Received 27 June 2022, accepted 1 August 2022, date of publication 16 August 2022, date of current version 9 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3199007

RESEARCH ARTICLE

On the Integration of Blockchain With IoT and the Role of Oracle in the Combined System: The Full Picture

ALIA AL SADAWI[1], (Member, IEEE), MOHAMED S. HASSAN[2], (Member, IEEE), AND MALICK NDIAYE[1]

[1]Engineering Systems Management Program, Industrial Engineering Department, American University of Sharjah, Sharjah, United Arab Emirates
[2]Electrical Engineering Department, American University of Sharjah, Sharjah, United Arab Emirates

Corresponding author: Mohamed S. Hassan (mshassan@aus.edu)

This work was supported in part by the Open Access Program from the American University of Sharjah.

ABSTRACT The remarkable increase in the number of interconnected smart devices in today's Internet of things networks introduces more challenges related to security, trust, and centralization, which require more effective solutions. Fortunately, blockchain technology has recently emerged as a potential rescuer for IoT-based solutions due to its decentralization and enhanced security features. It is usual for smart contracts to arise in handling and processing the generated data when IoT devices are combined with blockchain. However, blockchain and smart contracts need to interact with input data of the same level of trust to guarantee correct applications execution. This implies using oracles to provide trust compatibility between inserted information collected from IoT devices and blockchain and smart contracts. Therefore, this study adopts a methodology that was shaped based on current literature and design and experiments to provide a full narrative of the process of combining two of the most intriguing systems in today's world of technology, namely, blockchain and IoT including a very important part of the comprehensive system, viz. blockchain oracle. Moreover, it was found that the literature lacks a complete view of the IoT-blockchain integration process that covers all its important and related aspects. Therefore, this work is an attempt to fill the gap in literature and contribute to the body of knowledge by surveying the literature about existing IoT-blockchain architectures and shed light on the role of blockchain in addressing IoT issues while demonstrating the concept of oracles as well as their functions in addition to the main operating blockchain oracles. Additionally, this work illustrates a CO_2 measuring use case where a smart contract is developed and tested as part of two proposed oracle-based designs. The obtained results demonstrate a full picture of a practical integrated IoT-blockchain system architecture.

INDEX TERMS Blockchain, challenges, data authentication, enhanced performance, Internet of Things, integration, oracle, smart contract, trust.

I. INTRODUCTION

The internet of Things (IoT) provides everyday objects with sensing, identifying, processing and networking capabilities that enable them to communicate with one another and with other devices and services over the Internet to accomplish a desired set of objectives [1]. The strength of the IoT paradigm lies in its remarkable effect on users' behaviors which comes as a result of simplifying different aspects of their daily lives on different fronts. In particular, IoT technology influences the domestic and working experiences of its users in ways that enhance their living conditions. IoT-based systems promise to play a major role in e-learning, e-health and many more now and in the near future. As an example, for business users, advancements in IoT technology should improve automation and industrial manufacturing, process management, logistics, intelligent transportation systems (ITS) for people and goods, etc. [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Ibrar Yaqoob.

The interaction of IoT smart devices is expected to generate considerable data flow between stakeholders, especially in large-scale networks, which requires establishing trusted data sharing among the different participants [3]. This imposes a serious challenge on the performance of IoT-based systems. Also, the employment of IoT in the daily operations of private and business sectors raises many concerns about data security since information needs to be protected at all levels of the IoT ecosystems. While, achieving data security is relatively complex in general [4], [5], security in IoT -based solutions are not straightforward due to heterogeneity of the underlying devices, systems dynamics, unprotected environments, and the large-scale applications networks [4]. Unfortunately, other issues related to the high operational and maintenance costs of the IoT ecosystem [6] and the centralized structure of IoT networks additionally escalate these challenges and create the issue of a single point of failure, as well.

While the IoT is growing in popularity and becoming one of the biggest trends in modern technology, blockchain is slowly keeping up. Blockchain has many distinguishing features that enable providing trust and combat security breaches in IoT systems. Combining blockchain and IoT can elevate several industries such as the pharmaceutical industry, automotive industry, water management, supply chain and logistics [7]. Blockchain technology is capable of communicating with diversified IoT devices and supporting integrated IoT-blockchain automated architecture to perform efficient and safe data sharing and operations [8]. The automation part of the suggested combined IoT-blockchain system is fulfilled using smart contracts, which reside within decentralized blockchains and are written in computer programs that are automatically executed when predefined terms are met without the intervention of a third party. Smart contracts consist of transactions that are stored and replicated in the blockchain's ledger. Smart contracts extend blockchains' utility from simply keeping a record of financial transactions to automatically executing multi-participants agreement conditions [9]. Those agreements could be identification mechanisms of IoT entities, IoT devices services payments, peer-to-peer IoT messaging scheduling, or any other IoT system operations. In addition to the automation aspect, smart contracts performing these IoT system operations have the advantage of reduced risks of errors and manipulation. However, automating IoT operations using a combined IoT-blockchain system faces an abstract where the data generated by IoT devices cannot be injected directly into the smart contract but rather through trusted oracles. During practical implementations, oracles form the only means for blockchain to communicate with the real world. They are used to ensure trust in the collected IoT data making it eligible to be entered into the blockchain and get executed by smart contract to fulfill certain objectives. Blockchain oracles come in different architectures and designs [10] as will be demonstrated in this research.

From reviewing the literature, it was found that researchers acknowledged the benefits of integrating IoT with blockchain to enhance IoT-based applications' performance [11]. In this paper, we survey the different IoT-blockchain frameworks and architectures in the literature to enhance further exploration and understanding of possible integrated system forms, which shall in turn assist in choosing the proper combined system design for different applications. To the best of our knowledge, current review articles in the literature do not cover all issues related to the IoT paradigm nor reveal the full potential of blockchain and its capability to elevate IoT-based systems. In addition, they do not demonstrate all types of possible combined IoT-blockchain system architecture. Furthermore, this article explains the blockchain oracle concept and what is known as "the oracle problem". It also reviews the present oracles systems available in the business market. Additionally, this study discusses the impact of oracle blockchain on the performance and efficiency of IoT systems once they are combined with blockchain and smart contracts. Despite their heterogeneous nature, nearly all blockchain applications suffer from the oracle problem. Nevertheless, the adverse impact of the oracle problem differs based not only on the nature of the business case but on involved stakeholders, as well [10]. It is also noted that oracles and their role in smart contracts are not thoroughly addressed in the literature and hence leaving results and implications questionable [10]. Therefore, this study draws a full picture for combining IoT and blockchain that includes different design forms as well as the oracle layer that connects IoT devices with blockchain and smart contracts. Moreover, this work demonstrates a use case design of a carbon measuring system as part of a carbon pricing approach using IoT devices (carbon meters) combined with a smart contract that operates on top of the Ethereum blockchain in two different oracle-based architectures. Although many papers dealt with the integration of IoT- and blockchain systems while others discussed the oracle concept and its designs, to the best of our knowledge, the literature still lacks such an extensive review that incorporates IoT, blockchain and oracle combined technologies. Yet, there is no comprehensive study in the literature that covers both topics while providing a complete vision of the process of merging the IoT and blockchain technologies in addition to demonstrating a use case offering two oracle-based designs involving a smart contract implementation, testing and evaluation. This study contributes to the body of knowledge in the following ways:

1) Explain blockchain architecture and layers as well as the concept of smart contract and lifecycle to understand their values and cornerstone role in an integrated system with IoT.
2) Highlight the main challenges facing existing IoT networks and discuss the roles of blockchain and smart contracts in addressing them.
3) Review integrated IoT-blockchain systems frameworks and architectures available in the literature and summarize promising designs and forms to provide a vision of possible ways of integration.

4) Explain the concept of blockchain oracles and discuss the oracle problem and the proposed oracle systems in the literature and industry focusing on linking the outside world with blockchain by facilitating the insertion of trusted IoT devices' readings into the blockchain.
5) Additionally, this paper discusses the impact of oracle blockchain on IoT system performance and efficiency in an integrated IoT- blockchain, especially when using a smart contract to elevate combined performance.
6) Demonstrate a use case example of a carbon measuring system based on integrating IoT devices (carbon meter) with blockchain including a smart contract and adopting two different oracle types (hardware and software oracles) resulting in two system designs.

The rest of the paper is organized as follows. Section 2 explains blockchain technology structure, layers, and smart contract that resides on it to provide the automation feature. Section 3 demonstrates IoT challenges and blockchain as well as smart contract roles in addressing them. The surveyed literature articles are summarized in section 4. Section 5 defines the research methodology. Section 6 presents the oracle concept, related problems and reviews its available network designs. Section 7 demonstrates the carbon pricing use case, the two suggested oracle designs and the associated smart contract implementation and testing. Finally, the paper is concluded in section 8.

II. BLOCKCHAIN TECHNOLOGY

Blockchain is an immutable, secured and decentralized information recording system. It stemmed from merging several existing multi-disciplinary sectors including software engineering, distributive and decentralized computing, cryptographic science and economic game theory [12], [13], which shows how well related it is to IoT systems. It functions at the intersection of the above-mentioned fields to provide its distinguished characteristics of digital assets security, software infrastructure and peer-to-peer networking. Furthermore, blockchain protocol provides economic incentives for the network participants to discard malicious transactions and act honestly [13]. Therefore, the novelty of blockchain as an adopted approach is the combination of multiple technologies [14] as demonstrated in Fig. 1.

The spread of blockchain technology greatly resembles that of the Internet. Specifically, it is observed that the blockchain protocol is following the same steps with which the TCP/IP-based Internet has revolutionized the way businesses used to function [12], [15].

A. BLOCKCHAIN FEATURES AND ADVANTAGES

In today's world, central authorities and third parties are trusted to provide security and privacy for digital assets. Also, they are assumed to remain honest and effective throughout the verification process. Unfortunately, the reality is that those third entities could be hacked, compromised, or manipulated at any time or level. Consequently, the whole system could be jeopardized [13], [16]. This is where blockchain

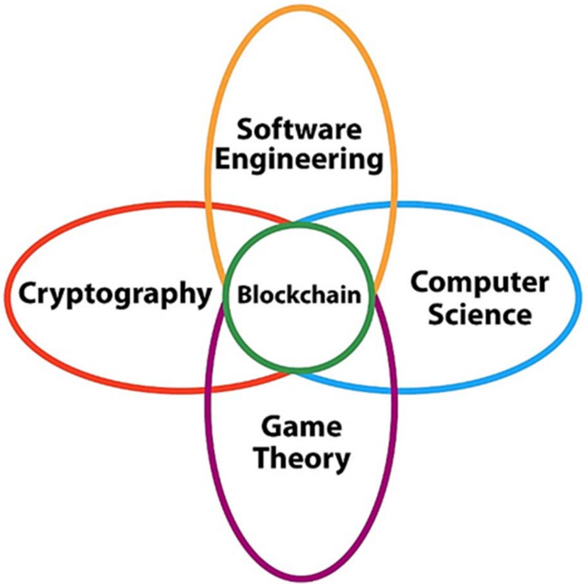


FIGURE 1. Multidisciplinary blockchain foundation.

characteristics promise as a permanent solution. Therefore, the core innovation of blockchain lies in its ability to verify and validate transactions publicly in a secured yet private manner and store them immutably in a chronicle order where all transactions could be easily traced without intermediaries. This, clearly, creates a trusted, reliable and robust system.

B. BLOCKCHAIN STRUCTURE

A blockchain is a flat network without a hierarchy [17]. Therefore, when a new block is generated, it is added to the end of a blockchain where it gets verified and stored linearly in chronological order [18]. A Blockchain network consists of nodes, each with a copy of the ledger [16]. A node is any entity connected to the blockchain [19]. All connected nodes are regularly synchronized to ensure that the same transaction is globally shared [18]. Therefore, a blockchain is owned and controlled by no one, yet it is shared and monitored by everyone [20]. Special nodes called miners are responsible for creating and validating blocks independently then adding them to the chain. The authentication and validation process is performed by miners. This process is called mining in which different type [19].

Researchers proposed an organized architecture to categorize the complexity of blockchain technology consisting of five layers [21] as shown in Fig. 2:

1) THE APPLICATION LAYER

This layer contains the applications used by the end-users such as cryptocurrency wallets, smart contracts, and decentralized applications (DApps) [22]. It consists of two sublayers: the presentation layer, which includes APIs that user interfaces use to connect the application layer with the blockchain network, and the execution layer, which includes

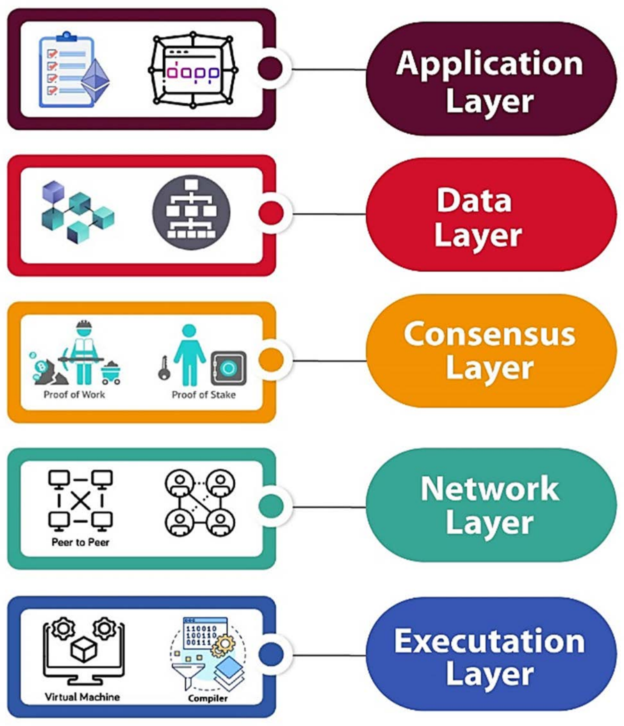


FIGURE 2. Blockchain technology layers.

smart contracts and (DApps) [21]. This layer is supported by all the lower layers since it is responsible for presenting the end results executed from the distributed ledger system [22].

2) THE DATA LAYER

In this layer, multiple data-related concepts are utilized such as transaction models, data structure, Merkel trees, hashing, and encryption algorithms in the form of digital signature and asymmetric key pair [22].

3) THE CONSENSUS LAYER

This is the core of any blockchain because it sets the rules all nodes need to follow to reach an agreement [22].

4) THE NETWORK LAYER

A peer-to-peer network performs peer discovery, transactions, and block propagation. It is responsible for providing speed and stability [22].

5) THE EXECUTION LAYER

This layer is responsible for executing smart contracts and/or low-level machine codes using a runtime environment installed on network nodes. For instance, the Ethereum blockchain has its special machine language and a virtual machine (EVM) utilized to run smart contracts. The runtime environment has to be efficient in a way that produces deterministic execution results to avoid uncertainty and inconsistency of transactions on all nodes. A transaction abortion due to inconsistent execution shall decrease the performance of the blockchain and waste energy [22].

C. BLOCKCHAIN APPLICATIONS

There is a wide spectrum of possible applications for blockchain due to the important characteristics featuring this technology. In addition to bitcoin and other cryptocurrencies, industrialists suggest different practical applications that involve mainly one or more of the following six elements: assets, trust, ownership, money, identity, and contracts [12]. However, to benefit from blockchain technology, these elements have to be programmable to allow blockchain to introduce new services. The following section explains the concept and lifecycle of smart contracts, which represent the automation and programmable aspect of blockchain.

D. SMART CONTRACT

The cryptographer Nick Szabo was the first to define a smart contract as "a set of promises, specified in digital form, including protocols within which the parties perform on the other promises" [23]. However, this concept was never implemented until the emergence of blockchain technology. In blockchain, a smart contract is defined as "a self-executing code on a blockchain that automatically implements the terms of an agreement between parties" [24]. A smart contract is not only an automatically executed program but rather a participant in the blockchain system by responding to received messages and getting, storing and sending data values. From a technical perspective, a smart contract is a web server that is not built on the internet but rather on the blockchain, therefore, specific contract applications can run on this server [25].

E. SMART CONTRACT LIFE CYCLE

The life cycle of a smart contract contains four main phases as follows. The creation of the smart contract, freezing of the smart contract, execution of the smart contract, and the completion of the smart contract [26] as shown in Fig. 3.

1) THE CREATION PHASE

The creation phase is divided into two phases: an iterative contract negotiation phase and an implementation or deployment phase. In the beginning, participating parties agree on the content and objectives of the contract. All parties should have a pseudonymous identifier in a wallet on the underlying blockchain platform for identification and transfer of funds purposes [27]. The next step is turning the agreement into a code depending on the expressiveness of the smart contract coding language [28]. After that, the smart contract code is submitted to the blockchain during the publication phase where participating nodes receive it as part of a transaction block. Once the block gets confirmed by the majority of nodes, the contract becomes ready for execution and thus is deployed to platforms on top of a blockchain where all parties can access it through this blockchain [9], [28].

