Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

A framework for blockchain-based management of IoT-driven data sharing

Abdulrahman Alreshidi *

College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia

ARTICLE INFO

Article history:
Received 31 August 2024
Received in revised form
21 December 2024
Accepted 9 January 2025

Keywords:
Blockchain architecture
IoT data management
Data integrity
Smart contracts
Decentralized systems

ABSTRACT

Big data systems rely on acquiring, analyzing, and processing diverse data sets to enable predictive analytics in decision support systems. The integration of Internet of Things (IoT) devices generates vast amounts of data, creating challenges related to security, scalability, and efficient data management in mobile environments. Traditional IoT data sharing platforms often depend on Trusted Third Parties (TTPs), which compromise security, transparency, and trust, while also facing constraints like limited storage, privacy concerns, and high energy consumption. This research proposes a blockchain-based IoT data management platform utilizing the InterPlanetary File System (IPFS) to address these issues. The platform defines data access roles through smart contracts, ensuring data integrity and security while modernizing legacy systems. Experimental evaluation demonstrates the platform's efficiency in energy consumption and storage management for smart communication contracts. Key contributions include a blockchain-based architecture, algorithms for implementation, and validation using satellite-sensed data, advancing decentralized data management solutions.

1. Introduction

The Internet of Things (IoT) represents a multitude of computational entities (things) that are interconnected via networking technologies, integrated via software services and sensors to facilitate data exchange with other systems and devices (Hammi et al., 2018). Traditional methods for secure data sharing often fall short in this context due to processing a massive volume of data, heterogeneity of devices, lack of trust, and the absence of transparency in data management (Liang et al., 2019). Blockchain systems and technologies can offer a viable solution for a diverse range of distributed applications where trust and transparency are critical. Consequently, there is growing interest in both industry and academia regarding the integration of IoT systems with blockchain technology. Several research initiatives propose connecting IoT systems directly to blockchain platforms to address secure data exchange challenges (Hammi et al., 2018; Liang et al., 2019). Such approaches apply hybrid storage mechanisms, where a provider maintains the data, and the blockchain ensures integrity and trustworthy distribution (Xia et al., 2017). For example, one approach involves storing access control policies on the blockchain. When a storage provider receives a request for data access, it can refer to these policies to determine whether access should be granted. This enables the storage provider as the central entity for policy enforcement, whereas the blockchain system can preserve policy integrity and allows for transparency and auditing of any policy changes (Liang et al., 2019).

In the last decade, the research community has achieved considerable technical advancements in the adaptation and enhancement of data-sharing methodologies (Razzaq, 2022). Facilitating collaboration and making informed decisions are essential for advancing research-based initiatives. Data sharing plays a central role in maximizing the benefits of scientific progress. However, the timing of data dissemination is critical and must be carefully considered before the exchange process begins. This study examines the utilization of blockchain technology as a secure means for data sharing and transactions. Blockchain, characterized by its distributed ledger system, has emerged as a significant innovation in information technology, with applications extending beyond financial domains to various non-financial sectors (Shrestha and Vassileva, 2018). Traditionally, centralized authorities, such as cloud servers, have managed

* Corresponding Author.
Email Address: ab.alreshidi@uoh.edu.sa
https://doi.org/10.21833/ijaas.2025.01.020
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-9034-3909

large volumes of data, which introduces risks, including the potential for single points of failure. To prevent such failures, third-party services are often employed to provide data backups. Blockchain, however, offers a trust-based and transparent framework, thereby removing the necessity for intermediaries. Despite its advantages, challenges persist, particularly concerning the storage and processing constraints of individual network nodes. To mitigate these challenges, the peer-to-peer architecture of the InterPlanetary File System (IPFS) is being adopted (Benet, 2014). Considering the context of peer-to-peer data access protocols IPFS as a content-based protocol protects IoT data by means of cryptographic techniques, as shown in Fig. 1. Specifically, by means of cryptographic hashing the data of text is preserved as unchangeable and temper-proof (Benet, 2014). IPFS is also useful as it provides efficient bandwidth utilization, allowing faster data transfer, and avoiding data duplication. By relying on IPFS, IPFS objects can allow up to 256 KB of binary data to be wrapped and shared over the network. In case of larger data volumes exceeding 256 KB, IPFS can split data to store them into multiple IPFS objects that are linked to the same IoT data. As a result, IPFS allows an immutable storage system where any attempts to modify or temper the value of a file hash are detectable and can be avoided. A hash string route is supported by IPFS for data transport for encryption and additional storage of the data. An overview of the proposed solution is illustrated in Fig. 1. Fig. 1 shows a synergy of IoT-based sensing of data and blockchain-driven security and management of IoT-sensed data. The system design, algorithmic implementations, and tool support represent the primary contributions as well as the building blocks of the proposed solution.

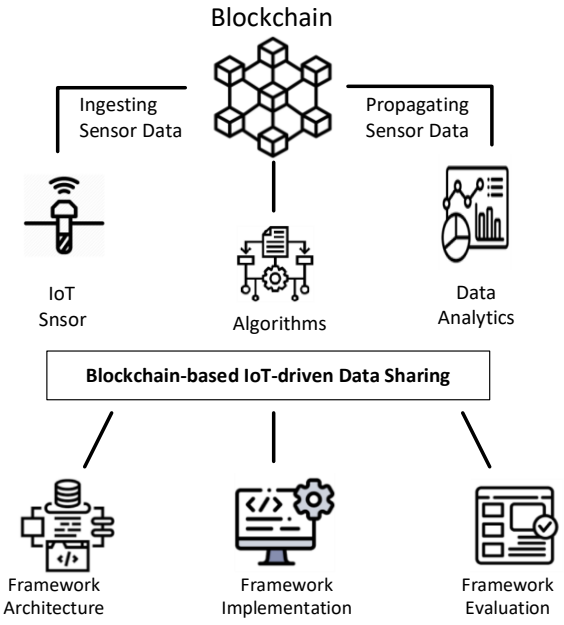


Fig. 1: A generic overview of the proposed solution

As shown in Fig. 1, the system is presented using a layered architecture design, that helps system designers and developers to maintain the layer of abstraction. The primary contributions of this research are:

- Blockchain-based framework and architecture that provides a blueprint and design principles to architect, implement, and validate IoT-driven data sharing platform.
- Algorithmic implementation modularizes the solution such that a set of algorithms can be developed to automate and parameterize the proposed solution.
- Experimental validation of the solution by evaluating a variety of scenarios to assess solutions' efficiency in terms of algorithmic execution performance, gas consumption, query execution, and response time.

