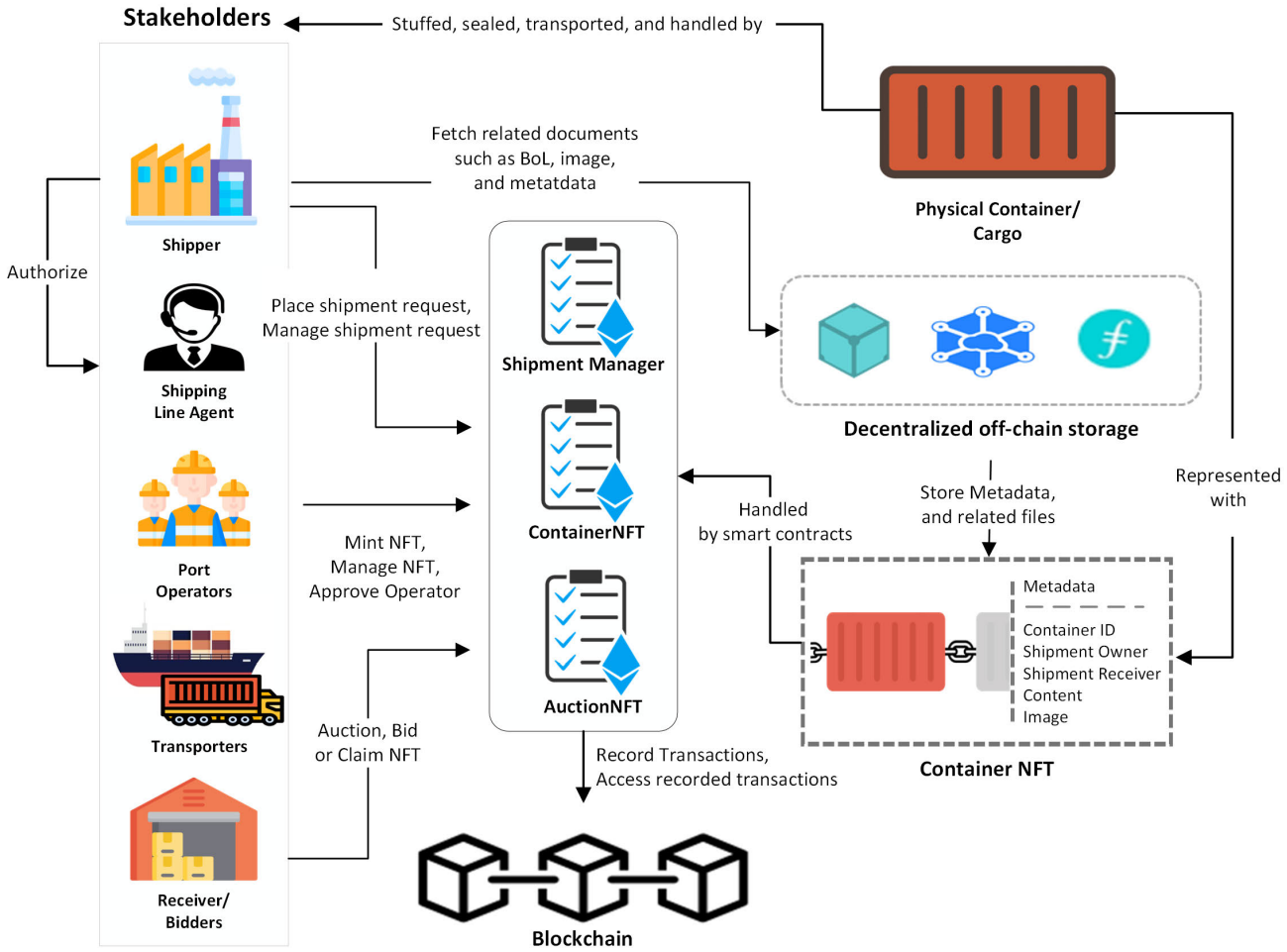

FIGURE 2. Proposed system architecture for NFT for shipping containers traceability and auction.

the physical shipments. The participants are able to track the real-time movement of the shipment by tracking the ownership record of its respective NFT, identified by a unique number. The stakeholders and participants in the proposed system are described as follows.

- **Shipper or Consignor**: The party who wishes to export items. Shippers can range from companies exporting full container load (FCL) to individuals with small packages shipped in less than container load (LCL).
- **Shipping Line Agent**: An agent representing a shipping line, or a container shipping company such as Maersk or Evergreen Marine Corporation. The aforementioned companies offer door-to-door delivery of cargo arranged and monitored by a shipping agent, while other companies require the shippers to transport their cargo to the origin port. In the latter cases, freight forwarders arrange and transport the shipments. In this paper, we assume the company offers door-to-door pickup.
- **Transporters**: Parties that facilitate the movement of the cargo from the shipper to the receiver. These include trucks, trains, cargo planes, and container vessels. In a typical shipping process, the first and last miles of the cargo transit are on container trucks.
- **Receiver or Consignee**: Entity located at the shipping process's destination. Upon reaching the destination port, this party either claims or abandons the cargo, leading to its auction or disposal.
- **Bidders**: Actors who are allowed to bid on auctioned cargo in the port.

The smart contracts are used to place shipping requests, manage shipments, transfer NFT ownership, store and fetch data from the database, auction and bid on abandoned containers, and record transactions on the blockchain. The smart contracts deployed in the proposed system are composed of the following elements.

- **Variables**: A data item that stores a value and is updated after a function execution or in response to logic flow.
- **Events**: Messages broadcast in the blockchain network informing participants of functions called, transactions completed, and states modified.
- **Modifiers**: Components used to restrict the access and execution of functions through access control.