It is worth mentioning that smart contract resides on the immutable blockchain and thus cannot be modified after

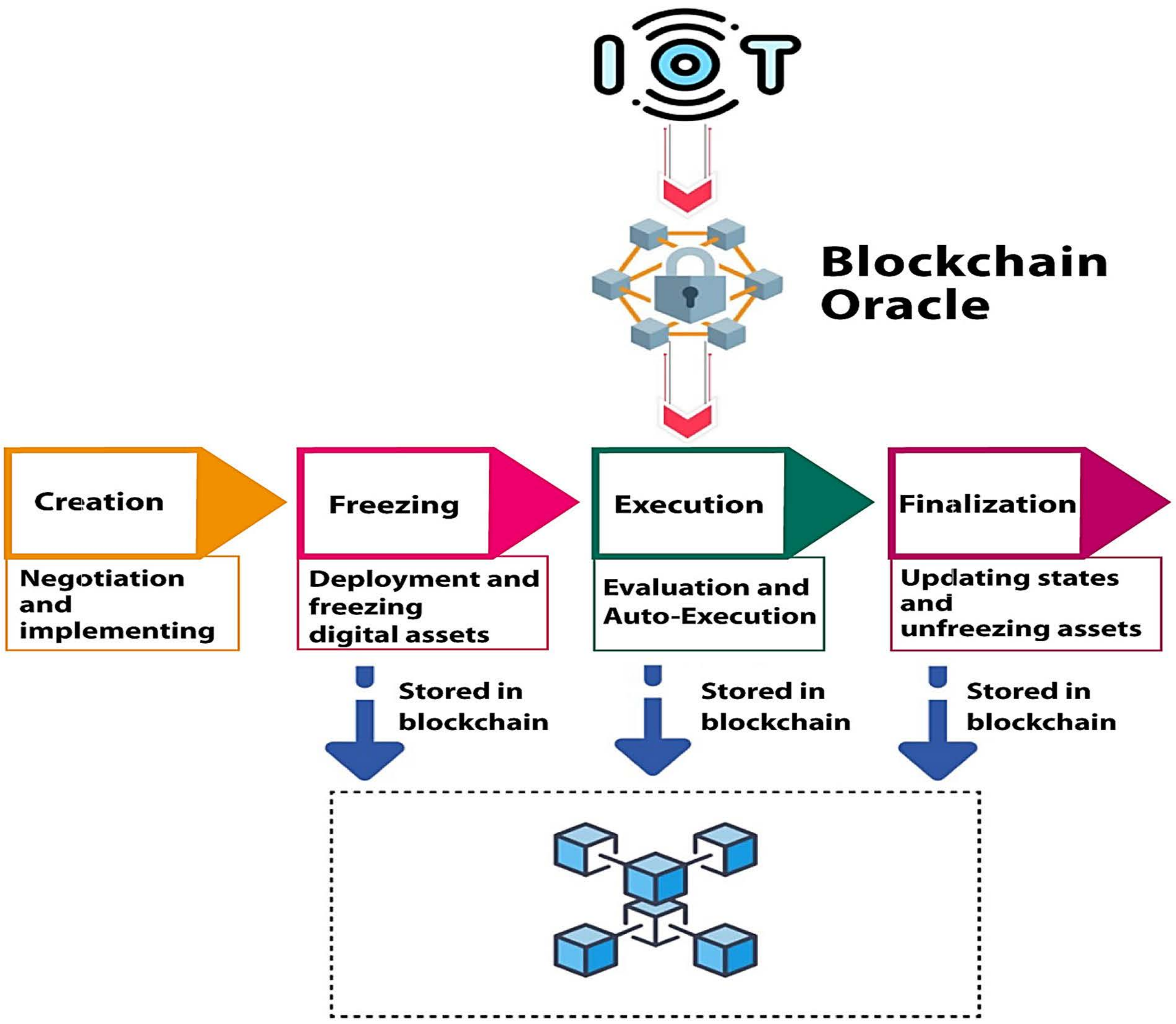

IOT
Blockchain Oracle
Creation
Negotiation and implementing
Freezing
Deployment and freezing digital assets
Execution
Evaluation and Auto-Execution
Finalization
Updating states and unfreezing assets
Stored in blockchain
Stored in blockchain
Stored in blockchain

FIGURE 3. Smart contract life cycle.

being deployed and added to a block. Consequently, a new smart contract has to be created [28].

2) THE FREEZING PHASE

After submitting a smart contract to the blockchain and after the confirmation of this contract by the majority of nodes, in exchange for this service, a fee has to be paid to miners. This shall help to prevent flooding of the ecosystem with smart contracts. During this phase, each transfer made to the smart contract wallet address is frozen and the nodes take on the governance role ensuring the predetermined conditions for executing the smart contract are fulfilled [26].

3) THE EXECUTION PHASE

The inference engines of the smart contract environment, such as compilers and interpreters, execute the functions of the smart contract. The inputs for the execution are collected from the oracles and involved parties. The smart contract execution results in a set of new transactions as well as a new state of the smart contract. [26].

4) THE COMPLETION PHASE

The results of the execution of the smart contract and the updated contract state information get stored in the blockchain after being validated using the consensus protocol. Meanwhile, the digital assets are transferred from one party to another. Consequently, the digital assets of the involved parties get unlocked. Hereby, the smart contract has completed the whole life cycle [9].

It is important to note that during the deployment, execution, and completion phases of a smart contract, a sequence of transactions is generated. Such a sequence corresponds to a statement or function in the smart contract and is stored in the blockchain. Thus, the above-described phases write data to the blockchain [9].

The smart contract features related to writing data to the blockchain that necessitate using oracles are immutability and deterministic. In more detail, smart contract immutability ensures that its code and execution transactions are permanently and unchangeably saved in the blockchain, while the deterministic component of the smart contract implies that its execution outcome remains the same for any node that runs it [29]. Therefore, oracles are utilized to fulfill the deterministic and immutability characteristics of the IoT devices' interaction transactions. Once oracles are used, the result of running any IoT operations smart contract is guaranteed to remain the same for every executing node and correctly saved in each node's immutable database.

The above discussion clarifies that the main point of failure for smart contracts is the communication channel with the real world, which is fulfilled by oracles [29]. The role and characteristics of which will be outlined later on in this research. Furthermore, although they are created by an agent on a node, smart contracts are self-owned to the extent that the contract creator has no exclusive right over them.

III. BLOCKCHAIN AND SMART CONTRACT ROLES IN ADDRESSING IOT CHALLENGES

IoT systems cannot successfully reach their full potential unless they are capable of handling the flow of massive amounts of generated data streaming from connected devices and sharing them in an efficient and trusted way. This means that IoT systems should support a trusted sharing of trusted information among trusted stakeholders which is crucial, especially for large-scale IoT applications [3]. On the ground, companies and entities involved in IoT systems usually lack trusted relationships, which creates the following three major challenges to the IoT systems [3]:

1) How to guarantee the credibility of the identities of the entities in the IoT paradigm.
2) How to ensure that the authentic data collected by IoT devices from the physical world can be inserted into the information world of IoT in an objective and true manner without being tampered with during transmission.
3) Specifically, how to maintain the credibility of entities' identity, the authenticity of data and the reliability of data transmission when a trusted third party fails to provide the expected trusted services.

It is known that IoT systems depend mostly on a central entity or a trusted third party for scheduling and decision-making processes, which results in multiple challenges such as [3]:

- The heavy and burdensome processes performed on central servers may affect total IoT system efficiency;
- Central entities represent a single point of failure, which jeopardizes the reliability of the IoT system;
- A central entity IoT system design adopts a central hierarchy where higher levels in the hierarchy are more comprehensive and lower levels are more detail-oriented. Such design is typically unattainable and costly in terms of maintenance and modification.

Blockchain could be integrated with IoT with the aid of smart contracts forming a combined and autonomous system that inherits the blockchain feature of providing trust between participants who do not fully trust each other. Additionally, blockchain oracles have the capability to address the first trust-related challenge and ensure the authenticity of IoT collected data that is fed to the combined blockchain-IoT integrated system. A comprehensive explanation of oracles is demonstrated in the following section.

The second above-mentioned challenge is resolved by using a smart contract due to its resilience to tampering and ability to automatically handle a secured and trusted entities' registration process. Additionally, smart contracts save service costs and improve the efficiency of entities' identification processes while reducing manipulation risks [9]. An IoT device in a combined IoT-blockchain system can operate a smart contract by delivering a transaction to its address, for instance, to pay for consumed IoT resources [27]. Additionally, smart contracts control and record all IoT interactions while offering a reliable and secured processing tool, which results in trusted actions. It is important to note that any device has the ability to call the functional code. This is true since on a blockchain a smart contract consists of functional codes and data with a specific address. Consequently, functions can trigger events resulting in applications that can listen to events and accordingly react to them [28]. As a result, smart contracts can securely define and model the logic that supports the underlying IoT applications [28]. Thus, connected IoT devices would send their measurement data in the form of transactions to the concerned smart contract [30], [31], [32]. Replacing third-party services with the distributed blockchain addresses the trust-related challenge and provides a robust and decentralized combined IoT–blockchain system that does not suffer from service failure issues.

Furthermore, the issues affecting the performance of IoT systems due to their central server-oriented architecture are totally addressed when integrated with blockchain. As previously explained, blockchain is a flat network without hierarchy, unlike the central entity system. Therefore, it is attainable with low maintenance and modification cost. Nodes can join and leave the blockchain anytime without affecting blockchain's overall operations and with minimal or almost no cost. This is in contrast to the case when IoT network profits do not meet its expectations, the IoT system cost seems too high [6]. While present IoT solutions are not operating as efficiently as expected, the maintenance costs of central cloud and large servers are considerably expensive. Luckily, blockchain technology can reduce these high costs through its decentralized structure and technical automating feature [30].

Moreover, the IoT-blockchain system could be reprogrammed easily by creating new smart contracts that act similar to web servers as demonstrated in the above smart contract section. Also, since blockchain and smart contracts are cryptographically secured and immutable, IoT networks can share data securely between stakeholders based on embedded and previously agreed-on terms and conditions to automate IoT devices services payment transactions, authenticate data sharing and verify the identification of participating devices. This results in a healthier IoT network because devices register and automatically validate themselves, execute contracts, and reduce the threat of cyber-attacks since there is no central system to be targeted by attackers. It is known that IoT is mainly associated with generating, gathering and analyzing data; however, less attention is being paid to automating the interactions between the things in the IoT. Since smart contracts are executable codes automating logic and actions, in the context of IoT, smart contracts could establish the foundation for novel solutions based on automated transactions triggered by sensors, actuators, RFIDS or any other IoT devices [31].

Looking at the security of IoT, the centralized architecture of existing networks makes them vulnerable to security risks [32]. Unfortunately, the current topology and resources constraints of conventional IoT systems render traditional security methods and technologies not totally applicable [33]. However, the decentralized and encrypted characteristics of blockchain are well-suited for establishing distributed and secured environment for IoT systems. Blockchain trust feature along with smart contracts enhance the trust mechanisms in IoT [6] and provide a safer and more dynamic solution [34]. Utilizing blockchain with IoT will make peer-to-peer data transactions between IoT devices more secure and private [35] and shall enhance data dispersion, encryption, and punctuality [36]. Furthermore, BaaS (blockchain as a service) management system, provides on-demand traceability, intelligent diagnostics and machine maintenance, product certification, customer-to-machine and machine-to-machine transactions, and asset registration while saving energy and cost and preventing attacks [35].

IV. LITERATURE REVIEW

Integration of blockchain with IoT systems is addressed in several papers in the literature. In what follows, we shed light on available combined IoT-blockchain frameworks and designs.

Starting with a three-layer blockchain-based architecture proposed in [11] which consists of an IoT device layer, a dew-blockchain layer, and a cloudlet-blockchain layer. This architecture utilized Dew computing, which is a modern computing model that emerged after the broad success of cloud computing. Blockchain usage usually comes in three different types: a decentralized storage database, a distributed ledger, and a supporting distributed services provided by smart contracts. Integrating blockchain with two out of three layers, precisely the dew layer and the cloudlet layer provides computation offloading, outsourced data storage, and management of network traffic. Also, a blockchain-enabled distributed framework consisting of edge, cloud and software-defined networking (SDN) was suggested in [37]. A security attack detection was incorporated at the cloud layer. Consequently, security attacks were decreased at the edge layer of the IoT system. The SDN-enabled gateway offered dynamic management of the network traffic flow. The proposed framework in [37] aimed to shed the light on the importance of integrating blockchain, edge cloud, and SDN to accomplish the needed confidentiality of the data in the IoT paradigm. Moreover, the work in [38] proposed a blockchain and smart contracts based design and prototype for edge-IoT framework called ''EdgeChain''. This work integrated a permissioned blockchain to connect the edge cloud resource pool with each IoT device's account. EdgeChain initiated its own internal currency or coin and utilized a credit-based resource management system to control the number of resources that IoT devices can obtain from edge servers. In EdgeChain, the edge servers perform the mining process since they have more resources than the resource-constrained IoT devices.

A blockchain-based collective Q-learning (CQL) approach was presented in [39] to tackle the challenges associated with combining machine learning (ML) with public IoT solutions. The research used lightweight IoT nodes to train parts of learning layers. Also, the approach used blockchain to share learning results in a verifiable and permanent manner. The study enhanced the traditional Proof of Work (PoW) consensus algorithm by regarding the learning process in the IoT node. Furthermore, an edge computing enabled mobile blockchain network was proposed in [40]. This system enables IoT devices and/or mobile users to use resources and computing services from an edge computing service provider to facilitate the operation of blockchain applications. The network prototype utilizes a workstation with Intel Xeon CPU E5-1630 as the edge computing server and Android devices as the mobile nodes. Nodes such as mobile or IoT devices run mining operations on the edge computing server. The nodes install a mobile blockchain client application capable of recording data using internal sensors such as accelerometer and GPS. Also, the BlockIoTIntelligence architecture proposed in [41] combines blockchain and artificial intelligence (AI) for IoT-based systems. This BlockIoTIntelligence architecture is divided into four intelligences: namely, cloud intelligence, fog intelligence, edge intelligence and device intelligence. The proposed BlockIoTIntelligence' design aims at demonstrating how to integrate Blockchain and AI to perform effective big data analysis while addressing security and centralization issues of IoT applications. Additionally, the research in [42] proposed a middleware for IoT applications, which facilitates the distribution of data via a blockchain while ensuring data integrity. The presented middleware supports data exchange via a second channel and enables data distribution in almost real-time. However, data integrity is not guaranteed during the on-chain data exchange. The researchers implemented their design in a fog setting, Ethereum blockchain and the Inter-Planetary File System (IPFS). A lightweight framework called FogBus is proposed in [43] with the objective of integrating IoT systems, Fog, and Cloud infrastructure to benefit from edge and remote resources. FogBus implements Blockchain to ensure the integrity of confidential data. For this reason, FogBus stores data in local repository nodes in a distributed manner rather than sending them to a cloud or any centralized data storage.

The architecture of FogBus consists of IoT devices, Fog Gateway Nodes (FGN), Fog Computational Nodes (FCN), and Cloud data centers. Also, a decentralized fog nodes reputation system combining blockchain and Ethereum smart contract is proposed in [44]. The architecture aimed at establishing trust between IoT devices and the fog nodes that are used for IoT data storage, computation, and communication with the cloud layer. The suggested decentralized integrated system solves the problem of single-point failure of existing centralized fog computing since it provides IoT client devices with the means to choose the most suitable fog nodes. The system includes the client IoT devices that evaluate the fog nodes, the fog nodes to be evaluated, and the Ethereum smart contracts governing the interaction between fog nodes and their client IoT devices.

The authors in [45] proposed a fog level integration of IoT with blockchain called DualFog-IoT. The fog level computing resources are virtually divided into Fog Cloud Cluster (FCC), which FCC communicates with cloud as in available IoT architecture and Fog Mining Cluster (FMC), which is dedicated for mining in blockchain-based applications. The suggested DualFog-IoT operates on three configurations describing the type of incoming request: Real-Time (RT), Non-Real-Time (NRT), and Delay-Tolerant Blockchain (DTB) applications. Also, the authors in [46] proposed a distributed blockchain-based cloud architecture model with fog computing and software-defined networking (SDN). The model aimed at the efficient management of raw IoT data streams at the edge of the network and the distributed cloud. Their proposed model consists of three layers: the IoT devices, the SDN controller network, which is based on blockchain for fog nodes, and a distributed cloud based on blockchain. Another work that integrated blockchain and the cloud computing technologies forming a secure and efficient smart home architecture is proposed in [47]. The architecture in [47] benefits from the decentralized nature of blockchain technology to serve the processing services and make the transaction copy of the collected sensible user data from the underlying smart home. Blockchain throughput was used to evaluate the performance of the proposed system and is also used to demonstrate that blockchain is an efficient security solution for future IoT networks. The study in [48] proposed a blockchain-based Industrial Internet of Things (IIoT) architecture. This work aimed at utilizing blockchain as a distributed ledger to keep records of all transactions in the IIoT systems. The proposed architecture divided the IIoT infrastructure into three layers: local IIoT networks, blockchain overlay network, and cloud infrastructure. The authors presented a novel blockchain storage structure that stores blocks in a hierarchical way to address the storage challenges in IIoT networks where most of the blockchain is stored in the cloud to leverage its abundant storage capacity while recent blocks are stored in the overlay network of the individual IIoT systems. The work in [49] developed an Efficient Lightweight integrated Blockchain (ELIB) model to address IoT security. The authors in [49] deployed their model in smart homes environment to test its applicability in different IoT scenarios. The presented ELIB model generates an overlay network where highly equipped resources can merge with public blockchain. Three optimizations were carried out in the proposed ELIB model, which are lightweight consensus algorithm, certificateless (CC) cryptography, and Distributed Throughput Management (DTM) scheme. The authors in [50] proposed a three-tier architecture system named IoTchain consisting of an authentication or certification layer, a blockchain layer, and an application layer. The integrated system aimed at providing identity authentication, IoT devices access control, privacy protection, regional node fault tolerance, DoS resilience, and storage integrity. The certification layer includes a certification center and a detection center whereas the application layer has regional nodes that perform mining and IoT devices management. IoTchain adopted the Practical Byzantine Fault Tolerance algorithm (PBFT) as a consensus algorithm.