To have an overview and streamlines the relevant related work on (i) blockchain systems for digital assets, and (ii) applications of blockchain in smart systems, the following summarization can be outlined:

A. Blockchain solutions for IoT-based digital asset management: In the context of asset digitization, the Swedish government employed blockchain for digitizing the real estate industry, particularly in managing land papers and titles. This approach aims to enhance the reliability and security of document updates and exchanges among stakeholders, enabling user identity confirmation through smart contracts (Liang et al., 2019). While the model is in testing, the recordskeeper has proposed a solution based on public access which is open-source and protects the documents (Steichen et al., 2018). This technology offers heightened security and accessibility across peer groups, unlike conventional database systems, as it creates immutable records (Razzaq, 2022). Records Keeper provides a rigorous framework for blockchain-based document storage (Nizamuddin et al., 2019), allowing validation at any time.

B. Blockchain in IoT and smart city systems: Iron Mountain, a global organization, employs blockchain technology to securely save and manage digital data and ensure the network storage is reliable. This blockchain-assisted framework serves as an audit trail or version tracking system, allowing approved network members to monitor changes, overcoming concerns of digital asset authenticity and reliance on untrustworthy third parties. Eleks Labs has created a unique technique using Ethereum to secure document transmission for various types of sensitive data, eliminating the need for third-party intermediaries. Their permissionless blockchain enables safe storage and transmission of documents, including legal agreements, with cryptographic technology and Ethereum-enabled smart contracts ensuring security and efficiency (Razzaq, 2022). Considering the recent research and development on big data for remote sensing, capable developments are emerging. A typical

example of such a case is Analysis Ready Data (ARD) which is being recommended by the Committee on Earth Observation Satellites (CEOS), although challenges remain in data aligning formats (Ahmad et al., 2023; 2024).

2. Background: Blockchain and IoT in big data systems

This section provides background information to contextualize and explain the key concepts of the proposed solution, based on the illustration in Fig. 2. It discusses the role of blockchain in big data systems (Section 2A) and explores the use of IoT in data analytics (Section 2B). The section introduces fundamental concepts and technical aspects that will be referenced throughout the paper.

A. Blockchain Systems in Big Data: First, we examine the main components of blockchain-based and IoT-driven platforms designed for the secure management of IoT-generated data. Additionally, we review relevant literature that supports and justifies the contributions of the proposed solution. Kokoris-Kogias et al. (2021) introduced CALYPSO, a system for auditable sharing of privacy-sensitive data. CALYPSO allows data to be stored on the blockchain, while data access is controlled through a smart contract. However, one of its limitations is its inability to handle large volumes of data, which is essential in real-world applications.

A similar approach by Shafagh et al. (2017) enables the sharing of time-series data. This solution allows data owners to define transaction policies each time they share data with external entities. Once the data is shared, only the data owner has the authority to update the transaction policy.

In real-world IoT data-sharing platforms, data volume and transmission speed are critical considerations. Existing research and industry solutions have proposed various data exchange policies, but few have explored incentive-driven mechanisms for data sharing.

To address this gap, Rowhani et al. (2017) investigated the incentivization of secure and private data sharing for health-critical data. Their study examined the impact of incentives on the quantity and success rate of shared medical records. The findings suggest that further research is needed to systematically and empirically evaluate incentive mechanisms for data sharing.

B. IoT for Data Analytics: Data-driven intelligence and automation in daily life increasingly depend on IoT systems. IoT devices, sensors, and nodes generate, consume, and exchange data over networks (Razzaq, 2020). For IoT systems to function effectively, they must be designed with built-in security and privacy, ensuring that end users and devices can share data securely and with trust.

To promote the adoption of secure-by-design IoT systems, Hammi et al. (2018) proposed a blockchain-based decentralized data-sharing model, known as the "bubble of trust." While this solution enhances secure data exchange, it has limitations, including challenges in real-time data exchange and a lack of adaptability to cryptocurrency rate fluctuations. Furthermore, storing IoT data on blockchain platforms presents issues such as data privacy risks, high maintenance costs, and difficulties in monitoring IoT networks.

To address these challenges, Liang et al. (2019) introduced a fabric-based data transport system. This system organizes and manages IoT data using dynamically linked records (chains of data) based on a data consensus mechanism. While this approach improves the security of real-time data transmission, it remains inadequate for processing large volumes of data.

There is a significant storage issue with the blockchain, especially when a lot of data needs to be kept on network nodes. The storage capacity of the terminal node is limited since it cannot support the storage of very large data. Many issues arise from this paradox, including the requirement for extremely powerful computers and the high expense of handling enormous volumes of data. Steichen et al. (2018) have introduced a decentralized storage approach utilizing IPFS to address these issues. Every node has segments containing the files. However, a file cannot be seen until users are authorized with the necessary permissions. This is a very resourceful way to safeguard private data. Due to blockchain involvement, the recommended schemes experience latency when receiving files from the server.

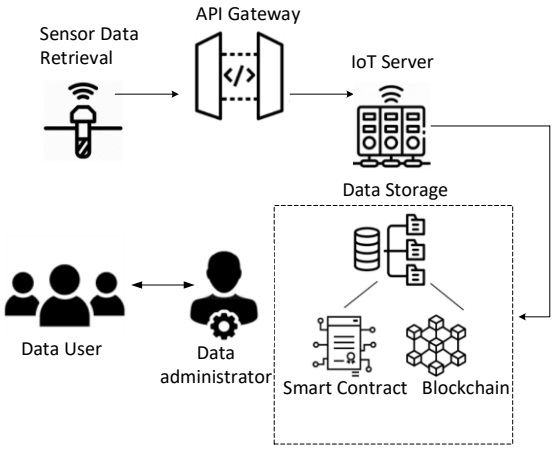


Fig. 2: Blockchain and IoT in data analytics systems

Nizamuddin et al. (2019) proposed a blockchain-based data sharing platform. The version management of the documents is aided by a framework, which allows for easy tracking of modifications. On a blockchain network, several users may collaborate to negotiate document modifications. Consequently, several versions of documents may be maintained on a network without the help of a third party. Additionally, a decentralized storage system called IPFS is used to store the data. Ethereum smart contracts facilitate communication between several parties, including users and document owners. Based on the aforementioned current research, we are motivated

to work on blockchain-based digital data exchange. Even though most researchers have worked in related fields, there is still much that can be done to advance and modify earlier studies to benefit the research community.

3. Research method and solution overview

This section details the research method that is used to conduct this study, presented in section 3A. The overview of the three phases of the research method is presented in Fig. 3. Fig. 3 shows three phases as an increment-driven method to enable the designing, implementation, and validation of the solution, with each step detailed below. We also present an overview of the proposed solution that provides a design framework for solution implementation. The solution overview is demonstrated in Fig. 4.

