Blockchain-Based Traceability for Shipping Containers in Unimodal and Multimodal Logistics

FERUZ KIFLAY ELMAY, KHALED SALAH, (Senior Member, IEEE), IBRAR YAQOOB, (Senior Member, IEEE), RAJA JAYARAMAN, (Senior Member, IEEE), AMMAR BATTAH, AND YASSINE MALEH, (Senior Member, IEEE)

1Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
2Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, United Arab Emirates
3ENSAK, Sultan Moulay Slimane University, Beni-Mellal 23000, Morocco

Corresponding author: Ibrar Yaqoob (ibrar.yaqoob@ku.ac.ae)

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

ABSTRACT An unprecedented amount of goods and commodities are shipped and transported globally each day through different modes of transport. Due to its complexity, the maritime industry suffers from a lack of trust and secure ownership evidence, protracted documentation procedures, and excessive data aggregation. These shortcomings are reflected in cargo processing delays and elevated costs in the shipping process. Most of today’s systems and technologies leveraged for managing shipping containers in unimodal and multimodal logistics fall short of providing transparency, traceability, reliability, audit, security, and trust features. In this paper, we propose a blockchain-based solution that allows users to trace and track their container shipments in a manner that is decentralized, transparent, auditable, secure, and trustworthy. We employ the InterPlanetary File System (IPFS) to overcome the limited data storage problem. We develop smart contracts and present algorithms along with their full implementation, testing, and validation details in both unimodal and multimodal logistics. We present security and cost analyses to show that the proposed solution is secure and cost-efficient. Furthermore, we compare our proposed solution with the existing solutions to show its novelty. All developed smart contract codes are made publicly available on GitHub.

INDEX TERMS Blockchain, ethereum, multimodal logistics, shipping containers, smart contracts, unimodal logistics.

I. INTRODUCTION Logistics is the integration of various complex processes for the efficient and effective transportation of goods in response to a customer’s requests. In realizing a successful logistics flow, numerous modes of transportation and their combinations on land, air, and sea are utilized, as briefly shown in Figure 1. The use of a single mode of transportation, usually trucking, is referred to as unimodal logistics, while hauling shipments using two or more modes of transportation falls under multimodal logistics [1]. Multimodal logistics is a transportation network that uses multiple modes of transportation to safely and efficiently transport cargo in accordance with a specific contract. This mode of logistics is characterized by a long service chain, a network of similar industries and hubs, and the high employment of available resources. Seaway and airway modes of transport explicitly fall under multimodal transport since trucking, railroads, or both must be used for the first and last miles of the shipping process. Moreover, multimodal transport can be categorized into a direct route, where the carrying vessel travels from the origin port to the destination port, or an indirect route, where one or more intermediary ports are used, as briefly described in Figure 2.

Maritime shipping is one of the fundamental aspects of globalization and a vital part of international trade. Seaborne transport alone is responsible for 90% of all commodities transported internationally and 70% of the global trade by value, with containerized cargo accounting for 50% of the sea-based shipments [2]. The Drewery Shipping
Consultant reported that container shipping surpassed 37 million twenty-equivalent units (TEUs) in 2018 alone, demonstrating the widespread use of containers as carrying units [2]. This is attributable to the availability of flexible means of transportation for container movement, which potentially reduces cargo handling and transit costs and time. The maritime industry is composed of complex operations with multi-echelon parties obliged to communicate despite their differences in customs and shipment handling processes. As a result, in such a setting, trust is crucial. However, due to the competitive nature of the sector, stakeholders are reluctant to share data, which is eventually reflected in the statistics on container losses and shipping process setbacks [2]. Furthermore, another vulnerable point in the shipping industry is the lack of reliable real-time visibility of freight status throughout its shipment process. Disputes and conflicts on account of shipment damage or loss arising from such factors result in higher costs for all parties involved. Moreover, as the volume of shipped goods and participation of the industry in all critical sectors, such as healthcare, grows, the amount of mislabeled, lost, and counterfeit cargo also increases correspondingly. According to the National Cargo Security Council, the losses encountered in the cargo industry due to mislabeling, theft, and lost shipments in the USA only account for $50 billion annually, equivalent to $40 per person daily [2]. CargoNet also documented 359 fraud and theft cases in the third quarter of 2021, mainly targeting warehouses and in-transit cargoes throughout the USA and Canada [3].

Several paper-based documents, including but not limited to cargo agreements, contracts of affreightment, customs clearance, cargo manifestos, and bills of lading, are required to be filled out and approved as part of port operations in order to successfully move goods in any international trade [4]. These documents are then shared with the respective stakeholders for approval through infeasible and expensive software programs such as electronic data interchange (EDI) or courier services [4]. The paperwork-related average waiting time extends from several days to weeks, and paperwork administration is estimated to reach 20% of the total cost of the shipping industry [5]. The delay in processing these documents results in congestion in the terminal, a delay in container departure time, and consequently, delivery time, which can adversely affect time-sensitive and perishable shipments. In addition to the increased waiting time due to paperwork processing, these paper-based documents are prone to forgery, data inconsistency, incomplete information, and human error [5]. For instance, fraudulent bill of lading documents is used to transport stolen cargo and unauthorized goods. While granting that the recent introduction of an electronic bill of lading has lowered the rate of fraud, it remains insecure since indistinguishable duplicates of the document can be created.

There is a need for a solution that can help track freight and related processes in a transparent and secure manner. Blockchain has the potential to resolve the aforementioned issues in shipping logistics. Through the advantage of a decentralized administration, blockchain can ensure traceability, transparency, and security of the transactions of the participating stakeholders, which strengthens the trust between the participants. In this paper, we propose a blockchain-based traceability solution for shipping containers in unimodal and multimodal logistics. The main contributions are as follows:

- We present a blockchain-based solution to trace and track shipping containers in both unimodal and multimodal logistics in a way that is decentralized, transparent, reliable, auditable, secure, and trustworthy.
- We demonstrate how the proposed system architecture utilizes the InterPlanetary File System (IPFS) to record all the transactions related to shipping containers in an off-chain manner to overcome the limited storage issue.
- We develop smart contracts and present algorithms along with their full implementation, testing, and validation details. Additionally, we present the sequence diagrams to explain the sequence of events and responses as a result of smart contract functions being executed in the shipping process.
- We evaluate our proposed solution through test cases and measure the performance and execution cost of the
The remainder of the paper is organized as follows. Section II discusses the key technologies and components used in enabling the proposed solution. The related work is presented in Section III. Section IV discusses the proposed system, its components, and the process flow. Sections V and VI present the implementation, testing, and validation details of the solution. In Section VII, we present cost and security analysis as well as a comparison table with the existing solutions. Finally, Section VIII provides the concluding remarks.

II. BACKGROUND
This section presents the key components and technologies used in designing and implementing our solution.