An integrated IoT-blockchain architecture called Di-ANFIS was proposed by the authors of [51]. The Di-ANFIS architecture consists of six layers: the data layer, connection layer, blockchain layer, smart layer, ANFIS layer, and application layer. The authors in [51] used smart contracts to create an intelligent system for performance evaluation to securely and immutably transfer and track information in order to enhance the service supply chain management. The research in [52] proposed a method to incorporate blockchain with IoT devices to provide authentication by employing a compatible cryptographic algorithm to the IoT data prior to transmission. In the suggested structure, each IoT device is connected to a Blockchain infrastructure that is linked to the router and the cloud service through a secure interface, which enables monitoring each of the generated requests. To tackle the integrity of smart homes' IoT in addition to confidentiality and their centralization problems, the work in [53] proposed the integration of deep reinforcement learning with blockchain. In [53], the smart home architecture is based on a combination of machine learning and blockchain that consists of an IoT layer, an application layer, a blockchain layer, and a machine learning-based access layer. Furthermore, the work in [54] integrated a hyperledger fabric permissioned blockchain with the deployed IoT system to secure edge devices by using a local authentication process. The suggested system also supports generated traceability of the IoT data. The proposed model consists of base station (BS) nodes, cluster head (CH) nodes (edge layer), and IoT devices as well as off-chain storage servers to enhance the data storage of IoT devices. Moreover, the study in [55] tackled the security risk of healthcare IoT devices connected to a local area network (LAN) or wide area network (WAN). The authors proposed using blockchain in combination with fog computing (FC) to extend the services of the cloud at network edges. The system proposed in [55] consists of a three-tier FC architecture and an advanced signature-based encryption (ASE) algorithm for the verification of IoT devices in a healthcare application and the authentication of patient health data (PHD).

A new solution called blockchain-enabled edge of things (BEoT) was suggested by [56] to support low-latency and high security for IoT applications. the edge-of-things (EoT) concept emerged from the integration of edge computing and the Internet-of-Things (IoT). The proposed architecture consists of a blockchain integrated with an EoT network that consists of IoT devices and multi-access edge computing (MEC) servers. Another architecture called blockchain-enabled IoT-BIM platform (BIBP) was developed by [57] where the blockchain was integrated with an IoT and a building information modeling (BIM) platform. The research targeted off-site production management in modular construction to tackle existing (BIM) problems. Also, the study in [58] tackled some of the challenges facing IoT by integrating two emerging artificial intelligence (AI) based technologies namely, blockchain and SDN. The authors proposed a new routing protocol applied to cluster IoT networks using blockchain architecture for software-defined network (SDN) controllers. The authors in [58] argued that their study achieved efficient data analysis and energy management for the proposed system by removing the proof of work consensus algorithm. Moreover, a new energy-efficient data aggregation mechanism (EEDAM) secured by blockchain was presented in [59]. The study used IoT devices, edge computing and a blockchain integrated with a cloud server. The edge computing layer is validated by the blockchain to support secured services for the IoT layer. A research by [60] combined three important technologies namely edge computing, blockchain, and AI to develop a secure, robust platform to support AI-enabled IoT applications. The architecture used edge and IoT devices without impacting the power consumption of these devices. As a result, the suggested system guarantees continuous AI prediction and eliminates the signal point of failure which enhances the decision-making process. Lastly, [61] proposed a practical architecture that integrates IoT and blockchain to support big data analytic services. The study used Federated Learning (FL) and fuzzy hashing to provide privacy and security in the system as well as train a proposed pharmaceutical-based model locally and transmit the encrypted output to an AI service situated at the edge layer. Table 1 summarizes the work in the literature in terms of utilized technologies and the features of the resulted integrated system.

Based on the above discussion, it is concluded that systems integrating blockchain and IoT demonstrate better performance compared to standard benchmark IoT systems with no blockchain integration [62]. It is clear that the reviewed articles did not only agree on the feasibility of the integration but also presented a variety of designs to achieve it. While some researchers concentrated on the general architectural prospects needed for the integration; others focused on mitigating specific challenges. Additionally, other researchers used integrated blockchain-IoT systems as a platform to deploy certain applications. However, despite the proposed design architecture for the combined IoT-blockchain system, an oracle layer must be included to secure data feed from IoT devices to the blockchain, especially if a smart contract is utilized. In the following section, blockchain oracles are explained to draw a complete picture of a comprehensive IoT–blockchain integrated system.

V. RESEARCH METHODOLOGY

This research aims at demonstrating the whole picture of the IoT – Blockchain integration process including all its aspects. To achieve the goal of the study, a methodological approach based on literature review and design and experiment is adopted as per the following details.

1) First step: Literature review which includes surveying the most relevant and appropriate studies related to blockchain technology, integrating IoT with blockchain and blockchain oracles from reputable academic journals, international conferences and professional websites. The extracted information was analyzed and evaluated. The deduced knowledge was used to fulfill the following research objectives:
   - Explain blockchain and the structure of smart contracts, life cycle and characteristics to assess their value to the IoT ecosystem.
   - Identify the challenges facing the IoT paradigm and the vital role of blockchain and smart contracts in addressing them.
   - Survey and summarize the main integrated IoT-blockchain designs and architectures proposed by researchers.
   - Define the oracles concept, their key functions, the blockchain oracles available in the industry as well as the impact oracles have on the efficiency of the IoT – blockchain integration process.
   - Identify and select a use case that is of interest and benefit to the body of knowledge.
   - Explore the available blockchains tools and practical techniques that are useful for developing, implementing and testing an illustrative use case such as IDEs, smart contract coding languages and security analysis software.
2) Second step: Design and experiment step, which includes designing an illustrative use case that utilizes IoT, blockchain and the appropriate type of oracle based on the knowledge extracted from the conducted literature review. It also includes developing a smart contract to implement the use case using the information found in the literature as well as a suitable IDE that supports the compiling and debugging operations. The steps of developing a smart contract include creating the message sequencing based on the logic of the use case and then coding it. Lastly, this step includes testing the smart contract in terms of functionality, security and vulnerability. Step 2 fulfills the following objective:
   - Provide a better view of the IoT-blockchain integration process by illustrating a useful and beneficial use case that covers all the IoT-blockchain

TABLE 1. Researches in terms of technologies and system features.

Research work		System design	Technologies used in the integrated IoT -Blockchain system	Integrated system features	Research limitations
Al Sadawi et al, 2021	[11]	Three layers	Dew computing, Cloudlet, Practical Byzantine Fault Tolerance (PBFT) consensus algorithm	Computation offloading, outsourced data storage, and management of network traffic	General architecture, no example use case, oracle not discussed
Medhane et al, 2020	[37]	Three layers	Edge, Cloud, Software-defined networking (SDN)	Security, authenticity and data confidentiality	General Framework without a practical use case, oracle not discussed
Pan et al, 2019	[38]	Three layers	Edge, Cloud, Credit-based resource management system	Enhanced security, scalability and cost efficiency	Suggested for general IoT application, designed specifically for IoT resources provision, oracle not discussed
Qiu et al, 2020	[39]	Two layers	Edge, Cloud, machine learning (ML)	Enhance governance, reliability and safety of learning layer	The newly introduced consensus algorithm Proof of Learning (PoL) did not explain the incentive mechanism, oracle not discussed
Xiong et al, 2018	[40]	Two layers	Mobile edge computing (MEC), economic edge computing re-source management	Enable Blockchain Application in mobile services	Focused on mobile IoT devices, oracle not discussed
Singh et al, 2020	[41]	Three layers	Artificial intelligence (AI), Edge, Fog, Cloud	Perform effective big data analysis, address security and centralization issues	the architecture does not address the computational power and latency issues associated with including AI, oracle not discussed.
Krejci et al, 2020	[42]	Two layers	Fog, Inter-Planetary File System (IPFS)	Facilitate data distribution, ensure data integrity	The security issues associated with proposing a middleware layer were not investigated, oracle not discussed
Tuli et al, 2019	[43]	Three layers	Fog (Gateway and Computational Nodes), Cloud	Enhance QoS, easy to deploy, scalability, integrity and authenticity	The proposed framework is purely experimental with no reflection on practical use, oracle not discussed
Debe et al, 2019	[44]	Three layers	Fog, Cloud, Reputation system	Enable decentralized trustworthy service provisioning for IoT devices, prevent single-point failure	Oracle not discussed
Memon et al, 2019	[45]	Two layers	Segregated Fog (Fog Cloud Cluster and Fog Mining Cluster), Cloud	Decrease system drop rate, minimize cost, reduce computation load	The dual fog architecture is of certain complexity that limits applications, oracle not discussed.
Sharma et al, 2018	[46]	Three layers	Software-defined networking (SDN) enable controller, Fog, Cloud	Provide low-cost, secure, and on-demand access to IoT network, enable cost-effective high-performance computing.	Designed to mostly tackle security challenges with less consideration to other IoT issues, oracle not discussed
Singh et al, 2019	[47]	Two layers	Cloud, Inter-Planetary File System (IPFS)	Achieve efficiency, scalability, and greener smart homes	Merely addressed security issues in smart homes only, oracle not discussed
Wang et al, 2019	[48]	Three layers	Cloud, Hierarchical blockchain storage structure	Handle massive data storage issues for Industrial IoT	Lacks performance evaluation especially in terms of latency and throughput, oracle not discussed.

TABLE 1. ***(Continued.)*** **Researches in terms of technologies and system features.**

Mohanty et al, 2020	[49]	Two layers	Certificateless cryptography (CC), Distributed Throughput Management (DTM), Cloud	Security, privacy, reduce processing time, minimize energy consumption	Tested in smart home environment only, oracle not discussed
Bao et al, 2018	[50]	Three layers	Certification layer (certification center and detection center), Application layer (regional nodes for mining and IoT devices management)	Provide identity authentication, IoT devices access control, privacy protection, regional node fault tolerance, DoS resilience, and storage integrity	Lacks empirical evaluation of the proposed architecture in a real-world settings by implementing a prototype, oracle not discussed.
Bamakan et al, 2021	[51]	Six layers	Data layer, connection layer, blockchain layer, smart layer, ANFIS layer, application layer	Enhance security and trust for Service supply chain management	Conceptual model for supply chain performance evaluation, no real world and practical exploration, oracle not discussed
Šarac et al, 2021	[52]	Four layers	Device layer, home server, blockchain layer, cloud layer	Added decentralization, authentication, and versatility to IoT infrastructure	Developed for home environment, oracle not discussed
Shahbazi et al, 2021	[53]	Four layers	IoT layer, application layer, blockchain layer, access layer	Improved IoT system performance in terms of security and privacy	Proposed for smart homes, oracles are not discussed
Pajooh et al, 2021	[54]	Four layers	Base Station (BS) nodes, Cluster Head (CH) nodes (edge layer), IoT devices	Supports traceability and authenticity of IoT data	Model developed and tested using Hyperledger blockchain only, oracle not discussed
Shukla et al, 2021	[55]	Three layers	IoT layer, Fog layer, cloud layer	Enhance IoT data identification and authentication	Proposed for Healthcare sector only, oracle not discussed
P et al, 2021	[56]	Three layers	IoT devices, MEC, blockchain	Elevated low-latency and high-security for IoT applications	Lacks detailed evaluation of the proposed architecture, smart contracts and oracle not discussed
Wu et al, 2022	[57]	Three layers	Infrastructure as a Service (IaaS), Blockchain BIM as a Service (BaaS), Software as a Service (SaaS)	Tackled BIM platform single point of failure issue	Targeted construction sector only, tested the proposed system on one pilot project only without analyzing cost, oracle not discussed
Latif et al, 2022	[58]	Three layers	IoT layer, SDN-blockchain layer, cloud layer	Achieve efficient energy management for the IoT network	Smart contract and oracles not discussed
Ahmed et al, 2022	[59]	Three layers	IoT layer, Edge layer, cloud-blockchain layer	Enhanced security for the IoT, extended the wireless sensor network (WSN).	Lacks detailed security and energy analysis of the proposed system, oracles not discussed
[60]	[60]	Three layers	Processing layer (blockchain-edge equipped with AI engine), network layer (connectivity), sensing layer	Support AI prediction with high power and throughput effectiveness	Lacks real-world evaluation of the proposed architecture, smart contract and oracle not discussed.
[61]	[61]	Two layers	Blockchain layer, applications layer	Supports big data analytics	Focused pharmaceutical use case, smart contract and oracle not discussed.

integration processes from architectural design to coding and testing a smart contract.

Following the above methodological approach produced the work found in the rest of the sections in this article.

VI. ORACLES

A blockchain is a closed environment in the information world that cannot ensure the objectivity and trustworthiness of data collected from the physical world and inserted into a blockchain's smart contract [63]. As previously demonstrated, smart contracts act on data that is available on the blockchain only and cannot operate on data existing outside the blockchain. Additionally, from looking at smart contract's life cycle, the execution phase should be deterministic for its outcome results to be publicly verified by the majority of nodes in a blockchain [64]. In other words, smart contracts are independently executed by every node in the blockchain during which external data other than transaction data is repeatedly and separately retrieved by each node. Since the external data source is outside the blockchain, there is no guarantee that every node will receive the same piece of information, which might get changed or altered between requests from different nodes or might be temporarily unavailable [65]. This might lead to a situation where some nodes could detect information as if it is coming from an "untrusted" source. Consequently, and as blockchain structure is built around reaching consensus among nodes, this consensus gets broken and the entire blockchain is terminated. The solution is that the insertions from the real world should come from a reliable, unquestionable, and indisputable source where every node will have an identical copy of the data, which could be used reliably to perform trustworthy smart contract computation. This kind of information source is what is known as "an oracle" [10].

The need for oracles to support and complement smart contracts was identified shortly after blockchain technology was introduced [64]. In blockchain, an oracle is "an external data agent that observes the real-world events and reports them back to the blockchain to be used by smart contracts" [66].

Blockchain oracle is not the data source itself, but rather the layer that queries, verifies, and authenticates external data sources and then relays that information. They are considered as an interface that delivers the data from an external source to the blockchain. To elucidate the oracle concept, smart contracts cannot pull data from IoT devices such as temperature, pressure or any other type of sensors, probes, barcode readers and RFIDs. For blockchain and smart contracts to be applicable to IoT, an oracle is used to validate the data obtained from the physical world and injected into a smart contract residing on a blockchain. Therefore, building an integrated IoT-blockchain system requires ensuring that the data collected by IoT devices are authentic and not tampered with bearing in mind that once data is inserted in the immutable blockchain, it cannot be corrected or modified [3]. In more general scenarios, oracles are also used as a gateway from the external world to smart contracts dealing with a decentralized mechanism involving weather, stock prices, or political events.

To draw a full picture of an IoT-blockchain combined system, an oracle connects and communicates with the application layer in a blockchain as shown in Fig. 4. This blockchain in turn could be integrated with one or more dew, fog, edge, and cloudlet or cloud layer depending on the design architecture. To call data from the external source, the smart contract has to be invoked where a transaction fee measured in a special unit used in blockchain called Gas has to be paid.

A. THE ORACLE PROBLEM

The oracle problem is the risk of oracles being compromised and feeding the blockchain with falsified and tampered data. In blockchain, the oracle problem is defined as "the security, authenticity, and trust conflict between third-party oracles and the trustless execution of smart contracts" [67]. Any node joining a blockchain has access to its blocks and their contents, this means that malicious acts are difficult to control. Therefore, it is necessary to ensure that no malicious operations can be successfully completed [3]. The risk of corruption, malicious operation, and falsified information increases in the case of using a centralized oracle as a single source data input, which revokes the decentralization fundamental of blockchain and increases the probability of the oracle problem [66]. The oracle problem does not only undermine the feasibility of a project but presents a severe threat to investors, consumers, and academics, as well [10]. From a game-theoretical perspective, the higher the value of the smart contract, the higher the incentive for the system to be compromised [68].

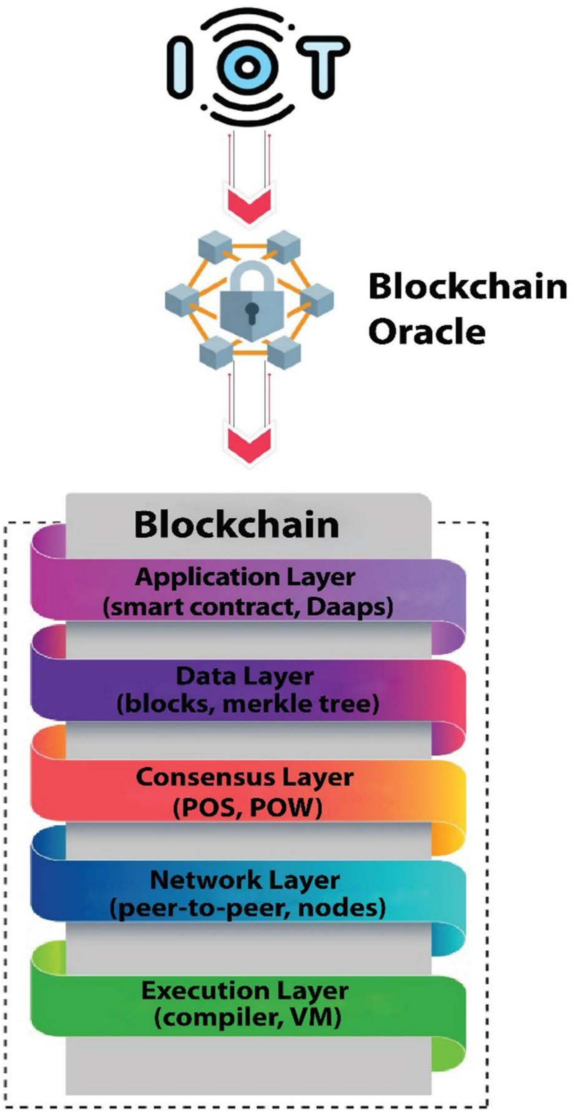


FIGURE 4. Oracle position in a comprehensive IoT-blockchain system.