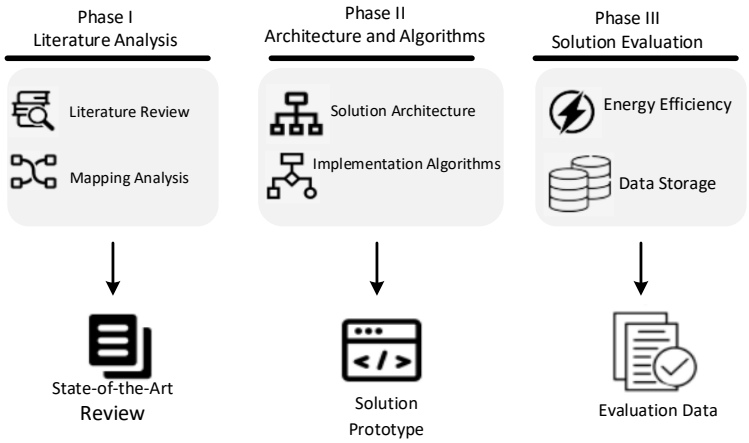


Fig. 3: Three phases of the research method

A. Research method: Fig. 3 presents a summary of the research methodology, illustrating the three distinct phases involved in the process. Each of the phases is detailed below, as per the illustration of methodological steps in Fig. 3.

- Phase I: Literature analysis: Our research journey commences with an extensive literature review to gain a profound understanding of the existing published research and analyze state-of-the-art in the field. This initial phase focuses on conducting a thorough comparative analysis between the existing solutions and our proposed approach. You can delve deeper into the findings of this literature review in Section VI, where we provide a comprehensive discussion.
- Phase II: Architecture representation and algorithmic design: In the subsequent stage, we transition from knowledge gathering to the creation of a blueprint for our solution's architecture. Simultaneously, we design the algorithms that constitute the core of our approach. Section III offers a detailed insight into the architectural aspects, while the modularized implementation of our algorithms is elaborated upon in Section IV.
- Phase III: Solution evaluation: The final stage of our research methodology focuses on rigorously evaluating the efficiency and suitability of our proposed solution. This phase, outlined in Section IV, ensures that our approach not only aligns with our research objectives but also proves its mettle in addressing real-world challenges.

B. The architecture of the proposed solution: This section discusses the overall architecture of the proposed solution that is demonstrated in Fig. 4. As per Fig. 4, the process of storing the data ingested by IoT sensors is preserved in the blockchain that is managed via a smart contract. When IoT data is passed to IPFS, a package file containing the hash key is generated that is uploaded to DApp. DApp allows two types of file upload that include i) manual upload by the administrator, and ii) automatic upload by the system. In the manual uploading type, the administrator initiates the upload of the IoT data package file to IPFS and receives the hash key along with other necessary data that is kept in the blockchain. Alternatively, if a new package file is received from any of the deployed IoT servers, the system uploads it straight. The system then manages to download the file from the deployed IoT server and uploads it to IPFS, and obtains the hash of the file that is stored on the blockchain via a smart contract. Both of these upload methods of execution are identical; however, one involves a human manually uploading data, while the other involves the machine automatically doing it.

The process of transferring digital data is initiated by the manual or system by generating metadata for the source file. Metadata encompasses information regarding the file, including its name, type, description, and size. When the metadata is finished, the whole data file is then uploaded to IPFS. Fig. 5 provides an illustration of a file upload to IPFS.

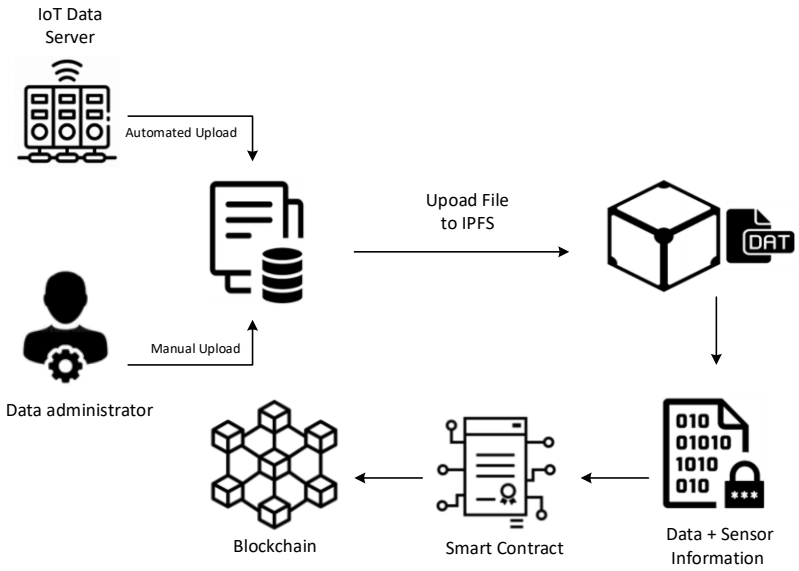


Fig. 4: Blockchain-based storage of the IoT-driven data

```
var UploadingType = ReactSession.get ("uploadingType")
var location = locationid.options[locationid.selectedIndex].value;
const Sensor = this.sensorid.value;
const Description = this.descriptionid.value;
ipfs.add(this.state.buffer, (error, result) => {
   console.log ('Ipfs result', result)
    if (error) {
       console.error(error)
       Return
   }
this.props.AddDataPackegeRecord(Sensor, location, result[0]. hash, Description, UploadingType)
})
```

Fig. 5: The source code snippet to upload data via IPFS

When a file is uploaded to IPFS and then returned to the administrator or system, a hash of its contents is created. When IPFS transmits the hash to the administrator or system, the hash key is mapped using the available parameters. If this is an administrator-initiated operation, they will choose the IoT data package file and use the assigned system to upload it to IPFS. IPFS will thereafter supply a hash key. The necessary parameters (sensor, location, and description) will be associated by the administrator using the supplied input form. This information will then be sent to a smart contract, which will store all of the data on the blockchain. The system uploading procedure will be initiated using the same execution method. The system will obtain the most recent file meant for upload from the IoT data server by navigating along the designated path. After that, it will upload the file directly to IPFS, where it will receive a hash key. This key will be connected to the existing data and stored on the blockchain through a smart contract.

