BLOCKCHAIN-BASED DEMAND-SUPPLY MATCHING SYSTEM FOR IOT DEVICE DATA DISTRIBUTION

Kenta Kawai[1], Wu Yuxiao, Yutaka Matubara, and Hiroaki Takada

Graduate School of Informatics, Nagoya University, Aichi 464-8601, Japan

ABSTRACT

The booming of IoT devices has attracted significant interest in data integration platforms that enable seamless utilization and control of sensor data across various applications. However, most existing platforms are centralized structure, aggregating data on specific companies' servers. This centralization raises privacy concerns and imposes limitations on data sharing with third parties. To address these challenges, this paper proposes a decentralized demand-supply matching system for IoT device data distribution using blockchain technology. The paper details the requirements for the entire matching system, including both users and IoT devices, and introduces a system concept alongside a practical implementation. Evaluation experiments conducted on a prototype system demonstrate the feasibility and effectiveness of the proposed approach.

KEYWORDS

Blockchain, Data Marketplace, Demand-Supply Matching, IoT Data

1. INTRODUCTION

Recent advancements in embedded device performance and network technologies have resulted in the widespread IoT integration of nearly all household electronic devices. These IoT devices no longer operate in isolation but instead interact with one another, providing more comprehensive information and enabling unified control. Moreover, the data and control functionalities of these devices have extended beyond local home and corporate networks, connecting to external networks to facilitate various services. To support this interconnectivity, data integration platforms play a crucial role. These platforms enable the aggregation, integration, and sharing of data generated by IoT devices, supporting not only data flow but also the integration and optimization of control mechanisms. For example, one such integration [1] involves a smart agriculture big data platform that uses IoT technology to collect real-time environmental data, such as temperature and soil moisture, enabling farmers to remotely manage and optimize their operations.

Data integration platforms for IoT devices are gaining attention across various fields. However, existing data integration platforms are predominantly centralized, with data being aggregated on servers managed by the manufacturers [2]. Figure 1 illustrates this traditional centralized architecture, where IoT devices installed in homes typically communicate with platforms provided by their manufacturers, enabling data transmission and control. Users leverage these platforms to utilize aggregated data for analysis and to exert control over external systems. These platforms support the interoperability of the manufacturer's IoT devices, allowing for complex services when devices are used in together. While this structure offers numerous benefits, it also

David C. Wyld et al. (Eds): DSML, ARIA, NLP, CSEN, BIBC, EDTECH - 2024
pp. 113-131, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.142408

presents several critical issues.

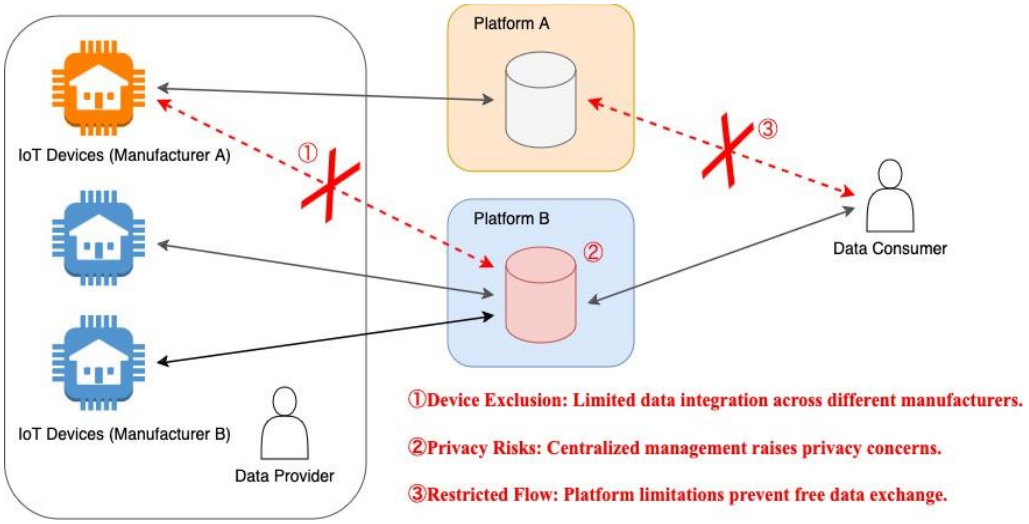


Figure 1. Centralized Data integration platforms

Firstly, as shown in Figure 1 (①), IoT devices from different manufacturers are typically incompatible with platforms other than their designated ones, which hinders the integration and exchange of data across devices. This limitation not only restricts the versatility of IoT ecosystems but also reduces the potential for comprehensive data analysis and control across multiple devices. Secondly, as indicated in Figure 1 (②), the centralized management of these platforms raises significant privacy concerns, as all data are collected and stored on manufacturer-controlled servers. This centralization can lead to the potential misuse or unauthorized access to sensitive user data [3]. Finally, Figure 1 (③) illustrates how the rigid structure of some platforms may prevent the free and unrestricted exchange of data, as the platform policies or technical limitations might inhibit users from sharing or utilizing their data beyond the confines of the platform.

To address the structural issues of centralized data platforms, research has increasingly focused on decentralized data integration platforms leveraging blockchain technology [4]. These decentralized systems distribute data ownership and access rights directly to users, reducing reliance on a central authority and promoting greater transparency and tamper resistance. Prior studies have explored encryption technologies to safeguard the confidentiality and integrity of data transactions, while also estimating the costs associated with blockchain-based data flows. These studies demonstrate the effectiveness of blockchain in enhancing security and transparency across decentralized ecosystems.

