Blockchain-based Secure Data Transmission for Internet of Underwater Things

Abdul Razzaq (dr.razzaq@zju.edu.cn)
Zhejiang University https://orcid.org/0000-0002-4465-6365

Research Article

Keywords: Internet of Things, Blockchain, Smart Contract, Ethereum, Decentralized Storage, Privacy, Data Security

Posted Date: June 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1710078/v1

Blockchain-based Secure Data Transmission for Internet of Underwater Things

Abdul Razzaq[1*]

[1]Ocean Engineering and Technology, Ocean College of Zhejiang University, Zhoushan, 316000, Zhejiang, China.

Corresponding author(s). E-mail(s): 11934071@zju.edu.cn;

Abstract

Internet of Things (IoTs) is an integrated network collection of heterogeneous objects which enable seamless integration between systems, humans, devices, and various other things, to support pervasive computing for smart systems. IoT-driven systems and sensors continuously ingest data resulting in an increased volume and velocity of information which can lead to critical concerns such as security of the data and scalability of the system. The Internet of Underwater Things (IoUTs) is a specific genre of IoTs in which data related to oceanic ecosystems is continuously sensed through underwater sensors. IoUT has emerged as an innovative paradigm to support smart oceans. However, there are several critical challenges which IoUT system designers must consider such as (i) scalability of the system to handle large volumes of oceanic data and (ii) security of data that is transmitted from IoT sensors deployed underwater. Blockchain as a newly emerged technology and an enabling platform allows decentralized and secure transmission of data among a wide group of untrustworthy parties. This research aims to exploit blockchain technology to secure IoUT data transmission by exploiting Interplanetary File System (IPFS) method. Additionally, this study also addresses the system's scalability in two aspects, (i) scalability, and (ii) security. We used a case study-based approach and performed experiments to evaluate the proposed solution's usability and efficiency in terms of query response (i.e., performance), and algorithmic execution (i.e., efficiency). The proposed solution unifies blockchain technologies to secure IoT-driven systems and provides guidelines to engineer and develop next-generation of robust and secure blockchain-aided distributed IoT systems.

Keywords: Internet of Things, Blockchain, Smart Contract, Ethereum, Decentralized Storage, Privacy, Data Security

1 Introduction

The Internet of Things (IoT) orchestrates heterogeneous things such as systems, services, devices, and individuals to coordinate autonomously in smart systems and environments [1]. According to a recent estimate by Statista, by the end of 2020, there were 8.74 billion IoT-connected devices on the planet, with the number expected to rise to 16.44 billion by 2025 (i.e., practically doubled in the next five years) [2]. IoT systems are comprised of hardware sensors which employ software tools and wireless networks, as well as other internet-based technologies for communicating and data sharing with corresponding devices and systems [3]. The Internet of Underwater Things (IoUTs) is a subset of the IoT which refers to an operation in an oceanic environment, exploiting data from underwater sources and delivering it to off-shore servers for data-driven intelligence and human decision support [4]. IoUTs are envisioned as being the ears and eyes of the deep blue (sea), providing vital information by operationalizing scenarios such as monitoring marine life, assessing underwater pollution, and observing the relationship with various parameters e.g., temperature and acidity on water [5]. Although the significance of IoTs in the development and implementation of a wide range of smart systems and infrastructures is critical, so far limited research activities have been performed on using IoTs for smart ocean solutions [5] [6]. Despite the predicted revenues of IoTs and strategic benefits of IoUTs, such sensor-driven systems include some critical challenges, according to a recently published report on 'Smart Ocean Technologies' [7]. These challenges include but are not limited to, sensor resource poverty, wireless network stability, data security and privacy, and other factors which can prevent the trustworthiness and ensure the extensive utility of IoUTs. The proposed solution for secure data sharing falls short in such cases due to the amount of data collected, the variety of devices used, a lack of trust among participants, and a lack of transparency in data management [8].

Cloud servers, as distributed nodes for data processing and management, process and preserve large volumes of data [9]. Single-point failure [10] is one of the potential risks which come across with a central authority for data processing and management such as cloud servers. All third parties have been involved to offer cloud storage in order to avoid such a disaster [9]. A blockchain is a decentralized digital ledger of transactions which is hard to alter or compromise. Blockchain technology is used to offer trust and transparency in the creation of a trust-based paradigm, removing the need for an intermediate third party [11]. As part of blockchain-based solutions, decentralized storage, also known as distributed ledger storage, is a system which allows users to store their data on several nodes of a network independently. The problem is that network nodes are limited in terms of storage and processing power [12]. IPFS (Interplanetary File System) which is based on peer-to-peer architecture is being used for limitations in the processing of network nodes and storage problems [13]. Blockchain is not only used in financial applications but also in non-financial applications which require secure data management

and transmission [14]. The risk of critical data (healthcare, personal identities, etc.) being misused or misinterpreted is the fundamental underlying issue in research data sharing [15]. It is due to the fact that data-sharing methodologies are still in their infancy in terms of the trust, which is gradually being developed in the research community. Various solutions have been offered to address this issue, such as the safeguarding of every individual's identity [16] and regulated access to data rather than providing open access to all data [17]. Despite the offered solutions, these systems are unable to guarantee trust, digital data immutability, or data usage traceability [18]. Recent research and development efforts have made technological advancements in the last decade to adapt data-sharing approaches, collaboration, and intelligent decision-making strategies which can boost research-based efforts [16]. However, it is critical to understand the methods, modes, and timings for transmission of such data. By utilizing the capabilities of blockchain [16], this research aims to offer safe data sharing and sale in the future.

Several types of research activities have been carried out with the aim to link IoT systems to a blockchain platform to tackle the challenge of secure data exchange between sensors and systems [8] [19]. The majority of the existing solutions adopt hybrid approaches for data storage and transmission like a cloud provider that maintains the data but the blockchain provides features such as integrity and trusted distribution of data [9]. Blockchain is a key decentralized and equally distributed system which assures the secrecy performance of the IoT and IoUT. It's a distributed ledger since all of the blocks are linked together. It can keep track and arrange transactions for billions of IoT devices, as well as data storage [20]. The most significant benefit of blockchain technology is its ability to decentralize data while featuring a peer-to-peer transaction with decentralized credits within a distributed network. It reduces costs and solves the major problem of unsecured data storage in centralized businesses. To investigate this, some key characteristics of blockchain decentralization such as asymmetric encryption, smart contracts, and access management were examined [21], including their possible applications to the IoT data-sharing platform to improve and secure the data. Finally, we present blockchain as an excellent and appropriate approach for the development of IoT and IoUT data.

In Interplanetary File System (IPFS), as a peer-to-peer content-based protocol, it works with smart contracts and data from blockchains, each IoUT dataset is assigned a cryptographic hash to make the text unchangeable [13]. Moreover, IPFS integrates an encryption mechanism to the hashes of submitted data on IPFS ensuring data security. When uploading to IPFS, the datasets are encrypted using 256-bit encryption, and the returning hash is also encrypted with a crypto algorithm signature-based approach. Another benefit to using IPFS, the storage is minimized by increasing the download speed of IoUT data, sharing huge volumes of data without duplication, and reducing bandwidth expense. To store data in a file, a structure consisting of data and connections called an IPFS object is used. For data larger than 256 KB, they

are split up and stored in the form of multiple IPFS objects, with a single empty object connected to the IoUT data file objects. IPFS is widely accepted as it supports a hash string route for data transfer and altering a hash value will impact the hash value. A hash could be used to access IoUT data at any time as the paths work in a similar approach to the standard universal resource locator on the internet.

Data security is achieved by integrating encryption scheme Advanced Encryption Standard (AES) and RSA

$$f(x) = \sum_{i=0}^{d} c_i \prod_{\substack{j=0 \\ j \neq i}}^{d} \left(\frac{x - p_j}{p_i - p_j} \right) \tag{1}$$

$$w \mapsto \sum_{i=0}^{7} \lambda_i w^{-2^i}.$$

This formula for the modified S-box may then be used to generate an expression for an AES encryption using a kind of continuing fractions. As a result, after five AES cycles, each byte of the state space, $\mathcal{S}_{i,j}^{(5)}$, is given by [22]

$$\mathcal{S}_{i,j}^{(5)} = K + \sum \frac{C_5}{K^* + \sum \frac{C_4}{K^* + \sum \frac{C_3}{K^* + \sum \frac{C_2}{K^* + \Sigma \frac{C_1}{K^* + p_*^*}}}}},$$

RSA: Bob computes the residual when m^e is divided by N to encrypt a message m. Encoding: $M = m^e \bmod N$. The square-and-multiply approach described below is an effective way to accomplish this. To decode the message M, The values p and q are used by Alice. d is calculated once N and e have been chosen: Decoding exponent: $d = e^{-1} \bmod (p-1)(q-1)$. The extended Euclidean Algorithm can efficiently compute this inverse. Alice then computes the message's decoding. Decoding: $m = M^d \bmod N$ [23].