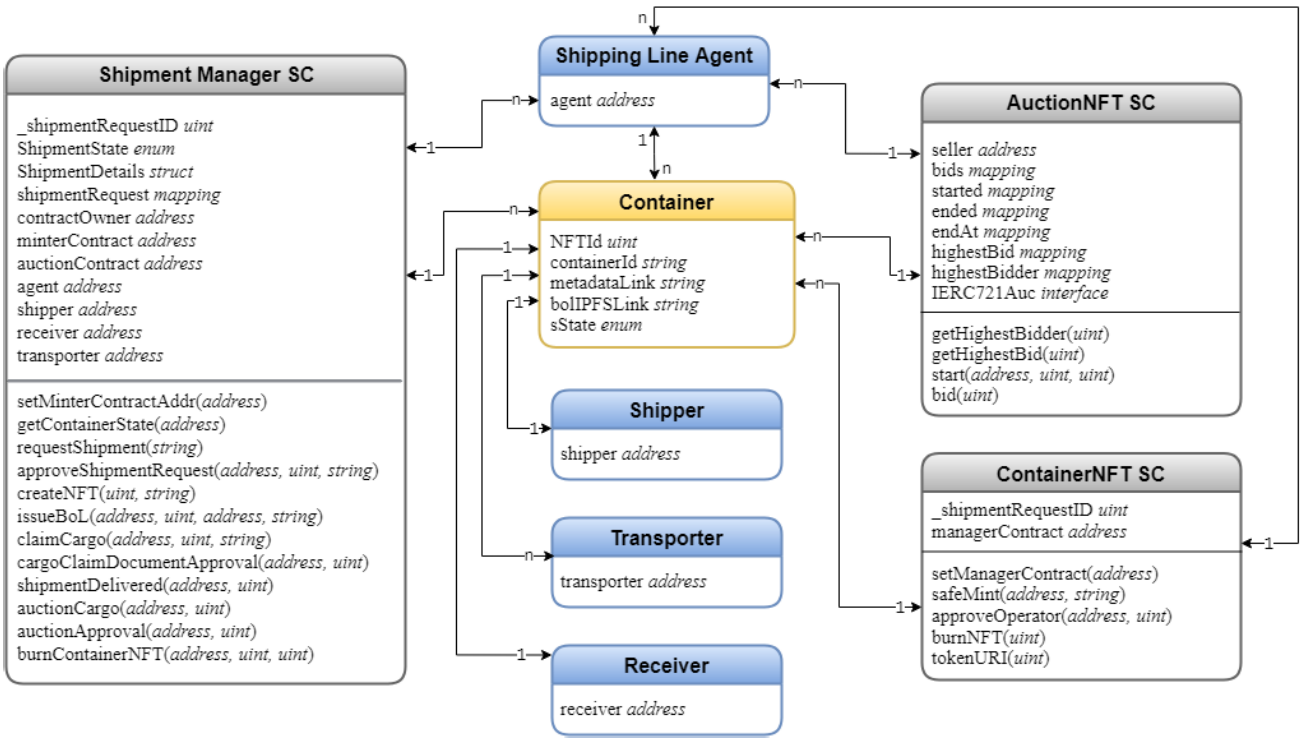

FIGURE 3. Entity relationship diagram.

- **Enum**: A collection of predefined constant string values. This data structure is used in this system to represent the container state, which is updated every time a function is invoked.
- **Struct**: A data structure used to group variables of different data types to represent an object. In this system, the shipment is represented by a struct containing an NFT id, container id, container state, metadata IPFS link, and BoL IPFS link.
- **Functions**: Methods that are used to execute logic and update states and variables.
- **Interface**: Abstract contracts identified by the keyword ''interface''. They contain only function definitions and are used to access the functions of a smart contract externally.

The system is composed of three smart contracts, namely the Shipment Manager, ContainerNFT, and AuctionNFT smart contracts, as briefly shown in the entity relationship diagram in Figure 3. The smart contracts are deployed once using the company's Ethereum address (EA), and in doing so, we ensure accountability of the employees and everyone who uses the functions as they are assigned different EAs. Every company can have more than one agent who manages n number of container shipments. Therefore, the relationship between the shipping agent and the shipments is 1 to n. However, since these agents use the same deployed smart contracts, the relationship between every smart contract and the agents is 1 to n. Every shipment is owned by one shipper and has one receiver, resulting in the cardinality of 1:1 in both cases. Finally, every shipment is transported by more than one transporter; therefore, it has a relationship of 1:n as shown in the diagram.

B. SYSTEM PROCESS FLOW

The interactions between each stakeholder and the smart contracts are demonstrated in a sequence of events and functions as shown in Figures 4 and 5. Initially, every stakeholder is required to register thereby owning an EA composed of a set of public and private keys. The public key acts as the user's public identifier while the private keys are used to digitally sign transactions.

The smart contracts are deployed by the same address, which we refer to here as the shipping line company's address. As shown in Figure 4, the shipping process begins when a shipper issues a request using the *requestShipment*() function, which results in an event broadcast on the network. Following the placement of the request, the shipping agent approves the required documents, books a container for the shipper, and assigns a shipping container number to the request. The shipping container number, according to the ISO standard, is a unique combination of letters and numbers used to identify a container internationally. The shipper then drafts and uploads the NFT metadata that includes details such as the assigned container number, shipment owner, shipment receiver, shipment content, images, and other related documents to IPFS. This upload returns a unique CID hash link that points to the data. The shipper then mints the NFT using the URI, a concatenation of the gateway and the CID, with the *createNFT*() method while also approving the shipping agent to manage the NFT ownership using

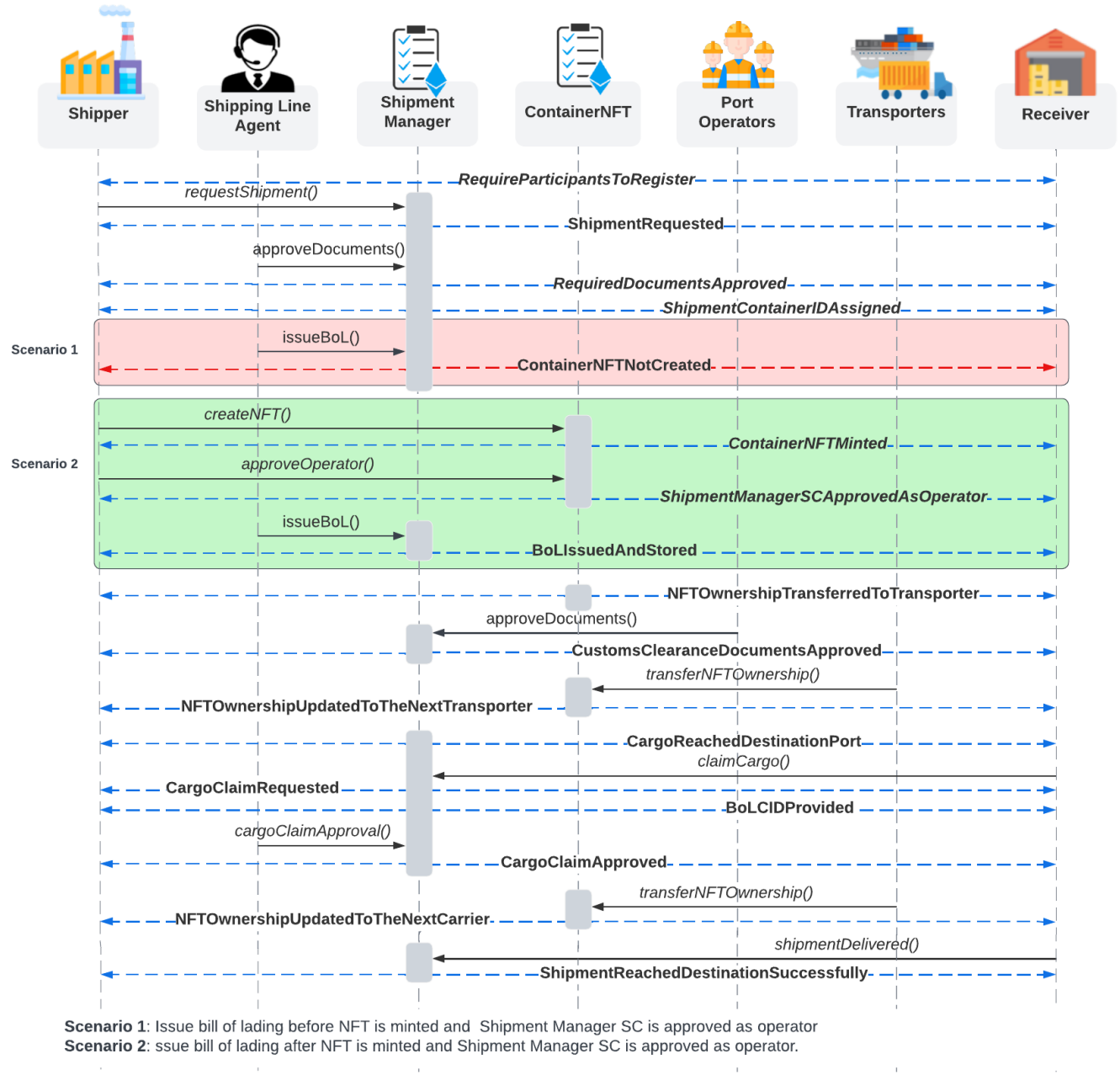

FIGURE 4. Sequence diagram depicting the communication between the participants and the smart contracts when cargo is claimed at the destination port.

approveOperator(). The shipping agent then proceeds with issuing the BoL, sets the container's BoL IPFS link, and transfers the NFT ownership to the first-mile transporter using the *issueBoL*() function. The NFT ownership is updated every time the cargo is handed off to the next transporter through the *transferNFTOwnership*() function.