However, despite these advancements, many prior studies fail to fully address the comprehensive requirements of data integration platforms, particularly those involving users and IoT devices in a complete matching system with concrete implementations. Additionally, many existing platforms limit the provision of data to only representative values, which mitigates the risk of low-cost resale by buyers but significantly restricts the scope of data that can be offered. While this approach works well in large-scale marketplaces, it limits flexibility. Our study addresses these shortcomings by focusing on smaller-scale environments, such as homes or small businesses, where the risk of malicious reselling is lower and the need for a wider range of raw data and control capabilities is more pressing.

To address the above-mentioned challenges, this paper proposes a decentralized demand-supply matching system utilizing blockchain technology to facilitate the secure and integrated flow of IoT device data. The principal contributions of this research are as follows:

- **Comprehensive System Requirements Organization**: We systematically define and organize the requirements for IoT device data distribution platforms, emphasizing user-centric considerations such as data privacy, security, scalability, and the seamless interoperability of heterogeneous IoT devices. This structured approach provides a robust foundation for addressing the complexities inherent in distributed data systems.
- **Development of a Dual-Mode Data Distribution Mechanism**: We introduce a novel mechanism that enables both active and passive data distribution by users. This dual- mode functionality allows users to either actively engage in data transactions or utilize rule-based automation to manage data flows in the background. This approach enhances the system's flexibility and accommodates diverse user preferences, contributing to its practical applicability in various contexts.
- **Portability and Open-Source Implementation**: In this study, we have improved the portability of the system by utilizing Docker, ensuring that it can be deployed across various environments without compatibility issues. Furthermore, we have made the source code publicly available, providing transparency and fostering community collaboration for further enhancements and adaptations of the system.

To evaluate the effectiveness of the prototype system, we conducted a series of experiments that measured CPU usage, memory utilization, and gas costs during operation. The results demonstrated that the system operates within practical limits in terms of both computational load and transaction costs, making it feasible for real-world deployment. Moving forward, we plan to further extend our work by conducting stress tests using actual IoT devices and exploring improvements in the metadata structure to enhance usability and system efficiency.

This paper is structured as follows: Chapter 2 introduces foundational concepts, including blockchain technology and smart contracts. Chapter 3 reviews related work, focusing on decentralized data integration platforms and highlighting the limitations of these studies, thereby clarifying the position of this research. Chapter 4 outlines the system requirements for IoT data distribution and presents the architecture and design of the proposed system. Chapter 5 describes the system's implementation and discusses the results of performance evaluations, demonstrating the system's efficacy. Chapter 6 reflects on the experimental findings, examines the limitations of the prototype system, and outlines directions for future improvements. Finally, Chapter 7 concludes the paper with a summary of findings and potential avenues for future research.

2. BACKGROUND KNOWLEDGE

2.1. Blockchain

Blockchain is a type of database that directly connects terminals on an information communication network and processes and records transaction data in a decentralized manner using cryptographic technology. When a transaction occurs on the blockchain, its content is verified by participants and recorded in the ledger only if consensus is reached. Multiple transaction records in the ledger are grouped into a new block. This block not only contains the transaction data but also stores the hash value of the previously generated block. This process of linking blocks in a chain-like manner gives blockchain its name. Despite its decentralized nature, blockchain is highly resistant to data destruction and tampering. If a user attempts to alter a transaction, the hash value of the block containing the altered transaction will change.

Consequently, the hash values of subsequent blocks will also change sequentially, allowing other users to detect the tampering. Therefore, to tamper with the blockchain, one would need to regenerate all subsequent blocks, which is practically impossible. Additionally, unlike traditional centralized systems governed by a specific authority, blockchain involves multiple systems, each holding information and constantly synchronizing with each other. As a result, even if some systems stop or fail, the impact on the overall operation and functionality of the system is minimal [5].

2.3. Ethereum

Ethereum, introduced by Vitalik Buterin in 2014, represents a major development in blockchain technology and is often referred to as a second-generation blockchain [6]. Unlike Bitcoin, which primarily functions as a digital currency, Ethereum stands out for its support of smart contracts. These smart contracts are self-executing agreements with terms directly written into code, allowing transactions to be automated without the need for intermediaries, thereby making processes faster and more secure. A key innovation of Ethereum is its use of a Turing-complete programming language, which enables developers to create a wide variety of smart contracts on its blockchain. This has made Ethereum the go-to platform for building decentralized applications (DApps), greatly expanding the potential uses of blockchain technology.

Originally, Ethereum used a Proof of Work (PoW) consensus algorithm, where miners solved complex puzzles to validate transactions and secure the network. However, Ethereum has recently transitioned to a Proof of Stake (PoS) model, which is more energy-efficient and better suited for scalability [7]. This shift helps Ethereum remain sustainable and capable of handling the growing demands of its network.

2.2. Ethereum Dapps

DApps, or decentralized applications, operate on blockchain technology. Unlike traditional applications, DApps do not have a central management body and are instead run by all nodes participating in the blockchain. This eliminates the need for intermediaries, enabling open information flow and allowing users to conduct highly transparent transactions. The key difference between DApps and traditional web services lies in how the data is processed—specifically, whether the backend services are decentralized. As a result, many existing web services can be built as DApps. Known fields where DApps are deployed include social networking [8], gaming [9], and finance [10].