A. BLOCKCHAIN 2.0
Blockchain is a decentralized and immutable ledger that enables peers to send and receive assets directly without the necessity of an intermediary [6]. It is characterized by features such as transparency and irreversibility of transactions, identity preservation, and permanent and tamper-proof storage of data [6]. Blockchain 1.0, the bitcoin blockchain, is not programmable or customizable for business applications as it is limited to financial transactions. This necessitated the introduction of Blockchain 2.0 [7]. Blockchain 2.0 utilizes the concept of smart contracts. Smart contracts are self-executing immutable programs that are triggered when a pre-defined condition is met or if invoked from another contract [8]. These contracts enable users to interact with the blockchain ledger by reading and writing values to the ledger through transactions. Some instances of Blockchain 2.0 include Ethereum and Hyperledger Fabric.

Ethereum is a blockchain platform that addresses several limitations of the bitcoin blockchain [9]. Ethereum networks are classified as either public or private. The public network allows anyone to join and transact, whereas private networks are accessible only to permissioned users [10]. The public network requires Ether, the Ethereum native token. Using this token, users may exchange financial assets or invoke custom smart contract functions for business applications [11]. The private network, however, does not require Ether tokens for non-financial transactions. The Ethereum-based smart contract is coded using Solidity, a JavaScript-like programming language. Moreover, it enables users to deploy decentralized
applications (DApps) on top of the blockchain network. This evolution has opened an opportunity for the wide adoption of blockchain technology for the traceability of processes in a variety of fields, including military and supply chain operations, as well as for securing corporate operations and artificial intelligence (AI) frameworks [12], [13].

B. DECENTRALIZED STORAGE SYSTEM

Even though blockchain technology provides a secure and transparent record of transactions, its storage capacity is limited. To tackle this shortcoming, we utilize a decentralized storage option, the InterPlanetary File System (IPFS). This storage system is a content-based distributed peer-to-peer file system [14]. Upon upload, IPFS returns a unique hash value, known as the content identifier (CID), of the uploaded file. The generated hash link points to the file, given that the nature or contents of the file have not been modified.

III. RELATED WORK

The application of blockchain in the maritime industry and shipping container supply chain is still in its infancy due to the low maturity of blockchain, the lack of implementation experiences, and the complexity of the shipping sector. Thus, very limited literature is available on the subject. To identify research gaps and shortcomings, it is crucial to look into how theoretically found projects link to or overlap with one another and the real-world applications. This section presents the related works, both conceptual as well as implemented frameworks in the supply chain industry.

Saberi et al. [10] discussed the advantages that blockchain technology can bring to building a sustainable supply chain and the possible barriers that currently discourage the integration of blockchain into the global supply chain. These barriers are categorized into inter-organizational, intra-organizational, technical, and external barriers. The application of smart contracts in the transfer of ownership, handling payments, and encouraging transparency in the information flow are also comprehensively discussed. The authors in [5] detail the application of blockchain in the maritime supply chain. The paper points out the shortcomings of the current management of the shipping process and how blockchain can reconstruct the inefficient and restricting management into a system that enhances trust between the stakeholders. They also discussed the use of blockchain in digitizing paperwork, container tracking and tracing, and finally customs clearance procedures, despite the technology’s main drawbacks of standardization and scalability.

Tsiulin et al. [15] reviewed various applications of blockchain in the shipping industry. In their report, they examine the practicality of incorporating blockchain into the existing supply chain workflow. The authors conclude that the examined papers fall into one of three categories: documentation process, financial process, or device connectivity, and that most of these applications highlight and address similar problems. The aforementioned research works mostly deal with the theoretical aspects of integrating blockchain with supply chains and logistics areas.

The authors in [6] conducted a comprehensive review of notable examples aiming to digitalize shipping processes using blockchain, among which are TradeLens and Blockchain Documentation Transaction System (BDTS). TradeLens is a joint blockchain solution between Maersk and IBM that employs a secure and automated documentation process [16]. The BDTS, a platform developed by CargoX, also allows its users to securely share documents, either individually or in bulk. It enables Smart B/L, a blockchain-based bill of lading transfer and sharing system. In light of the examples, the authors also highlight the significant improvement in process efficiency and security provided by blockchain. These frameworks, however, are mainly directed toward the documentation procedures of the shipping process.

To minimize the side effects of food contamination, such as endangering the livelihood of consumers, business disruptions, and increased losses, Walmart partnered with IBM to implement a blockchain-based system to track and trace products from their originating place until they finally reach consumers [17]. The system uses Hyperledger Fabric with IBM’s plug-and-play features for efficient data recording. Evergreen also utilizes blockchain technology to track the provenance of diamonds and their movement from mines to stores [18].

Hasan et al. implemented a smart contract-based shipment tracking framework that monitors and manages the shipment conditions, automates payments, and notifies participants in the event any of the agreed-upon conditions are violated [19]. They use an IoT device-equipped container that continuously communicates with an MQTT server by pushing the sensor measurements with which the conditions are compared. The server aggregates, stores, and publishes all sensor data generated by the IoT sensors installed in the shipment. Similarly, Alkhooori et al. [20] introduced CryptoCargo, a blockchain-powered container that continuously monitors the container for any violations that may endanger its contents. The violations, if any should occur, are recorded on the decentralized ledger via smart contracts. These secure transaction records, however, do not address the end-to-end stakeholders’ communication in the shipping procedure.

The Ethereum blockchain and smart contracts were used in the automation of port operations by digitizing the paperwork in ports in [4]. The authors created a DApp that enables users to issue the cargo agreement document, reducing the paperwork workload in the shipping process. PiChain, a conceptual framework utilizing the PI and blockchain technologies for smart ports, is introduced in [21]. The authors optimize the container route through AI algorithms at the start of the process while automating the payments through smart contracts.

Komathy discussed a decentralized shipping system in [22]. All business operations related to cargo transportation are integrated and connected to each other through the framework. Suppliers, shipping firms, banks, insurers, and other parties involved in the transaction are represented as
nodes and are able to transparently access the transaction data. Moreover, Toyoda et al. [23] presented an ownership traceability solution to prevent counterfeiting in the post-supply chain. The solution employs the idea of “proof of possession of balance,” where bitcoin balance is used as proof of ownership following 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.

The discussion of related works reveals the gap in the literature and the proposed solutions. The frameworks and discussions concentrate on either the documentation procedures or product and shipment traceability and fail to address end-to-end communication among stakeholders, which is a crucial part of a smooth shipping process.

IV. PROPOSED SOLUTION

The section presents the details of the proposed solution. In the downstream supply chain flow, Ethereum smart contracts are used to manage shipping container processes. The solution promotes transparency by enabling the concerned parties to track the real-time progress of the container. Figure 3 provides an overview of the proposed solution. The stakeholders communicate with the underlying smart contract via a decentralized application. The necessary documents are stored in IPFS, and each step of the process is recorded on the blockchain.

A. COMPONENTS AND STAKEHOLDERS

We considered six participating entities. Figure 4 depicts the relationship between the entities and the smart contracts. Each participant is required to register through the Register contract, thereby obtaining an Ethereum Address (EA). This address is a public key used for signing transactions and documents digitally. The participants are detailed below.