- Phase 1 refers to sensor data that IoT nodes have ingested. There are also many kinds of sensor data. For an IoT data package, the information has been compressed into a file for a predetermined duration of time—roughly ten minutes—but it might be longer or shorter based on the predefined parameter. This packet of IoT data was sent to the IoT server via the gateway service.
- Phase 2 relates to services at the system level: There are two distinct DApp categories where IoT data can be uploaded. In order to replace manual uploading with a system, we have developed a system uploading service, also referred to as an auto uploading service. The system service, which runs on the server's backend, asks the IoT server for the most recent package file containing IoT data. The system service obtains the package file from the server, uploads it to an IPFS server, and then returns the file's hash to the system. The system service provides the file hash and other required information to the smart contract, which oversees the blockchain's data storage process.
- Phase 3 comprises the system administrator's manual uploading of the data. The admin still performs this step manually, but the process for uploading an IoT package file to IPFS and having it saved on the blockchain through a smart contract is the same.
- Phase 4 relates to the uploaded data's public accessibility. On this public portal, users are able to explore, view, and download any IoT data package at no cost. One way to obtain IoT data is through a

web portal. The user will have several alternatives for getting access to the data, depending on what they need.

4. Algorithmic specifications and tools support for solution implementation

The details of the implementation are covered in this section. The private Ethereum blockchain network is the recommended solution. An open-source distributed network that effectively utilizes Solidity is called Ethereum. a language for programming that enables smart contracts.

A. Overviewing the algorithms-based implementation: In order to implement the algorithms, there is a need to set up a platform that allows us to execute the algorithms. This subsection outlines the technical details of the platform for algorithmic implementation and execution.

- Microsoft Visual Studio Code: It is commonly referred to as VSC, which is a lightweight code editor that operates across multiple platforms, including Windows, Linux, and macOS. This dual-licensed source-code editor is designed to enhance the coding experience with features such as code refactoring, integrated Git version control, syntax highlighting, intelligent code completion, and advanced debugging capabilities.
- Ganache: It provides an extensive array of tests and commands that act as an emulator built on the blockchain. It is possible to perform tests, launch apps, and create contracts using the personal Ethereum blockchain executed via Ganache. It controls how the blockchain operates by examining the system's states. Ganache is the new name for it; Test RPC was its previous name.
- Metamask: It is an add-on for a browser that enables access to a dispersed web. It does not execute the entire Ethereum node; instead, it facilitates the execution of Ethereum decentralized applications directly within the browser. Users can access their Ethereum wallets through a web browser.
- IPFS: Data can be moved across IPFS, a distributed open storage system, by using a hash string path. Data that is encrypted and contains additional information is kept there. The paths work in a similar manner as the traditional universal resource locator on the web. As a result, by employing their hash, all IoT data is always accessible.

B. Algorithmic specifications: The phases of computation, data storage, and algorithm flow are shown in Fig. 6. The suggested solution (Fig. 6) and algorithmic specifications remain consistent since the operations are mapped using algorithmic stages.

- Algorithm 1: Creation of the smart contract: Algorithm 1 displays the uploading feature, which is discussed in this section. This method involves uploading data to IPFS while storing the file hash and mapping several additional attributes within a smart contract. Many parameters (location of the sensor, description of sensor, type of upload, and date of upload) are mapped using a hash of the submitted data.

- Input(s): A hash key is assigned to the parameters based on the input received by the algorithm.
- Processing: Upon reading the IoT data file and converting it into a buffer, the hash key is returned. Subsequently, this buffer package is uploaded to IPFS, designated as an IoT data file. The hash key of the uploaded data, as well as the additional attributes such as Sensor, Location, Description, Uploading Type, and Date, is stored on the blockchain through a smart contract.
- Output: The ultimate goal is to store the mapped data within the blockchain.

- Algorithm 2: Uploading IoT-driven data to blockchain platform: This section describes and algorithm 2 validates the data accessing functionality. The program retrieves the data from the blockchain and makes it accessible to the user as public information. The user can retrieve data from the blockchain based on the necessary specifications.

- Input(s): The mapping of the parameters for data access is done through the algorithm's input.
- Processing: Based on many criteria, such as the location of the sensor, location, date, and sensor mapping with a place, the data might be retrieved from the blockchain.
- Output: The mapping data output is accessible to the general audience.

Algorithm 1: Creation of the smart contract

```
Input: σ, Ł,h(γ℘), Δp, ψ, ρD, ϕp          ▷ Sensor, Location, Hash, Description
                                          ▷ Uploading Type, Dat,e Blockchain Address
Output: bool
procedure SMARTCONTRACT
    If msg.sender is not ϕp then          ▷ Get Blockchain address to execute the smart
Contract
        throw;
    end if
    mapping h(γ℘) to (σ/Ł/ρD)             ▷ Map with each parameter
end procedure
```