Finally, when the shipment reaches the destination port, one of two options takes place. The receiver can either claim the cargo as in Figure 4 or abandon it as in Figure 5. If the case is the former, the receiver provides the BoL hash link and invokes the *claimCargo*() function. Upon getting notified through an event, the shipping agent approves the required documents and releases the cargo to the receiver using the *cargoClaimApproval*() method. The cargo is then delivered and the NFT ownership is transferred to the receiver. The receiver can then finally confirm the shipment's delivery with the *shipmentDelivered*() method. On the other hand, in the case of abandoned cargo, the receiver can notify the port using the *auctionCargo*() method as shown in Figure 5. The last transporter, who is the current owner of the NFT, then proceeds with approving the shipment manager contract to manage the NFT ownership transfers. The shipping agent is then able to approve the auction using the *auctionApproval*() function and transfer the NFT to the AuctionNFT contract

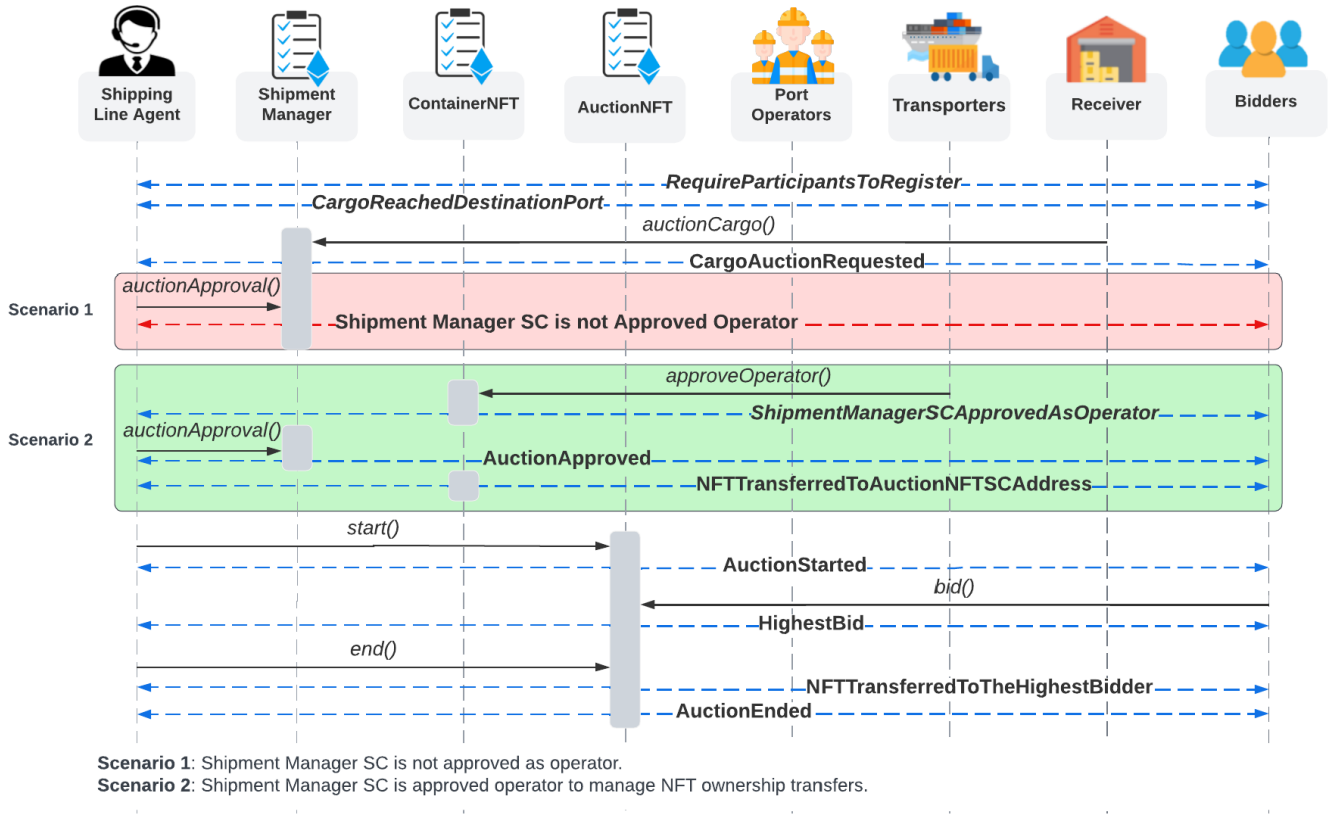

FIGURE 5. Sequence diagram depicting the communication between the participants and the smart contracts when cargo is auctioned or abandoned at the destination port.

address. The cargo is then auctioned off using the *start*() function and the registered bidders can participate using the *bid*() method. When the auction ends, the NFT is transferred to the highest bidder after approving the required documents and settling payment.

V. IMPLEMENTATION DETAILS

This section discusses the implementation details of the proposed blockchain-based system. The code was written using the Solidity programming language and was developed, compiled, and debugged using the Remix IDE, a web-based platform. The implementation was deployed on a test network on the Ethereum blockchain. The full code is made publicly available on GitHub.[1] A total of three smart contracts are deployed for the system, and each contract is separated from the others depending on the tasks it accomplishes. The permission and access control of the functions in the smart contracts are enforced using modifiers. These developed contracts are explained briefly below.

A. SHIPMENT MANAGER

This contract enables the registered stakeholders to manage their shipments and shipment requests. Using this contract, the shippers may place shipment requests, create the shipment NFT, and invoke functions from the ContainerNFT smart contract. Moreover, the shipping line agent can manage numerous shipment requests, issue the BoL, approve required documents, assign transporters, and announce an auction for a cargo. Finally, the receivers can track their shipment, place a claim request, and confirm the delivery of the cargo.

B. ContainerNFT

It is composed of inherited methods and properties from the *ERC721, ERC721URIStorage, ERC721Burnable, Ownable* contracts from OpenZeppelin, a library of secure and reusable contracts. The ContainerNFT contract is purely for minting and managing NFTs and allows only the owner of the token or an approved operator to transfer, approve, or burn a given NFT token. The majority of function executions are for requests submitted via the shipment manager contract, which in turn are restricted using modifiers.

C. AuctionNFT

It is used when the receiver decides to sell the cargo at auction. This contract communicates with the former two via an interface. The interface sets the contracts from which we want to invoke functions.