B. HOW ORACLES WORK

The external data provider such as an IoT device or a sensor is mapped to the blockchain in the form of oracle contracts [66].

This means that an oracle needs to deploy an oracle smart contract and include it in the user smart contract to activate the data access service using APIs. The external data collected by oracles can trigger actions in the blockchain. Oracles gather data from IoT devices and feed it to the blockchain, which ensures data safety using the security hardware. Also, each time the data is provided by an oracle to a blockchain, a certification document is generated, which can be verified by any third entity that needs to confirm the validity of the results. The evidence used for verification is resistant to imitation or modification by malicious nodes [3].

C. TYPES OF ORACLES

Oracles can be categorized based on: data source, number of nodes, design patterns or interaction model [66] as discussed in what follows.

1) Oracles can get data from different sources based on which oracles are classified into:
   - Software or deterministic oracles are those with the ability to interact with data sources on the Internet and inject the obtained information into the blockchain smart contract. The data sources could be databases, servers and websites. Due to their connection to the Internet, software oracles enable real-time data transfer to the blockchain, which makes this type of oracles very common. Online data provided by software oracles varies from stock rates, metal prices, digital asset prices, to real-time flight information, or any other information available online [65].
   - Hardware oracles are those that collect data about the physical world directly using IoT devices such as scanners, RFID, barcode/QR scanners, robot, and different types of sensors and transfer it to the smart contract. A hardware oracle translates real-world events into digital data that could be interpreted by smart contracts [65].
   - Human oracles are those depending on input from individuals with specialized knowledge or skills to provide data to blockchain networks. Experts serving as oracles can provide, verify and authenticate data from various sources and supply that information to smart contracts. Human oracles use cryptography to verify their identity, which makes the possibility of fraud very low.
2) The number of nodes used by oracles to get data to a smart contract divides the trust model used by oracles into:
   - A centralized model, which is controlled by a single entity making it the sole data provider and a single point of failure, as well. This model where data trustworthiness depends on the entity controlling the oracle is considered the riskier.
   - A decentralized model addresses the problem of single point of failure system but results in higher latency and relatively less efficiency in data processing especially when compared with the centralized model. Decentralized oracles share the same features as public blockchains, which increase the reliability of data transferred to smart contracts. Decentralized oracles or consensus oracles do not depend on a single source for truth but rather collect information from multiple oracles to ensure data validity. Many blockchain projects provide decentralized oracle services to other blockchains in different applications such as markets prediction [65].
3) The oracles design pattern may have the following three forms:
   - A request-response design pattern, which is utilized for huge data space that cannot be stored in a smart contract and participants are expected to use a limited part of the complete dataset at a time. This design pattern is implemented in systems consisting of an on-chain smart contract part, which initiates the request for the oracle and an off-chain infrastructure used to monitor requests and retrieve data.
   - A publish-subscribe design pattern, which is utilized for an oracle that provides a changing data broadcast service such as commodity prices and temperature readings. When the oracle is updated with new data, it indicates the availability of new information for users.
   - An immediate-read design pattern, which is utilized for oracles that provide data required for immediate decisions such as academic certificates or dial codes. This type of oracle stores data once in its contract storage and make it available for any other smart contract or blockchain-enabled applications using a request call to the oracle contract.
4) Oracles can have different interactions with external data sources. They either feed data to the blockchain or deliver data from the blockchain. The following further explains the two types of oracles based on external data sources interaction:
   - Inbound oracles feed data captured from the external world into the blockchain such as asset price, which can be purchased automatically when it reaches a certain level.
   - Outbound oracles allow smart contracts to deliver data to the external world such as a smart lock that unlocks automatically when a payment is received its address on the blockchain.
   - An inbound oracle example is the one providing sensor reading to a smart contract while an outbound oracle is the one sending information initiated by a smart contract to a mechanism that unlocks a smart lock [65]. Finally, a single oracle can fall into multiple categories. For instance, an oracle that provides data from an

organization's website is a centralized inbound software oracle [65].

D. AVAILABLE ORACLES

Due to the challenges facing smart contracts interacting with the outside world and considering the necessity for trustworthy oracles to support blockchain applications expansion, multiple oracle proposals with different designs and performance levels emerged in addition to several platforms and industrial models. The following are the main software oracles found in the industry:

- Astraea is a general-purpose decentralized blockchain oracle that enables a voting-based game and supports external data feeds from multiple oracles to determine the truth or falsehood of reported data presented as Boolean propositions. Users of Astraea can possibly take over one or more of the following three roles: submitters, voters, and certifiers. Submitters are the clients who want to learn the outcome of Boolean propositions. Thus, they provide the propositions to the system and pay fees. On the other hand, voters play a low-risk/low-reward game to vote on the truth or falsehood of the randomly selected propositions. Finally, certifiers play a high-risk/high-reward game to certify the propositions of their choice. Users are rewarded when their vote or certification matches the majority outcome and might be penalized otherwise [64], [69].
- Augur is an oracle platform for prediction markets. It transfers data from the real world to the blockchain in a decentralized way without a need for a trusted intermediary. Platform users specify the outcomes of Augur's prediction markets who hold Augur's native reputation token. Users stake their tokens on the actual observed outcome and, in return, receive settlement fees from the markets. The incentive structure on Augur is designed to guarantee that honest, accurate reporting of outcomes is always the most profitable option for reputation token holders [70].
- Town Crier is a data authentication oracle in the form of a bridge between smart contracts and existing trusted websites data sources. It combines a blockchain front end with a trusted hardware back end. Town Crier supports confidentiality and enables private data requests with encrypted parameters. Town Crier is implemented on Intel's Software Guard Extensions (SGX) as a trusted code on Town Crier server. This trusted hardware capability of SGX supports multiple feeds data collection, aggregates data securely, and compiles authenticity proofs before returning data to blockchain smart contracts [71].
- Witnet is a decentralized oracle network (DON) running on a blockchain with a native protocol token (called Wit). It links smart contracts to the real world and enables retrieving data with verifiable integrity proof without the need to trust any third party. Miners are called witnesses and are rewarded for retrieving, attesting and delivering web content for clients based on their reputation. The Witnet protocol creates a powerful incentive for witnesses to act honestly and maintain their reputations. Whereas, clients spend Wit as incentives to witnesses for their Retrieve-Attest-Deliver (RAD) work [72].
- Provable (previously known as Oraclize): is a leading oracle service provider for blockchain applications and smart contracts compatible with major public and private blockchain platforms such as Ethereum, Rootstock, R3, Corda, Eos, and Hyperledger Fabric. It provides a solution that ensures that data fetched from external sources such as Web APIs is untampered by attaching authenticity proof with the returned data. Provable uses a variety of authenticity-proof technologies such as Trusted Execution Environments (TEE) and auditable virtual machines. In addition to solving the oracle problem, data providers do not need to change their services to be compatible with blockchain protocols [73].
- ChainLink is a decentralized oracles system that enables trustworthy data feeds and connectivity between smart contracts and external data sources. ChainLink has a high-level design that distributes trust models at two layers between Blockchain (on-chain) and ChainLink Nodes (off-chain). It is based on reputation to reproduce the consensus mechanism of a blockchain. ChainLink considers the majority of oracles with the same data and the reputational level of each oracle to decide on inserting data into a blockchain. The data confirmed by the majority of the oracle is uploaded on the chain. This trust model ensures that ChainLink components maintain integrity, confidentiality, and authenticity of data for smart contracts while selecting external oracles, during data reporting sessions between smart contracts and ChainLink Nodes, and aggregating reported query results from multiple data feeds [74].
- Aeternity an open-source highly scalable public blockchain platform with smart contract language called Sophia, and cryptocurrency, named Aeternity token. It adopts two consensus algorithms: Proof-of-Work and Proof-of-Stake. Aeternity used one consensus algorithm to agree on system state while the other to agree on the external world data state. Aeternity implements integrated oracles to retrieve real-world data from various providers. Aeternity oracle operators register with its blockchain and receive and answer questions posed to an oracle by the smart contracts after paying some fees. Aeternity blockchain supports high security using type-safe virtual machines called Fast Aeternity Transactions Engine (FATE) [75].

Table 2 demonstrates the differences between the above-explained oracles.

Based on the above discussion on the main software oracles operating in blockchain networks, it is clear that only a limited number of platforms exist in the industry. This offers potential opportunities for new startups, therefore, the

TABLE 2. Summary of available oracles in the industry market.

Oracle	Architecture	Trust model	Currency/ Token
Astraea	Decentralized/ on-chain	Voting	None
Augur	Decentralized/ on-chain	Reputation	REP
Town Crier	Centralized/ off-chain	TEE (SGX)	None
Witnet	Decentralized/ on-chain	Reputation	Wit
Provable	Centralized/ off-chain	TLSNotary authenticity proof	None
ChainLink	Decentralized/ on and off-chain	Reputation	LINK
Aeternity	Decentralized/ on-chain	Consensus	Aeon

above-provided survey shall be of great help for such future ideas.

Looking at hardware oracles, as some applications require obtaining data readings from the physical world where IoT devices should be used, a different type of oracle is required for those cases. The difficulty in these applications is to find a secure and authenticated way of obtaining the data from IoT devices and transferring them to the smart contract blockchain. This is the role of hardware oracles, which are able to secure an authenticated data readings from sensors, meters, RFIDs by providing cryptographically attested anti-tampering reading data. A hardware oracle authenticates the origin and value of the reading data using attestation and identification private key for each IoT device to sign the outgoing readings. Decentralized applications for the industrial, supply chain and shared economy applications would not exist without reliable Hardware Oracles. Therefore, a massive deployment of these hardware oracle devices is predicted in the future. One of the leading technology enabling organizations for industrial and enterprise blockchain use cases is "Ledger", which provides hardware oracle and blockchain computer solutions.

Although there are considerable studies conducted on blockchain and its applications, they seldom discuss oracles. By carrying out a systematic literature review on blockchain oracles, we figured out that out of 142 journal papers discussing blockchain real-world applications, only 15% considered the role of oracles, and less than 10% underlined the oracle problem [76]. Obviously, the research on trusted oracles is in its infant stages [66]. Although real-world blockchain applications strongly rely on oracles, the roles and implementations of oracles are mostly overlooked in the literature. Overlooking the oracle problem may lead blockchain researchers, developers and users to attain unrealistic inquiry methods due to misconceptions [76]. However, only a few articles addressing the oracle problem in blockchain were found in the literature. For instance, the work in [3] proposed SLTA, which is a secure and lightweight triple-trusting architecture. It includes an oracle-based data collecting mechanism that guarantees that the data collected from edge devices of IoT cannot be modified using a transport layer security (TLS) certification technology. The suggested architecture also ensures the credibility of the users' identities without a trusted third party. The in [77] is another research to integrate Hyperledger blockchain with an existing industrial strength secure element for cryptographic software protection (Wibu CmDongle / the "dongle") that acts as a hardware-based oracle. The dongle enforces cryptographic access control to the function level of an application through a local daemon. Also, the study in [78] established an oracle reputation system using smart contracts. The proposed system is aimed at providing a decentralized, scalable, and secure management solution for accessing IoT data.

The above presentation of the main available oracles in the industry and literature provides a clear view for system designers to use what best fits in-hand use cases. However, in the following section, we demonstrate a useful use case to enhance the understanding of how oracles work and the different ways of implementation.

VII. CARBON SENSING AND MEASUREMENT: A USE CASE

In order to draw a full picture of combining IoT with blockchain utilizing oracles, a practical use case is presented in this section. Carbon sensing and measuring use case was chosen because it is a suitable illustrative example of transferring data from the physical world through IoT devices and injecting it into a blockchain's smart contract to perform logical functions that have sensitive implications and consequences. This use case demonstrates the importance of oracles in ensuring the authenticity, trustworthiness, and effectiveness of the overall system performance.

A. CARBON PRICING

Carbon pricing is an approach that applies cost or price on carbon pollution as a means to reduce the amount of carbon dioxide emissions released to the atmosphere and pollute the environment. It aims at encouraging polluting entities to invest in green processes via passing the cost of emitting on to emitters. Carbon pricing is applied using two forms: carbon taxes or emissions trading [79]. For both strategies, CO_2 emitting organizations and manufacturers are obliged to report their emissions to administrations and authorities [80]. The reporting process is carried out by companies using their own methods without automated monitoring of the amount of released CO_2. So far, there has not been any specific and clear sensing and measuring method for CO_2.

Problem Statement: To date, a trusted sensing and measuring system for the released CO_2 by emitting entities is not yet established. Emitters report their emissions independently and without any automated authentication follow-up system or administrative monitoring of the capturing process of the amounts of released CO_2. There is an immense need for CO_2 sensing and measuring platform to provide authenticated and trusted CO_2 readings that will be used in determining either carbon taxes or trading allowances for emitting parties as part of the carbon pricing approach aiming at reducing air pollution, mitigating climate change and saving the environment.

B. PROPOSED BLOCKCHAIN SYSTEM OVERVIEW AND DESIGN

In this section, an Ethereum blockchain solution that utilizes the key features of blockchain technology and Ethereum smart contract is presented. We propose using CO_2 meters mounted on required positions in emitting party's facilities such as manufacturing plants and premises to measure the quantity of released CO_2 emissions into the atmosphere. The CO_2 meter is an IoT device that is connected to a blockchain that supports a smart contract. Ethereum blockchain is suggested due to its robustness as the biggest smart contract supporting blockchain available in the market today. Ethereum uses a virtual machine known as EVM to execute smart contracts written in the Ethereum language "solidity". The EVM defines the rules that govern the change in the Ethereum state machine which is a large data structure holding Ethereum accounts and balances. Computing a new state machine from one block to another is controlled by the EVM since it defines the rules that validate the computation process. The EVM is considered a single virtual entity maintained by a huge number of connected computers acting as nodes running Ethereum clients [81]. Also, Ethereum uses the Keccak-256 cryptographic hashing technique for encryption and the Proof of Work (POW) mechanism to reach consensus. However, a highly anticipated transition from the Proof of Work (POW) consensus mechanism to Proof of Stake (POS) was proposed and has been delayed for some time but it is expected to take place by the end of the second quarter of 2022 [82].

Ethereum smart contract is used to create a decentralized, secured, and authenticated CO_2 measuring system that provides trust in CO_2 reading data, and uses immutable logs and events to save and declare system information. Using a smart contract facilitates the automation of the process and assists in saving the history of all transactions without alterations due to its traceability feature. Choosing Ethereum blockchain for this use case is because it is a promising technology that possesses a lot of features making it an ideal tracking platform and a perfect candidate for integrating with IoT devices to form a powerful traceable authenticated and trusted sensing and measuring solution. Also, since Ethereum smart contract made blockchain programmable, it supports the execution of code on top of blockchain, making the technology even more powerful.

Since the proposed solution is an integrated system combining a real-world IoT sensing device (CO_2 meter) outside the blockchain with smart contract residing on Ethereum blockchain, It requires an oracle to feed the sensing data to the smart contract. As previously explained, blockchain oracles are third-party services that provide smart contracts with external information.

The system setup depends on the details of the implementation, which should be tailored for the premises of the emitting organization. The IoT–Blockchain integration architectures demonstrated in section IV provide proper design directions for the required system of the CO_2 measuring use case and indeed any other use case in general. Based on the emitting premises, the system design could use any combination of dew, edge, fog, cloud computing from the surveyed studies to integrate blockchain with CO_2 sensing and measuring IoT devices. As per the type of oracle to be utilized in this use case, adequate guidance is presented in the "Available Oracles" subsection to select the oracle that fulfills the CO_2 measuring system needs, based on which two system design suggestions are provided.

C. SOFTWARE ORACLE-BASED DESIGN

In the first system design, a smart contract running on an Ethereum node requests the readings of CO_2 from a CO_2 meter by emitting a solidity event. The CO_2 meter stores the reading data on an off-chain database. An oracle listens to the events emitted from the smart contract and queries the off-chain database for the data. Once the response to the query is obtained, the requested data readings are transferred to the smart contract where the request is fulfilled. It is important to note that smart contracts cannot communicate directly with any platform, program or script outside the blockchain. Since the oracle is an off-chain service provider that resides outside the Ethereum blockchain, an API is required to provide an interface. Ethereum solidity uses two libraries for this purpose, namely "web3.js" and "ethers.js". The oracle type used in this design is an inbound software request-response oracle such as the service provided by provable oracle company.

The interaction and message sequence between the external IoT sensor and the blockchain smart contract is as follows:

1) An externally owned account (EOA) or a different contract account calls a function on the main smart contract (to be explained later) to initiate a request for reading the CO2 meter data.
2) The requested function is tied to an event. This event gets emitted during the execution of the requested function.
3) The software oracle, which is listening online for this particular event at the smart contract address, is notified through a library (such as web3.js) and starts processing the request for CO_2 data.