2.4. Ethereum Events

Events are a feature used in smart contracts to notify applications connected to the blockchain of specific processes [11]. By declaring events within a contract and using the reserved word **emit** in functions, event information is stored in the Ethereum Virtual Machine (EVM) log area. Applications connected to the blockchain periodically query this log area to check for issued events. When an event is detected, the application can perform specific actions using the internal data of the event.

3. RELATED WORK

In this chapter, we review previous research on blockchain-based data integration platforms and examine how these studies have addressed key challenges in the field. We also identify the current focus areas of existing research and highlight their limitations. Finally, we position our

work in relation to these studies, explaining how our approach addresses the challenges that remain unresolved.

Table 1. Comparison of Blockchain-Based Data Distribution Platforms

	[6]	[7]	[8]	[9]	Proposed System
Target Area	Cross Domain IoT	Personal ~ Organization IoT Data	Medical	Spatial	Home ~ Organization IoT Data
Privacy Preservation	✓	✓	✓	✓	✓
Device Compatibility	✓	✓	✓	✓	✓
Decentralized Manner	-	✓	✓	✓	✓
Tamper Resistance	-	✓	✓	✓	✓
High Throughput	✓	-	-	-	✓
User Entry Policy	✓	-	-	-	✓
Open-Source Availability	-	-	-	-	✓

Xinghui et al. [12] proposed a decentralized data integration platform using a consortium blockchain to address cross-domain data sharing challenges caused by incompatible communication protocols. Their system aggregates IoT data from various domains, such as urban environments and agriculture, reconciling domain differences and recording access information on the blockchain. A smart contract-based authentication system ensures secure cross-domain data requests. The platform's feasibility was demonstrated through a practical experiment conducted in Xi'an, China.

Vlasis et al. [13] introduced "Agora," a privacy-preserving decentralized data marketplace that combines blockchain and cryptographic techniques. In Agora, encrypted data is shared with brokers, who compute functions using functional encryption keys, revealing only the results while keeping raw data private. Smart contracts automate secure payments, but high verification costs limit Agora's scalability, and its reliance on weighted sums restricts complex data processing.

Alsharif et al. [14] developed a blockchain-based medical data marketplace focused on confidentiality and transparency. Their platform uses attribute-based encryption (CP-ABE) and zk-SNARKs to ensure that only authorized entities can access sensitive data. Smart contracts automate fair exchanges by requiring verifiable proof of payment and delivery, preventing fraud and ensuring data integrity.

Hui et al. [15] applied blockchain to spatial data markets, securing data with Interplanetary File System (IPFS) for storage and function encryption for privacy. Their system uses a Vickrey

Reverse Auction for fair data pricing, where sellers receive the second-lowest bid. Although the approach protects privacy through disposable accounts, the auction's high execution costs pose scalability challenges.

In this study, we created a comparison table (Table 1) to analyze the key features of prior works. The comparison is based on eight criteria: the six requirements defined in Section 4.1, along with Target Area and Open-Source Availability. These criteria enable us to evaluate and discuss the strengths and weaknesses of the prior research in comparison to our own study.

First, while the transparency of blockchain can sometimes pose challenges to privacy preservation, extensive research in cryptography has aimed to address these issues. Various methods have been developed to ensure privacy protection. Although the approaches and strengths of these methods differ, both prior works and our study adequately address privacy concerns.

Regarding device compatibility, all the reviewed studies meet this requirement. For instance, while the research by Alsharif et al. and Hui et al. focuses on specific domains, their mechanisms are not restricted to any particular type of IoT device, making them broadly applicable to a wide range of IoT data.

The work by Xinghui et al. shares many similarities with our study, particularly in its ability to handle large volumes of data at high speed. Their experiments demonstrated that users could easily engage with the system through user-friendly applications, making system participation straightforward. However, their approach relies on data-sharing components within each domain for data collection and distribution, rather than giving users direct control. Additionally, their study lacks a mechanism for verifying data integrity.

Overall, numerous studies on blockchain-based data distribution platforms have focused on leveraging cryptographic techniques to ensure privacy, confidentiality, and data integrity. However, few studies have fully implemented the entire process, from user participation to actual data distribution. Our research distinguishes itself by conducting a comprehensive analysis of the system requirements for a complete matching system that includes users and IoT devices, implementing these requirements, and releasing the solution as an open-source project.

4. SYSTEM REQUIREMENTS AND PROPOSED SYSTEM

4.1. System Requirements

In this section, we outline the key system requirements necessary to facilitate smooth data distribution among users and IoT devices participating in the platform. These users include not only the general public, who may lack advanced knowledge of IoT devices, but also corporations and organizations. The primary system requirements are as follows:

- Privacy Preservation
- Excellent Device Compatibility
- Decentralized Structure
- Tamper Resistance of Data
- High Throughput
- Flexible User Entry Policy

Privacy preservation is essential because data collected by IoT devices can potentially threaten privacy when aggregated. For example, data on the operation times of home appliances can reveal a user's daily routine. Therefore, safeguarding user privacy is a crucial requirement.

The system must ensure excellent device compatibility to accommodate the rapidly increasing number of IoT devices with diverse performance levels and data types. To achieve seamless data distribution across this varied ecosystem, a unified method that operates independently of device capabilities and data characteristics is essential.

The platform should be operated with a decentralized structure by its users. This approach eliminates the risk of improper data usage by a central authority, ensuring the platform remains transparent and trustworthy. By distributing control among users, the platform fosters a more open and secure environment for equitable data exchange, promoting a dynamic and inclusive data distribution market for IoT devices.