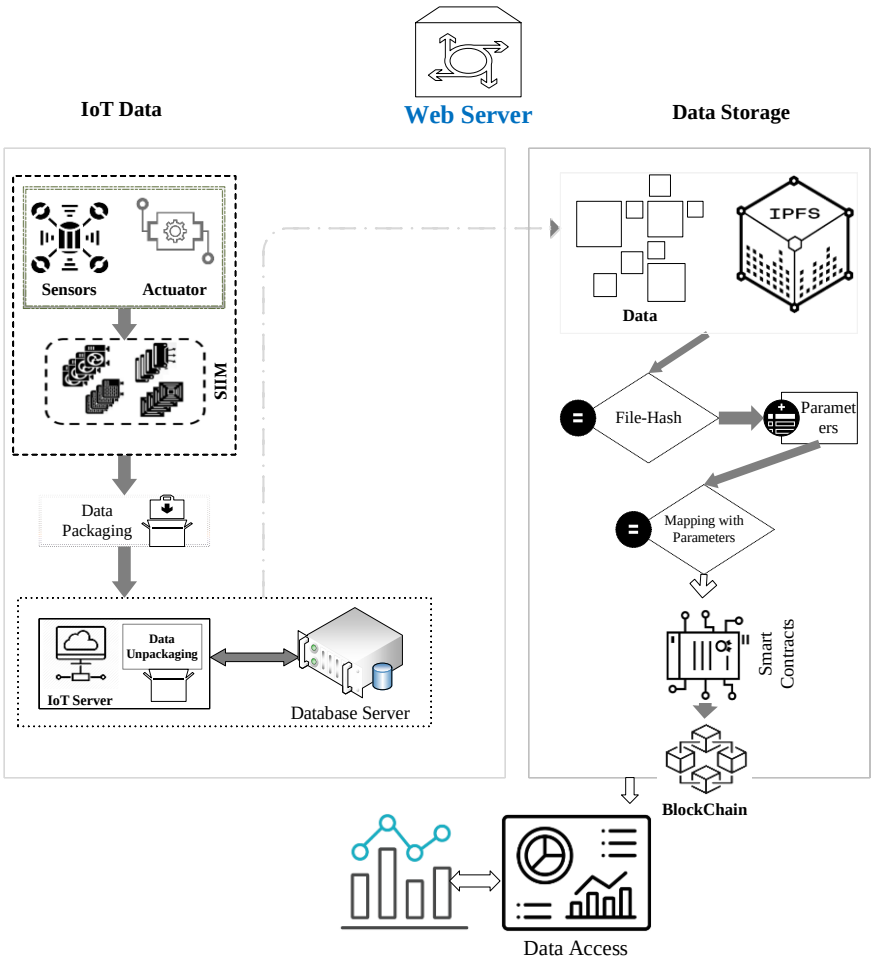


Fig. 6: A layered architecture view of the proposed solution

Algorithm 2: Uploading IoT-driven data to blockchain platform

```
Input: σ, ℒ, Δp, ψ, ρD, θλ                    ▷ Sensor, Location, Description
                                               ▷ Uploading Type, Date, Meta Data File
Output: ℛ                                      ▷ Uploading Message
procedure DATAUPLOADINGMODULE                  ▷ Event based function
    if ψ == User || ψ == System then           ▷ Uploading User OR System
        ℱS ← File(θλ)                          ▷ Get File stream ℱS
        ℱB ← Bffer.form(ℱS)                    ▷ Convert ℱS to Buffer ℱB
        ℱH ← IPFS.add(ℱB)                      ▷ Get Hash of Uploaded Data ℱH
        ℛ ← SBC (σ, ℒ, ℱH, Δp, ψ, ρD)         ▷ Store Data to Blockchain with file hash
    end if
end procedure
```

– Algorithm 3: Secure access of the blockchain-based data: The case study as shown in Fig. 7 demonstrates the platform where all scenarios are executed successfully together with the generated algorithms. Figs. 8a and 8b show how stakeholders can use custom queries to retrieve IoT data from an on-premises server and donate IoT datasets to IPFS's decentralized storage. When the custom query runs successfully, IPFS receives the data publication. The start and finish dates, the list of sensors, and the list of locations where each sensor is deployed are some of the custom parameters that the custom query makes use of. The history of stakeholder-created custom datasets is displayed in Fig. 7.

- Input(s): The sensor, its position, and the date the data was sensed are the inputs fed into the algorithm.
- Processing: The interface receives the processed data in order to deliver it to the end user. Based on many criteria, such as the location of the sensor, description of the sensor, type of upload, and date of upload, the data might be retrieved from the blockchain.
- Output: The updated data on the user interface is the output.

C. Tools and technologies used to implement the algorithms: This section summarizes the complementary role of relevant tools and technology for the proposed solution. The goal of this discussion is to increase the reader's general comprehension of technology. The technologies and tools are stacked, as seen in Fig. 9. The process of creating a hash key involves placing the sensor data into a CSV file, encrypting it, and then

uploading it to the IPFS network. The NodeJS framework provides numerous tools for the development of server-side applications. We employed VSC to launch the NodeJS application.

CONTRACT	ADDRESS
DatasetsStoring	0xc9Ef8B8B5339Be517392d5034185B3dfD89f72A1

FUNCTION
AddCustomDataset(_name:string, _loc: string, _sensor: string, _sdate: string, _edate: string, _edate: string, _fhash: string)

INPUTS
My Custome Dataset, All-Locations, All-Sensors, , , QmNb8B8t15DAkyVhVk1gCj9UyfEk8131o6ELhFWrCrX6rZ

Fig. 7: Representation of IoT data in blockchain-based ledger

Algorithm 3: Secure access to the blockchain-based data

```
Input: σ, ℒ, Δp,ρD                          ▷ Sensor, Location, Date
Output: ℛ, μ                                ▷ Display Data
procedure INTERFACEMODULE                   ▷ Event based function
    if σ == 𝒩 then
        μ ← GetData(σ)                      ▷ Get Data against Sensor
      ρD == 𝒩
        μ ← GetData(ρD)                     ▷ Get Data against given Location
      ℒ == 𝒩
        μ ← GetData(ℒ)                      ▷ Get Data against Location
        μ ← GetData(σ, ℒ)                   ▷ Get Data against Sensor Location
    end if
    ℛ ← UpdateDashboard(μ)                  ▷ Update available data on user screen
end procedure
```

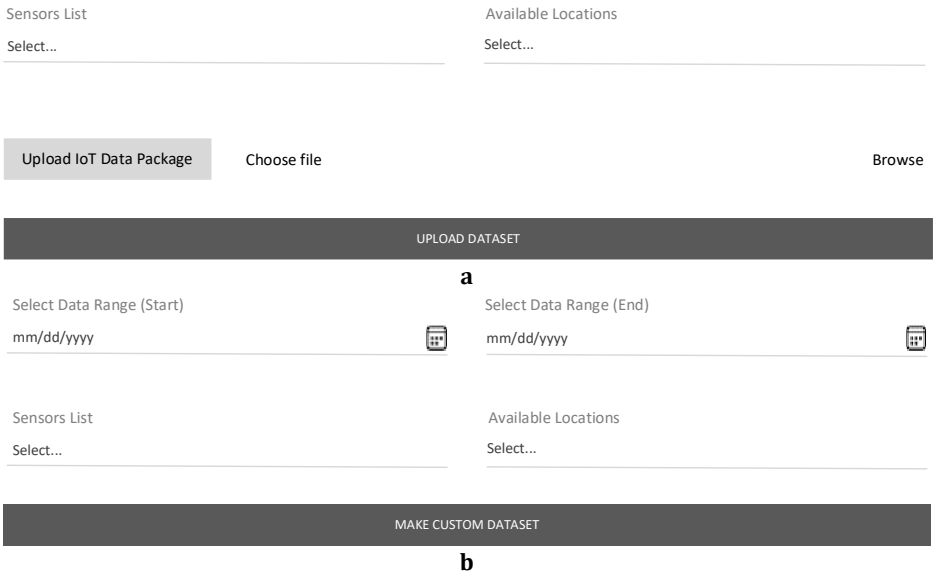


Fig. 8: Case study-based trial for (a) data packaging and (b) selection of IoT sensors