The algorithms described in this section use the smart contracts' functions according to the process flow. In Algorithm 1, the *requestShipment()* function is invoked by a party registered as an exporter. This function creates a shipment struct and sets its details, such as origin, destination, sender,

[1] https://github.com/fairouzK/Container-NFT.git

Algorithm 1 Algorithm for Booking and Approval of a Shipment Request

Function *requestShipment()*
  Input : origin, destination, size, content, receiver EA
  Require: caller to be registered
  if *caller ID is not exporter* **then**
    caller is not authorized to execute function
  end
  else
    Create a container struct
    */*A struct with the attributes described in Figure 3 */*
    Set container struct details
    Assign *requestId* to the current counter value
    shipment[caller EA][requestId].sStatus = *Requested*
    Increment counter
    */*Details of the container such as state, origin, and destination are updated in this algorithm*/*
  end
  Output : Emit ShipmentRequested Event
Function *approveShipmentRequest()*
  Input : requester EA, requestId, containerID
  Require: caller to be shipping line agent
  if *caller ID is not agent* **then**
    caller is not authorized to execute function
  end
  else
    shipment[requester][requestId].sStatus = *ContainerIdAssigned*
    shipment[requester][requestId].containerId = containerID
    */*containerId of the struct is assigned to a physical container identifier.*/*
    Send the containerId details to the exporter via event using the *requester EA, requestId and containerId*
  end
  Output : Emit ShipmentApprovedAndContainerIDAssigned Event

Algorithm 2 Algorithm for NFT Creation and Operator Approval

Function *createNFT()*
  Input : shipmentId, metadata uri
  Require: shipment[caller EA][shipmentId].sStatus == *ContainerIdAssigned*
  if *caller ID is not exporter* **then**
    caller is not authorized to execute function
  end
  else
    Mint NFT using *safeMint(caller ID, uri)*
    NFTId = *identifier generated by safeMint()*
    / when an NFT is minted, it is assigned a unique identifier.*/*
    shipment[caller EA][shipmentId].sStatus = *NFTMinted*
  end
  Output : Emit NFTmintedForContainer Event
Function *safeMint()*
  Input : minter EA, metadata uri
  Require: call made from the Shipment Manager SC
  if *caller ID is not ManagerSc* **then**
    caller is not authorized to execute function
  end
  else
    Assign nftId to the current counter value
    */*nftId is a unique NFT identifier*/*
    Increment counter
    _safeMint(to, nftId)
    _setTokenURI(nftId, uri)
    / _safeMint and _setTokenURI are private functions from the OpenZeppelin contracts.*/*
  end
  Output : return nftId
Function *approveOperator()*
  Input : Manager SC EA, nftId
  Require: caller is NFT owner
  approve(Manager SC EA, nftId)
  / a function from the OpenZeppelin contracts.*/*
  / the Manager SC is approved as operator since all calls are made from the SC.*/*
  Output : Emit operatorApproved Event

receiver, and content of the cargo, as per the given inputs. Every request is assigned an identifier, and when the request is approved as a shipment, it is recognized by the sender's EA and the request ID. This process sets the container status to *Requested*, increments the counter for the next request, and emits a request event. Following the request placement, the shipping agent approves the documents, assigns a container identifier to the request, and updates the container status to *ContainerIdAssigned* through the *approveShipmentRequest()* function.

Algorithm 2 details the process of NFT creation and operator approval for ownership transfers. Once the container ID is assigned, the shipper is able to mint an NFT representing the shipment using the *createNFT()* function. This function calls the *safeMint()* method from the ContainerNFT contract, which transfers the NFT from a zero address to the mint address, which is the shipper in this case. A successful execution of the function results in *NFTMintedForContainer* event. The shipper then approves the Manager contract as an operator to manage the transfer of the NFT ownership using *approveOperator()*. The ownership transfer requests can only be initiated by the NFT owner or an approved operator. Due to this restriction, the approval of the Manager contract is placed as a requirement for BoL issuance as shown in Algorithm 3. When approved, the agent issues the BoL, stores it in IPFS, broadcasts its URI to the blockchain, and transfers

Algorithm 3 Algorithm for Issuing BoL, and NFT Ownership Transfer

Function *issueBoL()*
  Input : shipper EA, shipmentId, transporter EA, BoL CID
  Require: shipment[sender][shipmentId].sStatus == *NFTMinted* AND Manager SC is approved operator
  if *caller ID is not agent* **then**
    caller is not authorized to execute function
  end
  else
    shipment[sender][shipmentId].bol = BoL CID
    / set the bol string of the struct to the IPFS link of the BoL document.*/*
    shipment[sender][shipmentId].sStatus = *BoLIssued*
    safeTransferFrom(exporter EA, transporter EA, nftId)
    / a function from OpenZeppelin contracts to transfer NFT ownership and emits Transfer event.*/*
    shipment[sender][shipmentId].sStatus = *departed*
  end
  Output : Emit BoLIssuedAndStored and Transfer Events
Function *safeTransferFrom()*
  Input : owner EA, newOwner EA, nftId
  Require: caller is NFT owner
  update NFT owner to newOwner EA
  / a function which is part of the OpenZeppelin contracts.*/*
  Output : Emit Transfer Event

the token ownership to the first-mile transporter using *issueBoL()*. Within this function, the *safeTransferFrom()* function from the ContainerNFT contract is invoked. The successful execution of these functions emits *BoLIssuedAndStored* and *Transfer* events. Finally, the container status is updated to *Departed*.

Following the completion of shipment haulage, either Algorithm 4 or Algorithm 5 is executed. The Algorithm 4 describes the process flow of a cargo claim when it reaches the destination port. If the receiver decides to claim the cargo, the *claimCargo()* function is invoked by providing the CID of the BoL. To approve the claim, the agent verifies the authenticity of the document and approves the claim through the *cargoClaimApproval()* method. This execution emits a *CargoClaimApproved* event and updates the shipment status to *ClaimApproved*. Finally, the receiver can confirm the delivery through *shipmentDelivered()* function. This function can only be invoked by the party registered as the receiver of the cargo. The status is updated to *DestinationReached* and a *ShipmentDeliveredSuccessfully* event is emitted.

Algorithm 4 Algorithm for Cargo Claim Process

Function *claimCargo()*
  Input : sender EA, shipmentId, BoL CID
  Require: shipment[sender][shipmentId].sStatus == *Departed*
  if *caller ID is not receiver* **then**
    caller is not authorized to execute function
  end
  else
    shipment[sender][shipmentId].sStatus = *Claimed*
    / BoL CID is used to verify the BoL document stored in IPFS.*/*
  end
  Output : Emit CargoClaimRequested Event
Function *cargoClaimApproval()*
  Input : sender EA, shipmentId
  Require: shipment[sender][shipmentId].sStatus == *Claimed*
  if *caller ID is not agent* **then**
    caller is not authorized to execute function
  end
  else
    */*documents related to claim process are verified.*/*
    / BoL CID is used to verify the authenticity of the BoL.*/*
    shipment[sender][shipmentId].sStatus == *ClaimApproved*
  end
  Output : Emit CargoClaimApproved Event
/ NFT ownership is transferred to the receiver after cargo claim approval.*/*
Function *shipmentDelivered()*
  Input : sender EA, shipmentId
  Require: shipment[sender][shipmentId].sStatus == *ClaimedApproved*
  if *caller ID is not receiver* **then**
    caller is not authorized to execute function
  end
  else
    shipment[sender][shipmentId].sStatus == *DestinationReached*
  end
  Output : Emit ShipmentDeliveredSuccessfully Event