- Exporter: Often referred to as the consignor. This is the party that intends to export goods out of the country.
- Importer: Also known as the consignee. This is the party that intends to import goods into the country.
- Freight Forwarders: Parties at both the source and destination who plan and coordinate the procedures and documents required to ship cargo. They assume responsibility for the container in the shipping process.
- Customs Administration: This is the office that deals with export or import taxes and duties of the cargo. This
party collects the taxes and duties and occasionally scans the outgoing and/or incoming cargo.

- **Shipping Line**: Also known as a sea carrier is the carrier that transports goods via a cargo ship.
- **Inland Carrier**: Party whose purpose is to transport the container to and from the origin port and destination port respectively. This party represents both trucks and rails.

For uniformity, the parties involved are referred to as the exporter, importer, freight forwarder, customs officer, sea carrier, and inland carrier, respectively. Each shipping container is represented by a smart contract referred to as the ContainerShipment contract. The deployed smart contracts include the following major components:

- **Variables**: These are data items used to store the details of the container and the addresses of the participating parties.
- **Functions**: Modules of code that automate the workflow of the process. In our case, these functions are used to verify documents, request carriers, request container hand-offs to the next carrier, etc.
- **Events**: Messages broadcast as a result of smart contract function execution.
- **Modifiers**: Special functions used to modify function behaviors. They define restrictions and limit permissions required to execute functions. Using these components, we are able to differentiate permissions between the involved stakeholders.
- **User-defined data types**: Data types such as struct and enum enable us to define a collection of variables of dissimilar data types and named values, respectively. Such data types are used to represent the transportation modes and the states of the container, respectively.

### B. PROCESS FLOW

Figures 5 - 7 detail the events and flow of actions involved in the shipping process. In these diagrams, the set of events and functions, as well as the information and container state update, are illustrated. Each stakeholder is required to register using the Register contract.

The entire shipping process is initiated by the exporter placing a request through the `requestShipment()` function. This request results in an event broadcast on the blockchain network notifying the stakeholders of the details of the request, including origin, destination, content, size, and the receiver’s address. Based on the details obtained, the freight forwarder then arranges either unimodal or multimodal transportation. In the case of a unimodal shipment, as illustrated in Figure 5, the freight forwarder verifies the required documents, creates a shipment instance, and issues a bill of lading by executing the `documentVerification()`, `createUnimodalContainerShipment()`, and `issueBoL()` functions respectively. Customs clearance is then requested, and approved by a customs officer. Finally, the container is loaded onto the inland carrier for highway haulage to the destination warehouse.

On the other hand, if the requested shipment requires ocean haulage, the freight forwarder creates an instance of a multimodal container shipment using the function `createMultimodalContainerShipment`. After verifying the required documents, as shown in Figure 6, the bill of lading is issued using the same function as in the unimodal shipment.
The issued bill of lading, in both unimodal and multimodal shipments, is stored in IPFS. The CID of the uploaded document is stored in the blockchain for the stakeholders to view and validate when necessary. When `issueBoL()` is executed, it also calls the `inlandCarrierRequest()` function, thereby initiating the transportation process. Following the carrier request, the next carrier approves by invoking the `approveCarrierRequest()` function. This process continues in a loop for the number of carriers in export haulage. If the approving carrier is not an authorized carrier, the function broadcasts an error event and reverts the approval request, as shown in Figure 6.

Upon reaching the origin port, the container is handed off to the port operator via the function `containerHandoffRequest()`. This function initiates export customs clearance and updates the state of the container to restrict further updates unless declared customs cleared. In the case of a direct shipment, the container is loaded onto the vessel and transported to the destination port when export customs are cleared. However, if one or more transshipments are required, the container is transported to the intermediary port where a transshipment permit is issued using the `transshipmentPermit()` function invoked by a customs officer on the port. The container is transferred to the next sea carrier when a transshipment permit is issued. The complete documentation request and approval processes are described in Figure 7. The implementation also considers events where an unauthorized participant attempts to issue or approve a document request, indicated by an error event broadcast to the network and request rejection. In the destination port, the same process as the export haulage is repeated. Finally, the importer can confirm the delivery using the `shipmentReachedDestinationSignal()` function, which notifies all the stakeholders of the safe arrival of the container to the intended destination.

V. IMPLEMENTATION DETAILS

This section presents the algorithms along with their full implementation details. The code is written using the Solidity language on a web-based compiler known as Remix [24]. Using compiler 0.8.7 and remix version 0.28.1. Each of the stakeholders is assigned an Ethereum Address (EA) and can
invoke functions in the smart contract depending on the permissions set on the function. The permissions for function execution are enforced using modifiers.

The `requestShipment()` function, described in Algorithm 1 is executed by a party registered as an exporter. A struct object, as described in Figure 4, is initialized with details such as origin and destination places, size of the required container, type of contents, and the receiver of the shipment. This request sets the state of the container to `Requested` which is a requirement for the verification of the required documents in Algorithm 2.

Algorithm 2 execution depends on the invoker’s role. If it is executed by a customs officer or broker, then it is used to verify documents required for export and import customs clearance. These documents include a bill of lading and a commercial invoice, among others, and vary depending on the countries of origin and destination. However, if invoked by the freight forwarder, the algorithm is used to verify
the required shipping documents such as the packaging list, commercial invoice, and certificate of origin. When the documents are in order, the freight forwarder then proceeds with the shipment creation detailed in Algorithm 3. Upon satisfying the requirements, the freight forwarder or the customs office approves the process, thereby updating the state of the container.

Following the document approval in the export haulage stage, Algorithm 3 is executed. The algorithm presents the functions used for shipment instance creation. Depending on the route requirements, the freight forwarder creates a shipment with either of the functions described in Algorithm 3. For instance, if the shipment is unimodal, the only input required is the number of highway haulage trucks that will transport the container to the destination. If, on the other hand, the shipment is multimodal, the container details such as the origin and destination ports, transshipment status, the number of vessels, and the number of trucks in the export haulage and import haulage are set.