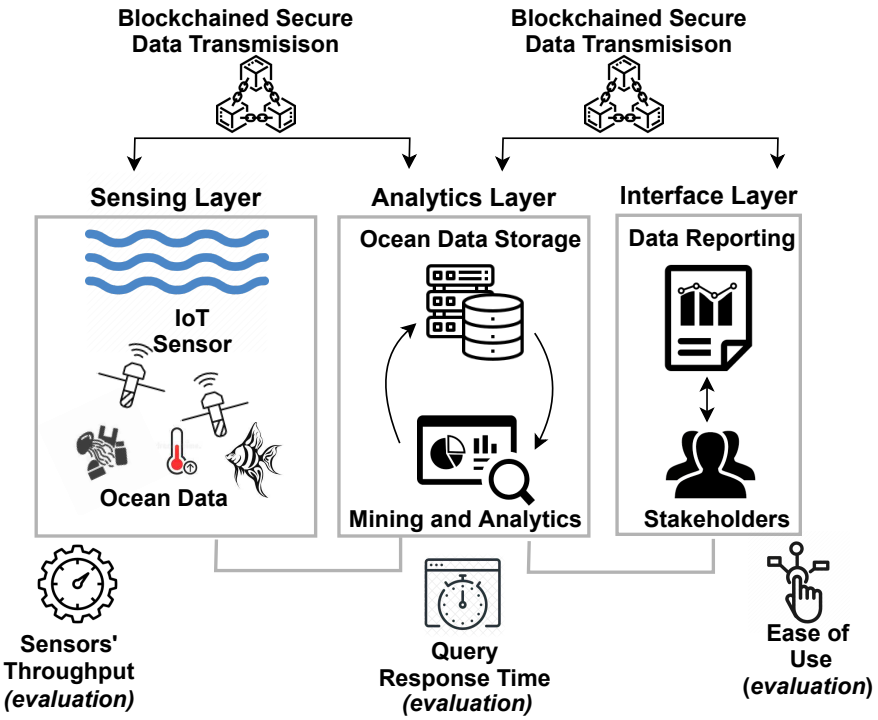

Fig. 1 Overview for the proposed Model

Figure 1 illustrates a high-level visual of the proposed solution, which streamlines the use of supplementary tool assistance to automate system development. As shown in Figure 1, the system has presented the structure of our developed platform, which guides system developers to maintain each module of abstraction, i.e., steps of module layering provide for the structuring of distinct operational features at distinct layers of the system, as well as the modularization of algorithmic specifications. The main objective of this research is to provide a secure decentralized platform for sharing IoUT data while likewise addressing privacy, security, and access flexibility problems". The following are some of the contributions that this research can make:

- Leveraging IoT-driven data sensing and blockchain technologies in the context of the Internet of Underwater Things for the efficient and secure transmission of oceanic data.
- Algorithms that provide automation and parameterized customization to ingest sensor-driven data (via IoTs) and its secure transmission (utilizing blockchain technologies).
- Case study-based validation of the solution for evaluating the system's quality in terms of sensor throughput, query response, and algorithmic execution.

This paper is structured as follows: The research context and methodology are presented in Section 2, while the related work is expressed in Section 3. In Section 4, the suggested scheme as a framework and smart contract design were explored in terms of algorithms, technology implementation, and system models. Section 5 provides the specifics of the evaluation and the simulation results. Finally, Section 6 concludes the paper.

2 Research Context and Methodology

This section includes research context with the methodology to assist in putting the building blocks of an IoT data sharing system based on blockchain technology into context. It also explains the use of software engineering to construct IoUTs.

2.1 Internet of Underwater Things

Figure 3 demonstrates the building blocks of IoUTs in an illustrated manner. The underwater things are a collection of wirelessly linked sensors (SA, SB, ..., SN) positioned at various locations, as well as a bridge node, the Scientific Instrument Interface Module (SIIM), which is wired or wirelessly connected to the backend server. Each sensor S captures a specific type of oceanic data, adds sensor identity, and location information, and sends it to the bridge. The bridge collects data from all sensors and operates as a link between the sensors (data collecting) and the server (data processing) such as data storage. Oceanic data can comprise a wide range of information, such as pollution types and levels, sunlight, temperature, dissolved oxygen, turbidity, conductivity, pressure, pH value, and water acidity. The bridge simply packages data from all sensors and sends it to the backend server. The server is controlled as a cloudlet, as shown in Figure 3, to send data for offshore processing and storage. Offshore processing collects and merges all data from relevant sensors into excel and performs the appropriate analyses in terms of computation. Finally, the data is delivered to end-users in the form of an excel dataset over a secure channel (encryption and decryption methods) to improve human decision-making in the context of smart oceans.

2.2 Challenges for IoUT System Design

Designing, operationalizing, and deploying underwater sensors despite the strategic capabilities of IoUTs, remains a difficult process. Pervasive sensors have various constraints at the hardware, network, and software levels [5] [6], as evidenced by the deployment of smart ocean technologies. The widespread and transportable nature of the sensors has resulted in a lack of computing, storage, and energy resources in terms of hardware. Instability in the network causes frequent disconnections and declining sensor throughput which can cause abnormal data transfer. Furthermore, the performance of IoT data analytics in terms of algorithmic efficiency and query processing are among the

critical concerns to be addressed from a software perspective [24]. Architects and developers can use the software engineering principles to (i) design IoUTs with blockchain technology by using processes for incremental and reusable development [24], (ii) develop parameterized algorithms to customize the solution [5], (iii) automate the solution using software tools and technologies [25], and (iv) evaluate system functionality and quality.

2.3 Research Method

Here we describe the research methodology, which includes the design specifics for the proposed solution. Figure 2 illustrates a summary of the research technique, which consists of four steps that follow an incremental approach to assess, design, execute, and validate the solution, as described below.

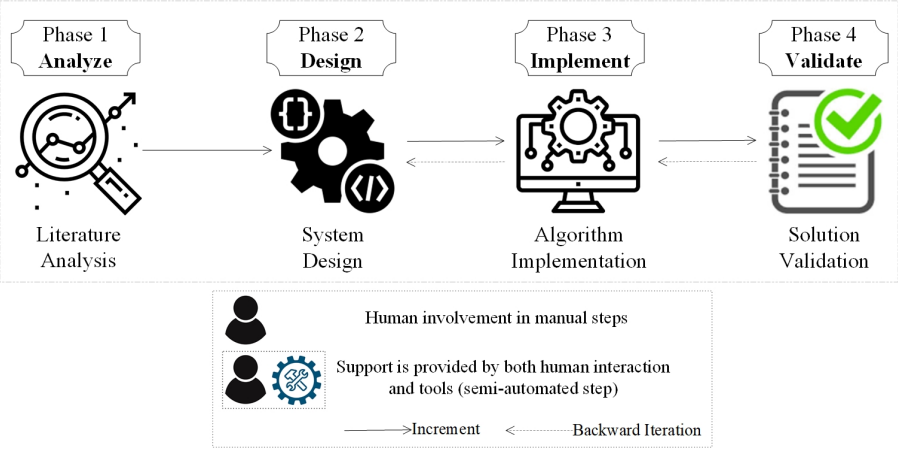

Fig. 2 An illustration of Research Methodology

- Phase 1 – The first step in the process is to conduct a critical analysis of a wide range of existing literature (e.g., peer-reviewed published research, technology road maps, technical reports, etc.) [26] [27] [28] in order to identify existing solutions and their limitations. To perform a systematic literature review [29] we must follow the recommendations to review the most relevant existing studies. We were able to streamline the desired solutions and specify the scope of this research by analyzing existing research and development solutions.
- Phase 2 – The design phase of a methodology seeks to model the solution before it is implemented, and it is the key element to design a software system. To model IoT systems, we have followed the standards and recommendations from [30] to design the proposed solution. According to a software design viewpoint, previous work has defined blockchain as a software connector [31] which offers a decentralized data storage and program

running infrastructure (known as smart contracts). Blockchain has particular features such as immutability, non-repudiation, the validity of records, openness, and fair rights.

- Phase 3 – Algorithm implementation provides the execution of a solution in the form of computational and storage-intensive phases. The algorithmic solution is a modular deconstruction of a solution which can be changed by the users by using parameterized inputs. Algorithmic intricacies and underlying source code result in the design of executable standards.
- Phase 4 – The final step, Validation of Solution, evaluates the functionality and quality of the proposed solution to assess the entire system's quality. In particular, we concentrate on evaluating a variety of aspects of system usability and efficiency by using a set of well-established evaluation measures. Although blockchain technology is an opportunity to incorporate decentralized components, currently there is a limited general understanding of the effect of design decisions in this field on software quality characteristics such as protection, management, efficiency, or cost. As in Figure 2, the first two steps are entirely manual tasks which require human intelligence and decision. In contrast, the last two processes require human intervention and tool support to (semi-) automate solution development. Solution validation, for example, may advise algorithm refinement to improve efficiency or change functionality.

3 Related Research

This section highlights the most important relevant work in order to evaluate current solutions, their underlying methodology, and limitations in order to justify the proposed solution's scope and contributions. We also evaluate and objectively compare the most relevant existing research in order to justify the scope and contributions of the suggested model.

3.1 IoTs for Mining Oceanographic Data

In terms of the smart systems, traditional oceanographers are utilizing new tools for collecting synoptic measurements of ocean surface phenomena at previously unimaginable measurement points scales [32]. Oceanographers have gathered information on critical variables such as the sea surface, temperature, global high geographical and temporal resolution, Chlorophyll, sea surface height (SSH), and ocean circulation across time. The Ocean-of-Things (OoT) endeavor is a project of technology development and research that aims to fill the gap of knowledge in the maritime industry [33] via installing thousands of low-cost, intelligent vessels across broad ocean expanses that operate as a distributed sensor network. Osen et al. [34] present a method for coordinating IoT data to enable a web-based platform for data collection, retrieval, interpretation, and visualization of oceanic data from an IoT perspective. The web-based platform was developed to deliver distributed, interactive real-time data streams for locating underwater oil detectors. However, the capture,

transmission, and retrieval of data in the underwater marine environment have raised privacy and security concerns. The authors of [35] presented an IoT-based architecture for a secure and efficient data compression algorithm to address these concerns.

3.2 Blockchain-based Secure Data Communication for IoT Systems

Blockchain enables coordination between devices, exchanging information and data in the decentralized peer-to-peer networks generated through IoT devices. To update data information is delivered to the whole network, and any key decision is made with the approval of users in the majority rather than a single centralized server and is broadcasted to the whole network. Blockchain creates a transparent architecture that reduces the odds of a false input. In case of IoT solutions based on blockchain, attackers attempting to get control of the centralized server in order to steal personal information or take over the whole system will be unable to cause any risk to the network. Moreover, by implementing blockchain, the additional cost of IoT server security checking can be effectively minimized [36]. Furthermore, in the event of a malfunction, blockchain technology keeps IoT device data in a format which grants permission to track easily.