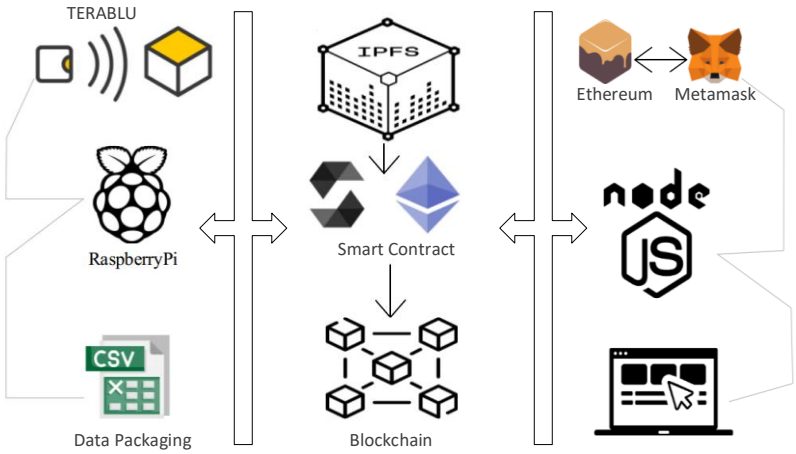


Fig. 9: A layered architecture view of tools and technologies for algorithmic implementation

5. Evaluation and validity threats

This section outlines the outcomes associated with the proposed solution. First, the assessment environment; next, the smart contract's fuel utilization functioning is evaluated. Next, we quantify and assess query response (i.e., performance), algorithmic execution employing criteria (i.e., efficiency), and data uploading and storage to the blockchain. The ISO/IEC-9126 model (Estdale and Georgiadou, 2018), which is used to evaluate the quality of software-intensive systems, serves as the foundation for the assessment criteria. Potential boundaries that need to be considered are examined, along with risks to the validity of the research.

A. Setting up the evaluation environment for the solution: In the evaluation environment, the solution is executed on a variety of hardware and software resources, tracking all of the execution steps and outcomes along the way. Specifically, hardware evaluation tests were conducted for both automatic IoT data uploading and manual user input using the Windows Platform (core i7 with 16 GB of runtime memory). System testing is automated by execution evaluation, which is also known as evaluation scripts in the software industry. Visual Studio Code was used to run similar NodeJS scripts built in the ReactJS programming language. Furthermore, the review process leverages several pre-existing libraries, such as (react, web3, ipfs.http), among others. For example, tracking CPU utilization during data uploads to IPFS and blockchain storage, as well as during data retrieval from the blockchain, is done via a JavaScript performance library script. A Ganache suite serves the purpose of creating a local Ethereum blockchain environment, and links to distributed websites are made possible with the use of a browser extension called Metamask. Using gas transaction fees, The Ganache suite, along with the Metamask extension, facilitates the connection of local Ethereum accounts to perform system activities.

B. Evaluating the energy efficiency–gas consumption: In order for the Ethereum smart contract to function, it is necessary to compensate for the fuel utilized. The fuel consumption associated with the initial data upload was measured to facilitate a comparison of the fuel demands between the two distinct methods of uploading the data. The smallest Ethereum currency, known as Gwei, is used to quantify fuel use. 109 Wei is referred to as Gwei. In our suggested solution, the cost of contract migration execution is shown (Table 1). The cost is stated in Ether and includes the used Gas. One unit of ether is equal to the quantity of gas used multiplied by the price of gas. The gas used in this arrangement represents the ongoing expense of computing. In order to adjust for differences in the value, the network has adjusted the cost of the system prototype that is now in use and has an automatic gas restriction set. The total amount of gas used is 2027188, and the cost of generating the contract is 0.04054376 in ether. Contract creation is necessary for the migration, and it only consumes 28636 gas and costs 0.1926345 (Ether). The overall costs can be further decreased if the input data is small.

Table 1: Cost analysis for data storage (price of gas=2 Gwei)

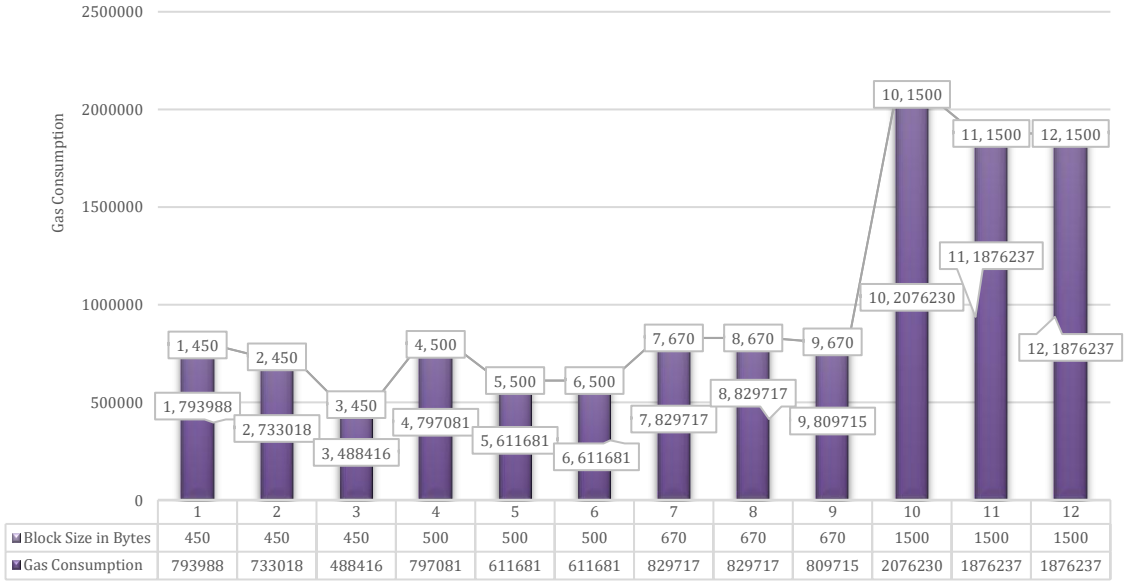
Type of execution	Utilization of gas	Ether cost
Creation of contract	2027188	0.04054376
Call for contract migration	28636	0.1926345
Cost of contract (initial)	24723	0.0549561
Cost of migration (initial)	44656	0.0192457

C. Evaluation environment: The final test item examined how long it takes users to share information with others. Data sharing time is defined as the total amount of time spent reading, remembering, and sharing data. Fig. 10 shows the results of multiple sets of experiments we conducted using an average data size. The average fuel consumption is around 671,807 Gas for uploading 450 bytes of data and 1,942,901 Gas for storing 1500 bytes of data. This illustrates how fuel consumption increases in tandem with data size. But when the IoT data was moved to IPFS using the recommended method, there was no appreciable change in fuel use, despite the fact that the volume of data increased.

A sequence diagram illustrating each object in the network and their interactions is presented in Fig. 11. Five distinct entities are present. Only the administrator, who has direct access to the dataset, uses the manual upload entity. The dataset-based time cycle can be uploaded into the system more easily thanks to the system upload entity. Any stakeholder with access to the public domain may be regarded as a user entity and could retrieve the data using their own unique queries. The execution flow is depicted in Fig. 12 to aid in the explanation of how the system works.