Tamper resistance of data is critical for maintaining the reliability of services that depend on it. If data is tampered with, it could undermine trust in the services using it. Therefore, the system requires mechanisms to ensure that the data collected by IoT devices remains identical to the data exchanged.

The platform must support high throughput to efficiently manage large volumes of data. Applications like traffic monitoring require real-time processing, while accurate data analysis and machine learning rely on substantial data sets. Therefore, the system must be designed to handle large-scale data rapidly and efficiently to meet the demands of these use cases.

A flexible user entry policy is crucial for encouraging widespread adoption of the platform. Since the proliferation of IoT devices is expected to be driven largely by general users, their participation will be key to the platform's success. To engage and retain these users, the system must include mechanisms that simplify the data distribution process for individuals without advanced technical expertise, fostering active and sustained participation.

These system requirements were derived from our research group's analysis of common demands in previous studies on data distribution platforms and existing services. To build a more practical and robust system, it is essential to incorporate insights from experiments involving technicians and users experienced in IoT device development, and to continuously improve the system based on these insights.

4.2. System Overview

This subsection outlines the proposed demand-supply matching system, with its overall architecture depicted in Figure 2.

1. **IoT Devices**: Controlled by the Data Provider, these devices transmit data and execute operations based on the provider's requirements.
2. **Data Provider**: The owner of the IoT devices, responsible for managing the data they generate. The Data Provider can be an individual or organization, seeking social or financial benefits from providing this data.
3. **Data Provider’s Mediator**: Acting as the Data Provider's agent, the Mediator manages IoT devices, processes data, and accesses the blockchain. It also automatically generates and deploys metadata to the Data Exchange DApps based on the Data Provider’s settings to facilitate smooth data distribution.

4. **Data Consumer**: The user who enters a transaction with the Data Provider via a Data Request and utilizes the data for purposes such as analysis or service improvement.
5. **Data Consumer's Mediator**: Managed by the Data Consumer, this Mediator acts as their agent, handling the Data Request, decryption, Data Integrity Verification, and payment settlement upon completion of the data acquisition process.
6. **Data Storage**: The cloud service used by the Mediators of both the Data Provider and Data Consumer to exchange data. This can include cloud storage services like Dropbox, decentralized storage using IPFS, or data brokerage servers with pub/sub messaging. By not limiting data mediation to the blockchain, the system can handle large-scale data efficiently, though additional measures for data integrity and access control are necessary.
7. **Data Exchange DApps**: A set of Blockchain Smart Contracts and web applications that facilitate data transactions. DApps manage metadata provided by the Data Provider, including the provider's ID, data volume, type, distribution status, and a hash value for later Data Integrity Verification. The web application offers an intuitive interface, enabling Data Consumers to make Data Requests and engage in transactions without blockchain expertise.

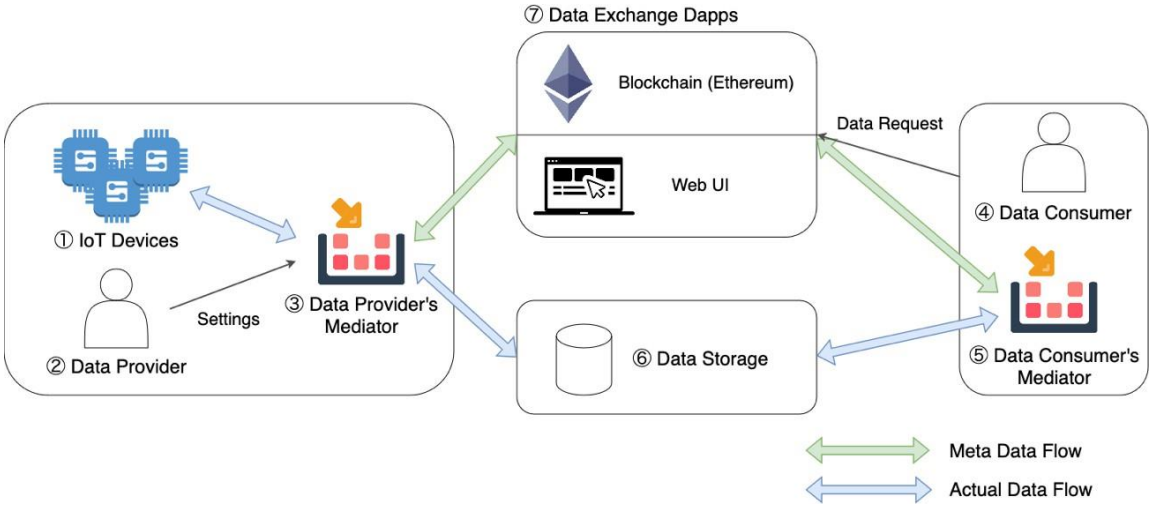


Figure 2. System Overview

4.3. Data Distribution Flow

This subsubsection provides a detailed explanation of the data transaction flow within the proposed demand-supply matching system, as illustrated in the sequence diagrams shown in Figures 3 and 4. Figure 3 depicts the flow from system setup, through the Mediator's deployment of metadata, to the Data Consumer's Data Request, while Figure 4 illustrates the process from the Data Request to the completion of the transaction. It is important to note that the Upload and Data Request steps, as shown in Figure 3, can occur repeatedly throughout the transaction process.

The system is designed with the assumption that users operate multiple IoT devices within a local network at home or within an organization. These devices continuously collect sensor data, which is then transmitted to a local system managed by the user. To deploy the system, the market owner initiates the deployment of Blockchain Smart Contracts on the blockchain, establishing them as Data Exchange Dapps, and launches the corresponding web application. Additionally, Ethereum's source code verification tools allow the market owner to validate the

integrity of the deployed smart contracts, effectively preventing unauthorized use. This process not only ensures the reliability of the system but also enhances user confidence in participating on the platform.