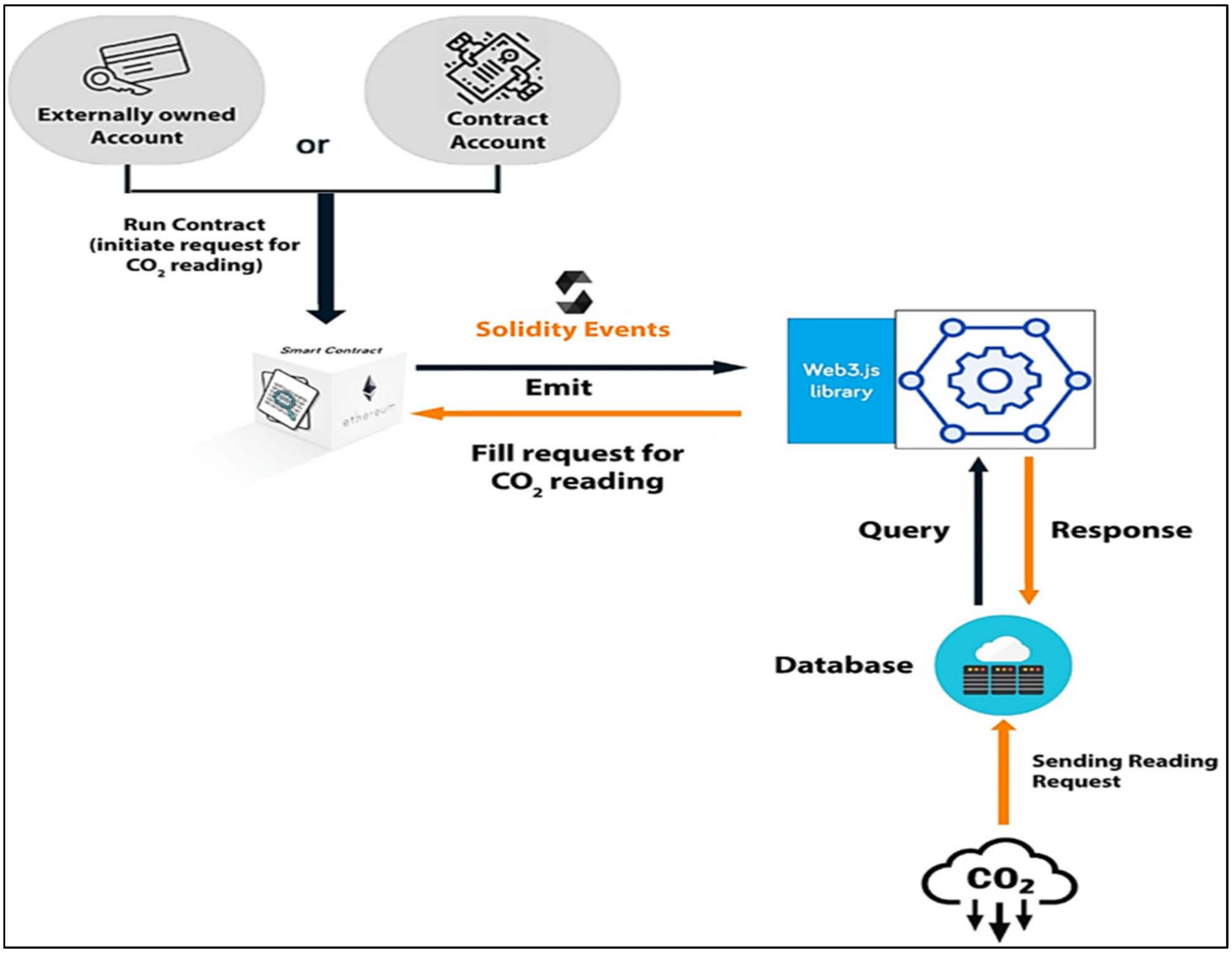


FIGURE 5. The process sequence for CO_2 measuring smart contract utilizing software oracle.

4) Once request processing is done, the oracle fulfills the data request of the calling smart contract through the library by running a method on the smart contract address. At this step, another event is emitted to notify the caller of the fulfilled request.

Fig. 5 further clarifies the whole process steps for the first suggested design.

D. HARDWARE ORACLE-BASED DESIGN

As per the second system design, a smart contract running on an Ethereum node requests the readings of CO_2 from a CO_2 meter by emitting a solidity event. The CO_2 meter is a Cryptographically attestable anti-tampering IoT device that securely reports the CO_2 readings. The cryptographic attestation of the meter reading aims at authenticating the origin of the CO_2 measures by signing outgoing computations using a private key in combination with a nonce to avoid digital signature repetition. This IoT device with secure reading represents a hardware Oracle, which is a gateway from the physical world to the blockchain ecosystem. An event listener listens to the events emitted from the smart contract through the library and a fetching script fetches the reading data. The requested data reading is then transferred to the smart contract where the request is fulfilled. The oracle type used in this design is an inbound hardware request-response oracle. The interaction and message sequence is as follows:

1) An externally owned account (EOA) or a different contract account makes a call to a function on the main smart contract (to be explained later) to initiate a request for reading the CO2 meter data.
2) The called function is tied to an event. This event gets emitted during the execution of the requested function.
3) An event listener is notified through the library (such as web3.js). This event listener is an event that listens online for a particular event at the smart contract address.
4) A fetching script (e.g. written in python) is then notified about the smart contract emitted event. This is done by the event listener. The fetching script fetches the encrypted sensor data from the CO_2 meter hardware oracle and writes it on the blockchain by calling a smart contract function.

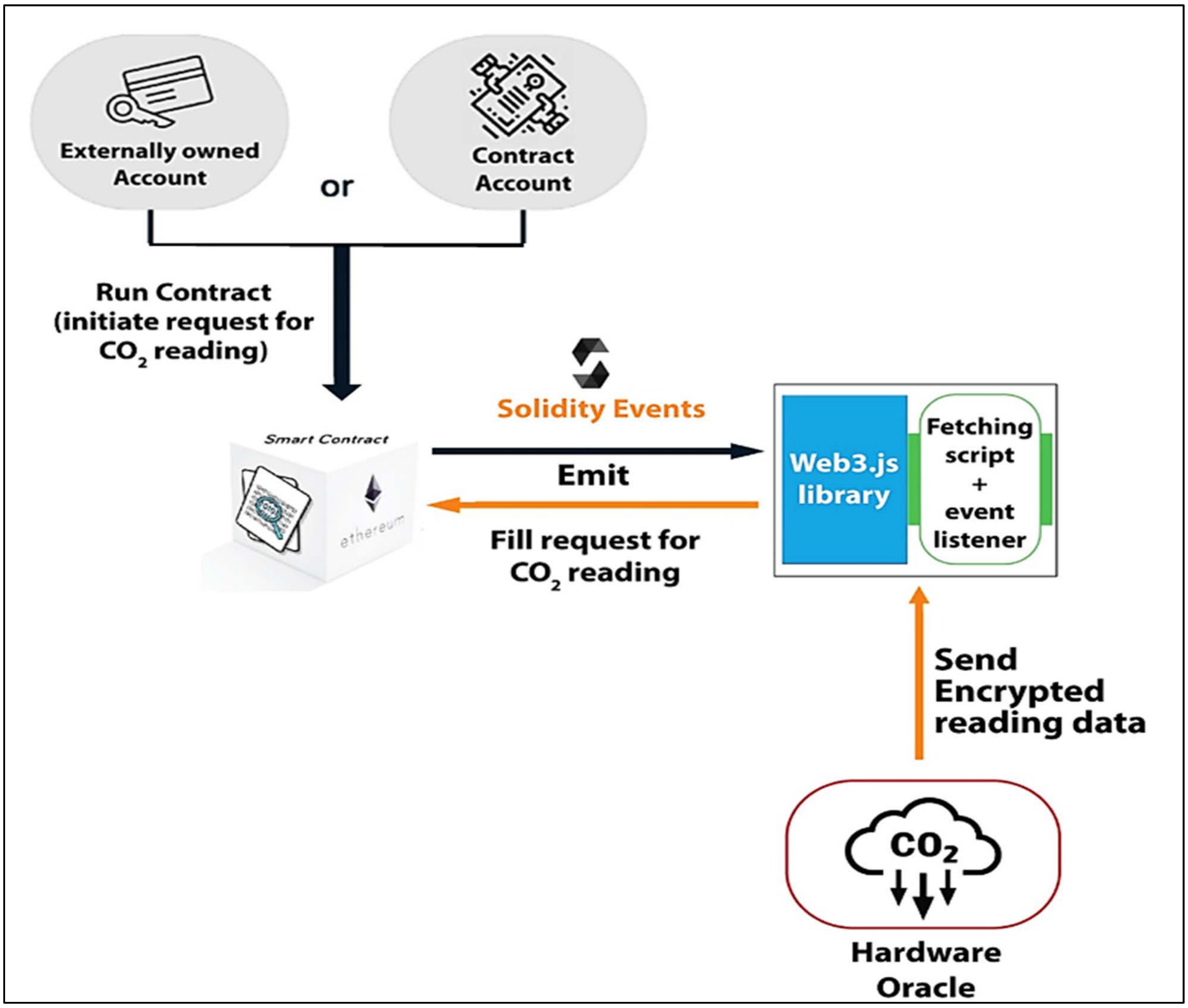

Externally owned Account
or
Contract Account
Run Contract (initiate request for CO_2 reading)
Smart Contract
ethereum
Solidity Events
Emit
Fill request for CO_2 reading
Web3.js library
Fetching script + event listener
Send Encrypted reading data
CO_2
Hardware Oracle

FIGURE 6. The process sequence for CO_2 measuring smart contract utilizing hardware oracle.

5) Once the encrypted sensor data is obtained, the fetching script completes the data request of the calling smart contract through the library by running a method on the smart contract address. At this step, a different event is emitted to notify the caller of the fulfilled request.

Fig. 6 further clarifies the process steps for the second suggested design.

E. SMART CONTRACT IMPLEMENTATION

For any selected design architecture, a smart contract is required to set and store trusted and authenticated CO_2 readings. However, implementing and executing smart contracts on blockchain is associated with certain costs or fees, therefore, blockchain developers use a testing environment to code, debug, compile, run and test their smart contract. One of the best testing environments is the browser-based compiler and IDE ‘‘Remix’’ which was used to develop our CO_2 measuring smart contract. Remix IDE rich features provide the required tools for testing and debugging [83] which makes modifying and correcting the code easier and the programming process more convenient. The language used for creating the smart contract is Ethereum Solidity. The process of developing and implementing our smart contract included determining the logic for the use case as demonstrated in Algorithm 1, based on which a message sequencing for the smart contract was created as shown in Fig. 7.

The next step was to write the code in Solidity using Remix. After which, an iterative process of debugging and modifying the smart contract code is conducted till reaching the ideal final version of the smart contract that supports the required functions. In this final version of the smart contract, the creator is the administrating authority that monitors and supervises the carbon pricing scheme covering a certain region or country. The administrator has a unique Ethereum Address (EA) and interfaces with the smart contract either

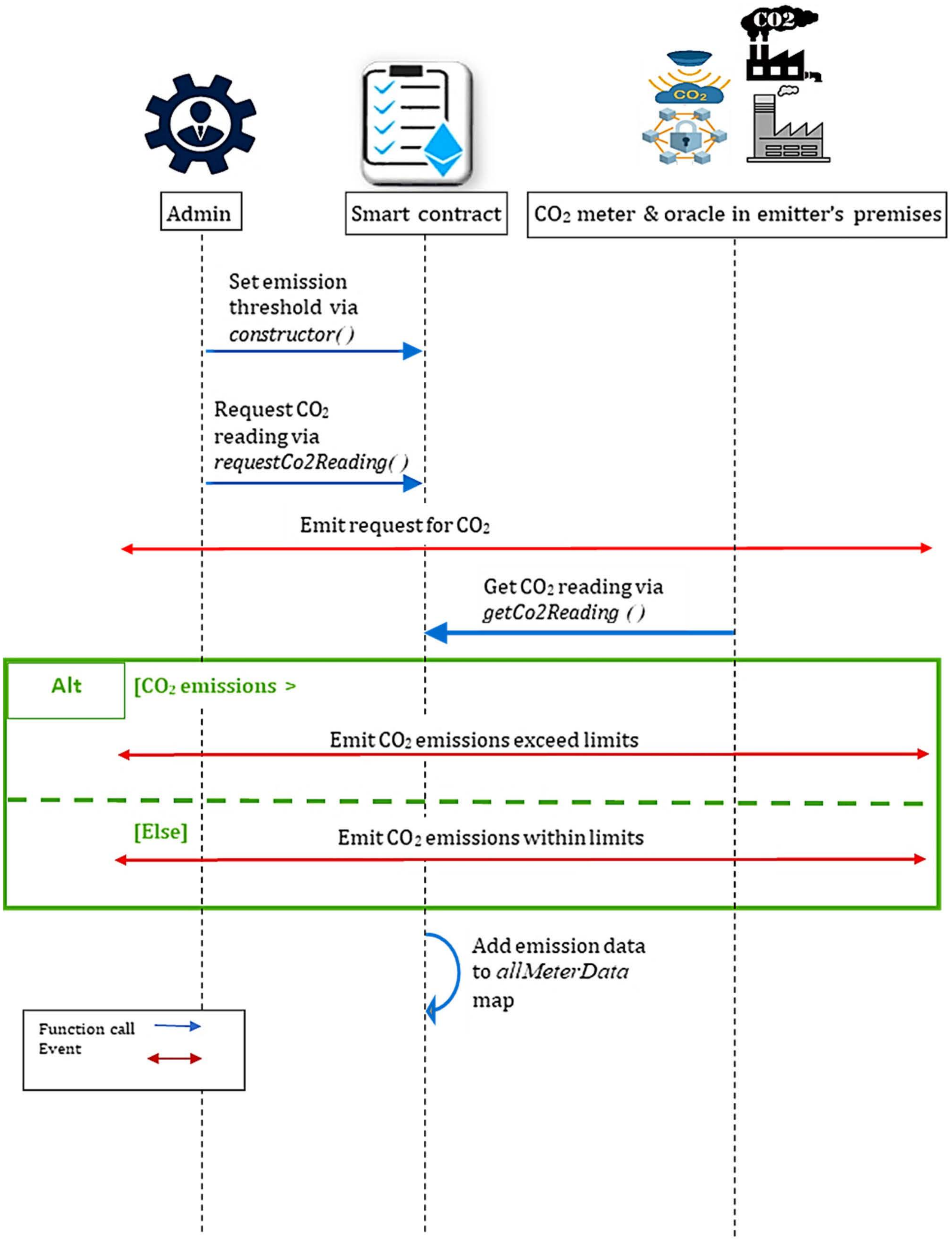
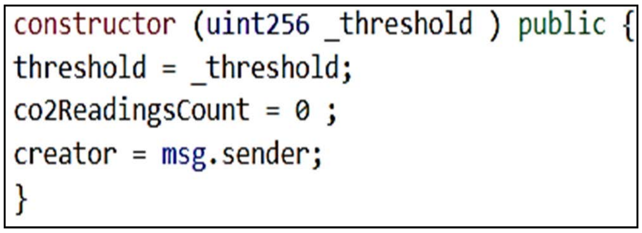

CO2
Admin
Smart contract
CO₂ meter & oracle in emitter's premises
Set emission threshold via *constructor( )*
Request CO₂ reading via *requestCo2Reading( )*
Emit request for CO₂
Get CO₂ reading via *getCo2Reading ( )*
Alt
[CO₂ emissions >
Emit CO₂ emissions exceed limits
[Else]
Emit CO₂ emissions within limits
Add emission data to *allMeterData* map
Function call
Event

FIGURE 7. Message sequencing of the developed CO_2 measurement smart contract.

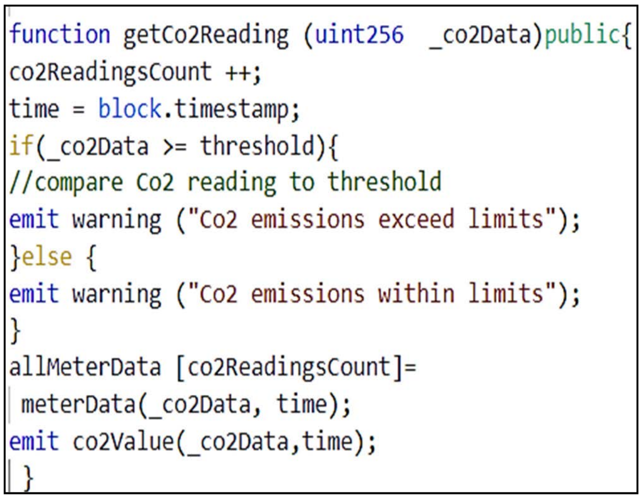
Algorithm 1 Carbon Emissions Measurement

```
Input: Administrator EA, $CO_2$ emissions threshold,
       $CO_2$ readings count
1  Emit an event requesting $CO_2$ reading.
2  Event captured by event listener.
3  Get $CO_2$ reading obtained by IoT sensor from an oracle.
4  Increase $CO_2$ readings count.
5  If $CO_2$ reading >= threshold
6      Emit warning event "Co2 emissions exceed limits".
7  Else
8      Emit warning event "Co2 emissions within limits".
9  Insert $CO_2$ reading & block time stamp to "allMeterData"
   Map list.
10 Emit $CO_2$ value event"reading & block time stamp".
```



```
constructor (uint256 _threshold ) public {
threshold = _threshold;
co2ReadingsCount = 0 ;
creator = msg.sender;
}
```

FIGURE 8. CO_2 measuring smart contract constructor and modifier.



```
function getCo2Reading (uint256  _co2Data)public{
co2ReadingsCount ++;
time = block.timestamp;
if(_co2Data >= threshold){
//compare Co2 reading to threshold
emit warning ("Co2 emissions exceed limits");
}else {
emit warning ("Co2 emissions within limits");
}
allMeterData [co2ReadingsCount]=
 meterData(_co2Data, time);
emit co2Value(_co2Data,time);
}
```

FIGURE 9. CO_2 measuring smart contract function.

directly as an externally owned account (EOA) or through another triggering smart contract. Anyways, the administrator sets the threshold of carbon emissions during the deployment stage of the smart contract using a constructor. Also, a modifier is added to the smart contract code to limit the function call to the administrator as shown in Fig. 8.