Researchers are attempting to apply blockchain on a larger scale because of its unique characteristics. A blockchain provides a safe, dependable, and trustworthy platform for data sharing [8]. It keeps track of any unwanted data access and hence offers traceability. However, because of its scattered structure, networks' control capabilities are harmed. Furthermore, the immutability of blockchain creates the risk of a network major attack [19]. The IoT data sharing schemes based on blockchain have flaws, such as security concerns, costly maintenance, and a huge amount of data from IoT devices. Authors in [19] presented a secure data transfer technique to address the aforementioned problems. A data consensus method, which overlooks the requirement of power data security and hence it is recommended only for small-scale applications, is used to save the data in a dynamic linked-based storage system. [7] Kokoris-Kogias et al. proposed CALYPSO which stores data on-chain and access control restrictions are enforced by a collective authority formed via the blockchain for auditable private data sharing. However, for the massive amount of data originating from the large collection of IoT devices, this technique is ineffective. Shafagh et al created a system in which the user has control by setting policies of sharing time-series IoT data and by issuing transactions to create policies for each time data is exchanged with a third party [37].

3.3 Challenges for Blockchain-based Systems

In the case of blockchain, there are several challenges in managing massive volumes of data on network nodes; storage since the inability to store very large files limits the capability of the terminal node, the expensive cost of processing

bulks of data, and computational capacity. Considering these issues, Steichen et al. presented a decentralized storage method based on IPFS and access control mechanisms in [38]. On each node, files are organized into pieces. A file, on contrary, cannot be viewed until the appropriate rights are provided to users. It is a smart strategy for keeping confidential data private. Due to blockchain interaction, the suggested schemes experience delays while obtaining files from the server and do not support real-time data storing. Maintenance (firmware or software upgrades) and network connectivity are both expensive, making it difficult to reap the benefits of IoT services that have been deployed. Using a typical security-by-obscurity method to ensure security and privacy in today's IoT systems is very complicated. In terms of security and transparency, a more practicable approach does not require full confidence in the internet because the system's flaws are public and can be confirmed.

Shangping Wang et al. presented a decentralized storage method based on IPFS and sharing mechanisms in [11], but this approach does not support securing the secret key. In [39], Fahad Ahmad Al-Zahrani et al. proposed a multichain mechanism to control blockchain access, but they only used the private and public key concept, which is not feasible in the context of data exchange without a secret key or without securing a secret key because data is not encrypted with a secret key. In [40], Bao Le Nguyen et al. utilized a different technique to protect a user's file in the cloud, but they simply employed file cyphertext encryption to save the file and saved the record in a blockchain ledger, but they did not secure the secret key. Parminder Singh et al. presented a cloud-based cross-data exchanging platform in [41], but sharing data is not secured, just verifying the user or organization to exchange the data.

Table 1 is a structured catalog that is a criteria-based comparison between the existing and proposed solution in order to illustrate the scope and contributions of the proposed work objectively. We used the existing study classification recommendations (IoUTs [6], Blockchain [16] [21]) to narrow down thirteen criteria that include (1) Data Encryption techniques are used to secure the data transmission on a network, (2) Source of Data Storage present the type of data storage like decentralized storage and on-premises storage (MSSQL) (3) Large Data Support to handle the big data, the cost does not affect due to use of decentralized storage and local storage on machine (4) Database Management shows that system is centralized or decentralized, but the system is using both type of storage decentralized and also centralized (5) Server Attack Resistance, decentralized storage is more secure from the several attacks than centralized (6) Roles Customization to grant or revoke the roles to any user, (7) Uploading Process Automation, user just make a query to access the custom data, but the data comes in the form of cipher based on AES encryption of 256bit with dynamic secret key, (8) Access and Controls Management provides the flexible system which allow to manage the roles dynamically and secure the accessing level (9) Dynamic Secret Key is created by the use who could be custom data user or data owner, the user set the dynamic key for sharing each dataset or accessing custom data, (10) Securing Secret Key with Double Encryption

Table 1 Existing Research vs. Proposed Solution: A Criteria-based Comparative Analysis

Schemes	[11]	[39]	[40]	[41]	**Proposed**
Data Encryption	Yes (Single)	Yes (Single)	Yes (Single)	Yes (Single)	Yes (Double Encryption)
Source Data Storage	Immutable IPFS Storage	Blockchain node	Cloud	Cloud	Immutable IPFS Storage
Large Data Support	Yes	No	Yes	Yes	Yes
Database Managing	Decentralized	Centralized	Centralized	Centralized	Decentralized
Attack Resistance on the Server	Yes	No	No	No	Yes
Roles Customization	No	No	No	No	Yes
Uploading process automation	No	No	No	No	Yes
Access & Controls Management	Yes	No	No	Yes	Yes
Dynamic Secret Key	No	No	No	No	Yes
Securing Secret Key with Double Encryption	No	No	No	No	Yes
Multi Storage flexibility	No	No	No	No	Yes
No Involvement of Third-party Authentication	Yes	No	No	No	Yes
Custom Data Filtering & Securing	No	No	No	No	Yes

this mechanism is used with RSA technique to secure the dynamic secret key which can be only decrypted by the requester, (11) Multi Storage flexibility is that system enables the different types of storage, MSSQL Server is used to store the data from the sensors as on-premises server which is accessed securely based on symmetric techniques, the dataset is stored on decentralized storage by data owner or custom data user from the on-premises database and the

transactions records are save in third storage is called blockchain ledger, (12) No Involvement of Third-party Authentication, (13) Custom Data Filtering and Securing, system is more flexible to connect the on-premises databases to access the data based on custom query through secure channel. To design and evaluate IoUT systems that offer real-time data collection and analysis. The system, however, lacks data security and privacy. Based on the overall comparison criteria in Table 1, we can conclude that blockchain has been used in a number of research studies for IoUT data exchange in the context of smart oceanic systems.

4 Solution Overview and Algorithmic Specifications

This section discusses the framework, including the algorithms and technologies that were employed to achieve the goal. The implementation technologies section is included to complement the algorithmic requirements by streamlining the tools and frameworks which software and system developers can consider to efficiently implement the proposed solution.

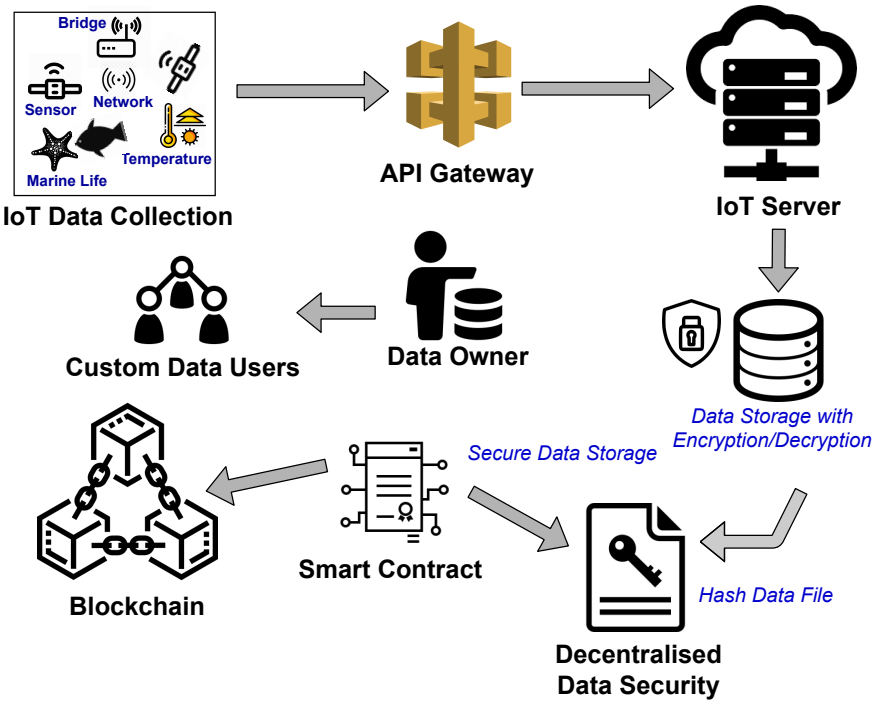

Fig. 3 A brief outline of the suggested solution

Table 2 Deployed sensors list

Sensors	Intent
Temperature (C)	To read the current temperature
Dissolved Oxygen (mg/L) (DO)	To measure the dissolved oxygen in the water
pH (moles/L)	To measure the acidity or alkalinity of the water
Salinity (ppt)	To measure the "saltiness" of seawater
Turbidity (NTU)	To measure the amount of light that is scattered by suspended solids in water.
Chlorophyll (mg/L)	The fluorometer gives readings of levels of Chlorophyll in water.
Sea Level	To measure the depth.

Figure 3 represents an abstract level summary of the proposed system. This model shows all the used tools with encryption techniques. The communication between all modules and stakeholders is clearly visualized. Each module in the system design clearly shows the concept of IoUT in terms of data and its computation. For example, the data sensing module corresponds to capturing, representing the data which is ingested from underwater things, transmitted to the server, and also data saved in the database. System design facilitates the developers to design and evolve the system while abstracting out certain implementation details (i.e., algorithmic specifications) which can be provided with appropriate tools.

4.1 Framework Layers

A brief execution flow for the developed system is visualized to demonstrate system's operation.

4.1.1 System Actors

We have seven actors by demonstrating the system flow (User, Web, Data Owner, RSA, AES, IPFS, Smart Contract) including two humans and five systems. Before using this system, firstly the user must apply for registration. When a user requests to join this system, the request is bundled with all essential information and saved in a blockchain ledger by using a smart contract. All requests will go to the Admin, who is in-charge of the User Management control panel. Admin has authority to grant or deny the user's request to access any role (Custom Data, Data Owner, Admin). A user can utilize the system after being allocated a role and successfully activating it. Users are not required to log in because the system confirms their identity automatically when they access the platform by using their blockchain address. After granting access to the user as a Custom Data; can query the database for data using a custom query (date, sensor(s), location(s). This query filters data from the database and writes it in excel form, after that the encryption method picks the excel file and encrypts it with a secret key provided by the user. After successfully uploading, IPFS returns the dataset file hash key, which is kept in

the blockchain ledger along with other associated parameters with the assistance of a smart contract. With the binding of the file hash key of a dataset, this list from the blockchain ledger is provided to a user on a custom data list page. The dataset can be downloaded from the custom list, the file hash key is sent to the IPFS server, and the encrypted dataset file is being downloaded. Later, the user has to choose this file by entering the secret key to decrypt it. The data owner has the option of sharing his datasets with the public or uploading them privately to decentralize them. After the data owners upload the dataset and pass the secret key; the rest of the procedure is to decentralize the dataset on IPFS after encryption. If the data owner makes a dataset which is available for public and private access it will appear on the list, where the user can access the request. To get access to a public dataset, initially, a user requests a specific dataset, which must be accompanied by a User Public Key which needs to be generated by the system using the RSA algorithm and also sent a Private Key to the user. This request is sent to the data owner, who decides to grant or deny the access. If the permissions are granted, then the data owner will be enquired to provide a secret key, that will be encrypted with the requestor's Public Key, and the user will be notified of the response to download the dataset. The user will decrypt the secret key by providing a Private Key and then decrypt the downloaded dataset with a secret key.