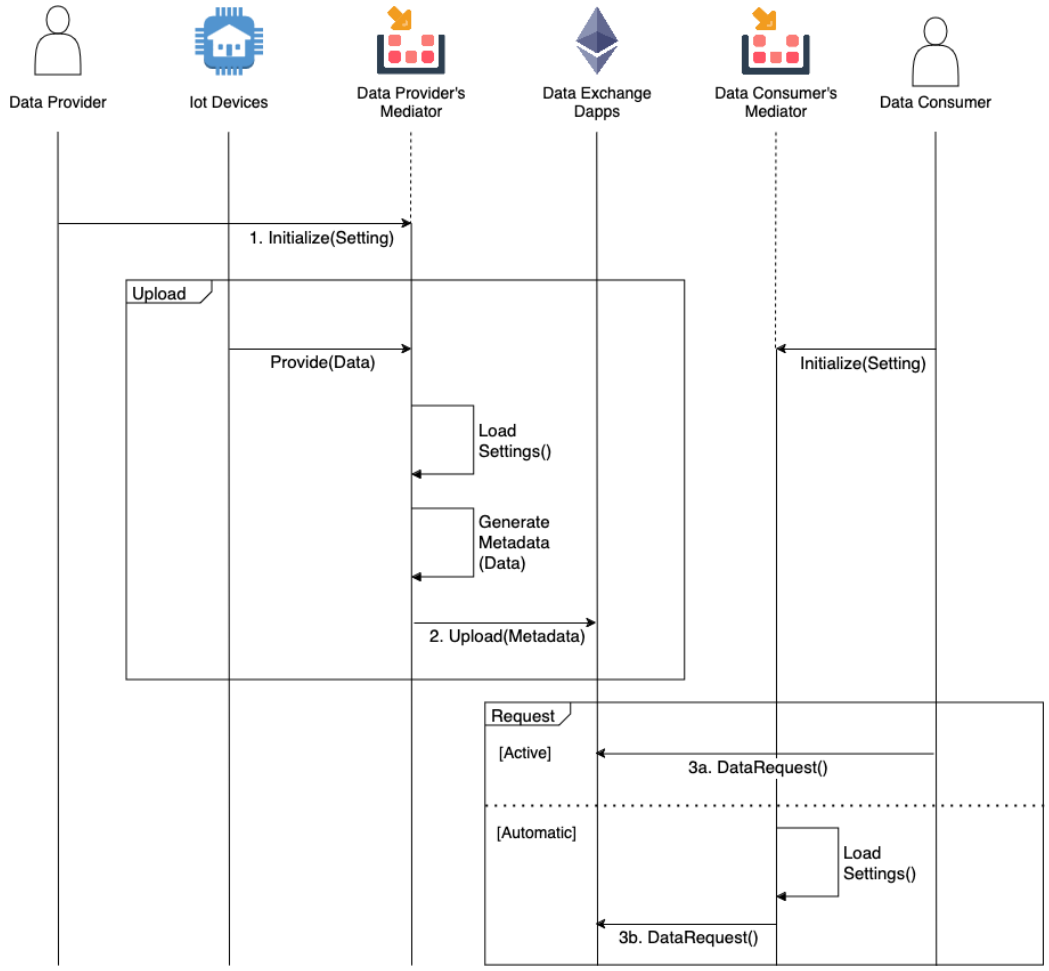


Figure 3. Data Distribution Flow (Setup ~ Data Request)

Once the system is deployed, users can set up and configure Mediator nodes within their local network (Step 1 in Figure 3). This configuration includes defining how the Mediator will access the IoT devices, providing blockchain account details, and setting up automation rules for transaction processes. These automation rules serve two primary functions: the first is for data provision, where the Mediator automatically deploys smart contracts when certain data conditions are met, thereby initiating the process of data provision. The second function is for data acquisition, where the Mediator automatically monitors metadata and requests data that meets specified conditions.

As the system operates, the Mediator continuously monitors data from connected IoT devices. When the data meets the predefined conditions for provision, the Mediator generates metadata based on the collected data. This metadata includes the data type, volume, the Data Provider's ID, the current distribution status, and a hash value used for Data Integrity Verification later in the process. The metadata is then deployed to the Data Exchange Dapps, making it available for other participants in the system to access (Step 2 in Figure 3).