Algorithm 4 encompasses three document-issuing functions. The functions are executed depending on the caller and the state of the container. Following the creation of the shipment instance, a bill of lading is issued by the freight forwarder using the function \texttt{issueBoL} described in Algorithm 4. This function then forwards a request to an inland carrier by invoking the function \texttt{inlandCarrierRequest}() described in Algorithm 5. If the container state is set to customs clearance request, the \texttt{customsClearance} function is used to issue the export or import customs clearance of the shipment. If, on the other hand, the container state is set to transshipment permit request, the customs officer in the intermediary port will approve and issue a transshipment permit using \texttt{transshipmentPermit}(). These functions are executed after a certain document, such as customs clearance or

\begin{figure}
\centering
\includegraphics[width=\textwidth]{sequence_diagram.png}
\caption{Sequence diagram showing the information flow in multimodal transport with a bill of lading, customs clearance, and transshipment permit issuance.}
\end{figure}
Algorithm 1 Algorithm for Initiating a Container Shipment Request

| Line |
|------|
| 1    |
| 2    |
| 3    |
| 4    |
| 5    |
| 6    |
| 7    |
| 8    |
| 9    |
| 10   |
| 11   |
| 12   |

**Input:** origin, destination, size, content, receiver EA  
**Output:** Emit ShipmentRequested Event  
**Require:** caller to be registered

```plaintext
1 Function requestShipment()
2 if caller ID is not exporter then
3 caller is not authorized to execute function
4 end
5 else
6 Create a container struct
7 /*A struct with the attributes described in Figure 4 */
8 Set container struct details
9 /*Details of the container such as state, origin, and destination are updated in this algorithm */
10 cState = Requested
11 Send order request details to the freight forwarder via events using the exporter EA and container ID
12 end
```

Algorithm 2 Algorithm for Verifying Required Documents

```plaintext
1 Function documentVerification()
2 if caller == freightfrw then
3 /*freight forwarder at the start of the shipment process*/
4 Require: cState == Requested
5 cState = Ready
6 end
7 else
8 /*These are the customs offices*/
9 if cState == ExportCustomsClearanceReq OR multimodal == false then
10 cState = ExportCustomsCDocumentsApprv
11 end
12 if cState == ImportCustomsClearanceReq OR multimodal == false then
13 cState = ImportCustomsCDocumentsApprv
14 end
15 if cState == TransshipmentPermitReq then
16 cState = TransshipmentDocsApproved
17 end
18 Output: Emit RequiredDocumentsVerified Event
```

transshipment permit, is requested. The state of the container is set to the requested document, which is then updated when the responsible party approves. For example, if import customs clearance is needed when the container gets to the destination port, the container is not released to the inland carrier until the customs officer approves and the container state is updated. Following the approval and issuance of the document(s), each of these functions, depending on the process flow, places a request for an inland or sea carrier.

Algorithm 5 demonstrates the carrier request, container hand-off, and carrier approval processes and the underlying implementation that differentiates between land-ocean haulage and ocean-land haulage transitions. The inlandCarrierRequest and seaCarrierRequest functions are used to place land and sea carrier requests respectively. The approveCarrierRequest function is then used to approve these requests given the conditions of the container state are satisfied. Moreover, the containerHandoff is invoked by the transporter to hand off the container to the next carrier. This function checks the container state and triggers the next step of the process flow.

**VI. FUNCTIONALITY TESTING AND VALIDATION**

In this section, we test and validate the implementation of the proposed system. We demonstrate that the state of the container is updated correctly by the appropriate functions, thereby allowing a smooth process flow. Additionally, we execute the code to test the permissions of the stakeholders to update and read the container status, as well as verify documents and approve requests when required. All the Ethereum addresses used in this testing and validation are shown in Table 1.

**A. STATE OF SHIPPING CONTAINER**

By introducing the state of the container as a requirement before every function execution, we test the correct container state update in the process flow. The changes are broadcast as events for every stakeholder to observe. Figure 8 and Figure 9 show instances of the state update. Figure 8 highlights the event broadcast when an exporter places a shipment request through the requestShipment() function. After a shipment request is placed, the responsible freight forwarder approves the required documents using the documentVerification() function. This function required a specific shipment state value. This execution, then, updates the container state and results in an event broadcast, as shown in Figure 9.

**B. SHIPMENT CREATION**

When the freight forwarder is creating the shipment instance, either unimodal or multimodal, the details are cross-checked to be in line with the type of shipment being requested. For
Algorithm 3 Algorithm for Creating a Container Shipment

/*functions executed by the freight forwarder depending on the requirement of the shipment*/

Function createUnimodalShipment(haulTruckC)

Require: cState == Ready
multimodal = false
exportHaulageC = haulTruckC
cState = ShipmentCreated

Output: Emit ContainerShipmentCreated Event

Function createMultimodalShipment(originPort, destinationPort, exportHaulCount, isTransshiped, vesselsC, importHaulCount)

Require: cState == Ready
multimodal = true

if isTransshiped then
/*If is transshipped, more than one vessel is required*/
Require: vesselsC > 1
numOfTransshipments = vesselsC - 1
end
else
Require: vesselsC == 1
numOfTransshipments = 0
end

cState = ShipmentCreated

Output: Emit ContainerShipmentCreated Event

Algorithm 4 Algorithm for the Issuing of the Bill of Lading, Customs Clearance, and Transshipment Permit

Function issueBoL(hashIPFS)

/*executed by the freight forwarder at the start of the shipment process*/

Require: cState == ShipmentCreated
IPFShash["bill of lading"] = hashIPFS
cState = BoLIssued

inlandCarrierRequest()

Output: Emit BillOfLadingIssued Event

Function customsClearance(hashIPFS)

/*for import and export customs clearance*/

Require: cState == ExportCustomsCDocumentsApprv OR cState == ImportCustomsCDocumentsApprv

if cState == ExportCustomsCDocumentsApprv then
IPFShash["export customs"] = hashIPFS
seaCarrierRequest()
end
else
IPFShash["import customs"] = hashIPFS
inlandCarrierRequest()
end

cState = CarrierRequested

Output: Emit CustomsClearanceApproved Event

Function transshipmentPermit(hashIPFS)

/*transshipment permit for transshipment in intermediary port(s)*/

Require: cState == TransshipmentDocsApproved

IPFShash["transshipment permit"] = hashIPFS

Output: Emit TransshipmentPermitIssued Event

logs

"from":
"0x0a029345e0cE3351a8650c0d71a34d286a2833f3a",
"topic":
"0x7c3702156d83931d8e67b526f43519393c9ab23b91a6
7a345ad175b0db9d",
"event":
"RequiredDocumentsVerified",
"args":
"
"0x0a029345e0c481177ec7e8f571cecaea80e22c20db",
"0x0a029345e0c481177ec7e8f571cecaea80e22c20db",

FIGURE 9. Required documents verified by the freight forwarder, and shipment state is set to RequiredDocumentsVerified.

instance, if transshipment is set to false in a multimodal shipment, the function expects the value of vessels to be 1. In cases where the details are not correct, as shown in Figure 10, the function execution reverts, and an error event is broadcast reminding the freight forwarder to check the details of the shipment. On the other hand, when the details are set right, the function is executed successfully, as shown in Figure 11.

C. APPROVE CARRIER-REQUEST

This function is executed by the transporters only, which are identified by transporter IDs. To successfully execute the function, the state of the container and the ID of the caller are checked. Figure 12 shows a reverted function execution when the approveCarrierRequest() is invoked by a random address. Figure 13, however, shows a successful execution of the function when triggered by a registered inland transporter.