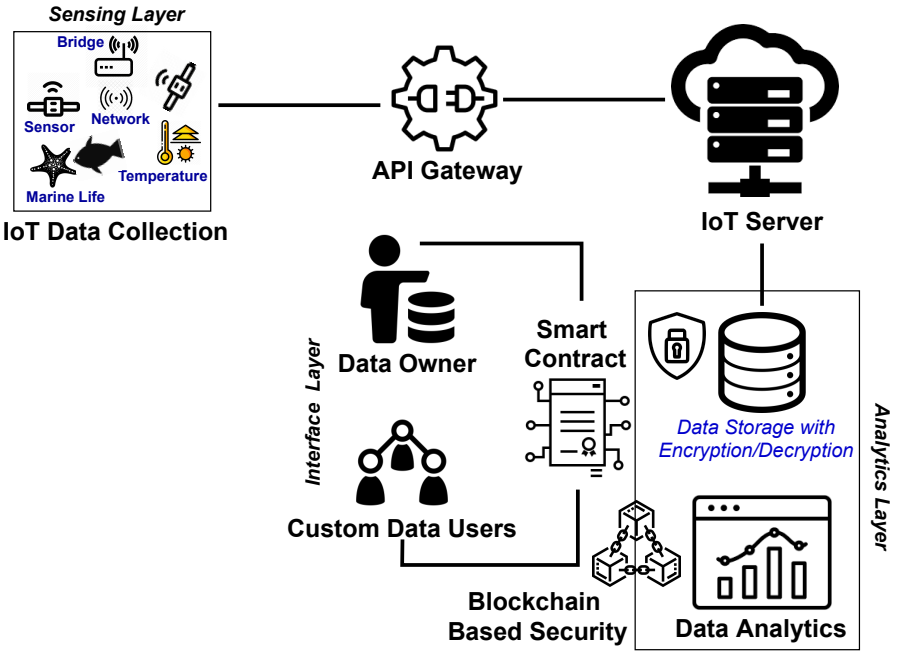

Fig. 4 An overview of the detailed solution

Figure 4 shows two views of the domain and system design across multiple modules of the solution. Particularly, the domain view allows filtering of the data from the database of the system in terms of several functional features and system building elements. The system's processes are representing computing and storage-intensive units, while connections represent component interconnections. The modules set of processes represent a logical separation of different functional aspects of the solution in an engineering process, which we have discussed next.

4.1.2 Data Sensing Layers

As per technical perspective, the data sensing module deals with the collaboration of data by the deployed sensors in the sea. Figure 4 shows data sensing with different deployed sensors (Sensor-ID) and includes their data collection. Each deployed sensor has its own unique identifier referred to as Sensor-ID A or Sensor-ID B for the collection of different types of oceanic data. The list of sensors already highlighted in Table 2 and presents the collected data type, unit for data representation, and the individual sensor is being deployed for data collection. For example, the sensor referred to as Sensor-ID A collects data termed DO, which is used to measure the dissolved oxygen in the water. Sensor deployment and data collection are supported by Scientific Instrument Interface Module (SIIM), acting as a bridge between sensors (data collection) and data server (data management). SIIM as a bridge, between two layers, unifies hardware with its control software to accumulate data collected from the deployed sensors, packages it with SIIM information, and transmits it to the server. In Figure 4, the SIIM collects data from Sensor A as dissolved oxygen at a specific time (Dissolved Oxygen (DO): 7.85, DateTime: 20-02-21::13:05:37) and packages it with SIIM identity and geo-location of collected data (SIIM-ID X, GeoLocation). The process of data collection and transmission both are periodic and take place on the basis of time intervals in minutes.

4.1.3 Data Filtering, Securing and Decentralization Layer

This module primarily focuses on filtering and securing the data from the database server which collects the data from the underwater sensors (i.e., data transmitted from SIIM). The main two types of data securing algorithms are performed on the basis of types of data including (i) historical data determined by specific data collected and said custom data. the custom data type is derived from an IoT sensor database. The Custom Data user makes a query by selecting parameters (Date, Sensor, and Location) to filter data that is stored on IPFS in an encrypted excel file by providing the dynamically secret key. (ii) data owner to share self-datasets to public or private. Moreover, the data owner has the right to grant permission to any user or revoke access from any user for his datasets. The data sharing stakeholder that owns the data will be shared with consumers as public data with double encryption. To give access to the dataset to the requester, the secret key is encrypted by using the RSA signature method with the requestor's public key and can only be decrypted

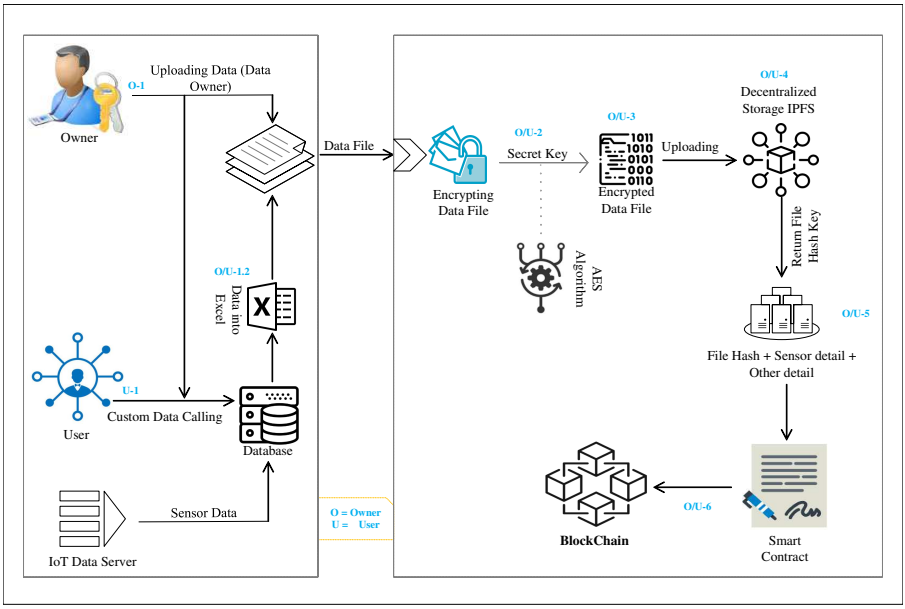

Fig. 5 Data storing process

with the requestor's private key, which is generated with the public key during the request. In both types (Custom Data and Data Owner), data is stored in decentralizing storage of IPFS.

4.1.4 Data Presentation

The last module in the proposed solution presents key insights and results of data to the end-users through a secure channel. End-users are stakeholders including ocean explorers and others interested in analyzing oceanic data. This module is also named the user interface layer which is developed as a web system to generate customized reports and visualizes various statistics that empowers end-users (stakeholder and decision-makers) with their decision. For example, the presented visualizations can help users to view the level of dissolved oxygen, underwater temperature, and water acidity over a specific period of time. As described in Figure 4, this interface layer uses the web template [42] design for front-end interface style after customizing the design, the system component named Dashboard which helps end-users to create views of the data in form of excel file. Users can make a request to be a Custom Data User or Data Owner. If the user is a custom data user then it can access the historical data up to date using different filters (Date range, sensors, and locations based), but the data owner user can only use the functionality as Custom Data User and also share their datasets with different access roles to a user (see Figure 6 of data accessing process).

Figure 5 depicts a process for saving IoT data on the blockchain, using a smart contract as a method of storing IoT data. There are two types of

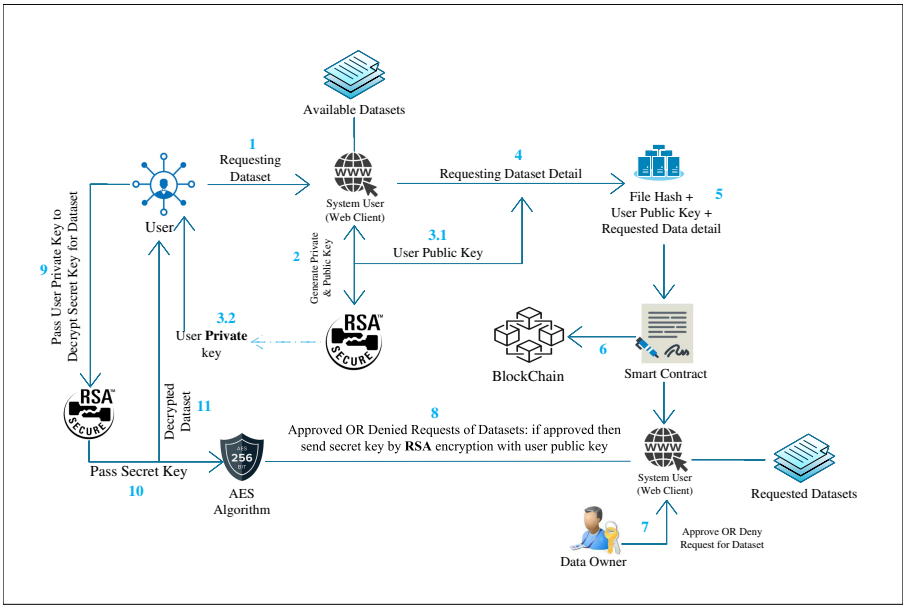

Fig. 6 Data accessing process

data uploading categories on the platform: Custom Data and Data Owner. Data owners can upload their datasets and custom data that comes from the database in excel format. Both are kept on IPFS in encrypted form and saved in the blockchain ledger as a record with a file hash.