The main two functions in the smart contract shown in Fig. 9 are "requestCo2Reading" and "getCo2Reading". "requestCo2Reading" function is restricted by "onlyAdmin" modifier. This function emits an event named "co2Request" stating "request for Co2 reading sent" which will be picked up by an event listener through the library.

The other function "getCo2Reading" is fulfilled by the oracle which obtains the CO_2 reading data. This function stores the reading value and time, compare the reading to the threshold, emit warning events about the status of CO2 emissions compared to the threshold, and store the reading data in "allMeterData" map list.

F. SMART CONTRACT TESTING

To test the smart contract. The code was compiled and an Ethereum address (EA) was chosen for the Admin which is: "0 × 5.38Da6a701c568545dCfcB03FcB875f56beddC4".

The smart contract was successfully deployed by the Admin using its (EA) where the administrator specified the threshold input parameter through the constructor. In science and industry, CO2 is measured in parts-per-million (ppm). For testing purposes, the threshold was set to 400 ppm. The transaction log shown in Fig.10 demonstrates the successful deployment of the smart contract. Also, the function "requestCo2Reading" was executed and the smart contract successfully emitted the event "request for co2 reading sent" as shown in Fig. 11.

To implement the first test case, the CO2 meter reading was set to 300 ppm which is above the 400ppm threshold by calling the function "getCo2Reading". The smart contract was executed and the reading value and time were saved in the "allMeterData" map and the transaction was successfully mined into a block as shown in the execution transactions shown in Fig. 12.

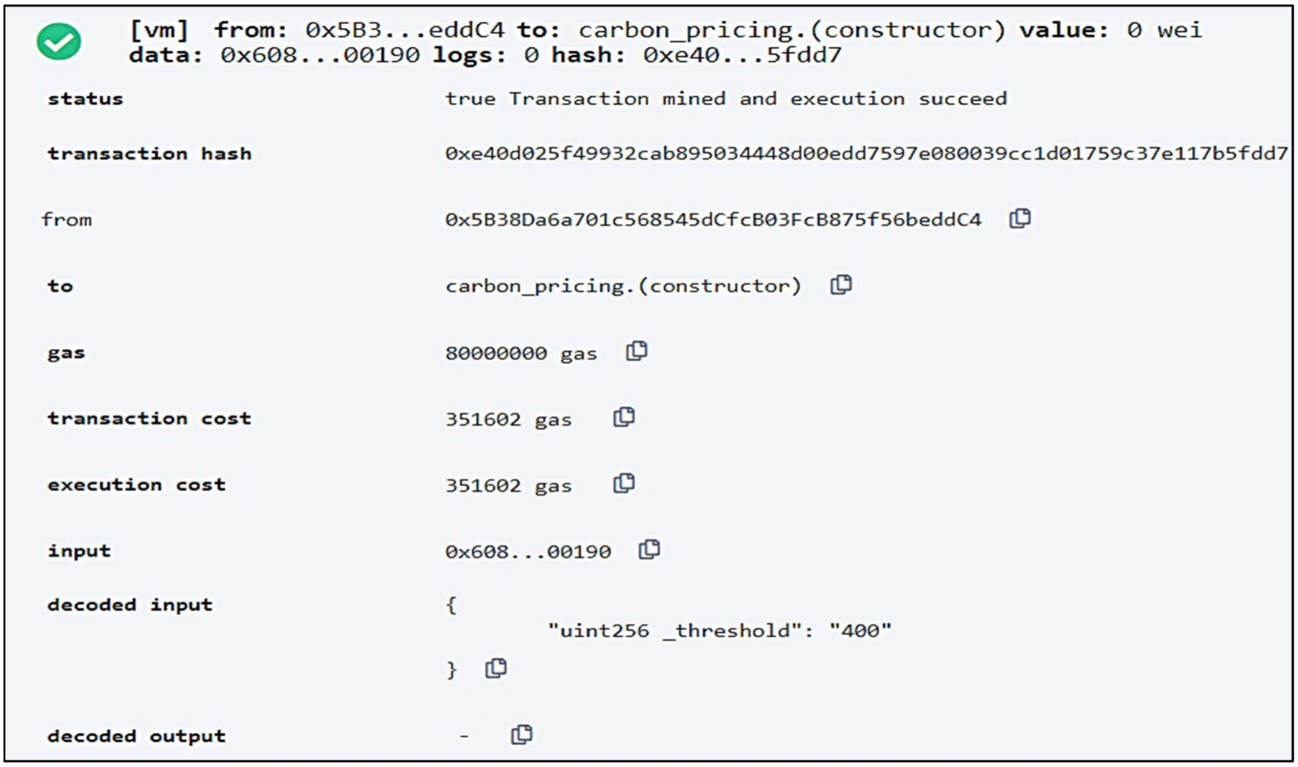
To implement the second test case, the CO2 reading was set to 500 pmm which is above the 400ppm threshold by calling the function "getCo2Reading". The smart contract was executed and the reading value and time were saved in the "allMeterData" map and the transaction was successfully mined into a block in the blockchain as shown in the deployment and execution transactions shown in Fig. 13.

The transactions log demonstrated that the smart contract emitted the correct events. As for the first test case, the emitted event stated that "CO2 emissions within limits". Whereas for the second test case, the emitted event stated that "CO2 emissions exceeding limits" as shown in Fig. 14 and Fig. 15 respectively.

As seen from the above implementation, the smart contract was deployed and functions were successfully executed to request and obtain the CO2 meter's data. All the gathered readings are stored immutably in the tamper-proof blockchain. This demonstration benefits in understanding how the whole picture of IoT, blockchain and oracle fits and operates together which is clearly a valuable illustration for real-world applications that systems designers can benefit from.

G. ANALYSIS AND DISCUSSION

In this section, a discussion is provided for the proposed carbon measurement use case to demonstrate the developed



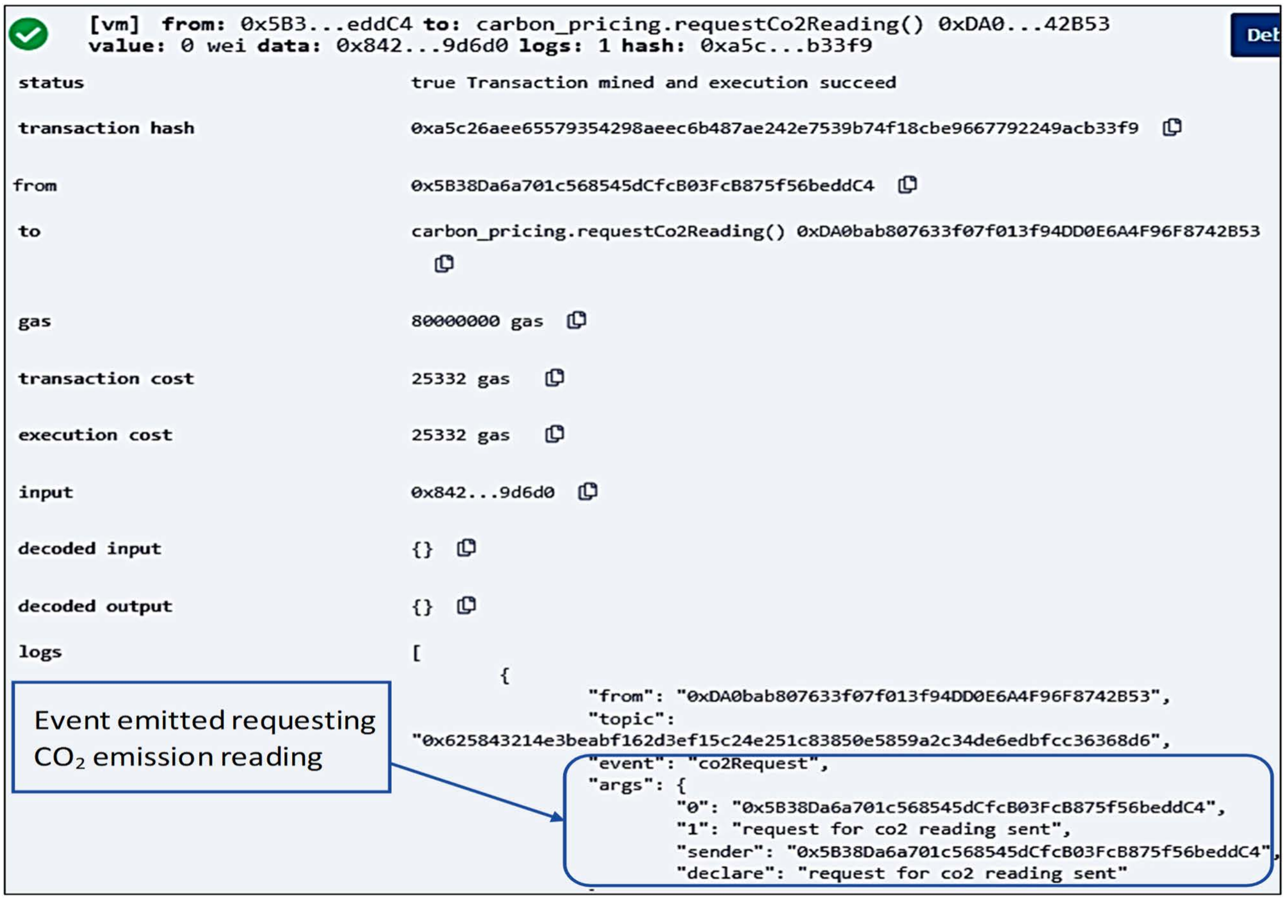
```
[vm] from: 0x5B3...eddC4 to: carbon_pricing.(constructor) value: 0 wei
data: 0x608...00190 logs: 0 hash: 0xe40...5fdd7

status                  true Transaction mined and execution succeed

transaction hash        0xe40d025f49932cab895034448d00edd7597e080039cc1d01759c37e117b5fdd7

from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to                      carbon_pricing.(constructor)

gas                     80000000 gas

transaction cost        351602 gas

execution cost          351602 gas

input                   0x608...00190

decoded input           {
                                "uint256 _threshold": "400"
                        }

decoded output           -
```

FIGURE 10. Successful deployment of the smart contract.

smart contract's robustness and practicality. From the implementation subsection, it was proved that our smart contract code for carbon measurement successfully executed all logical functions where all transactions were securely and transparently saved in the Ethereum blockchain and the correct events were emitted. However, to further investigate the performance of the smart contract, a security, vulnerability and cost analysis are conducted.

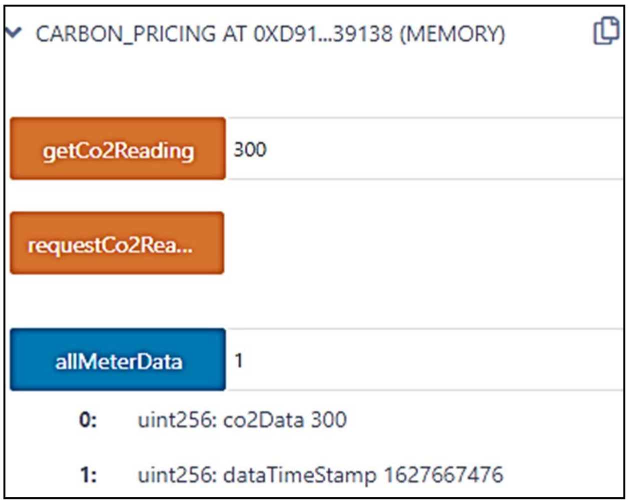
Security-wise, the performed analysis demonstrated that the considered carbon measurement use case leverages the blockchain's characteristics to fulfill the following security aspects:

- Availability: the fact that the smart contract is stored on the Ethereum blockchain makes its resulting execution transactions and logs available at all times. This is due to the distributed structure of Ethereum which guarantees its operability at all times and prevents the occurrence of the single point of failure problem.
- Authorization: the smart contract of the investigated use case designates each function to its caller using modifiers. These Ethereum modifiers are immutably saved in the blockchain ledger since they are part of the smart contract code. This ensures that the functions designation process is secured and tamper-proof.
- Accountability: is achieved by time-stamping all transactions and immutably saving them in the ledger. Therefore, emitters releasing CO2 emissions will be easily held accountable for their excess emissions.
- Confidentiality: is maintained when emission data is not disclosed to unauthorized entities or processes. Confidentiality is preserved for emitters since the measurement smart contract for each entity could be accessed only by the administrator in addition to the emitter itself.
- Integrity: is fulfilled when CO2 emissions readings get saved immutably without any chance of alteration. This is due to the cryptographic structure of the Ethereum blockchain where valid transactions containing CO2 emissions data are encrypted then stored on-chain.
- Resistance to cyber attacks: Ethereum blockchain is characterized by its ability to resist known threats and cyber attacks such as distributed denial of service (DDoS) and Man-in-The-Middle (MITM) attacks. The DDoS is a vicious attack that gets initiated by malicious nodes where the attacker floods the network with fake traffic. Ethereum blockchain decentralized structure facilitates defeating this attack by filtering transactions or specifying a bandwidth to absorb the attack [56]. As per the MITM attack where a malicious party interferes with the communication between two authentic parties, blockchain resists this attack through cryptography. It is known that all transactions must be cryptographically signed by the sender's private key after which they get encapsulated in a block that gets chained with the previous and lateral blocks. Consequently, interfering between transactions' sender and



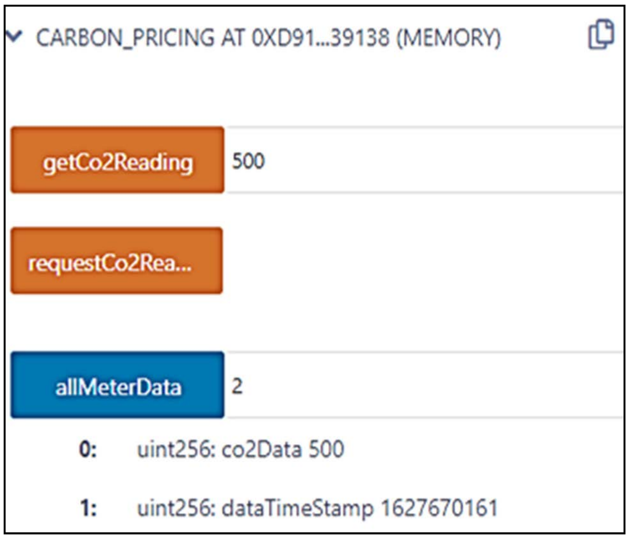
```
[vm]  from: 0x5B3...eddC4 to: carbon_pricing.requestCo2Reading() 0xDA0...42B53
value: 0 wei data: 0x842...9d6d0 logs: 1 hash: 0xa5c...b33f9
status              true Transaction mined and execution succeed
transaction hash    0xa5c26aee65579354298aeec6b487ae242e7539b74f18cbe9667792249acb33f9
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  carbon_pricing.requestCo2Reading() 0xDA0bab807633f07f013f94DD0E6A4F96F8742B53
gas                 80000000 gas
transaction cost    25332 gas
execution cost      25332 gas
input               0x842...9d6d0
decoded input       {}
decoded output      {}
logs                [
                        {
                            "from": "0xDA0bab807633f07f013f94DD0E6A4F96F8742B53",
                            "topic":
                    "0x625843214e3beabf162d3ef15c24e251c83850e5859a2c34de6edbfcc36368d6",
                            "event": "co2Request",
                            "args": {
                                "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                "1": "request for co2 reading sent",
                                "sender": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                "declare": "request for co2 reading sent"
```

FIGURE 11. Successful execution of the function (requestCo2Reading) and invocation of associated event.



```
CARBON_PRICING AT 0XD91...39138 (MEMORY)
getCo2Reading    300
requestCo2Rea...
allMeterData     1
    0:   uint256: co2Data 300
    1:   uint256: dataTimeStamp 1627667476
```

FIGURE 12. First test case smart contract deployment and execution transactions.