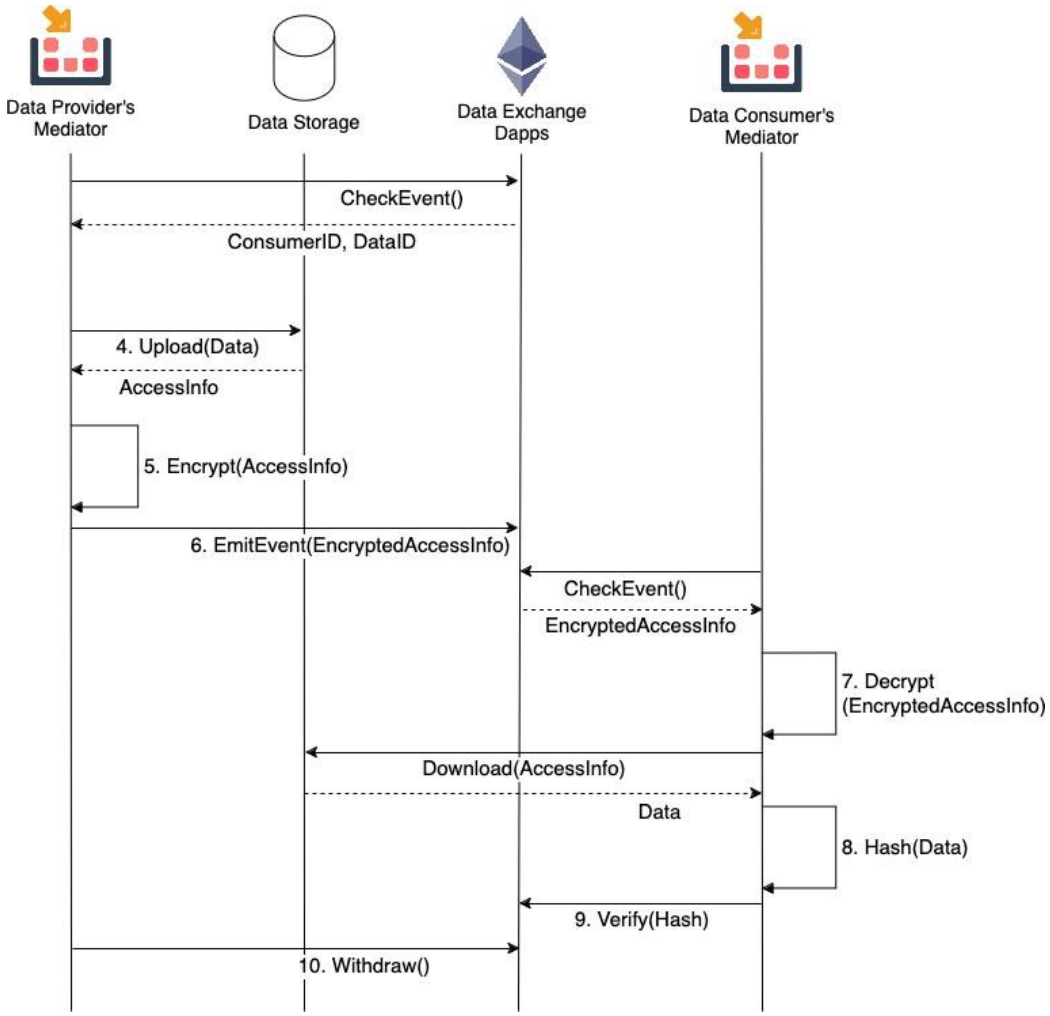


Figure 4. Data Distribution Flow (After Data Request)

Data Consumers can request the necessary data based on this deployed metadata. There are two primary methods for making a Data Request: one method involves the Data Consumer actively requesting data through the web application (Step 3a in Figure 3), and the other involves the Mediator automatically making a Data Request based on preconfigured rules (Step 3b in Figure 3). When using the web application, the Data Consumer can search through the available metadata, select the required data, and submit a Data Request. In the rule-based method, the Mediator monitors the metadata deployed to the Data Exchange Dapps in the background, and if it finds a match, it automatically initiates a Data Request. In certain cases, the system may require a Digital Currency payment at the point of request. However, the Data Provider cannot withdraw the Digital Currency until the subsequent Data Integrity Verification is successfully completed.

Once a Data Request is made, the Data Exchange Dapps issue events to both Mediators, instructing them to proceed with the data exchange. The Data Provider's Mediator then uploads the actual data to the specified Data Storage and retrieves the access information (Step 4 in Figure 4). This access information is encrypted using the Data Consumer's public key and broadcasted as an event on the blockchain (Step 5,6 in Figure 4). The Data Consumer's Mediator monitors the blockchain, captures the event, and decrypts the access information using its private key (Step 7 in Figure 4). This allows the Data Consumer to securely retrieve the requested data from the Data Storage.

To ensure data integrity, the Data Consumer's Mediator calculates a hash value of the retrieved data and sends it to the Data Exchange Dapps for verification (Step 8,9 in Figure 4). The Dapps compare this hash value with the original hash value recorded in the metadata to confirm that the data has not been tampered with. If the verification is successful, the Data Provider is then permitted to withdraw the Digital Currency (Step 10 in Figure 4). If the verification fails, the process is restarted, ensuring that both parties fulfill their obligations before completing the transaction.

To ensure data integrity, the Data Consumer's Mediator calculates a hash value of the retrieved data and sends it to the Data Exchange Dapps for verification (Step 8,9 in Figure 4). The Dapps compare this hash value with the original hash value recorded in the metadata to confirm that the data has not been tampered with. If the verification is successful, the Data Provider is then permitted to withdraw the Digital Currency (Step 10 in Figure 4). If the verification fails, the process is restarted, ensuring that both parties fulfill their obligations before completing the transaction.

These processes collectively ensure the integrity and transparency of data transactions within the system, making data distribution reliable and secure. The automation rules set in the Mediator significantly reduce the user's manual effort, enabling both Data Providers and Data Consumers to efficiently conduct secure data transactions. Furthermore, by employing a straightforward encryption and decryption mechanism, the system can accommodate various types of Data Storage, providing flexibility without being constrained to a specific storage solution. Additionally, the design of the system minimizes the incentive for either party to engage in dishonest behavior, as neither the Data Consumer receives verified data nor the Data Provider withdraws Digital Currency until the entire transaction is successfully completed. This transaction mechanism ensures that the system adheres to the requirements established in Section 4.1.