4.2 Algorithms

Figure 7 demonstrates algorithmic steps in terms of computational elements, data storage operations, and flow of the control. We illustrate the execution of algorithms as three layers to streamline the consistency between the proposed architecture (Figure 4) and its implementation shown in Figure 7.

Algorithm 1 User Management: The oceanic data elements from Table 2 act as parameterized input to algorithms to support customization and user input. For example, to trigger data collection from a specific sensor, the sensor identity In is passed as a parameter to Algorithm Input: In at Line 01. In the rest of the section, we present each of the algorithms, across all modules, as far as inputs, processing, and outcomes are concerned. For clarity and readability, comments are added to all the algorithms (e.g., >Verify Admin Blockchain Address, Line 4). The functionality for User Management is seen in Algorithm 1. This algorithm is used to handle the users who have expressed an interest in joining the platform. After submission of the user request for registration, the request is sent to the admin panel that manages the users. Admin has the power to grant, remove, and block the user's permissions.

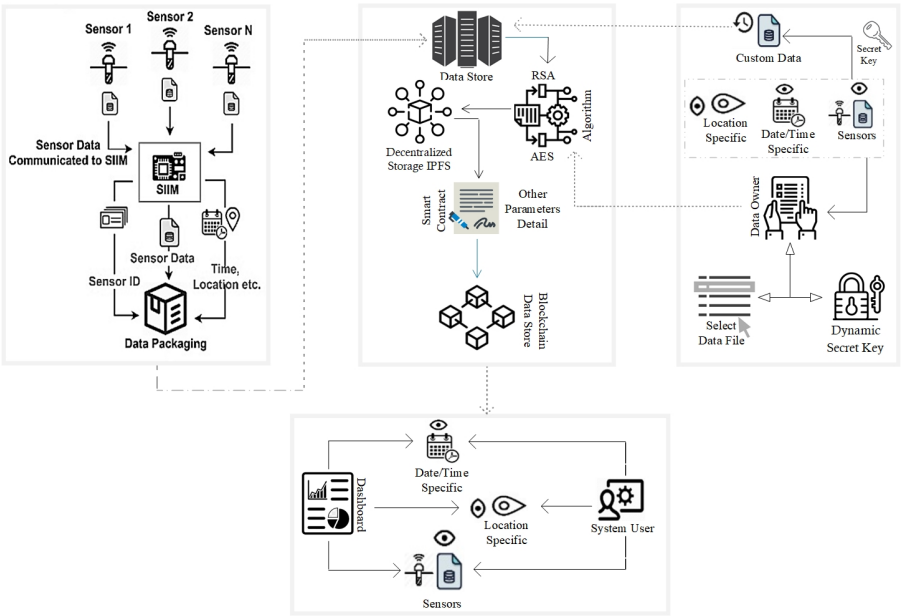

Fig. 7 A envisioned overview of the algorithms

- *Input(s)*: The input to the algorithm is the User's blockchain address, select role, and status for an account to be activated now or later.
- *Processing*: first verify the admin address if it is valid then the admin can take an action against a user to register or block. Admin passes the input and user's address parameter which is checked before further processing whether it is already available or not. If not, then the user is added to a new role, if it is already available, the user is updated, and stored in the blockchain using a smart contract.
- *Output*: The output from the blockchain ledger is as successfully granted or revoked by the user.

Algorithm 1 User Management

```
Input: Δwp, γ, Φ                                   ▷ User Address, Role, Status
Output: bool
procedure UserRoles
    if msg.sender == Admin then                    ▷ Verify Admin Blockchain Address
        if Δwp != Exist then                       ▷ Check User Address already not exist
            AddUserToRole (Δwp, γ, Φ)              ▷ Role Custom Data/Data Owner OR Blocked
        else
            UpdateUserRole (Δwp, γ, Φ)             ▷ Update User Role
        end if
    else
        UnAuthorized;
    end if
end procedure
```

Algorithm 2 Decentralizing and Securing the Data: The functionality of uploading data is illustrated in algorithm 2 and described in this section. First, we check whether the user is authorized to visit this page or not. There are two types of data uploading and securing possibilities for this method. The initial data uploading scenario is Custom Data, which allows any registered user to filter data from the database depending on a number of variables including date range, sensor/s, and location/s. The second sort of data uploading is called Data Owner, and it allows users to share their own datasets, which can be private or open to the public. The data is only viewable to the data owner if the dataset is set to private access; however, if the dataset is set to public access, the data is visible to anybody for open access. A public dataset does not imply that it is available for immediate download by anyone. Public datasets are those that are visible to all users and can be accessed by contacting the owner of datasets. If the data owners agree on the access to their datasets, then the user receives the secret key from the data owner, which is used to decrypt the dataset. Both types of datasets (Custom and Data owner datasets) are uploaded to IPFS storage to decentralize the dataset, but only in encrypted form with a secret key. The algorithm is used to upload data to IPFS and save the file hash in the blockchain ledger using smart contracts, along with mappings of various other parameters (see Figure 5). A hash of the uploaded dataset is used for the mapping of various parameters (Name, Sensor, Location, Uploading Type, Date, Uploading By, etc.).

- *Input(s)*: The input to the algorithm is sent Sensor/s, Location/s, Secret Key, and Date.
- *Processing*: If the data type is custom data, then data is filtered and queried from a database as per selected parameters. Now the data is written into an excel file but if the data type is data owner, then the dataset has to be attached. The next execution function is the same for both data types custom data or data owner user set the secret key, both types of dataset file are read and converted into a buffer, which passes to the encryption method to encrypt the dataset. This encrypted dataset is uploaded to IPFS which returns the dataset file hash key. The hash key of the uploaded dataset is mapped with additional parameters Sensor, Location, Uploading Type, Date, etc., and stored in the blockchain using a smart contract.
- *Output*: The output is to store the mapped data into the blockchain and available list on the dashboard.

Algorithm 3 Data Request to access the datasets: This algorithm validates the dataset request functionality. If a user is authentic, then any datasets uploaded and shared by the data owner as public access will be exposed to all users. The user selects a dataset and asks for access to download the required dataset. This request is sent to the data owner who uploaded the dataset for public access with the user public key by using the RSA signature technique. If the data owner accepts the user's request to download the dataset, the data

Algorithm 2 Decentralizing Data & Securing

Input: σ, $\mathcal{L}$, Ξ, ∞ ▷ Sensor, Location, Secret-Key, Date
Output: $\mathcal{D}_b, E_x, E_n$ ▷ Set Result (Database/Excel/Encryption)
procedure UploadingData(σ, $\mathcal{L}$, Ξ, ∞ = Null)
  if msg.sender is Valid **then** ▷ Verify User
    if DataType == Custom **then**
      if $\sigma_l > 0$ **then** ▷ Sensor is not null
        if $\mathcal{L}_l > 0$ **then** ▷ Location is not null
          while j < σ_l **do**
            while i < $\mathcal{L}_l$ **do**
              $\mathcal{D}_b \leftarrow$ Database(σ_l[j], $\mathcal{L}$[i],∞)
              if $\mathcal{D}_b$!= null **then**
                $\mathcal{E}_x \leftarrow$ GetData($\mathcal{D}_b$) ▷ Write Data into Excel
              end if
              i++
            end while
            j++
          end while
          if $\mathcal{E}_x$ $\mathcal{E}xist$ **then**
            $\mathcal{E}_n \leftarrow$ Encryption_AES($\mathcal{E}_x$, Ξ) ▷ Excel File Path & User Secret Key To Get Encrypted File
            $\mathcal{FS} \leftarrow$ File($\mathcal{E}_n$) ▷ Pass Encrypted File Path To Get File stream
            $\mathcal{FB} \leftarrow$ Buffer.form ($\mathcal{FS}$) ▷ Convert $\mathcal{FS}$ to Buffer $\mathcal{FB}$
            $\mathcal{FH} \leftarrow$ IPFS.Add ($\mathcal{FB}$) ▷ Get Hash of Uploaded Data $\mathcal{FH}$
            $\mathcal{SBC}$(name, σ, $\mathcal{L}$, ∞, $\mathcal{FH}$, msg.sender) ▷ Store Data to Blockchain Ledger
          end if
        end if
      end if
    end if
    if DataType == Data Owner **then**
      if $\sigma_l > 0$ **then** ▷ Sensor is not null
        if $\mathcal{L}_l > 0$ **then** ▷ Location is not null
          if $\mathcal{E}_x$ $\mathcal{E}xist$ **then**
            $\mathcal{E}_n \leftarrow$ Encryption_AES($\mathcal{E}_x$, Ξ) ▷ Excel File Path & User Secret Key To Get Encrypted File
            $\mathcal{FS} \leftarrow$ File($\mathcal{E}_n$) ▷ Pass Encrypted File Path To Get File stream
            $\mathcal{FB} \leftarrow$ Buffer.form ($\mathcal{FS}$) ▷ Convert $\mathcal{FS}$ to Buffer $\mathcal{FB}$
            $\mathcal{FH} \leftarrow$ IPFS.Add ($\mathcal{FB}$) ▷ Get Hash of Uploaded Data $\mathcal{FH}$
            $\mathcal{SBC}$(name, σ, $\mathcal{L}$, ∞, $\mathcal{FH}$, msg.sender, Status) ▷ Store Data to Blockchain Ledger
          end if
        end if
      end if
    end if
  end if
end procedure

owner will be asked to provide the secret key for the requested dataset which will be granted to the user.

- *Input(s)*: The input will be passed to the algorithm by the system during the request of the dataset (Requested Dataset ID, Address of Data Owner's dataset, and Address who made the request).
- *Processing*: The requested datasets record will automatically be stored in the blockchain ledger and mapped with the dataset file hash key.
- *Output*: The output is updated and available in the requested datasets list.