D. ISSUING DOCUMENTS

This function is invoked only by the freight forwarder and customs clearance officers and requires different container

FIGURE 10. Declined shipment creation due to incorrect details (istransshiped = false, vessels = 2).
Algorithm 5 Algorithm for a Carrier Request, Container Hand-Off, and Request Approval

Function inlandCarrierRequest()  
Require: exportHaulageC > 0 OR importHaulageC > 0
1. if exportHaulageC > 0 then
2. exportHaulageC = exportHaulageC - 1
3. end
4. else
5. Require: vesselsC == 0
6. importHaulageC = importHaulageC - 1
7. end
8. cState = CarrierRequested
9. Output: Emit RequestForInlandCarrier Event
10. Function seaCarrierRequest()  
Require: vesselsC > 0
11. vesselsC = vesselsC - 1
12. cState = CarrierRequested
13. Output: Emit RequestForSeaCarrier Event
14. Function containerHandoff(nextCarrierID, location) 
Require: cState == InlandCarrierApproved OR cState == seaCarrierApproved
15. if callerID == inlandCarrier then
16. if nextCarrierID == inlandCarrier then
17. Require: exportHaulageC > 0
18. inlandCarrierRequest()
19. end
20. else
21. Require: exportHaulageC == 0 AND vesselsC > 0
22. cState = ExportCustomsClearanceReq
23. end
24. end
25. if callerID == seaCarrier then
26. if nextCarrierID == inlandCarrier then
27. Require: vesselsC == 0 AND importHaulageC > 0
28. cState = ImportCustomsClearanceReq
29. end
30. else
31. Require: numOfTransshipments > 0
32. cState = TransshipmentPermitReq
33. end
34. end
35. Function approveCarrierRequest()  
Require: cState == CarrierRequested
36. if callerID == inlandCarrier then
37. cState = InlandCarrierApproved
38. end
39. else
40. cState = SeaCarrierApproved
41. end
42. Emit carrierRequestApproved event

states depending on the requested document. For instance, when the last inland carrier requests a container hand-off, the container state is set to ExportCustomsClearanceReq, while the state is updated to ImportCustomsClearanceReq when the container is unloaded in the destination port. These states are not updated until the responsible parties approve. Figure 14 depicts a declined execution of the customsClearance() by the customs officer due to an unsatisfied container state requirement. Figure 15, on the other hand, shows the successful execution of the function when the state of the container matched the required state.

VII. DISCUSSION
This section presents the cost and security analysis of the implemented smart contracts, as well as a comparison with the existing solutions. Furthermore, we discuss the generalization aspect of the proposed solution.

TABLE 1. Ethereum addresses of participants used for testing.

| Actors                  | Ethereum Addresses               |
|-------------------------|----------------------------------|
| Exporter                | 0x5383Da6a701e5e655545C03Fca887575b6beddC4 |
| Freight forwarder       | 0x485f093c4811777e78571ee5A8E22C026b  |
| Importer                | 0xA8843F6469C61Ed9F9b849Ae6776D33158535cb2 |
| Inland carrier          | 0x87f31D3C5ca687E134aC0F824c2a7ec18A495cbabB  |
| Sea carrier             | 0x617F261D72D7D95D50197092fa1865E7f2  |
| Customs office          | 0x17F6AD8E982297597C203069C1D3bFEB4348c372  |

FIGURE 11. Successful shipment creation with correct details (istransshiped = false, vessels = 1).

FIGURE 12. Reverted function execution when invoked by a random address.

FIGURE 13. Successfully executed function when invoked by an inland transporter’s address.
TABLE 2. Gas cost of functions in USD in public Ethereum blockchain.

| Function name                          | Transaction cost (Gas) | Execution cost (Gas) | Slow and Avg. Costs (USD) | Fast Cost (USD) |
|----------------------------------------|------------------------|----------------------|---------------------------|-----------------|
| register()                             | 53119                  | 46190                | $1.05                     | $3.89           |
| requestShipment()                      | 186334                 | 162029               | $3.68                     | $13.67          |
| documentVerification()                 | 33514                  | 29142                | $0.66                     | $2.45           |
| createUnimodalShipment()              | 61896                  | 53822                | $1.22                     | $4.54           |
| createMultimodalShipment()            | 196299                 | 170694               | $3.88                     | $14.40          |
| issueBoL()                             | 121482                 | 100836               | $2.29                     | $8.50           |
| approveCarrierRequest()               | 31622                  | 27497                | $0.63                     | $2.32           |
| containerHandoff()                    | 40296                  | 35040                | $0.79                     | $2.95           |
| customsClearance()                    | 122957                 | 102119               | $2.32                     | $8.62           |
| transshipmentPermit()                 | 121619                 | 100955               | $2.29                     | $8.51           |
| shipmentDelivered()                   | 31192                  | 27123                | $0.62                     | $2.28           |

As of the current rate on Nov 19, 2022, the gas prices for slow, average, and fast transactions are 14.3, 14.3, and 53 in Gwei respectively, as reported in Eth Gas Station [25]. According to the current Ether price of $1211, these rates equate to $0.000022, $0.000022, $0.000084 in USD, respectively.

Table 2 shows the cost of execution and transaction for all the functions in the smart contracts. The amount of gas spent per function is the same every time the function is executed. As detailed in the table, the requestShipment() and createMultimodalShipment() functions contribute to the most gas spent. However, this cost is reasonable given that these functions are executed only once in the container’s journey. Moreover, these functions significantly cut the costs incurred in the current shipment request handling processes. For instance, a bill of lading costs between $50 - $100 excluding courier service fees, which is issued by only $10 in this solution with at least an 80% decrease. Furthermore, the security aspect of storing the document in decentralized storage greatly complements the cost decrease shown in the table. However, due to the fluctuating nature of the Ether token and the gas fees in the network, these calculated values might differ when recalculated after some period of time. In the cases where the gas fees are higher, the high cost can be minimized by integrating Web 2.0 functionalities to define, initialize, and update values while broadcasting these changes into the network using smart contract functions. These operations can be carried out at a minimal or nonexistent cost. Moreover, utilizing private networks such as ZkSync is a high-cost and scalability issue mitigation alternative [26].

B. SECURITY ANALYSIS OF THE BLOCKCHAIN-BASED CONTAINER TRACEABILITY SOLUTION

Our proposed solution enables the stakeholders to transparently and securely track and trace the container throughout its journey by providing:

- **Authenticity**: The solution prevents identification attacks by requiring every participant to register and thereby own an EA. Each involved participant is
exclusively identified by their EA, which is managed by an Ethereum wallet. Every transaction has a signature to ensure its authenticity, where a message hash is signed using the sender’s private key. The signature can then be verified to ensure the appropriate EA sent the transaction.

- **Immutability:** The participants can store issued and required documents in the decentralized storage system, and broadcast their respective CIDs, thereby eliminating the risk of document manipulation, theft, and losses that paper documents face regularly. By saving the CID on the blockchain, the data is linked to the participant and immutably stored.

- **Availability:** After these documents are stored securely, the concerned parties are able to access them when necessary. The secure documentation process extends beyond the mentioned documents and stakeholders to include letters of credit issued by financial institutions, export declarations filed by the exporter, and every other required document in the decentralized storage system, the smart contracts were analyzed using Oyente and Figure 16 shows the report generated. The report indicates the possibility of an attack on the specified vulnerability using a boolean value.