D. Evaluations of query response time: Data querying is necessary for storing IoT data packages to IPFS and recording details in the blockchain. The query response time can be used to assess how well the solution stores and retrieves data from the blockchain. We examined two distinct approaches: the query response time for IoT data stored on IPFS and the query response time for blockchain records including file hashes. The query response time in milliseconds is shown in Fig. 13, where the vertical axis shows the response time in milliseconds and the horizontal axis shows the two distinct execution routines. "Complete function" describes how the entire process is carried out up until the point of storing the IoT data package on IPFS and recording the record detail on the blockchain along with the IoT data file hash. The

delay caused by the Smart Contract execution call made using Metamask is displayed by the "Smart Contract Function." We also assess the performance of CPU use while a set of functions is being executed using smart contracts. Similar to data exchange, we evaluated every phase of every tactic. There are a few methods, such as uploading to IPFS storage, calling encryption of the dataset, and saving information in the blockchain ledger.



	1	2	3	4	5	6	7	8	9	10	11	12
Block Size in Bytes	450	450	450	500	500	500	670	670	670	1500	1500	1500
Gas Consumption	793988	733018	488416	797081	611681	611681	829717	829717	809715	2076230	1876237	1876237

Fig. 10: Gas used in relation to transaction count and block size

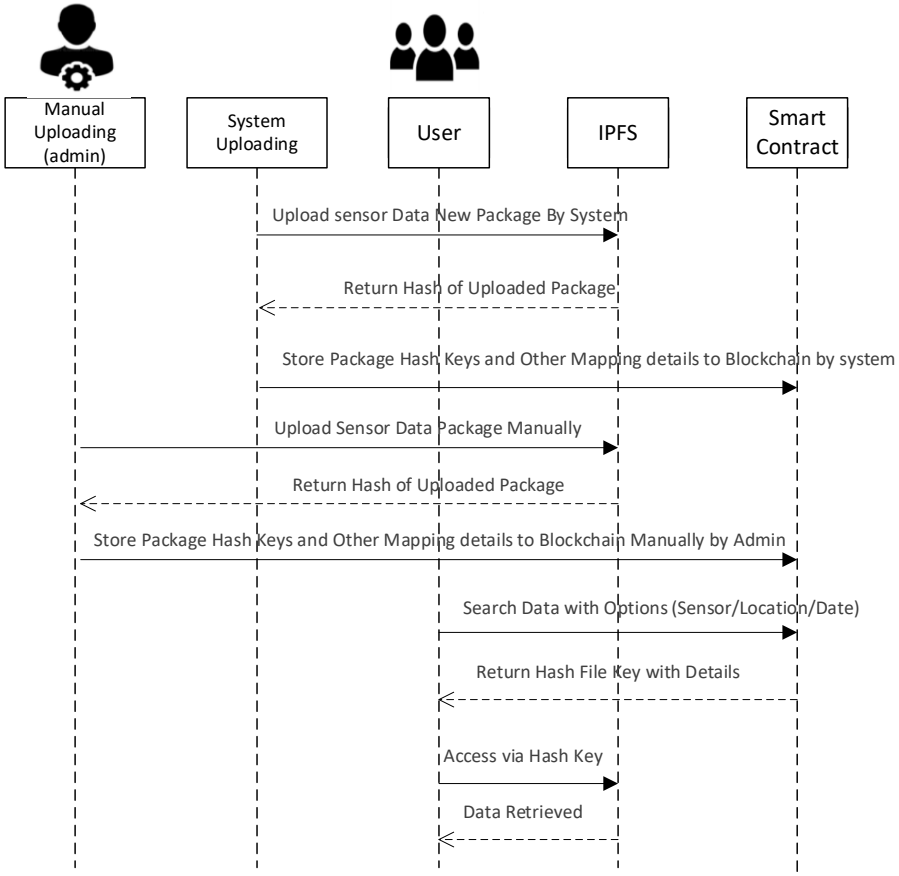


Fig. 11: UML sequence diagram for data sharing process as a sequence of interactions

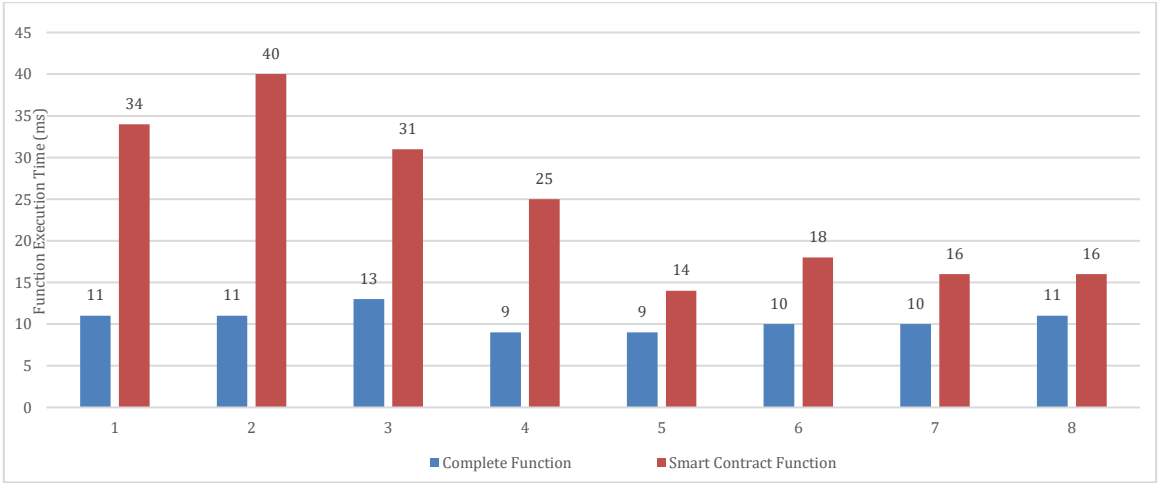


Fig. 12: Trials to assess the computational time taken for storing data in IPFS and blockchain