Algorithm 4 User Interface Layer: This layer shows the accessing functionality of the user interface. The data owner's response will be displayed in the user's requested list, either granting or denying access. If the user is permitted to download the dataset, they will be given an encrypted secret key and asked

Algorithm 3 Data Request

```
Input:                                                  ▷ Dataset ID, Data Owner Address
Output: R, μ                                                                  ▷ Data
procedure RequestingData
    if msg.sender is Valid then
        μ ← AvailableDataList()   ▷ Get Dataset ID, Data Owner Address from Available Datasets by making request to access
        Key ← RSA()                                          ▷ RSA Generate User Public Key
        R ← SBC(μ,Key)            ▷ Pass requested datasets parameters & User Public Key
    end if
    R ← UpdateDashboard(μ)                          ▷ Update available data on user screen
end procedure
```

to pass their private key to decode the secret key, which may then be used to decrypt the dataset. The downloaded dataset is then encrypted, and it must be decoded with the help of a secret key (see Figure 6).

- *Input(s)*: The input to access, the requested dataset in this algorithm is a secret key, private key, and dataset file hash by the system during the download of the dataset.
- *Processing*: The dataset will be available on a list from the blockchain ledger and linked with the dataset file hash key, then the requested dataset file hash key is passed to the IPFS storage server to download the dataset if access is granted by the data owner. The decrypted secret key will be passed to the AES algorithm method to decrypt the dataset.
- *Output*: The output is updated and available datasets list.

Algorithm 4 Interface Layer

```
Input: Ξ, P_k                                           ▷ Secret Key, User Private Key
Output: R                                                                     ▷ Data
procedure DataAccess
    if msg.sender is Valid then
        if Data Type == Custom Data then
            FH ← CustomDataList()                  ▷ Get file hash to download custom data
            F_i ← IPFS.DOWNLOAD(FH)                        ▷ Download Encrypted Data File
            R ← DECRYPT.AES(F_i,Ξ)                 ▷ File path & Secret Key to Decrypt Data
        end if
        if Data Type == Public Data then
            if File Access == Allow then                          ▷ Check data access if granted
                FH ← PublicDataList()              ▷ Get file hash to download public data
                F_i ← IPFS.DOWNLOAD(FH)                    ▷ Download Encrypted Data File
                Key ← SECRET.KEY(P_k)              ▷ Get Secret Key by private signature key
                R ← DECRYPT.AES(F_i,Key)           ▷ File & Secret Key to Decrypt Data
            end if
        end if
    end if
end procedure
```

5 Implementation and Evaluation of the Solution

In this section, we present the details for (i) implementing the solution via tools and technologies that we exploited (Section 4.1), and (ii) solution validation (Section 4.2), both are discussed below.

5.1 Tools and Technologies for Implementation

This section summarizes the complementary role of relevant tools and technologies for the proposed solution. The main intent of the discussion here is to provide a better solution for understanding the reader from a technology perspective. The tools and technologies are layered as shown in Figure 8. For example, the sensor's data is packaged into a CSV file and then sent to the IoT server where the data is processed and saved into the database. The requested data is filtered from the database as per a custom data query by a user and written in an excel file which is encrypted using the AES algorithm method and uploaded to IPFS to decentralize storage and return back a dataset hash key. This hash is mapped with other parameters to save the record in a blockchain ledger. With Smart Contracts, we leveraged Ethereum technology. Ethereum is a decentralized application platform built on a worldwide open-source blockchain. The NodeJS platform includes a variety of tools used to create a server-side application. We utilized Visual Studio Code (VSC) to run the NodeJS app. The Ganache Truffle Suite package is used to create a local Blockchain environment for running tests and executing commands.

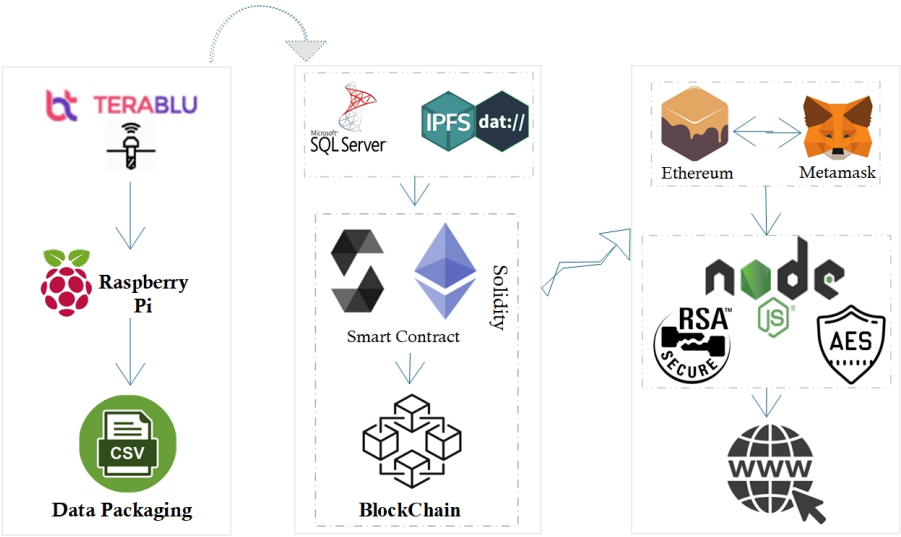

Fig. 8 Overview of Tools and Technologies for System Implementation

5.2 Solution Validation

The evaluation environment is a set of hardware and software resources that are used to run the solution and track various execution steps and outcomes. On the hardware side list of sensors (see Table 2) and Raspberry Pi with Scientific Instrument Interface Module (SIIM) is configured to retrieve the data from the sensors to the IoT server in excel/CSV packaging form where all the data is processed. Then it is saved into a database, and the evaluation trials were carried out by the different users of the blockchain. There are two major types of data performing categories, the first one is custom data which comes from the database where all the IoT data is stored, while the second one is self-datasets sharing to open access for the public. All trails are performed on the Windows 10 [43] Platform (SSD, Core i7 with 16 GB of runtime memory). This system can also be deployed on a Linux platform with little modifications because its open-source development can be run on any OS. Execution evaluation, commonly known as evaluation scripts in the software world and automates system testing. Scripts like this were created in NodeJS using ReactJS [44] language and programmed in Visual Studio Code [45]. Several existing libraries are also employed in the review process, including but not limited (react, web3, ipfs.http, etc.). For example, JavaScript performance profiling [46] is used to monitor CPU usage while uploading data to IPFS and storing it into the blockchain ledger with its retrieving the data. Ganache [47] suit is used to make the local Ethereum [48] environment of blockchain and Metamask [49] extension is used in a browser that allows connecting the distributed web. Metamask extension connects the local Ethereum accounts with Ganache suit to execute the system functionality by using gas transaction fees.

Fuel Consumption When Uploading Data Fuel should be paid for the Ethereum smart contract to be executed. As a result, the fuel consumption while uploading the original data was measured and compared to the fuel consumption using the suggested approach. The lowest unit of Ether, the Ethereum cryptocurrency is Gwei which is used to measure fuel usage. Gwei is the name given to 10^9 Wei.The data is secured by using the AES [50] encryption technique with a dynamic secret key mechanism, and the signature is verified using the RSA [51] algorithm for securing the secret key with a double encryption mechanism. In our proposed system, we listed the cost of contract migration execution (see Table 3). The cost is listed in Ether with mentioned Gas used. Ether is equal to the gas used * gas price. In this system, the gas reflects the continuous computing cost of this system. The network [48] adjusted the gas price to reflect changes in Ether's value. In the implemented prototype of our system, we set a gas limit by default. The contract creation is executed once with a cost of 0. 05,333,758 in Ether and used the total gas is 2,666,879. The migration calls for Contract creation is used the cost which is very little 0. 0054726 (Ether) and the gas consumption is only 27363. The costs are based on the amount of the data input; if the data input is small, the cost may be reduced, because datasets are uploaded to decentralized IPFS storage. The cost has no influence on the size of datasets. We can observe that

Table 3 Data Storing Cost analysis (gas price = 2 Gwei)

Execution Type	Gas Used	Cost in Ether
Requested Data SC Creation	2666879	0.05333758
Contract Migration Call	27363	0.0054726
Datasets Storing SC Creation	2874719	0.05749438
Contract Migration Call	27363	0.0054726
Data Logic SC creation	2112346	0.04224692
Contract Migration Call	27363	0.0054726
Initial Contract	225237	0.0450474
Initial Migration Call	42363	0.0084726
Total	4 SC Deployments	0.15758362

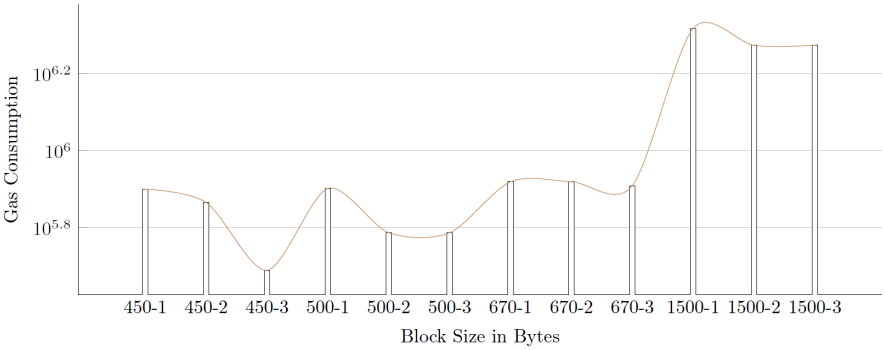

Fig. 9 Gas used based on block size and transaction count

the cost is less than that of alternative storage options; supplied by the third parties MedChain [52].

The last item tested, was the amount of time it took users to share data to public access with some extra detail as input of dataset name. Data sharing time measures the total time spent sharing data, recalling shared data, and reading data. We conducted multiple sets of experiments with average data sizes. The results are shown in Figure 9. the fuel consumption is on average about 671,807 While uploading data of size 450 bytes, for storing the data size of 1500 bytes the fuel consumption remains on average about 1,942,901. It shows that the fuel consumption increases as the data size increases. On the other hand, when the IoUT data were uploaded to IPFS through the proposed system there was no significant difference in fuel consumption, even if the data size increased.