Algorithm 5, on the other hand, is executed if the receiver decides to abandon the cargo. The auction request is submitted using *auctionCargo()* function from the AuctionNFT contract. This function requires the container status to not be set to *Claimed* or *ClaimApproved* and updates the status to *Auctioned*. When the receiver places an auction request, the last transporter approves the shipping agent to transfer the ownership of the token to the AuctionNFT smart contract's address. This is due to the restriction that the AuctionNFT

Algorithm 5 Algorithm for Cargo Auction

Function *auctionCargo()*
  Input : sender EA, shipmentId
  Require: caller ID is agent or receiver
  Require: shipment[sender][shipmentId].sStatus != *Claimed* AND shipment[sender][shipmentId].sStatus != *ClaimApproved*
  shipment[sender][shipmentId].sStatus = *Auctioned*
  Output : Emit CargoAuctionRequested Event
/ the last owner(transporter) needs to approve the Manager SC as an operator */*
Function *auctionApproval()*
  Input : sender EA, shipmentId, nft owner EA, nftId, auction SC EA
  Require: caller ID is agent
  Require: shipment[sender][shipmentId].sStatus == *Auctioned* AND Manager SC is approved operator
  transferFrom(nft owner, auction SC EA, nftId)
  / a function from OpenZeppelin contracts.*/*
  shipment[sender][shipmentId].sStatus = *AuctionApproved*
  Output : Emit AuctionApprovedAndNFTTransferredToAuctionSC Event
Function *start()*
  Input : nftId, starting Bid
  Require: *caller ID is agent* AND *auction[nftId] not started*
  highestBid[nftId] = starting Bid
  started[nftId] = true
  Output : Emit AuctionStarted Event
Function *bid()*
  Input : nftId, amount
  Require: *auction[nftId] started* AND *auction[nftId] not ended*
  if *amount > highestBid[nftId]* **then**
    highestBid[nftId] = amount
    highestBidder[nftId] = initiator
  end
  Output : Emit Bid Event
Function *end()*
  Input : nftId
  Require: *caller ID is agent* AND *auction[nftId] not ended*
  if *highestBidder[nftId] != seller* **then**
    transfer(highestBidder[nftId], nftId)
    / a function from OpenZeppelin contracts to transfer nft to highest bidder*/*
  end
  ended[nftId] = true
  Output : Emit AuctionEnded Event

contract does not allow users to auction a token that they do not own or are not approved to. When starting the auction, the starting amount and the auction duration are set. The smart contract allows bids that offer a higher amount than the current highest bid while reverting lower offer transactions. After the auction ends, the highest bidder and the highest bid are announced. In the end, the NFT ownership is transferred to the highest bidder after settling the documentation and payment requirements.

TABLE 1. Ethereum addresses of participants used for testing.

Actors	Ethereum Addresses
Company	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Shipper	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
Agent	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
Receiver	0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
Transporter	0x617F2E2fD72FD9D5503197092aC168c91465E7f2

FIGURE 6. Reverted function execution when invoked by a random address.

VI. TESTING AND VALIDATION

In this section, we verify and validate the implementation of the proposed solution. The Remix IDE enables testing of the developed contracts and offers a list of random addresses to test and validate the system with. The addresses used are listed in Table 1. The system is evaluated in terms of access permission, successful execution of functions, accurate process flow logic and container status updates, and broadcast events. We demonstrate three scenarios, which include minting, transfer of ownership, and auctioning off the container NFT.

The process flow is controlled by the container states, where one stage requires a certain enum value set by the previous stage. After the shipment request is placed by the party registered as a shipper and the required documents are verified by the agent, the shipping request is assigned a container number. This container identifier, a requirement for minting the NFT, is set only by the agent, and the permission is reinforced using a modifier. This identifier is also a prerequisite for minting the NFT. Figures 6 and 7 below demonstrate the cases where the container identifier is not set prior to minting or parties other than the approved agent attempt to set it. Finally, Figure 8 illustrates the successful creation of NFT by the shipper as a result of a valid container identifier setting. The minting process in the NFT ecosystem commences with a 0×00 address, as indicated by the yellow box, which then transfers the NFT to the party invoking the mint function. For testing, we upload the metadata directly to IPFS using the IPFS desktop application. This upload returns a CID and a share link in the format of *https://ipfs.io/ipfs/CID*.

FIGURE 7. Reverted function execution when createNFT() function is invoked prior to shipment approval and container identifier setting.

FIGURE 8. Successful minting of container NFT by the shipper.

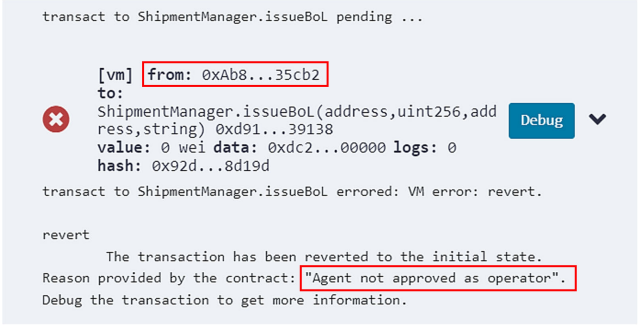

FIGURE 9. Issuing BoL reverted due to unauthorized access.

The next test case is the approval of an operator, which is required at the issuance of the BoL and auction stages. The BoL is successfully issued only after the shipper approves the Manager smart contract to manage the NFT ownership. The agent then sets the BoL CID and transfers the NFT ownership to the first-mile transporter within the same function as the Manager contract. This ensures the shipper and the agent that the exchange of the BoL and the NFT ownership happens at the same time, transparently and securely. Figure 9 depicts a scenario in which the agent attempts to issue the BoL despite not being approved as an operator, whereas Figures 10 and 11 depict the function being successfully executed when the agent is approved.

Unlike the previous case, the approval of the operator in the auction stage is initiated by the transporter, the current owner. By approving the Manager contract, the agent is able

FIGURE 10. Successful execution of issueBoL() function after agent is approved as operator.

FIGURE 11. Successful transfer of NFT ownership to the first-mile transporter.

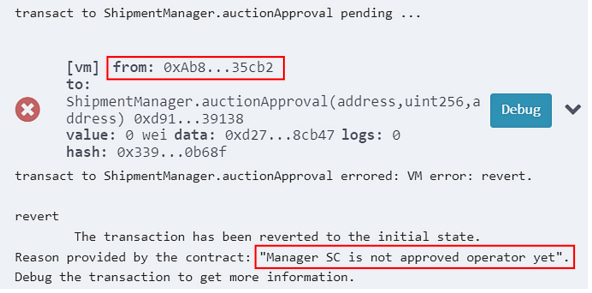

FIGURE 12. Reverted approval of auction due to lack of permission to manage the NFT ownership.

FIGURE 13. Successful approval of auction after the agent is approved as an operator.

to transfer the NFT ownership to the AuctionNFT smart contract's address from within the Manager contract using the *approveAuction*() function. Figures 12 and 13 illustrate the reverted and successful execution of the function explained above.