In Figure 16, EVM code coverage specifies the percentage of the bytecode Oyente scanned [28]. The tool analyzed up to 96.8% and 99.2% of the code of the ContainerShipment and Registration contracts, respectively. Integer underflow and overflow refer to the errors caused when the value set to a variable is outside the range that its data type can hold [27]. The counters used in our implementation are imported from OpenZeppelin contracts [29], a library of secure and audited contracts. This library prevents overflow and underflow attacks using safeMath, a contract that validates arithmetic operations, throws an exception if the operation causes an integer overflow or underflow, and reverts such operations. Moreover, Solidity compilers of version 0.8 and above offer these functionalities. In conclusion, our implementation has proven to be safe from the aforementioned attacks.

Timestamp dependency is a security flaw associated with the use of a block timestamp for function operations [30]. In our code, a block timestamp is not used or required. Transaction ordering dependency occurs as a result of incorrect ordering of transactions by miners [31]. This is controlled through the use of enum data structure. Every function requires a specific enum value which can only be set by the preceding function. Hence, if this requirement is not met, the function is not executed, making it impossible for incorrect transaction ordering.

Finally, Re-Entrancy is a well-known and frequent attack that happens when a function is repeatedly invoked before it completes its previous executions and variable updates [32]. This is prevented by the require statements in the code. When a require condition is evaluated to true, the enum variable used in the condition is updated to a different value. Therefore, re-entrancy is impossible since a second invocation of the same function would result in the require condition.

C. SMART CONTRACT SECURITY ANALYSIS

The smart contracts developed for the proposed system are analyzed using a security analysis tool that examines the code and localizes the vulnerable points in the contracts. Because of the immutable nature of smart contracts, it is necessary to eliminate any security flaws before deploying them into the blockchain network. Even though the Remix IDE provides tools for debugging, it does not sufficiently work through the code to detect problems beyond the syntax errors, hence the necessity for additional analysis tools. These tools include Oyente, MythX, and SmartCheck, to name a few [27]. Oyente directly works on the Ethereum virtual machine (EVM) bytecode without the need to access the high-level language and generates call graphs corresponding to every smart contract [28]. Depending on the results, the vulnerable implementations can be fixed and improved.
TABLE 3. Comparison between the existing system and proposed solution in terms of performance and cost.

| Criteria                        | Existing Systems                  | Blockchain-based Proposed Solution |
|---------------------------------|-----------------------------------|-----------------------------------|
| BoL processing and transmission | Physical: 5 - 13 days Digital: 2 days | Upload: 436 ms for 16MB Retrieve: 70 ms for 16MB |
| Process approvals               | 24 hours - Days                  | 25 sec - 2 min                    |
| Communication flow              | 15 to 240 min                    | 25 sec - 2 min                    |
| Cost                            | BoL Transmission $100             | $2.29 - $8.50                     |
| Re-issuance for lost BoL        | $50 - $100                       | N/A                               |
| Delay                           | Documentation and Approvals Days - weeks | Seconds to hours                 |

N/A: not applicable

TABLE 4. Comparison between our solution and other existing blockchain-based solutions.

| Criteria                        | CryptoCargo [20] | TradeLens [6] | BDTS [6] | Hasan et al [19] | Our Solution |
|---------------------------------|------------------|---------------|----------|------------------|--------------|
| Blockchain platform             | Ethereum         | Hyperledger   | Ethereum | Ethereum         | Ethereum     |
| Permission                      | Public           | Private       | Public   | N/S              | Public or private |
| Container Traceability          | ✓                | ✓             | ×        | ✓                | ✓            |
| Secure Document Digitizing      | ×                | ✓             | BoL only | ×                | ✓            |
| End-to-end Communication        | ×                | ×             | ×        | ✓                | ✓            |
| Scope                           | National         | Both          | Both     | N/S              | Both         |
| Container SC                    | ×                | N/S           | ×        | ✓                | ✓            |

N/S: not specified

evaluating to false and reverting the transaction. In conclusion, according to the analysis report and the proofs discussed above, our contract codes are secure from these attacks and security flaws.

D. COMPARISON WITH EXISTING SOLUTIONS

We compare our solution with the existing proposed systems, which follow a common pattern, as well as existing blockchain-based approaches. Due to the shortage of available details on the similar systems discussed in this paper, the comparison is performed in two categories. Table 3 details the latency and cost comparisons between the current systems and the proposed solution. Table 4 then compares the proposed solution with other similar blockchain-based solutions and their implementations.

The responsiveness comparison of the existing systems and the proposed solution highlights the time consumed in document processing and transmission, process approvals, and communication flow in both systems. The current system analysis used as a reference is detailed in [33] and [34]. The proposed solution’s responsiveness analysis is estimated based on similar Ethereum-based implementation in [35]. Furthermore, the cost is estimated based on the cost analysis shown in Table 2 and Eth Gas Station [25].

The traditional systems handle either physical documents, which is one of the main causes of delay in the process, or electronic documents through the EDI [36]. In [6], the authors present the concept of a Port Community System (PCS) which ports use to share digital documents within the community, excluding outside actors. These electronic documents, though faster, are still prone to fraud and alterations. The comparison table focuses on the BoL document to represent the document handling procedures in the shipping process. Process approvals and end-to-end communication among the stakeholders also contribute to the overall delay. Some permits, such as customs clearance, are only issued after the cargo has undergone a physical inspection, thereby requiring processing time. However, the involvement of a large number of stakeholders and third-party actors in a given consignment lengthens the already lengthy approval processes. The delay due to pending approvals and documentation processing usually ranges from several days to weeks. As summarized in the table, smart contracts significantly minimize the latency for all the given criteria. The documentation handling, process approvals, and communication flow are compared for both systems to determine the improvement in the incurred costs. Integrating blockchain into the shipping process provides cost cuts, efficient document and process handling, and efficient stakeholder communication flow.

The comparison with other blockchain-based solutions is detailed in Table 4. Blockchain implementations by leading firms introducing TradeLens and the BDTS platforms are discussed. However, these projects focus on the documentation processes and neglect the traceability of the cargo and end-to-end communication of the involved parties. CryptoCargo and Hasan’s frameworks are also discussed. Most of these solutions target national shipping, that is, shipping within the country, and none of these solutions differentiate and address the unimodal and multimodal transportation of the cargo. In conclusion, these systems target either secure documentation or container traceability in national shipping. Our solution, on the other hand, integrates both of these features while
also addressing international shipping. Additionally, we use a decentralized data storage system to store and securely share documents among the stakeholders. This enables us to handle documentation separately without overwhelming the system or obstructing information flow. While most of the discussed solutions restrict the joining of new customers through the use of a private blockchain, our solution grants its users the choice of platform type, which is either a public, private, or consortium blockchain.