5. IMPLEMENTATION AND EVALUATION

5.1. Implementation

5.1.1. Implementation Overview

To demonstrate the feasibility of the proposed system, we implemented a proof of concept (PoC) that supports active data transactions facilitated by users via a web application. This implementation focuses on evaluating the core functionality of the system, specifically its efficiency and security in handling data transactions.

The overall architecture of the prototype system is illustrated in Figure 5. In this implementation, we focused on realizing four key components from the proposed system described in Chapter 4. These components are the Data Provider's Mediator, Data Consumer's Mediator, Data Storage, and Data Exchange Dapps. The Data Provider's Mediator manages data collected from IoT devices on behalf of the data provider, while the Data Consumer's Mediator is responsible for receiving the data and verifying its integrity on behalf of the data consumer. The Data Storage component supports secure data exchange between the Mediators, and the Data Exchange Dapps mediate data transactions through smart contracts.

For the implementation, we selected an optimal technology stack for each component. The Data Provider's Mediator and Data Consumer's Mediator were implemented using Rust [14], chosen for its high performance and memory safety, which are crucial for real-time data processing. The blockchain-related parts of the Data Exchange Dapps were developed using Solidity for smart

contract development, with Hardhat serving as the integrated development environment. Hardhat is optimized for smart contract development, testing, and private chain deployment.

Additionally, the UI was developed using SvelteKit, a web application framework, implemented with TypeScript for efficient and robust user interface development.

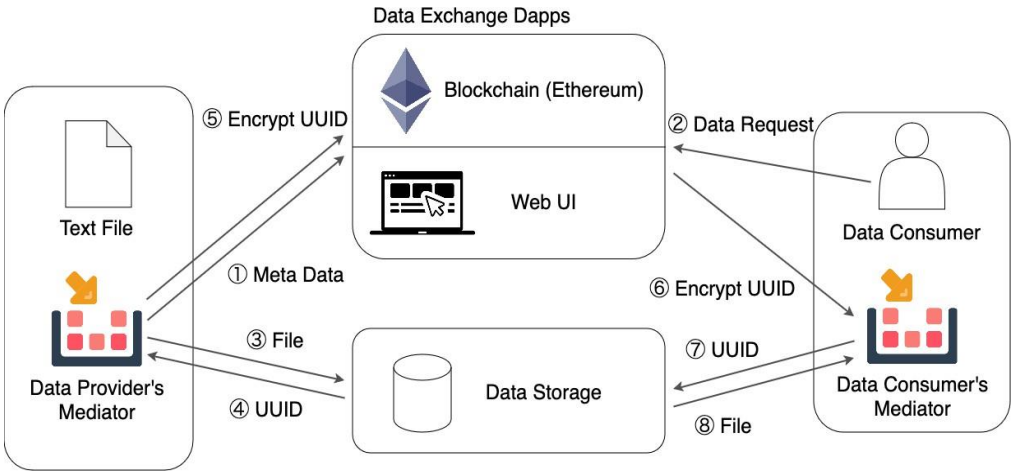


Figure 5. Implemented System and Data Flow

The development environment was established using VSCode's DevContainer [14], allowing for the creation of isolated development environments for each component. DevContainer leverages container technology to ensure consistency across development environments by encapsulating the dependencies and toolchains required for each component. To facilitate interaction between components, these containers were configured to operate on the same network, enabling seamless testing of inter-component communication.

The implementation of the system has been made available as open-source software. The source code, along with detailed documentation, can be accessed at (https://github.com/ertlnagoya/Blockchain_IoT_Marketplace/). This availability encourages further experimentation and collaboration within the research community.

5.1.2. Component Implementation

Data Provider's Mediator and Data Consumer's Mediator share the same source code, with configuration files determining the specific account used by each. In this proof of concept, the sample text file within the Data Provider's Mediator project serves as the target for distribution. Once the system is initialized, metadata is deployed to the Data Exchange Dapps. When a Data User makes a data request via the UI application, the Data Consumer's Mediator accesses the Data Storage, retrieves the requested text file, and saves it within the project. Additionally, the UUID generated during the data upload to Data Storage is encrypted and decrypted to prevent unauthorized third-party access, ensuring secure transactions.

Data Storage functions as a web server, providing the following handlers: / responds to GET requests for health checks, returning a 200 OK status. The */upload* endpoint responds to PUT requests, handling *multipart/form-data* files and saving them. To test the integrity verification feature, there is a 50% chance that the file will be tampered with during this process. After the file is saved, a randomly generated UUID is issued and returned as a response, which serves as the file path for subsequent downloads through the */download* endpoint. The UUID is generated

using a secure method, making it resistant to external guessing.

Data Exchange Dapps are composed of smart contracts running on the blockchain and a connected web application. The smart contracts include a contract managing the list of metadata and individual contracts for each metadata entry, enabling efficient data transaction management. The UI application features both a metadata listing page and a detailed view, allowing users to review details and make data requests. Furthermore, the UI application supports MetaMask, enabling users to easily and securely perform data transactions by leveraging their digital wallets within the web browser. MetaMask enhances the transparency and security of transactions by allowing users to manage digital currencies and sign blockchain transactions directly.

5.1.3. Deployment and Configuration

This subsection outlines the deployment and configuration steps required to initialize each component of the proposed system and ensure the entire system operates as intended. These steps follow the data distribution flow described in Chapter 4.

The development environment utilizes VSCode's Dev Container, providing isolated execution environments for each component. These environments are configured to operate on the same network, facilitating easy testing of communication between components.