Finally, when the auction is conducted, the agent is able to end the auction after settling the required documents and

```
},
{
        "from": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
        "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
        "event": "Transfer",
        "args": {
                "0": "0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47",
                "1": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                "2": "0",
                "from": "0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47",
                "to": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                "tokenId": "0"
        }
},
{
        "from": "0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47",
        "topic": "0xd2aa34a4fdbbc6dff6a3e56f46e0f3ae2a31d7785ff3487aa5c95c642acea501",
        "event": "AuctionEnded",
        "args": {
                "0": "0",
                "1": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                "2": "100",
                "NFTId": "0",
                "winningBidder": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                "highestBid": "100"
        }
}
```


FIGURE 14. The successful transfer of container NFT ownership to the highest bidder.

payment. The NFT is transferred to the highest bidder through the function end() in the AuctionNFT smart contract, as shown in Figure 14.

VII. DISCUSSION

This section discusses the cost and security analysis of the deployed smart contracts. We also provide a comparison with existing approaches in the maritime shipping industry and highlight the generalization of the proposed system.

A. COST ANALYSIS

Smart contracts, when triggered, use gas for transactions and executions on the Ethereum blockchain. The amount of gas consumed depends on various factors such as the code size, function input, and output size, variable value setting and updating, and data types used. The estimated gas in the Remix IDE reflects the actual cost of execution and transactions on the blockchain. There are two types of costs in the Ethereum blockchain: execution and transaction costs. The execution cost is associated with performing operations in the Ethereum virtual machine (EVM). On the other hand, the transaction cost is associated with deploying and executing a piece of code on the Ethereum blockchain. In conclusion, the transaction cost includes the execution cost and the cost of deploying the smart contract on the blockchain.

The cost of gas is calculated by multiplying the amount of gas consumed by the price of gas. The gas price varies depending on the demand of miners and network congestion. According to the ETH Gas Station (2022), as of today, August 8, 2022, the gas prices for slow, average, and fast executions are 27, 30, and 37 in Gwei, respectively. Based on the current price of Ether of $1597, these values amount to $0.000068, $0.000076, and $0.000094. These prices indicate the priority by which the transactions are validated, where fast executions are given a higher priority than slow executions. The gas costs of every function executed in the process flow of the proposed system are shown in Table 2 in USD. As shown in the table, the *creteNFT*() and *issueBoL*() functions cost the most. This is due to the fact that these functions require a string input from the user, which is the URI of the NFT in the case of *createNFT*() and the CID of the BoL in the case of *issueBoL*(). These functions additionally modify the mapping variables' data in the container struct, which adds to the cost.

These fees are primarily imposed to reward miners for validating transactions and to encourage more participants into the network, as the more participants, the more stable the network grows. However, the Ethereum ecosystem has struggled because of the high gas rates. To combat these rising gas prices, several options have been introduced. One of the solutions is zkSync, which offers low gas prices, fast transactions, and adds scalability to the Ethereum network [32]. Utilizing a private blockchain, where transactions are free of charge, is also another choice. Additionally, we can introduce Web 2.0 functionalities for setting, modifying, and passing variables while using smart contract code to record the history of these changes by broadcasting them to the network.

B. SECURITY ANALYSIS

Our proposed solution incorporates the advantages of blockchain technology into the shipping process through the concepts discussed in this section.

The transactions recorded in the blockchain network are immutable. This is made possible due to the blocks being linked together to form a continuous line of blocks. Every block includes the hash of its preceding block in calculating its own hash value. Hence, any change of content in any preceding block invalidates the current block's hash, consequently affecting all the following blocks. Therefore, any confirmed transaction in the system data cannot be modified or reversed, offering data integrity and security in the shipping process.

Moreover, the documents such as BoL issued in the shipping process are stored in IPFS. It is a decentralized storage system where this upload returns a hash of the document known as the CID. This hash value is broadcast throughout the network for the approved stakeholders to see. Altering or deleting any part of the file results in a completely different hash value. As a result, if a party provides a different CID when the BoL is requested, the responsible authority would flag the document as fraudulent and revert any request. Additionally, uploading these documents to a secure database offers flexibility in the process. The delays resulting from BoL processing can be minimized, and the resulting cargo release delays and demurrage fee accumulations can be reduced. Since the document can be securely accessed online, the additional costs of sending the document and/or fees incurred as a result of lost BoL can be minimized.

Furthermore, the system enforces accountability by requiring every participant to register. Every transaction is signed with the private key of the initiator, which can be verified using the public key. This concept enables non-repudiation, which prevents parties from denying actions after misusing resources. If a fraudulent document or transaction is introduced into the process, the party is held responsible as a result of the digital footprint. Finally, the use of events to broadcast

TABLE 2. Gas cost of invoked functions in USD in a public ethereum blockchain.

Parent SC	Method Name	Transaction Cost (Gas)	Execution Cost (Gas)	Slow Execution	Avg. Execution	Fast Execution
Shipment Manager	requestShipment()	57644	50125	$4.23	$4.70	$5.79
	approveShipmentRequest()	84591	73557	$6.21	$6.90	$8.51
	creatNFT()	235433	204724	$17.28	$19.20	$23.68
	approveOperator()	59248	51520	$4.34	$4.83	$5.96
	issueBoL()	216496	178657	$15.08	$16.75	$20.67
	claimCargo()	39613	34446	$2.90	$3.23	$3.98
	cargoClaimDocumentsApproval()	34569	30060	$2.53	$2.81	$3.47
	auctionCargo()	40376	35109	$2.96	$3.29	$4.06
	auctionApproval()	104631	81383	$6.87	$7.63	$9.41
	shipmentDelivered()	36384	31638	$2.67	$2.96	$3.66
	burnContainerNFT()	99952	58114	$4.90	$5.45	$6.72
ContainerNFT	approveOperator()	32680	28417	$2.39	$2.66	$3.28
AuctionNFT	start()	106779	92851	$7.83	$8.71	$10.74
	bid()	64587	56162	$4.74	$5.26	$6.49
	end()	145438	101767	$8.59	$9.54	$11.77

```
INFO:root:contract ContainerNFT.sol:AuctionNFT:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                  99.4%
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 ContainerNFT.sol:ContainerNFT:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                  92.0%
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 ContainerNFT.sol:ShipmentManager:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                  99.3%
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 ======
root@691255a2e00e:/oyente/oyente#
```

FIGURE 15. Report generated by oyente security analyzer tool on the three deployed smart contracts.

changes and updates in the states enables the stakeholders to view the state of the container transparently. The ownership of the container NFT can be tracked by noting the transfers recorded in the network.

The code executed on the Ethereum blockchain is the same as any other code and has its own vulnerabilities. Since the code cannot be fixed once deployed, careful optimization and logic design are required for a secure smart contract. Smart contracts deal with financial assets and are therefore in a constant loop of attacks. Some of the recent attacks include buffer overflow, arithmetic bugs, reentrancy, time dependency, handling exceptions, and denial of service (DoS) [23]. In order to test our system against such vulnerabilities, we used a security analyzer to pinpoint the vulnerabilities and fix them. There are numerous smart contract security analyzers available, including Oyente, Securify, tractFuzzer, and Sereum, to name a few [33]. These analyzers vary depending on the stage of code development at which they are utilized. Static analyzers, such as Oyente and Securify, deal with the code pre-deployment to the EVM, while dynamic analyzers such as Sereum are involved at run-time, post-deployment of code [23]. In our system, we used Oyente to analyze the developed code. It examines the structure of the code that may be vulnerable and returns the possible bugs in the code. This tool generates a report which is shown in Figure 15. As illustrated in this figure, our code is secure and has no major problems.