E. Generalization

A shipment process may comprise a unimodal or multimodal mode of transport. In our solution, we address both shipping modes. In unimodal logistics, the assumed route is the direct transportation of containers through the highway. In this process, there can be a transfer of cargo from one truck to another, but not to a different mode of transport. Multimodal logistics, on the other hand, uses trucks, trains, and vessels to transport freight, as illustrated briefly in Figure 1 and Figure 2. For this implementation, it is assumed that a container shipment originates from Shanghai port, is transshipped to another vessel in Hong Kong port, and finally heads to its destination, Rotterdam port. This arrangement is commonly known as the Pendulum route, a famous type of inter-range route. In this paper, we focus on the pendulum route, then scale up the implementation to accommodate any number of intermediary ports in the shipping route, thereby demonstrating that the implementation can adapt to any number of transshipments.

The implementation is designed in such a manner that allows any combination of modes of transport. Therefore, in cases where air haulage is involved, this development can be incorporated without major modifications in the code. Moreover, in multimodal shipping, we assume that a freight forwarder company arranges the transportation in the exporter’s stead. However, this system also accommodates the shipping process arranged by either the exporter or the shipping line company.

In a private blockchain model, the deployer would have to specify certain constraints and make minor adjustments to appropriately handle the deployment case. We focused on the more general blockchain implementation, mainly the public blockchain, but the private blockchain would be faster and more cost-effective primarily due to its smaller size. However, this puts security at risk if proper policies are not implemented. The main concern would be the access control policies that determine the participants in the network and their authority. Furthermore, within the private network, several methods could be implemented to bolster its security, including determining the consensus algorithm, the authority of each participant, the inclusion of reputation, incentive and penalty mechanisms, and granular access control [37].

Our proposed approach is generic enough to be adopted by various industries and use cases with minimal modifications. A change to the participating parties can be made depending on the system requirements. Figure 3 can be used as a base template to highlight what modifications have to be made to the proposed solution in order for it to be utilized in a given industry.

VIII. Conclusion

In this paper, we have proposed a blockchain-based solution to provide traceability of shipment containers in both unimodal and multimodal logistics in a decentralized, transparent, reliable, auditable, secure, and trustworthy manner. We employed IPFS to enable the involved parties to securely store and access data in an off-chain manner to overcome the limited blockchain data storage issue. We developed Ethereum smart contracts to issue and validate documents securely in an automated manner. We presented algorithms as well as their implementation, testing, and validation details. The evaluation results revealed that the proposed solution is cost-effective and minimizes the unexpected costs incurred in the current shipping process. The security analysis showed that the proposed solution is secure against security threats and attacks. The comparison results highlighted that the proposed solution is novel. The cases where higher costs might occur due to the fluctuating nature of gas prices in the Ethereum blockchain are discussed. The mitigation suggestions, integrating Web 2.0 functionalities and using L2 blockchain frameworks are also described. As a future work, we plan to deploy and test our proposed solution on the real Ethereum network and build end-to-end Decentralized Applications (DApps) with customized interfaces for each system user.

References

[1] J. H. Xiao and X. Liu, “Development of multimodal transport in China,” in Contemporary Logistics in China. Singapore: Springer, 2018, pp. 137–160.
[2] D. Song, “A literature review, container shipping supply chain: Planning problems and research opportunities,” Logistics, vol. 5, no. 2, p. 41, Jun. 2021.
[3] M. Pantina. (1999). MS Windows NT 2021 Third Quarter Supply Chain Risk Trends Analysis|CargoNet. [Online]. Available: https://www.cargonet.com/news-and-events/cargonet-in-the-media/2021-q3-theft-trends
[4] P. S. Sangeeth and K. V. Lakshmy, “Blockchain based smart contracts in automation of shipping ports,” in Proc. 6th Int. Conf. Inventive Comput. Technol. (ICICT), Jan. 2021, pp. 1248–1253.
[5] C.-S. Yang, “Maritime shipping digitalization: Blockchain-based technology applications, future improvements, and intention to use,” Transp. Res. E, Logistics Transp. Rev., vol. 131, pp. 108–117, Nov. 2019.
[6] M. Iović, M. Filipović, E. Tijan, and M. Jardas, “A review of blockchain technology implementation in shipping industry,” Pomorstvo, vol. 33, no. 2, pp. 140–148, Dec. 2019.
[7] M. Uliuer, “Blockchain 2.0 and beyond: Adhococracies,” in Banking Beyond Banks and Money. Cham, Switzerland: Springer, 2016, pp. 297–303.
[8] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, “Untangling blockchain: A data processing view of blockchain systems,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 7, pp. 1366–1385, Jul. 2018.
[9] G. A. Oliva, A. E. Hassan, and Z. M. Jiang, “An exploratory study of smart contracts in the Ethereum blockchain platform,” Empirical Softw. Eng., vol. 25, no. 3, pp. 1864–1904, May 2020.
[10] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, “Blockchain technology and its relationships to sustainable supply chain management,” Int. J. Prod. Res., vol. 57, no. 7, pp. 2117–2135, Apr. 2019.
[11] D. Vujicic, D. Jagodic, and S. Randic, “Blockchain technology, Bitcoin, and Ethereum: A brief overview,” in Proc. 17th Int. Symp. INFOTEH-JAHORINA (INFOTEH), Mar. 2018, pp. 1–6.
[12] M. H. U. Rehman, K. Salah, E. Damiani, and D. Svetinovic, “Towards blockchain-based reputation-aware federated learning,” in Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), Jul. 2020, pp. 183–188.

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

[14] J. Benet, “IPFS—Content addressed, versioned, P2P file system,” 2014, arXiv:1407.3561.

[15] S. Tsiulin, K. H. Reinau, O.-P. Hilmo, N. Goryaev, and A. Karam, “Blockchain-based applications in shipping and port management: A literature review towards defining key conceptual frameworks,” Rev. Int. Bus. Strategy, vol. 30, no. 2, pp. 201–224, Jun. 2020.

[16] M. Jovanovic, N. Kostic, I. M. Sebastian, and T. Sedej, “Managing a blockchain-based platform ecosystem for industry-wide adoption: The case of TradeLens,” Technol. Forecasting Social Change, vol. 184, Nov. 2022, Art. no. 121981.

[17] R. Kamath, “Food traceability on blockchain: Walmart’s pork and mango pilots with IBM,” J. Brit. Blockchain Assoc., vol. 1, no. 2, p. 3712, 2018.

[18] R. C. Koirala, K. Dahal, and S. Matalonga, “Supply chain using smart contract: A blockchain enabled model with traceability and ownership management,” in Proc. 9th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence), Jan. 2019, pp. 538–544.

[19] H. Hasan, E. AlHadhami, 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.

[20] O. Alkhoori, A. Hassan, O. Almansoori, M. Debe, K. Salah, R. Jayaraman, J. Arshad, and M. H. U. Rehman, “Design and implementation of CryptoCargo: A blockchain-powered smart shipping container for vaccine distribution,” IEEE Access, vol. 9, pp. 53786–53803, 2021.