5.3 Evaluations of Query Response Time

For storing IoUT data packages to IPFS and recording details in the blockchain ledger, data querying is required. The query response time can be used to evaluate how efficiently the solution stores and retrieves data from the blockchain. We tested query response time for storing IoUT datasets to IPFS and saving records to the blockchain ledger by using dataset hashes with other detail. The results of query response time in milliseconds are shown in Figure 10 is divided

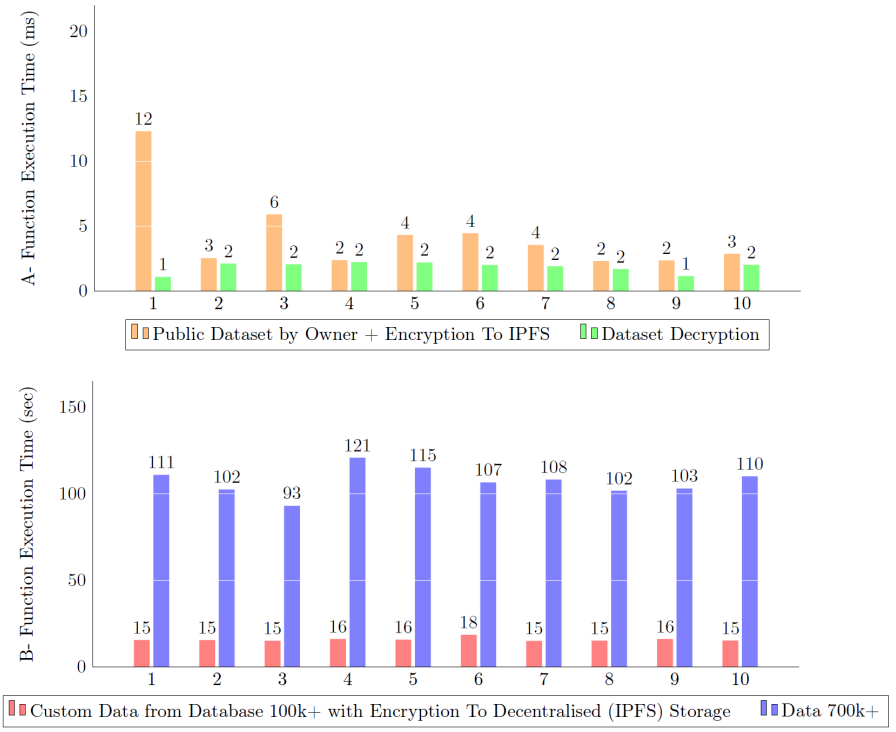

Fig. 10 The time it takes to compute the results of a function

into two parts 'A' and 'B', while the vertical axis displays the response time in milliseconds and the horizontal axis highlights the multiple different execution functions (Data Sharing by Data Owner, Custom Data access from the database, uploading to IPFS decentralize storage and encryption/decryption of datasets).

The experimental results in Figure 10 describe the experimental findings of three separate sets of activities used to verify execution performance in milliseconds. The custom data 'B' in figure 10 represents by the blue and red color bar, takes a longer time to complete data operations than regular data operations. The reason to consume more time than others is a set of actions of different methods and call data from the database server with 100k to 700k records. The second reason is to take more time in custom data action that this method has multiple sub-methods to filter the data from the database. Write the data in excel and then encrypt the dataset which is finally uploaded to IPFS storage, which saves the record to blockchain ledger with dataset hash returned by IPFS. Therefore, we mitigate the throughput gap by separating the smart contract logic of custom data. The dataset decryption method takes very less time to decrypt any size of the dataset because it is a standalone self-function that is executed directly in the system.

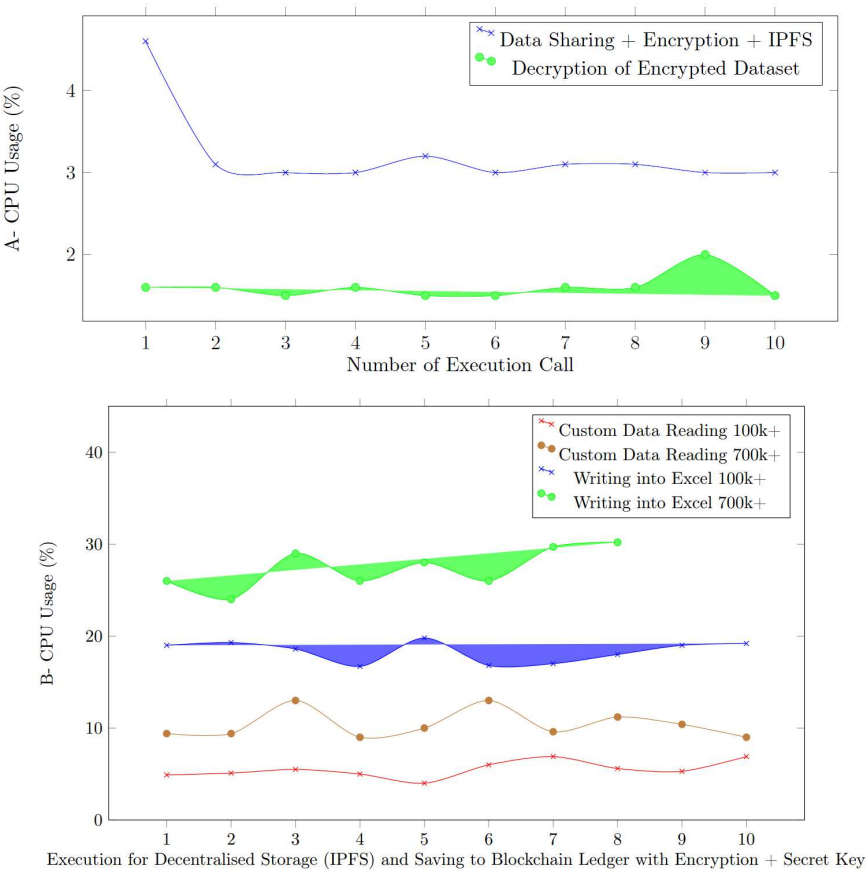

Fig. 11 CPU utilization computation time

In Figure 11, we also analyze the performance of CPU usage during the execution set of functions with smart contracts. we evaluated every step of each method like in data sharing. There are different methods such as calling encrypting Dataset, Uploading to IPFS storage, and saving records to the blockchain ledger. All these actions are measured in one cycle with a set of calls. After completing the cycle with a set of calls, we recorded the CPU usage%. The most usage of CPU is in Custom Data calls due to a lot of sub-methods in one cycle of a custom data call. The decryption method uses the CPU very less, the CPU is increasing the usage at A number 9 calls due to a large dataset encrypted file. Figure 11 depicts the performance of two alternative CPU utilization scenarios, A and B. Part A depicts the CPU utilization when sharing one's own datasets. The performance for custom data is shown in Part B. Custom data is a little more substantial and comes with a variety of processing options (dynamic secret key with symmetric encryption and custom query on-premises database). Upon performing custom data operations, the CPU use

increases, yet it has no effect when decentralizing the data. The activities are carried out on our deployed list of sensors (Table 2) with recordings ranging from 100k to 700k. In two different cases, we tested the performance of custom data (Data Reading, and Excel Writing)

5.4 Threats to Validity

A few possible risks to the research validity are briefly elaborated here. Validity risks are restrictions or limitations which cause a substantial impact on the solution's design, implementation, and validation. As part of future efforts to improve the solution and its ramifications, vulnerabilities to validity must be reduced.

- *Threats to Internal Validity*: refer to limitations or constraints which can put an impact the design and implementation of the proposed solution. For instance, the number of sensors used to collect oceanic data, and a number of trials to evaluate sensor data. Decentralizing the blockchain-based storage in IPFS using the Ethereum platform with smart contract and platform used to evaluate the solution may result in internal validity. It means that altering the platform of Ethereum can produce different results in the evaluation. In future works, the research fraternity must consider different platforms instead of the Ethereum platform, which can help to minimize the internal validity.
- *External Validity*: refers to the validation of a solution on various connected systems and case studies. We used a case study-based approach to demonstrate and validate the solution, such as we performed in the research method (see Figure 2) and the evaluation section. A single case study, on the other hand, may be constrained in terms of justifying the solution's generality and validity consistency. In order to minimize the effects of external validity; future works must incorporate more case studies and other systems.

6 CONCLUSIONS

IoTs represent a class of pervasive systems which exploit embedded sensors (hardware), applications (software which manipulates the hardware), and networks (interconnecting things to transmit data) in smart systems and environments. IoUTs as a specific genre of IoTs rely on underwater sensors which ingest oceanic data in the context of smart oceans. The oceanic data ingested from a multitude of deployed sensors are useful to carry out analysis of multifaceted information relating to marine life, water pollution, or to find water quality parameters. Consequently, it's critical to provide a platform which allows efficient sharing and storage of IoT data in ad-hoc, unsecured environments. To provide a distributed and trustworthy access control mechanism, we have considered Blockchain technology, specifically Ethereum smart contract to share IoT data. This article offers a system which uses Ethereum blockchain and IPFS for efficient and secure storage of IoT data. It also offers

smart contracts to make it easier for users to store and manage access roles for corresponding IoT data. The suggested solution encrypts IoT data sent to IPFS's decentralized storage and records the hash value among different parameters in a blockchain ledger. System evaluation is focused on assessing the performance of data transmission. The solution focuses on decentralizing and securing the data of IoUTs with IoTs system development as a specific genre of the IoTs. This research aims to provide a solution along with a set of guidelines for IoUT practitioners and researchers which can support in the context of the smart ocean to develop the next-generation (software-intensive) of robust and secure blockchain-aided distributed IoT systems. The main purpose of IoUT application is used for decentralizing and sharing the data in an untrustworthy environment. The primary contributions of this research are as follows:

- *Contribution 1*: The access level of data (access management) is implemented to enable data governance, trust-based customizable roles, trustworthiness, and security.
- *Contribution 2*: This research unifies the Internet of Things (IoTs) for effective data sharing, mining, and analyzing oceanic data with Blockchain technology which enables secure management and transmission of relevant data while ensuring performance and efficiency of the proposed solution.