C. COMPARISON WITH EXISTING SOLUTIONS

To the best of our knowledge, there is no other solution that implements NFT traceability for shipping containers in the maritime industry. However, there exist similar suggestions and implementations that utilize NFTs for ownership data provenance as shown in Table 3. These solutions are explained in Section III briefly. The comparison metrics are the blockchain platform employed, user permissions in the blockchain, ERC721 implementation, auction and bidding services, off-chain storage systems, and document security.

The compared solutions are in different sectors such as food and art supply chains, invoice financing, pharmaceuticals, physical internet (PI) containers, and shipping containers. The majority of the solutions use the Ethereum blockchain's NFT implementation, whereas Chiacchio et al. [22] use the VeChain Thor blockchain for product traceability. In addition to that, the majority of the solutions use IPFS for off-chain data storage, while Wang et al. [29] employ LevelDB, a key-value-based storage system.

Moreover, our proposed solution and [28] allow the users to auction off their cargo. On the other hand, the rest of the implemented solutions do not provide users with these options and do not discuss the situations where the product is no longer required. Finally, the compared solutions do not offer cost analysis or allow their users to choose the type of platform, while our solution enables its users to make a calculated decision based on the cost analysis.

D. GENERALIZATION

Our approach demonstrates how one can leverage the security and transparency provided by blockchain technology in the

TABLE 3. Comparison between our system and other existing solutions in the supply chain industry.

	Mezquita et al. [28]	Wang et al. [29]	Guerar et al. [30]	Meyer et al. [21]	Chiacchio et al. [22]	Our Solution
Sector	Food supply chain	Art supply chain	Invoice Financing	PI containers	Pharmaceuticals	Shipping containers
Use of NFT	Yes	Yes	No	Yes	Yes	Yes
Auction and Bidding	Yes	Yes	Yes	No	No	Yes
Blockchain platform	Ethereum	Ethereum	Ethereum	Ethereum	VeChain	Ethereum
Permission	Private	Private	Public	Private	Private	Private or Public
Off-chain storage	No	LevelDB	IPFS	IPFS	N/S	IPFS
Document security	N/S	N/S	Yes	N/S	N/S	Yes

N/S : not specified

shipping supply chain. Even though the implementation is focused on the FCL container, it can be customized for small packages in LCL containers. Every package in the container will have its own NFT without affecting the system functionality or the traceability of other packages.

In the actual shipping process, the cargo being transported may or may not require transshipment at an intermediary port. That is, shipping processes can either have a direct route or a pendulum. A pendulum route is when cargo is transshipped between one or more intermediary ports, as opposed to a direct route, when the cargo is transported by a single vessel. In our solution, we assume a direct shipment where a container is shipped from the origin to the destination on one vessel. In cases where transshipment is taken into account, the system can function well. Scalability is not a problem in terms of paperwork because all relevant papers are stored in IPFS. Moreover, a mapping variable for any number of documents related to the shipment, including transshipment permits and/or customs clearance paperwork, is included in the container struct. The system can, therefore, be used in more complicated operations by updating the struct value with the CID of the uploaded document. Furthermore, since all the participants are required to register, the NFT ownership transfer can easily be traced as before.

The system can also be customized to the type of supply chain we want to use it in. Figure 2 can be used as a base template to determine what has to change in the system. In sectors where the processes are not paper-intensive, we can exclude IPFS storage and store the data on-chain. Finally, the functions and events can easily be altered to fit the system's functionalities and process flow.

VIII. CONCLUSION

In this paper, we introduce a blockchain-based solution for tracking container shipments in the maritime industry. We utilized smart contracts and NFTs to develop a system that provides traceability, security, and authenticated process records in the shipping process. We demonstrated how our proposed system makes use of the IPFS storage system to allow the participating parties to securely and safely store and share documents. We presented test cases and scenarios to validate the full functionality and applicability of the system. We showed that our method reduces the elevated costs associated with the shipping process and port management and minimizes the delays caused by lengthy paperwork processing. Through the added analysis and auction services, we highlighted how our solution differs from the existing solutions. The solution is flexible enough to be used on both public and private blockchain platforms depending on the system requirements. The implementation code for blockchain smart contracts has been made publicly accessible on GitHub along with detailed instructions and steps. As a future work, we intend to deploy the system with its smart contracts on the real Ethereum blockchain main network and develop relevant front-end user decentralized applications (DApps) to interact with these smart contracts.

REFERENCES

[1] D. Song, "A literature review, container shipping supply chain: Planning problems and research opportunities," *Logistics*, vol. 5, no. 2, p. 41, Jun. 2021.

[2] A. Papathanasiou, R. Cole, and P. Murray, "The (non-)application of blockchain technology in the Greek shipping industry," *Eur. Manage. J.*, vol. 38, no. 6, pp. 927–938, Dec. 2020.

[3] N. Radonic and R. Beck, "Using blockchain to sustainably manage containers in international shipping," in *Proc. 41st Int. Conf. Inf. Syst.*, India, 2020.

[4] J. Sunny, N. Undralla, and V. M. Pillai, "Supply chain transparency through blockchain-based traceability: An overview with demonstration," *Comput. Ind. Eng.*, vol. 150, Dec. 2020, Art. no. 106895.

[5] A. Musamih, K. Salah, R. Jayaraman, J. Arshad, M. Debe, Y. Al-Hammadi, and S. Ellahham, "A blockchain-based approach for drug traceability in healthcare supply chain," *IEEE Access*, vol. 9, pp. 9728–9743, 2021.

[6] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin*, vol. 4, p. 2, Jun. 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[7] M. Nassar, K. Salah, M. H. ur Rehman, and D. Svetinovic, "Blockchain for explainable and trustworthy artificial intelligence," *WIREs Data Mining Knowl. Discovery*, vol. 10, no. 1, p. e1340, Jan. 2020.

[8] N.-P. Abdellatif, "An ethereum bill of lading under the UNCITRAL MLETR," *Maastricht J. Eur. Comparative Law*, vol. 27, no. 2, pp. 250–274, Apr. 2020.

[9] J. Olsson, "Undelivered goods under the law of carriage of goods by sea," M.S. thesis, Dept. Law, Lund Univ., Lund, Sweden, Sep. 2013.

[10] V. Buterin, "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, no. 37, pp. 1–2, 2014.

[11] M. Westerkamp, F. Victor, and A. Küpper, "Blockchain-based supply chain traceability: Token recipes model manufacturing processes," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1595–1602.

[12] A. A. Aigner and G. Dhaliwal, "UNISWAP: Impermanent loss and risk profile of a liquidity provider," 2021, *arXiv:2106.14404*.

[13] M. Kuhn, F. Funk, G. Zhang, and J. Franke, "Blockchain-based application for the traceability of complex assembly structures," *J. Manuf. Syst.*, vol. 59, pp. 617–630, Apr. 2021.

[14] D. S. Bloch, "Non-fungible tokens: A solution to the challenges of using blockchain bills of lading in the international sales of goods," *J. Law, Market Innov.*, vol. 1, no. 1, pp. 44–65, 2022.

[15] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," 2021, *arXiv:2105.07447*.