[21] E. Hirata, D. Watanabe, and M. Lambrou, “Shipping digitalization and automation for the smart port,” Tech. Rep., 2022.

[22] K. Komathy, “Verifiable and authentic distributed blockchain shipping framework for smart connected ships,” J. Comput. Theor. Nanosci., vol. 15, no. 11, pp. 3275–3281, Nov. 2018.

[23] K. Toyod, P. T. Mathiopoulou, 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.

[24] Remix—Ethereum IDE & Community. Accessed: Oct. 1, 2022. [Online]. Available: https://remix-project.org/

[25] ETH Gas Station. Accessed: Oct. 11, 2022. [Online]. Available: https://ethgasstation.info/

[26] 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. 1–12.

[27] S. S. Kashwaha, 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.

[28] 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.

[29] OpenZeppelin. Accessed: Oct. 1, 2022. [Online]. Available: https://www.openzeppelin.com/

[30] S. Badruidkola, R. Dantu, Y. He, K. Upadhyay, and M. Thompson, “Making smart contracts smarter,” in Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC), May 2021, pp. 254–269.

[31] M. D. Angelo and G. Salzer, “A survey of tools for analyzing Ethereum smart contracts,” in Proc. IEEE Int. Conf. Decentralized Appl. Infrastructure. (DAPPCON), Apr. 2019, pp. 69–78.

[32] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, “ReGuard: Finding reentrancy bugs in smart contracts,” in Proc. 40th Int. Conf. Softw. Eng. Companion, May 2018, pp. 65–68.

[33] N.-P. Abdellatif, “An Ethereum bill of lading under the UNCITRAL MLET,” Maastricht J. Eur. Comparative Law, vol. 27, no. 2, pp. 250–274, Apr. 2020.

[34] L. Li and H. Zhou, “A survey of blockchain with applications in maritime and shipping industry,” Inf. Syst. e-Bus. Manage., vol. 19, no. 3, pp. 789–807, Sep. 2021.

[35] P. Kostamis, A. Sendros, and P. Efraimidis, “Exploring Ethereum’s data stores: A cost and performance comparison,” in Proc. 3rd Conf. Blockchain Res. Appl. Innov. Netw. Services (BRAINS), Sep. 2021, pp. 53–60.

[36] S. K. Paik and A. Gharahgozli, “Teaching case: Electronic data interchange in port information systems,” J. Inf. Technol. Case Appl. Res., vol. 2022, pp. 1–18, Mar. 2022.

[37] R. Neisse, G. Steri, and I. Nai-Fovino, “A blockchain-based approach for data accountability and provenance tracking,” in Proc. 12th Int. Conf. Availability, Rel. Secur., Aug. 2017, pp. 1–10.

FERUZ KIFLAY ELMA 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. She is also a full-time Researcher and a Graduate Student with the Department of Electrical and Computer Engineering, Khalifa University, where she is also a Research and Teaching Assistant. Her research interests include blockchain solutions in maritime shipping, supply chain, and logistics.

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 with the Department of Electrical and Computer Engineering, Khalifa University, United Arab Emirates. He has over 220 publications and three U.S. 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 is also leading a number of projects on how to leverage blockchain for healthcare, 5G networks, combating deepfake videos, supply chain management, and AI. He is a member of the IEEE Blockchain Education Committee. He served as the Chair of the Track Chair for IEEE GLOBECOM 2018 on Cloud Computing. He is an Associate Editor of IEEE Blockchain Tech Briefs.

IBRAR YAQOOG (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Malaya, Malaysia, in 2017. He is currently a Research Scientist with the Department of Electrical Engineering and Computer Science, Khalifa University, United Arab Emirates. Previously, he worked as a Research Professor at the Department of Computer Science and Engineering, Kyung Hee University, South Korea, where he completed the Postdoctoral Fellowship. He worked as a Research Assistant at the Centre for Mobile Cloud Computing Research (C4MCCR), University of Malaya. His numerous research articles are very famous and are among the most downloaded in top journals. His research interests include blockchain, mobile cloud-edge computing, the Internet of Things, big data, and computer networks. He is also serving as an editor for various IEEE, Elsevier, and Springer journals. He has been involved in a number of conferences and workshops in various capacities. He was selected as a Highly Cited Researcher for 2021 and 2022 by Clarivate-Web of Science.
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 with the Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, United Arab Emirates. His expertise is in multi-criteria optimization techniques applied to diverse applications including supply chain and logistics, healthcare, energy, environment, and sustainability. His postdoctoral research was centered 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 in the area of supply chain data standards adoption in the U.S. healthcare system. His research has appeared in top rated journals, including *Annals of Operations Research, IIEE Transactions, Energy Policy, Applied Energy, Knowledge Based Systems, IEEE Access, Journal of Theoretical Biology*, and *Engineering Management Journal*. His research interests include blockchain technology, systems engineering and process optimization techniques to characterize, model and analyze complex systems with applications to supply chains, maintenance operations planning, and healthcare delivery.

AMMAR BATTAH received the B.Sc. degree in computer engineering from Khalifa University, Abu Dhabi, United Arab Emirates, in 2019, where he is currently pursuing the degree in computer science. He is also a Researcher and a Teaching Assistant with Khalifa University. His current research interests include blockchain technologies, the Internet of Things (IoT) security, and information security.

YASSINE MALEH (Senior Member, IEEE) is currently an Associate Professor of cybersecurity and IT governance with Sultan Moulay Slimane University, Morocco. He has made contributions in the fields of information security and privacy, the Internet of Things security, wireless, and constrained networks security. He has published over 100 papers (book chapters, international journals, and conferences/workshops), 17 edited books, and three authored books. His research interests include information security and privacy, the Internet of Things, networks security, information systems, and IT governance. He is a member of the International Association of Engineers IAENG and the Machine Intelligence Research Laboratories. He received Publons Top 1% reviewer award for the years 2018 and 2019. He is the Founding Chair of IEEE Consultant Network Morocco and the Founding President of the African Research Center of Information Technology and Cybersecurity. He was the Publicity Chair of BCCA 2019 and the General Chair of the MLBDACP 19 Symposium and ICT2C 2021 Conference. He is the Editor-in-Chief of the *International Journal of Information Security and Privacy* and the *International Journal of Smart Security Technologies* (IJSSST). He serves as an Associate Editor for *IEEE Access* (2019 Impact Factor 4.098), the *International Journal of Digital Crime and Forensics* (IJDCF), and the *International Journal of Information Security and Privacy* (IJSIP). He is a Series Editor of *Advances in Cybersecurity Management* (CRC Taylor & Francis). He was also a Guest Editor of a Special Issue on Recent Advances on Cyber Security and Privacy for Cloud-of-Things of the *IJDCF* (Volume 10, Issue 3, July-September 2019). He has served and continues to serve on executive and technical program committees and as a Reviewer of numerous international conferences and journals, such as *Ad Hoc Networks* (Elsevier), *IEEE Network* magazine, IEEE *SENSOR JOURNAL*, *ICT Express*, and *Cluster Computing* (Springer).