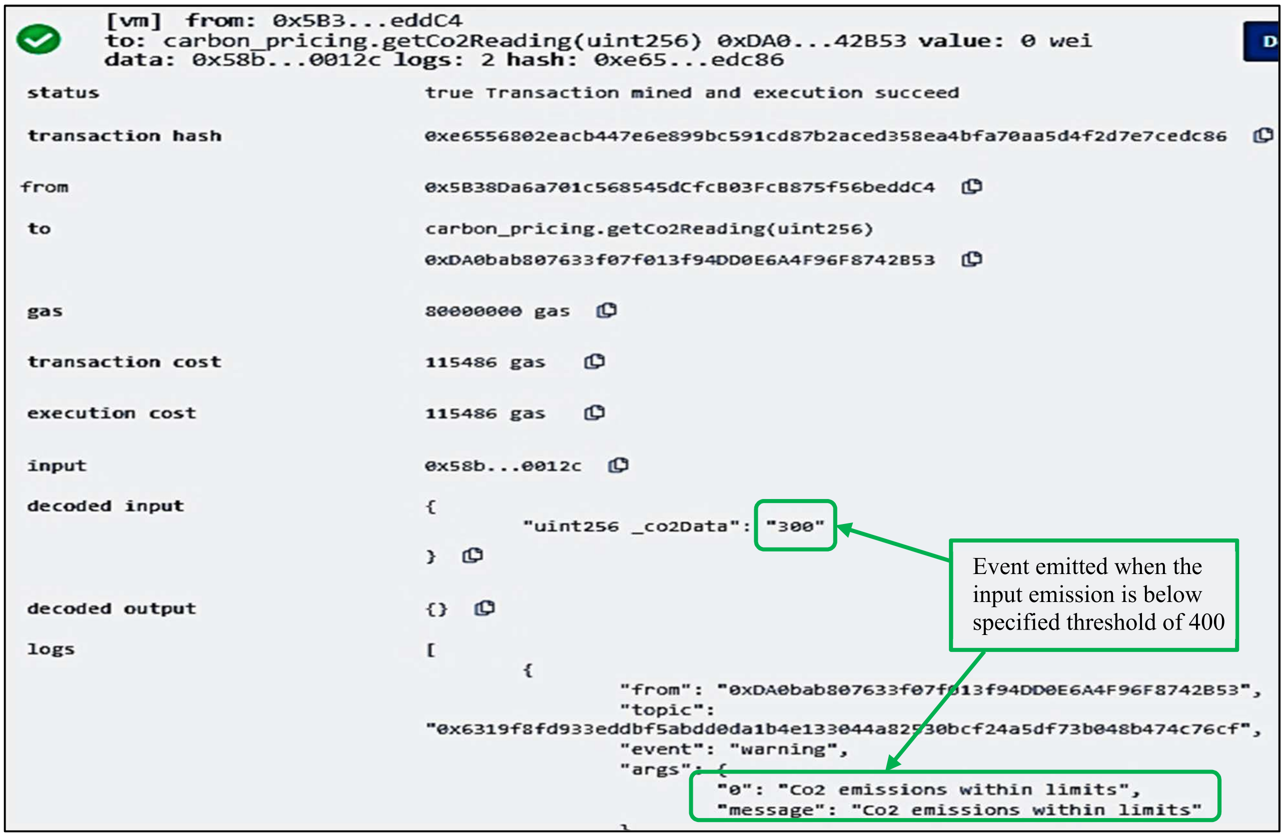


```
CARBON_PRICING AT 0XD91...39138 (MEMORY)
getCo2Reading    500
requestCo2Rea...
allMeterData     2
    0:   uint256: co2Data 500
    1:   uint256: dataTimeStamp 1627670161
```

FIGURE 13. Second test case smart contract deployment and execution transactions.

receiver is not possible especially since private keys cannot be forged.

Additionally, a smart contract security and vulnerability analysis was conducted as it is very important to guarantee that the smart contract does not suffer from any vulnerabilities or bugs that could result from any programming inefficiencies. It is crucial to perform a security and vulnerability check



```
[vm] from: 0x5B3...eddC4
to: carbon_pricing.getCo2Reading(uint256) 0xDA0...42B53 value: 0 wei
data: 0x58b...0012c logs: 2 hash: 0xe65...edc86
status                    true Transaction mined and execution succeed
transaction hash          0xe6556802eacb447e6e899bc591cd87b2aced358ea4bfa70aa5d4f2d7e7cedc86
from                      0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                        carbon_pricing.getCo2Reading(uint256)
                          0xDA0bab807633f07f013f94DD0E6A4F96F8742B53
gas                       80000000 gas
transaction cost          115486 gas
execution cost            115486 gas
input                     0x58b...0012c
decoded input             {
                                  "uint256 _co2Data": "300"
                          }
decoded output            {}
logs                      [
                                  {
                                          "from": "0xDA0bab807633f07f013f94DD0E6A4F96F8742B53",
                                          "topic":
                          "0x6319f8fd933eddbf5abdd0da1b4e133044a82530bcf24a5df73b048b474c76cf",
                                          "event": "warning",
                                          "args": {
                                                  "0": "Co2 emissions within limits",
                                                  "message": "Co2 emissions within limits"
```

Event emitted when the input emission is below specified threshold of 400

FIGURE 14. First test case smart contract transactions log.

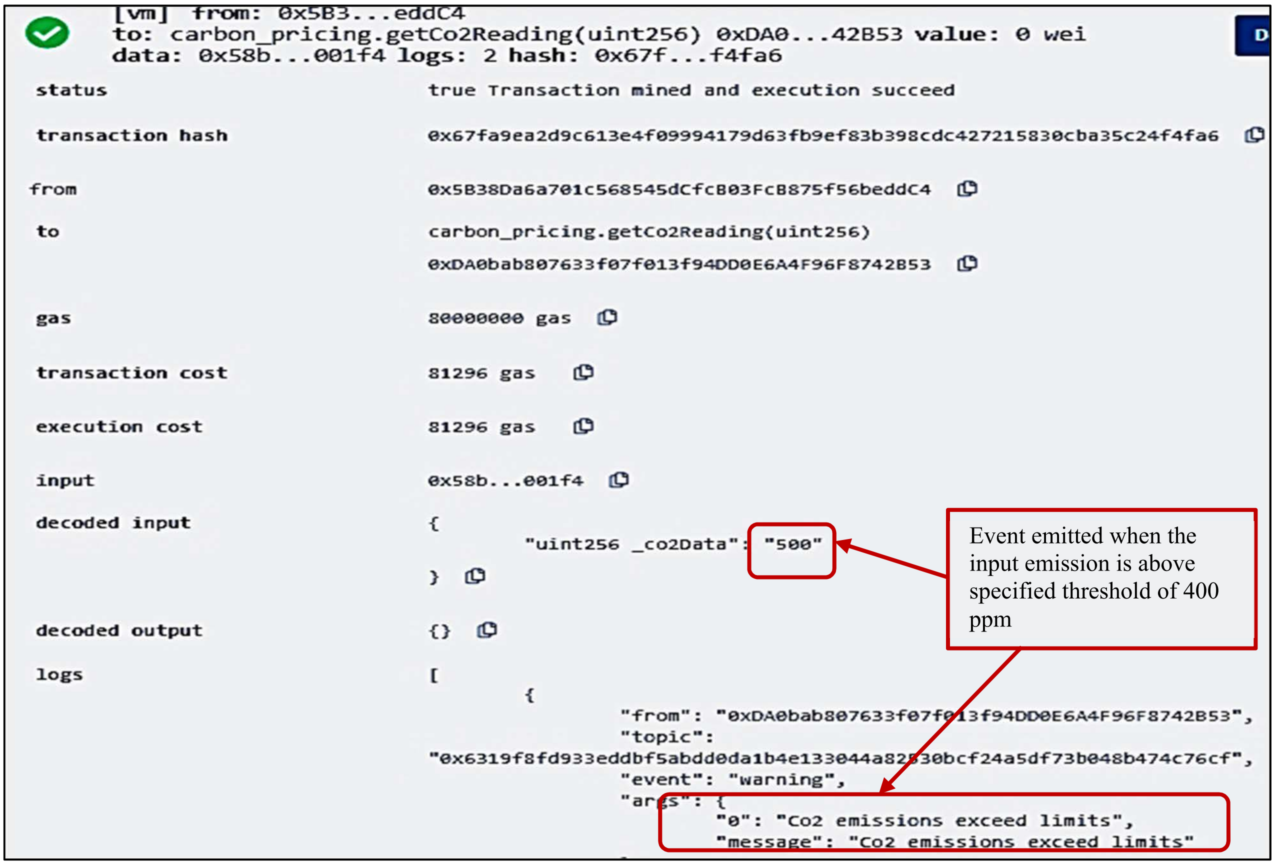
on the Ethereum smart contract before deploying it because once a smart contract is launched it cannot be erased or altered. This is due to the immutable structure of blockchain. The current practices for developing any smart contract imply running a vulnerabilities analysis before deploying it on the main net to ensure reliability and trust in the decentralized application. Although the Remix IDE utilized to develop our carbon measuring smart contract has the capability to detect run time and syntax errors in the solidity code, professional security tools support a higher level of analysis. Multiple smart contracts security and vulnerability analysis tools exist in the market, however, Oyente is the software tool that was chosen to analyze our smart contract.

Oyente runs on Linux operating system and checks the solidity smart contract code for security vulnerabilities such as:

1) Integer underflow: when the attacker subtracts a positive integer from zero resulting in a big value [84].
2) Integer overflow: when the attacker adds a positive integer to the maximum value resulting in zero [84].
3) Callstack depth attack vulnerability: when external call fails because it exceeds the limit of numbers of calls to a contract method allowed by the call stack [85].
4) Transaction ordering dependence: when the order of transactions becomes inconsistent with the time of their invocations [85].
5) Timestamp dependency: when miners control the time of certain transactions to manipulate their execution output [84].
6) Re-entrancy vulnerabilities: when a function call occurs Recursively from a fallback function [84].
7) Parity multi-signature bug: when public functions can be called by anyone because the access modifier is not used correctly [84].

The result of analyzing our smart contract using Oyente is shown in Fig. 16 where the generated security report demonstrates that all the above-mentioned vulnerabilities do not exist and are given "False" output. This proves that our developed smart contract is secure, bug-free and does not suffer from any vulnerabilities.

Furthermore, a cost analysis was conducted for the smart contract of the considered use case. As known, deploying and executing smart contracts involves miners who consume resources to encapsulate transactions into blocks. Therefore, certain fees are associated with the deployment of the smart contract as well with each time a function is executed [86]. The unit of cost for smart contracts is called "Gas" and it is used to measure and pay miners for their mining process [87]. Therefore, it is important to consider Gas fees when developers write their smart contract code since cost is a factor to evaluate the effectiveness and practicality of the developed smart contract [86].



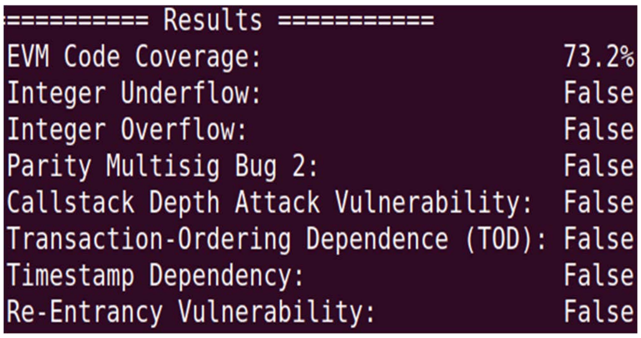
```
[vm] from: 0x5B3...eddC4
to: carbon_pricing.getCo2Reading(uint256) 0xDA0...42B53 value: 0 wei
data: 0x58b...001f4 logs: 2 hash: 0x67f...f4fa6
status                 true Transaction mined and execution succeed
transaction hash       0x67fa9ea2d9c613e4f09994179d63fb9ef83b398cdc427215830cba35c24f4fa6
from                   0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                     carbon_pricing.getCo2Reading(uint256)
                       0xDA0bab807633f07f013f94DD0E6A4F96F8742B53
gas                    80000000 gas
transaction cost       81296 gas
execution cost         81296 gas
input                  0x58b...001f4
decoded input          {
                               "uint256 _co2Data": "500"
                       }
decoded output         {}
logs                   [
                               {
                                       "from": "0xDA0bab807633f07f013f94DD0E6A4F96F8742B53",
                                       "topic":
                       "0x6319f8fd933eddbf5abdd0da1b4e133044a82[illegible]30bcf24a5df73b048b474c76cf",
                                       "event": "warning",
                                       "args": {
                                               "0": "Co2 emissions exceed limits",
                                               "message": "Co2 emissions exceed limits"
```

D
Event emitted when the input emission is above specified threshold of 400 ppm

FIGURE 15. Second test case smart contract transactions log.

TABLE 3. Smart contract cost analysis.

Function	Tx (GAS)	Fastest transaction fee (40 Gwei)		Fast transaction fee (37 Gwei)		Average transaction fee (30 Gwei)		Slow transaction fee (27 Gwei)	
		Ether	USD	Ether	USD	Ether	USD	Ether	USD
Deployment	351,602	0.0141	43.978	0.013	40.68	0.0105	32.984	0.0095	29.685
requestCo2Reading	25,332	0.001	3.1686	0.0009	2.931	0.0008	2.377	0.0007	2.139
getCo2 Reading	115,486	0.0046	14.4449	0.0043	13.3617	0.0035	10.833	0.0031	9.750



```
========== Results ===========
EVM Code Coverage:                       73.2%
Integer Underflow:                       False
Integer Overflow:                        False
Parity Multisig Bug 2:                   False
Callstack Depth Attack Vulnerability:    False
Transaction-Ordering Dependence (TOD):   False
Timestamp Dependency:                    False
Re-Entrancy Vulnerability:               False
```

FIGURE 16. Oyenete security tool smart contract test result.

The remix IDE provides the transaction Gas cost for each transaction; however, Gas price is not fixed compared to the fiat currency. Our CO_2 measurement smart contract functions transactions costs were calculated at the time of this writing (Ether = 3008.90 $) using the ETH Gas Station [88] and were found to be as shown in Table 3.

It is clear that the maximum cost is endorsed by the administrator of the smart contract at deployment. This is due to using a constructor at deployment that includes writing the carbon emissions threshold variable and storing it on-chain for the first time. The maximum deployment transaction cost incurred to achieve the fastest transaction is around $43.978 depending on the price of Ether. This price is considered reasonably economical to start up a measurement system that supports capturing and storing carbon emissions input readings.

The rest of the functions' fastest transaction costs fall below $15. These are fairly moderate costs to be endorsed by emitters each time they run processes, therefore, the smart contract operation is proved to be cost-effective in addition to its efficiency and reliability as previously demonstrated.

The above discussion proved that the developed smart contract is practical and efficient in terms of cost and security. The selected use case gave a candid demonstration of the IoT-blockchain integration process tackling all related aspects.

VIII. CONCLUSION

The significant advancement of the IoT systems opens the door for endless applications but unfortunately at increased risks and obstacles. More specifically, the growth of IoT networks brings up challenges related to security and trust in the generated and transmitted data. These issues exacerbate due to the adopted centralized IoT structure. With the emergence of blockchain technology, many researchers investigated the integration of blockchain with IoT to eliminate challenges and enhance performance. Blockchain's special features of security, trust, and decentralization address IoT issues through integration that comes in different architectures. However, the full integration requires an intermediate layer called oracle. The main purpose of blockchain oracles is to facilitate the transmission of authenticated IoT devices' data to the blockchain network and its smart contract. The need for oracles stems from the sophisticated and contained structure of blockchain that requires equally trusted and verified input data to interact with. Looking at the body of knowledge, it was noticed that no research covered the entire process of integrating IoT with blockchain including all its relevant and necessary parts. Therefore, our work utilized a two-aspect methodological approach consisting of literature review and designing and experimenting to provide a comprehensive view of the process of combining the important technologies of IoT, and blockchain through oracles. This study fills the gap in the literature and provides a key contribution to the world of research by discussing the role of blockchain and smart contracts in addressing the challenges facing the IoT paradigm. It further contributes by surveying and summarizing the main IoT-Blockchain combined system architectures found in the literature as well as shedding the light on the concept and functions of blockchain oracles and the main oracles that exist in the industry. Moreover, our research provides an illustrative CO_2 measuring use case where two oracle types, namely hardware and software oracles, were utilized to design an IoT-blockchain integrated system. The selected use case was implemented using Ethereum smart contract to create a carbon measuring and sensing system. The smart contract was developed using Solidity language and Remix IDE and its functionality was executed and tested successfully. Additionally, security and vulnerability analyses were conducted to ensure the robustness of the smart contract. The outcome of this study is providing a whole picture of the IoT-blockchain integration process covering all its aspects.