[16] L. Kugler, "Non-fungible tokens and the future of art," *Commun. ACM*, vol. 64, no. 9, pp. 19–20, Sep. 2021.

[17] E. Hirata, D. Watanabe, and M. Lambrou, "Shipping digitalization and automation for the smart port," in *Supply Chain—Recent Advances and New Perspectives in the Industry 4.0 Era*, T. Banyai, A. Banyai, and I. Kaczmar, Eds. IntechOpen, 2022, doi: 10.5772/intechopen.102015.
[18] H. Hasan, E. AlHadhrami, A. AlDhaheri, K. Salah, and R. Jayaraman, "Smart contract-based approach for efficient shipment management," *Comput. Ind. Eng.*, vol. 136, pp. 149–159, Oct. 2019.
[19] K. Komathy, "Verifiable and authentic distributed blockchain shipping framework for smart connected ships," *J. Comput. Theor. Nanoscience*, vol. 15, no. 11, pp. 3275–3281, Nov. 2018.
[20] 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.
[21] T. Meyer, M. Kuhn, and E. Hartmann, "Blockchain technology enabling the physical internet: A synergetic application framework," *Comput. Ind. Eng.*, vol. 136, pp. 5–17, Oct. 2019.
[22] F. Chiacchio, D. D'Urso, L. M. Oliveri, A. Spitaleri, C. Spampinato, and D. Giordano, "A non-fungible token solution for the track and trace of pharmaceutical supply chain," *Appl. Sci.*, vol. 12, no. 8, p. 4019, Apr. 2022.
[23] Y. Z. Lim, J. Zhou, and M. Saerbeck, "Shaping blockchain technology for securing supply chains," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, 2021, pp. 3–18.
[24] J. Arcenegui, R. Arjona, and I. Baturone, "Secure management of IoT devices based on blockchain non-fungible tokens and physical unclonable functions," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, 2020, pp. 24–40.
[25] V. Valastin, K. Kost'al, R. Bencel, and I. Kotuliak, "Blockchain based car-sharing platform," in *Proc. Int. Symp. ELMAR*, Sep. 2019, pp. 5–8.
[26] R. B. dos Santos, N. M. Torrisi, and R. P. Pantoni, "Third party certification of agri-food supply chain using smart contracts and blockchain tokens," *Sensors*, vol. 21, no. 16, p. 5307, Aug. 2021.
[27] T. Le, Y. Kim, and J.-Y. Jo, "Implementation of a blockchain-based event reselling system," in *Proc. 6th Int. Conf. Comput. Sci., Intell. Appl. Informat. (CSII)*, May 2019, pp. 50–55.
[28] Y. Mezquita, A. González-Briones, R. Casado-Vara, P. Chamoso, J. Prieto, and J. M. Corchado, "Blockchain-based architecture: A mas proposal for efficient agri-food supply chains," in *Proc. Int. Symp. Ambient Intell.* Cham, Switzerland: Springer, 2019, pp. 89–96.
[29] Z. Wang, L. Yang, Q. Wang, D. Liu, Z. Xu, and S. Liu, "ArtChain: Blockchain-enabled platform for art marketplace," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 447–454.
[30] M. Guerar, L. Verderame, A. Merlo, and M. Migliardi, "Blockchain-based risk mitigation for invoice financing," in *Proc. 23rd Int. Database Appl. Eng. Symp. (IDEAS)*, 2019, pp. 1–6.
[31] J. Martins, M. Parente, M. Amorim-Lopes, L. Amaral, G. Figueira, P. Rocha, and P. Amorim, "Fostering customer bargaining and E-procurement through a decentralised marketplace on the blockchain," *IEEE Trans. Eng. Manag.*, vol. 69, no. 3, pp. 810–824, Jun. 2022.
[32] A. A. Pandey, T. F. Fernandez, R. Bansal, and A. K. Tyagi, "Maintaining scalability in blockchain," in *Proc. Int. Conf. Intell. Syst. Design Appl.* Cham, Switzerland: Springer, 2022, pp. 34–45.
[33] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Ethereum smart contract analysis tools: A systematic review," *IEEE Access*, vol. 10, pp. 57037–57062, 2022.

FERUZ K. ELMAY received the B.Sc. degree in computer engineering from Khalifa University, Abu Dhabi, United Arab Emirates, in 2020, where she is currently pursuing the M.S. degree in computer engineering with the Department of Electrical and Computer Engineering. She is also a full-time Researcher at the Department of Electrical and Computer Engineering, Khalifa University, where she is also a Research and Teaching Assistant. Her research interests include blockchain and NFTs applications in the supply chain.

KHALED SALAH (Senior Member, IEEE) received the B.S. degree in computer engineering with a minor in computer science from Iowa State University, USA, in 1990, and the M.S. degree in computer systems engineering and the Ph.D. degree in computer science from the Illinois Institute of Technology, USA, in 1994 and 2000, respectively. He is currently a Full Professor at the Department of Electrical and Computer Engineering, Khalifa University, United Arab Emirates. He joined Khalifa University, in August 2010, and is teaching graduate and bachelor's courses in the areas of cloud computing, computer and network security, computer networks, operating systems, and performance modeling and analysis. Prior to joining Khalifa University, he worked for ten years at the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia. He has over 190 publications and three patents, has been giving a number of international keynote speeches, invited talks, tutorials, and research seminars on the subjects of blockchain, the IoT, fog and cloud computing, and cybersecurity. He was a recipient of the Khalifa University Outstanding Research Award, in 2014 and 2015, the KFUPM University Excellence in Research Award, in 2008 and 2009, and the KFUPM Best Research Project Award, in 2009 and 2010, and also a recipient of the departmental awards for Distinguished Research and Teaching in prior years. He serves on the editorial boards of many WOS-listed journals, including *IET Communications*, *IET Networks*, *JNCA* (Elsevier), *SCN* (Wiley), *IJNM* (Wiley), *J.UCS*, and *AJSE*. He is the Track Chair of IEEE Globecom 2018 on Cloud Computing. He is an Associate Editor of IEEE Blockchain Newsletter and a member of IEEE Blockchain Education Committee.

RAJA JAYARAMAN received the bachelor's and master's degrees in mathematics from India, the Master of Science degree in industrial engineering from New Mexico State University, and the Ph.D. degree in industrial engineering from Texas Tech University. He is currently an Associate Professor at the Department of Industrial & Systems Engineering, Khalifa University, Abu Dhabi, United Arab Emirates. His research interests include application of blockchain technology, systems engineering and process optimization techniques to characterize, model and analyze complex systems with applications to supply chains, maintenance planning, and healthcare delivery. His post doctoral research was on technology adoption and implementation of innovative practices in the healthcare supply chains and service delivery. He has led several successful research projects and pilot implementations of supply chain data standards in the U.S. healthcare system.

ILHAAM A. OMAR received the bachelor's degree in electrical and electronic engineering and the master's degree in engineering systems and management from Khalifa University, in 2017 and 2020, respectively. She and her team built a talking digital clock for the visually impaired along with her team, in 2014, and switch mode power supply for her graduation project. She focused on the fabrication of microscale memristors in which she participated in the undergraduate research competition, in 2016. Her research interests include the field of blockchain technology and how it impacts the healthcare sector. She found this research area intriguing as it combines both a cutting-edge technology and healthcare both of which she finds captivating.