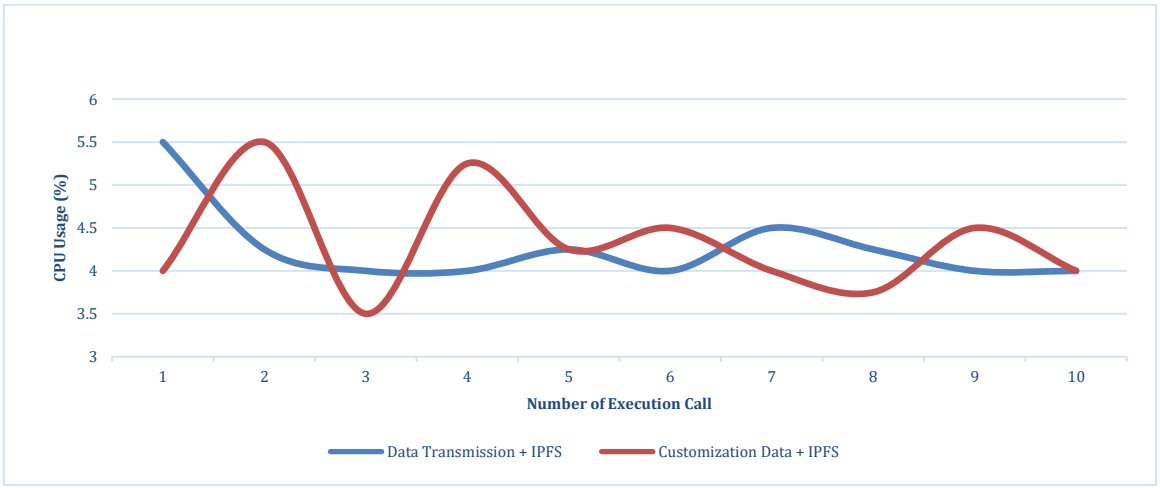


Fig. 13: CPU time spent performing calculations

6. Conclusion

The IoT consists of widespread systems that integrate embedded networks, sensors, and applications to develop intelligent environments and systems. Establishing an efficient framework for exchanging and storing IoT data in dynamic and potentially hazardous environments is crucial. This study explores the use of Ethereum smart contracts to facilitate IoT data exchange through blockchain technology, thereby creating a decentralized and reliable access control framework.

An Ethereum smart contract was implemented to provide a secure and distributed access control system for IoT data sharing. The proposed approach utilizes the Ethereum blockchain and the InterPlanetary File System (IPFS) to ensure secure storage of IoT data. Smart contracts simplify the management of access permissions, allowing users to control and preserve access roles effectively. In this method, IoT data is encrypted and transferred to IPFS for decentralized storage. A hash value, along with additional metadata, is then recorded on the blockchain ledger for security and verification.

To evaluate the system's performance, an experiment was conducted using datasets of different sizes to analyze data upload and access efficiency. The results showed that as data volume increased, the upload process became faster and more efficient. Additionally, it was observed that the fuel consumption associated with the blockchain-based upload method remained constant regardless of data size. Future studies will focus on the following areas:

- Expanding the proposed solution to other application domains, such as smart healthcare, financial technology (FinTech), and smart city systems.
- Conducting diverse case studies to systematically evaluate the applicability of the proposed framework. Empirical validation will be necessary, relying on experimentation and real-world use cases to establish practical guidelines for designing, implementing, and validating blockchain-based solutions for decentralized data management.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Ahmad A, Altamimi AB, and Aqib J (2024). A reference architecture for quantum computing as a service. Journal of King Saud University-Computer and Information Sciences, 36(6):102094. https://doi.org/10.1016/j.jksuci.2024.102094

Ahmad A, Waseem M, Liang P, Fahmideh M, Aktar MS, and Mikkonen T (2023). Towards human-bot collaborative software architecting with ChatGPT. In the 27th International Conference on Evaluation and Assessment in Software Engineering, Association for Computing Machinery, Oulu, Finland: 279-285. https://doi.org/10.1145/3593434.3593468

Benet J (2014). IPFS - Content addressed, versioned, P2P file system. Arxiv Preprint Arxiv:1407.3561. https://doi.org/10.48550/arXiv.1407.3561

Estdale J and Georgiadou E (2018). Applying the ISO/IEC 25010 quality models to software product. In the 25th European Conference on Systems, Software and Services Process Improvement, Springer International Publishing, Bilbao, Spain: 492-503. https://doi.org/10.1007/978-3-319-97925-0_42

Hammi MT, Hammi B, Bellot P, and Serhrouchni A (2018). Bubbles of trust: A decentralized blockchain-based authentication system for IoT. Computers and Security, 78: 126-142. https://doi.org/10.1016/j.cose.2018.06.004

Kokoris Kogias E, Alp EC, Gasser L, Jovanovic PS, Syta E, and Ford BA (2021). CALYPSO: Private data management for decentralized ledgers. Proceedings of the VLDB Endowment, 14(4): 586-599. https://doi.org/10.14778/3436905.3436917

Liang W, Tang M, Long J, Peng X, Xu J, and Li KC (2019). A secure fabric blockchain-based data transmission technique for industrial Internet-of-Things. IEEE Transactions on Industrial Informatics, 15(6): 3582-3592. https://doi.org/10.1109/TII.2019.2907092

Nizamuddin N, Salah K, Azad MA, Arshad J, and Rehman MH (2019). Decentralized document version control using Ethereum blockchain and IPFS. Computers and Electrical Engineering, 76: 183-197. https://doi.org/10.1016/j.compeleceng.2019.03.014

Razzaq A (2020). A systematic review on software architectures for IoT systems and future direction to the adoption of microservices architecture. SN Computer Science, 1: 350. https://doi.org/10.1007/s42979-020-00359-w

Razzaq A (2022). Blockchain-based secure data transmission for Internet of Underwater Things. Cluster Computing, 25(6): 4495-4514. https://doi.org/10.1007/s10586-022-03701-4

Rowhani-Farid A, Allen M, and Barnett AG (2017). What incentives increase data sharing in health and medical research? A systematic review. Research Integrity and Peer Review, 2: 4. https://doi.org/10.1186/s41073-017-0028-9 **PMid:29451561 PMCid:PMC5803640**

Shafagh H, Burkhalter L, Hithnawi A, and Duquennoy S (2017). Towards blockchain-based auditable storage and sharing of IoT data. In the Proceedings of the 9th ACM Cloud Computing Security Workshop (CCSW 2017), Dallas, USA: 45-50. https://doi.org/10.1145/3140649.3140656

Shrestha AK and Vassileva J (2018). Blockchain-based research data sharing framework for incentivizing the data owners. In the Blockchain–ICBC 2018: First International Conference, Held as Part of the Services Conference Federation, Springer International Publishing, Seattle, USA: 259-266. https://doi.org/10.1007/978-3-319-94478-4_19

Steichen M, Fiz B, Norvill R, Shbair W, and State R (2018). Blockchain-based decentralized access control for IPFS. In the 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), IEEE, Halifax, Canada: 1499-1506. https://doi.org/10.1109/Cybermatics_2018.2018.00253

Xia QI, Sifah EB, Asamoah KO, Gao J, Du X, and Guizani M (2017). MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. IEEE Access, 5: 14757-14767. https://doi.org/10.1109/ACCESS.2017.2730843