Future research directions: In the future, we will primarily focus on the diversity of data evaluation with more case studies which can further enhance the rigor of evaluation. In addition to the consideration of more case studies and use cases, we also aim to enhance the scalability of the system which can accumulate data from multiple oceanic sites, and centralize the processing and analytics of the data which is being ingested from various distributed sensors.

This research work was supported by any Zhejiang University

Declarations

- Funding: The authors did not receive support from any organization for the submitted work. The authors have no relevant financial or non-financial interests to disclose.
- Data availability: Available on request.
- Code availability: Available on request Declarations
- Conflict of interest: All Authors declare that they have no conflict of interest.
- Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Ejaz Ahmed, Ibrar Yaqoob, Abdullah Gani, Muhammad Imran, and Mohsen Guizani. Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges. *IEEE Wireless Communications*, 23(5):10–16, 2016.

[2] Statista. Iot and non-iot connections worldwide 2010-2025, 2021.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: a survey. computer networks. *doi*, 10:1016, 2010.

[4] Chien-Chi Kao, Yi-Shan Lin, Geng-De Wu, and Chun-Ju Huang. A comprehensive study on the internet of underwater things: applications, challenges, and channel models. *Sensors*, 17(7):1477, 2017.

[5] Tie Qiu, Zhao Zhao, Tong Zhang, Chen Chen, and CL Philip Chen. Underwater internet of things in smart ocean: System architecture and open issues. *IEEE transactions on industrial informatics*, 16(7):4297–4307, 2019.

[6] Fraunhofer. Smart ocean technologies solutions for responsible ocean use, 2021.

[7] Eleftherios Kokoris-Kogias, Enis Ceyhun Alp, Sandra Deepthy Siby, Nicolas Gailly, Linus Gasser, Philipp Jovanovic, Ewa Syta, and Bryan Ford. Calypso: Auditable sharing of private data over blockchains. *IACR Cryptol. ePrint Arch., Tech. Rep*, 209:2018, 2018.

[8] Mohamed Tahar Hammi, Badis Hammi, Patrick Bellot, and Ahmed Serhrouchni. Bubbles of trust: A decentralized blockchain-based authentication system for iot. *Computers & Security*, 78:126–142, 2018.

[9] Liehuang Zhu, Yulu Wu, Keke Gai, and Kim-Kwang Raymond Choo. Controllable and trustworthy blockchain-based cloud data management. *Future Generation Computer Systems*, 91:527–535, 2019.

[10] Alexander Yakubov, Wazen Shbair, Anders Wallbom, David Sanda, et al. A blockchain-based pki management framework. In *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[11] Shangping Wang, Yinglong Zhang, and Yaling Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *Ieee Access*, 6:38437–38450, 2018.

[12] Mingjun Dai, Shengli Zhang, Hui Wang, and Shi Jin. A low storage room requirement framework for distributed ledger in blockchain. *IEEE Access*, 6:22970–22975, 2018.

[13] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[14] Gaoqi Liang, Steven R Weller, Fengji Luo, Junhua Zhao, and Zhao Yang Dong. Distributed blockchain-based data protection framework for modern power systems against cyber attacks. *IEEE Transactions on Smart Grid*, 10(3):3162–3173, 2018.

[15] QI Xia, Emmanuel Boateng Sifah, Kwame Omono Asamoah, Jianbin Gao, Xiaojiang Du, and Mohsen Guizani. Medshare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access*, 5:14757–14767, 2017.

[16] Ajay Kumar Shrestha and Julita Vassileva. Blockchain-based research data sharing framework for incentivizing the data owners. In *International*

Conference on Blockchain, pages 259–266. Springer, 2018.

[17] Axin Wu, Yinghui Zhang, Xiaokun Zheng, Rui Guo, Qinglan Zhao, and Dong Zheng. Efficient and privacy-preserving traceable attribute-based encryption in blockchain. *Annals of Telecommunications*, 74(7):401–411, 2019.

[18] Zhaofeng Ma, Ming Jiang, Hongmin Gao, and Zhen Wang. Blockchain for digital rights management. *Future Generation Computer Systems*, 89:746–764, 2018.

[19] Wei Liang, Mingdong Tang, Jing Long, Xin Peng, Jianlong Xu, and Kuan-Ching Li. A secure fabric blockchain-based data transmission technique for industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, 15(6):3582–3592, 2019.

[20] Tiago M Fernández-Caramés and Paula Fraga-Lamas. A review on the use of blockchain for the internet of things. *Ieee Access*, 6:32979–33001, 2018.

[21] Oscar Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, 2018.

[22] Carlos Cid, Sean Murphy, and Matthew Robshaw. Equation systems for the aes. *Algebraic Aspects of the Advanced Encryption Standard*, pages 71–85, 2006.

[23] Henry Williams. A modification of the rsa public-key encryption procedure (corresp.). *IEEE Transactions on Information Theory*, 26(6):726–729, 1980.

[24] Aakash Ahmad, Mahdi Fahmideh, Ahmed B Altamimi, Iyad Katib, Aiiad Albeshri, Abdulrahman Alreshidi, Adwan Alownie Alanazi, and Rashid Mehmood. Software engineering for iot-driven data analytics applications. *IEEE Access*, 9:48197–48217, 2021.

[25] Brice Morin, Nicolas Harrand, and Franck Fleurey. Model-based software engineering to tame the iot jungle. *IEEE Software*, 34(1):30–36, 2017.

[26] Xabier Larrucea, Annie Combelles, John Favaro, and Kunal Taneja. Software engineering for the internet of things. *IEEE Software*, 34(1):24–28, 2017.

[27] Franco Zambonelli. Towards a discipline of iot-oriented software engineering. In *WOA*, pages 1–7, 2016.

[28] Kwame Opuni-Boachie Obour Agyekum, Qi Xia, Emmanuel Boateng Sifah, Jianbin Gao, Hu Xia, Xiaojiang Du, and Moshen Guizani. A secured proxy-based data sharing module in iot environments using blockchain. *Sensors*, 19(5):1235, 2019.

[29] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009.

[30] Abdulrahman Alreshidi and Aakash Ahmad. Architecting software for the internet of thing based systems. *Future Internet*, 11(7):153, 2019.

[31] Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran, and Shiping Chen. The blockchain as a software connector. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 182–191. IEEE, 2016.

[32] David Halpern. *Satellites, oceanography and society*. Elsevier, 2000.

[33] Augustin A Saucan and Moe Z Win. Information-seeking sensor selection for ocean-of-things. *IEEE Internet of Things Journal*, 7(10):10072–10088, 2020.

[34] Ottar L Osen, Hao Wang, Karina B Hjelmervik, Halvor Sch, et al. Organizing data from industrial internet of things for maritime operations. In *OCEANS 2017-Aberdeen*, pages 1–5. IEEE, 2017.

[35] Chunqiang Hu, Yuwen Pu, Feihong Yang, Ruifeng Zhao, Arwa Alrawais, and Tao Xiang. Secure and efficient data collection and storage of iot in smart ocean. *IEEE Internet of Things Journal*, 7(10):9980–9994, 2020.

[36] Marc Pilkington. Blockchain technology: principles and applications. In *Research handbook on digital transformations*. Edward Elgar Publishing, 2016.

[37] Hossein Shafagh, Lukas Burkhalter, Anwar Hithnawi, and Simon Duquennoy. Towards blockchain-based auditable storage and sharing of iot data. In *Proceedings of the 2017 on cloud computing security workshop*, pages 45–50, 2017.

[38] Mathis Steichen, Beltran Fiz, Robert Norvill, Wazen Shbair, and Radu State. Blockchain-based, decentralized access control for ipfs. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1499–1506. IEEE, 2018.

[39] Fahad Ahmad Al-Zahrani. Subscription-based data-sharing model using blockchain and data as a service. *IEEE Access*, 8:115966–115981, 2020.

[40] Bao Le Nguyen, E Laxmi Lydia, Mohamed Elhoseny, Irina Pustokhina, Denis A Pustokhin, Mahmoud Mohamed Selim, Gia Nhu Nguyen, and K Shankar. Privacy preserving blockchain technique to achieve secure and reliable sharing of iot data. *Computers, Materials & Continua*, 65(1):87–107, 2020.

[41] Parminder Singh, Mehedi Masud, M Shamim Hossain, and Avinash Kaur. Cross-domain secure data sharing using blockchain for industrial iot. *Journal of Parallel and Distributed Computing*, 2021.

[42] CreativeTim. Web interface template design, 2021.

[43] Gale Fritsche. Understanding windows 10. In *Proceedings of the 2015 ACM SIGUCCS Annual Conference*, pages 75–78, 2015.

[44] Robin Wieruch. *The Road to GraphQL: Your journey to master pragmatic GraphQL in JavaScript with React. js and Node. js*. Robin Wieruch, 2018.

[45] Microsoft. Configuring visual studio code for ethereum blockchain development, 2021.

[46] Thomas Nägele, Jozef Hooman, Remco Zigterman, and Mark Brul. Client-side performance profiling of javascript for web applications. *Universidad de Radbound*, 2015.

[47] Ganache. Ganache documentation, 2021.

[48] Ethereum. A secure decentralised generalised transaction ledger, 2021.

[49] Metamask. Metamask-a crypto wallet gateway to blockchain apps, 2021.

[50] AkoMuhamad Abdullah. Advanced encryption standard (aes) algorithm to encrypt and decrypt data. *Cryptography and Network Security*, 16:1–11, 2017.

[51] Xin Zhou and Xiaofei Tang. Research and implementation of rsa algorithm for encryption and decryption. In *Proceedings of 2011 6th international forum on strategic technology*, volume 2, pages 1118–1121. IEEE, 2011.

[52] Bingqing Shen, Jingzhi Guo, and Yilong Yang. Medchain: Efficient healthcare data sharing via blockchain. *Applied sciences*, 9(6):1207, 2019.