REFERENCES

[1] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things—A survey of topics and trends," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 261–274, Apr. 2015.
[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: https://doi.org/10.1016/j.comnet.2010.05.010.
[3] P. Shi, H. Wang, S. Yang, C. Chen, and W. Yang, "Blockchain-based trusted data sharing among trusted stakeholders in IoT," *Softw., Practice Exper.*, vol. 51, no. 10, pp. 2051–2064, 2019.
[4] E. Bertino, "Data security and privacy in the IoT," in *Proc. EDBT*, 2016, pp. 1–3.
[5] Y. L. Zhao, "Research on data security technology in Internet of Things," *Appl. Mech. Mater.*, vols. 433–435, pp. 1752–1755, Oct. 2013.
[6] Y. Lu, "The blockchain: State-of-the-art and research challenges," *J. Ind. Inf. Integr.*, vol. 15, pp. 80–90, Sep. 2019, doi: 10.1016/j.jii.2019.04.002.
[7] I. A. Omar, H. R. Hasan, R. Jayaraman, K. Salah, and M. Omar, "Implementing decentralized auctions using blockchain smart contracts," *Technol. Forecasting Social Change*, vol. 168, Jul. 2021, Art. no. 120786, doi: 10.1016/j.techfore.2021.120786.
[8] M. Samaniego, U. Jamsrandorj, and R. Deters, "Blockchain as a service for IoT," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Dec. 2016, pp. 433–436.
[9] S. Jani, "Smart contracts: Building blocks for digital transformation," Indira Gandhi Nat. Open Univ., New-Delhi, India, 2020.
[10] G. Caldarelli, "Understanding the blockchain Oracle problem: A call for action," *Information*, vol. 11, no. 11, p. 509, Oct. 2020.
[11] A. A. Sadawi, M. S. Hassan, and M. Ndiaye, "A survey on the integration of blockchain with IoT to enhance performance and eliminate challenges," *IEEE Access*, vol. 9, pp. 54478–54497, 2021.
[12] W. Mougayar, *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology* (Online Access With DDA: Askews (Economics). Hoboken, NJ, USA: Wiley, 2016.
[13] K. Sultan and R. Lakhani, *Conceptualizing Blockchains: Characteristics & Applications*. Ithaca, NY, USA: Cornell University Library, 2018.
[14] D. Macrinici, C. Cartofeanu, and S. Gao, "Smart contract applications within blockchain technology: A systematic mapping study," *Telematics Informat.*, vol. 35, no. 8, pp. 2337–2354, 2018, doi: 10.1016/j.tele.2018.10.004.
[15] K. Bheemaiah. *Block Chain 2.0: The Renaissance of Money*. Accessed: 2020. [Online]. Available: https://www.wired.com/insights/2015/01/block-chain-2-0/
[16] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, nos. 6–10, p. 71, 2016.
[17] A. Kawa and A. Maryniak, *SMART Supply Network*. Cham, Switzerland: Springer, 2019, pp. 2193–4614.
[18] J. F. Galvez, J. C. Mejuto, and J. Simal-Gandara, "Future challenges on the use of blockchain for food traceability analysis," *Trends Anal. Chem.*, vol. 107, pp. 222–232, Oct. 2018, doi: 10.1016/j.trac.2018.08.011.
[19] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Informat.*, vol. 36, pp. 55–81, Mar. 2019, doi: 10.1016/j.tele.2018.11.006.
[20] M. Swan, *Blockchain: Blueprint for a New Economy*. Sebastopol, CA, USA: O'Reilly, (in English), 2015.
[21] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, "A survey on the adoption of blockchain in IoT: Challenges and solutions," *Blockchain, Res. Appl.*, vol. 2, no. 2, Jun. 2021, Art. no. 100006, doi: 10.1016/j.bcra.2021.100006.
[22] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126927–126950, 2020, doi: 10.1109/ACCESS.2020.3006078.
[23] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, pp. 1–21, 1997.
[24] V. J. Morkunas, J. Paschen, and E. Boon, "How blockchain technologies impact your business model," *Bus. Horizons*, vol. 62, no. 3, pp. 295–306, May 2019, doi: 10.1016/j.bushor.2019.01.009.
[25] L. Yu, W.-T. Tsai, G. Li, Y. Yao, C. Hu, and E. Deng, "Smart-contract execution with concurrent block building," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2017, pp. 160–167.
[26] C. Sillaber and B. Waltl, "Life cycle of smart contracts in blockchain ecosystems," *Datenschutz und Datensicherheit*, vol. 41, no. 8, pp. 497–500, Aug. 2017.
[27] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.
[28] H. Surden, "Computable contracts," *UCDL Rev.*, vol. 46, p. 629, 2012.

[29] A. Kumar, R. Liu, and Z. Shan, "Is blockchain a silver bullet for supply chain management? Technical challenges and research opportunities," *Decis. Sci.*, vol. 51, no. 1, pp. 8–37, Feb. 2020.

[30] A. M. Saghiri, M. Vahdati, K. Gholizadeh, M. R. Meybodi, M. Dehghan, and H. Rashidi, "A framework for cognitive Internet of Things based on blockchain," in *Proc. 4th Int. Conf. Web Res. (ICWR)*, Apr. 2018, pp. 138–143, doi: 10.1109/ICWR.2018.8387250.

[31] G. Schmitt, A. Mladenow, C. Strauss, and M. Schaffhauser-Linzatti, "Smart contracts and Internet of Things: A qualitative content analysis using the technology-organization-environment framework to identify key-determinants," *Proc. Comput. Sci.*, vol. 160, pp. 189–196, Jan. 2019.

[32] S. Li and L. Da Xu, *Securing the Internet of Things* (Syngress Books). Amsterdam, The Netherlands: Elsevier, 2017.

[33] Y. Lu and L. D. Xu, "Internet of Things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2103–2115, Apr. 2019, doi: 10.1109/JIOT.2018.2869847.

[34] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014, doi: 10.1109/TII.2014.2300753.

[35] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, no. 10, pp. 533–546, Oct. 2016.

[36] M. Conoscenti, A. Vetro, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2016, pp. 1–6, doi: 10.1109/AICCSA.2016.7945805.

[37] D. V. Medhane, A. K. Sangaiah, M. S. Hossain, G. Muhammad, and J. Wang, "Blockchain-enabled distributed security framework for next-generation IoT: An edge cloud and software-defined network-integrated approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6143–6149, Jul. 2020, doi: 10.1109/JIOT.2020.2977196.

[38] J. Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu, and Y. Zhao, "EdgeChain: An edge-IoT framework and prototype based on blockchain and smart contracts," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4719–4732, Jun. 2019, doi: 10.1109/JIOT.2018.2878154.

[39] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud–edge–end in IoT: A blockchain-assisted collective Q-learning approach," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12694–12704, Aug. 2021, doi: 10.1109/JIOT.2020.3007650.

[40] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018, doi: 10.1109/MCOM.2018.1701095.

[41] S. K. Singh, S. Rathore, and J. H. Park, "BlockIoTIntelligence: A blockchain-enabled intelligent IoT architecture with artificial intelligence," *Future Gener. Comput. Syst.*, vol. 110, pp. 721–743, Sep. 2020, doi: 10.1016/j.future.2019.09.002.

[42] S. Krejci, M. Sigwart, and S. Schulte, "Blockchain- and IPFS-based data distribution for the Internet of Things," in *Service-Oriented and Cloud Computing*. Cham, Switzerland: Springer, 2020, pp. 177–191.

[43] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A blockchain-based lightweight framework for edge and fog computing," *J. Syst. Softw.*, vol. 154, pp. 22–36, Aug. 2019, doi: 10.1016/j.jss.2019.04.050.

[44] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic, "IoT public fog nodes reputation system: A decentralized solution using ethereum blockchain," *IEEE Access*, vol. 7, pp. 178082–178093, 2019.

[45] R. A. Memon, J. P. Li, M. I. Nazeer, A. N. Khan, and J. Ahmed, "DualFog-IoT: Additional fog layer for solving blockchain integration problem in Internet of Things," *IEEE Access*, vol. 7, pp. 169073–169093, 2019, doi: 10.1109/ACCESS.2019.2952472.

[46] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2018, doi: 10.1109/ACCESS.2017.2757955.

[47] S. Singh, I.-H. Ra, W. Meng, M. Kaur, and G. H. Cho, "SH-BlockCC: A secure and efficient Internet of Things smart home architecture based on cloud computing and blockchain technology," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 4, Apr. 2019, Art. no. 1550147719844159, doi: 10.1177/1550147719844159.

[48] G. Wang, Z. Shi, M. Nixon, and S. Han, "ChainSplitter: Towards blockchain-based industrial IoT architecture for supporting hierarchical storage," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 166–175, doi: 10.1109/Blockchain.2019.00030.

[49] S. N. Mohanty, K. C. Ramya, S. S. Rani, D. Gupta, K. Shankar, S. K. Lakshmanaprabu, and A. Khanna, "An efficient lightweight integrated blockchain (ELIB) model for IoT security and privacy," *Future Gener. Comput. Syst.*, vol. 102, pp. 1027–1037, Jan. 2020, doi: 10.1016/j.future.2019.09.050.

[50] Z. Bao, W. Shi, D. He, and K.-K. Raymond Chood, "IoTChain: A three-tier blockchain-based IoT security architecture," 2018, *arXiv:1806.02008*.

[51] S. M. H. Bamakan, N. Faregh, and A. ZareRavasan, "Di-ANFIS: An integrated blockchain–IoT–big data-enabled framework for evaluating service supply chain performance," *J. Comput. Design Eng.*, vol. 8, no. 2, pp. 676–690, Apr. 2021, doi: 10.1093/jcde/qwab007.

[52] M. Šarac, N. Pavlović, N. Bacanin, F. Al-Turjman, and S. Adamović, "Increasing privacy and security by integrating a blockchain secure interface into an IoT device security gateway architecture," *Energy Rep.*, vol. 7, pp. 8075–8082, Nov. 2021, doi: 10.1016/j.egyr.2021.07.078.

[53] Z. Shahbazi, Y.-C. Byun, and H.-Y. Kwak, "Smart home gateway based on integration of deep reinforcement learning and blockchain framework," *Processes*, vol. 9, no. 9, p. 1593, Sep. 2021, doi: 10.3390/pr9091593.

[54] H. H. Pajooh, M. Rashid, F. Alam, and S. Demidenko, "Hyperledger fabric blockchain for securing the edge Internet of Things," *Sensors*, vol. 21, no. 2, p. 359, Jan. 2021, doi: 10.3390/s21020359.

[55] S. Shukla, S. Thakur, S. Hussain, J. G. Breslin, and S. M. Jameel, "Identification and authentication in healthcare Internet-of-Things using integrated fog computing based blockchain model," *Internet Things*, vol. 15, Sep. 2021, Art. no. 100422, doi: 10.1016/j.iot.2021.100422.

[56] P. B, N. Deepa, Q.-V. Pham, D. C. Nguyen, T. Reddy, P. N. Pathirana, and O. Dobre, "Toward blockchain for edge-of-things: A new paradigm, opportunities, and future directions," *IEEE Internet Things Mag.*, vol. 4, no. 2, pp. 102–108, Jun. 2021, doi: 10.1109/IOTM.0001.2000191.

[57] L. Wu, W. Lu, F. Xue, X. Li, R. Zhao, and M. Tang, "Linking permissioned blockchain to Internet of Things (IoT)-BIM platform for off-site production management in modular construction," *Comput. Ind.*, vol. 135, Feb. 2022, Art. no. 103573, doi: 10.1016/j.compind.2021.103573.

[58] S. A. Latif, F. B. X. Wen, C. Iwendi, L.-L.-F. Wang, S. M. Mohsin, Z. Han, and S. S. Band, "AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems," *Comput. Commun.*, vol. 181, pp. 274–283, Jan. 2022, doi: 10.1016/j.comcom.2021.09.029.

[59] A. Ahmed, S. Abdullah, M. Bukhsh, I. Ahmad, and Z. Mushtaq, "An energy-efficient data aggregation mechanism for IoT secured by blockchain," *IEEE Access*, vol. 10, pp. 11404–11419, 2022, doi: 10.1109/ACCESS.2022.3146295.

[60] S. M. Alrubei, E. Ball, and J. M. Rigelsford, "A secure blockchain platform for supporting AI-enabled IoT applications at the edge layer," *IEEE Access*, vol. 10, pp. 18583–18595, 2022, doi: 10.1109/ACCESS.2022.3151370.

[61] D. Unal, M. Hammoudeh, M. A. Khan, A. Abuarqoub, G. Epiphaniou, and R. Hamila, "Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of Things," *Comput. Secur.*, vol. 109, Oct. 2021, Art. no. 102393, doi: 10.1016/j.cose.2021.102393.

[62] N. Kshetri, "Can blockchain strengthen the Internet of Things?" *IT Prof.*, vol. 19, no. 4, pp. 68–72, 2017, doi: 10.1109/MITP.2017.3051335.

[63] R. Van Mölken, *Blockchain Across Oracle: Understand the Details and Implications of the Blockchain for Oracle Developers and Customers*. Birmingham, U.K.: Packt, 2018.

[64] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain Oracle," 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. 1145–1152.

[65] A. Beniiche, "A study of blockchain Oracles," 2020, *arXiv:2004.07140*.

[66] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain Oracles: Review, comparison, and open research challenges," *IEEE Access*, vol. 8, pp. 85675–85685, 2020.

[67] B. Curran, "What are oracles? Smart contracts, chainlink & the oracle problem," Blockonomi, U.K., 2021. Accessed: May 2021. [Online]. Available: https://blockonomi.com/oracles-guide

[68] P. Sztorc. *Blockchain: The Oracle Problems*. Accessed: May 2021. [Online]. Available: https://www.infoq.com/presentations/blockchain-oracle-problems

[69] R. Berryhill and A. Veneris, "ASTRAEA: A decentralized blockchain oracle," IEEE Blockchain, USA. Accessed: Mar. 2021. [Online]. Available: https://blockchain.ieee.org/technicalbriefs/march-2019/astraea-a-decentralized-blockchain-oracle

[70] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: A decentralized Oracle and prediction market platform," 2015, *arXiv:1501.01042*.

[71] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 270–282.

[72] A. S. de Pedro, D. Levi, and L. I. Cuende, "WitNet: A decentralized Oracle network protocol," 2017, *arXiv:1711.09756*.
[73] Provable. *Provable Documentation*. Accessed: 2021. [Online]. Available: https://docs.provable.xyz/#background
[74] S. Ellis, A. Juels, and S. Nazarov, "Chainlink a decentralized oracle network," White Paper, 2017, p. 2018, vol. 11. [Online]. Available: https://link.smartcontract.com/whitepaper
[75] Z. Hess, Y. Malahov, and J. Pettersson. (2017). *Eternity Blockchain*. [Online]. Available: https://aeternity.com/aeternity-blockchainwhitepaper.pdf
[76] G. Caldarelli, "Real-world blockchain applications under the lens of the Oracle problem. A systematic literature review," in *Proc. IEEE Int. Conf. Technol. Manage., Operations Decisions (ICTMOD)*, Nov. 2020, pp. 1–6.
[77] A. Schaad, T. Reski, and O. Winzenried, "Integration of a secure physical element as a trusted Oracle in a hyperledger blockchain," in *Proc. 16th Int. Joint Conf. e-Business Telecommun.*, 2019, pp. 498–503.
[78] H. Al Breiki, L. Al Qassem, K. Salah, M. Habib Ur Rehman, and D. Sevtinovic, "Decentralized access control for IoT data using blockchain and trusted Oracles," in *Proc. IEEE Int. Conf. Ind. Internet (ICII)*, Nov. 2019, pp. 248–257.
[79] K. Mintz-Woo, F. Dennig, H. Liu, and T. Schinko, "Carbon pricing and COVID-19," *Climate Policy*, vol. 21, no. 10, pp. 1272–1280, 2021.
[80] C. Saka and T. Oshika, "Disclosure effects, carbon emissions and corporate value," *Sustainability Accounting, Manage. Policy J.*, vol. 5, no. 1, pp. 22–45, Feb. 2014.
[81] P. Wackerow. *Ethereum Virtual Machine (EVM)*. Ethereum Developer Resources. Accessed: Apr. 13, 2022. [Online]. Available: https://ethereum.org/en/developers/docs/evm/
[82] M. Ghosh. *Ethereum Moves Closer to Proof-of-Stake After Kiln*. Accessed: Apr. 10, 2022. [Online]. Available: https://forkast.news/headlines/ethereum-closer-to-pos-after-kiln/#::text=Ethereum's%20merge%20on%20the%20Kiln,the%20end%20of%20Q2%2C%202022
[83] Remix. *Welcome to Remix's Documentation!*. Accessed: 2021. [Online]. Available: https://remix-ide.readthedocs.io/en/latest/
[84] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, "Smart contract vulnerability analysis and security audit," *IEEE Netw.*, vol. 34, no. 5, pp. 276–282, Sep. 2020, doi: 10.1109/MNET.001.1900656.
[85] H. Ermawan. *Vulnerabilities and Attacks of Smart Contracts*. Accessed: Apr. 12, 2022. [Online]. Available: https://medium.com/hryer-dev/vulnerabilities-attacks-of-smart-contracts-9f112ea6c52c
[86] *GAS and FEES*. Ethereum Docs. Accessed: Jan. 19, 2022. [Online]. Available: https://ethereum.org/en/developers/docs/gas/
[87] Ethos. *What is Ethereum Gas?*. ETHOS, The People-Powered Blockchain Platform. Accessed: Jan. 19, 2022. [Online]. Available: https://www.ethos.io/what-is-ethereum-gas/#
[88] *E. G. Station*. Accessed: Jan. 19, 2022. [Online]. Available: https://legacy.ethgasstation.info/calculatorTxV.php

ALIA AL SADAWI (Member, IEEE) received the M.Sc. degree in engineering systems management and the Ph.D. degree from the American University of Sharjah, United Arab Emirates, in 2016 and 2022, respectively. She is currently a Researcher at AUS. Her researches are oriented around blockchain technology. She is a certified blockchain and smart contract developer. She is specialized mainly in Ethereum blockchain. She has implemented multiple DApps (decentralized applications) projects related to decision making, sustainability in smart city and blockchain application in supply chain, logistics and carbon trading, auctioning, and healthcare. She is experienced in blockchain oracles applications, especially using Chainlink. Her research interests include blockchain integration with the IoT and its application in smart sustainable industrial sector.

MOHAMED S. HASSAN (Member, IEEE) received the M.Sc. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2000, and the Ph.D. degree in electrical and computer engineering from The University of Arizona, USA, in 2005. He is currently a Full Professor of electrical engineering with the American University of Sharjah. He was involved in multiple projects related to free space optical communications, electromagnetic shielding, demand response and smart grids, anti-static flooring, and fiber optic sensors for infrastructure health monitoring applications in addition to EV wireless charging systems. His research interests include multimedia communications and networking, wireless communications, cognitive radios, resource allocation and performance evaluation of wired networks, and next generation wireless systems.

MALICK NDIAYE received the M.S. degree in quantitative methods in economics, optimization and strategic analysis from the University of Paris 1 Sorbonne, France, and the Ph.D. degree in operations research from the University of Burgundy, France. He has worked with the University of Birmingham, U.K., and the King Fahd University of Petroleum and Minerals, Saudi Arabia, before joining the American University of Sharjah. His recent scholarly work focuses on developing last-mile delivery routing solutions, vehicle routing optimization in cold supply chain, and the use of emerging technology to improve logistics systems. His research interests include operations research, supply chain, and logistics systems management. He is a Certified Supply Chain Professional from the American Association for Operations Management (APICS).