The deployment begins with the initialization of the private blockchain within the Data Exchange Dapps to provide an environment where smart contracts can operate. Subsequently, a script included in the project is executed to deploy the smart contracts that manage the metadata list, establishing the foundation for metadata deployment.

Next, the account information from the private blockchain is copied and reflected in the configuration files of both the Data Provider’s Mediator and the Data Consumer’s Mediator. While the default settings are sufficient for operation, the configuration files can be modified if necessary. The MetaMask extension is then installed in the web browser and set up to connect the user to the private blockchain. Utilizing MetaMask ensures secure data requests.

Data Storage is activated using the cargo run command, placing the system in a state where it is ready to handle requests. Subsequently, both the Data Provider’s Mediator and Data Consumer’s Mediator are also initiated using the cargo run command. Upon startup, each Mediator performs a health check on the Data Storage and the private blockchain. Following this, the Data Provider’s Mediator reads a text file, creates metadata, and deploys it to the blockchain, preparing the system to operate according to the data distribution flow.

Finally, the web application is accessed via a browser, where a request is made for the deployed data. MetaMask facilitates the signing of the request, initiating a secure data transaction. This process enables the Data Provider’s Mediator and Data Consumer’s Mediator to collaborate, executing the data distribution and saving the text file within the Data Consumer’s Mediator project. This sequence of steps confirms the coordinated operation of the entire system.

5.2. Evaluation

5.2.1. Introduction and Experimental Environment

To quantitatively evaluate the performance of the implemented prototype system, we conducted a series of experiments focusing on three key aspects:

1. CPU and memory usage of the Mediator nodes
2. Costs for executing and deploying functions on the blockchain

These experiments were performed on a MacBook Air with the following specifications: Apple M3 chip (8-core CPU, 10-core GPU, 16-core Neural Engine), 16 GB of unified memory, and a 512 GB SSD. The operating system used was macOS Sonoma (version 14.6.1). Docker Desktop (version 4.28.0) was utilized for containerization, and Rust (version 1.76) was employed for compiling the programs.

The prototype system's Mediator was implemented in Rust, and the build was conducted using Rust's cargo build system in release mode. The resulting binary for the Mediator node was approximately 3.3 MB in size, with the code segment occupying around 3.25 MB, the data segment around 111 KB, and the BSS segment 713 bytes. The system relies on standard libraries such as libssl.so.1.1 and libcrypto.so.1.1 for SSL/TLS communication, as well as libc.so.6, libm.so.6, and libpthread.so.0 for standard C operations, mathematical computations, and threading support. These widely available libraries ensure compatibility across general- purpose Linux environments. Furthermore, the system is designed to run within Docker containers, ensuring portability and ease of deployment across various environments.

5.2.2. Evaluation of CPU and Memory Usage

The CPU and memory usage of the Mediator nodes were continuously monitored during system operations using a script that executed the docker stats command every second in the background. This script created a new log file for each measurement, using the current timestamp as the file name, and logged the Mediator nodes' performance metrics at each interval. This approach enabled continuous tracking of resource consumption throughout the experiment.

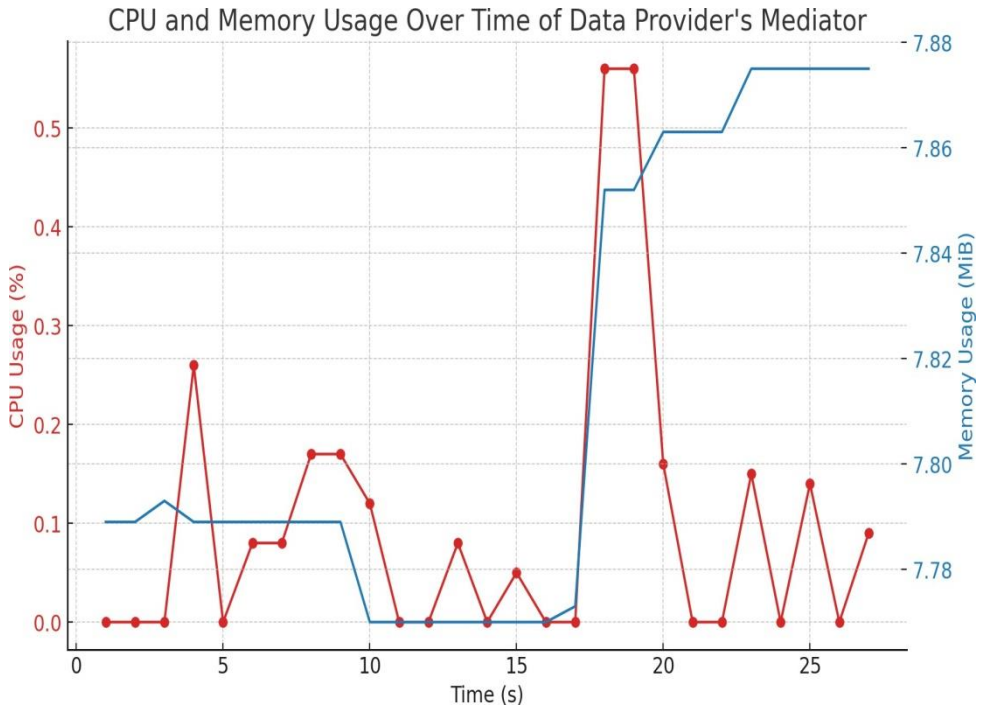


Figure 6. CPU and Memory Usage of Data Provider's Mediator

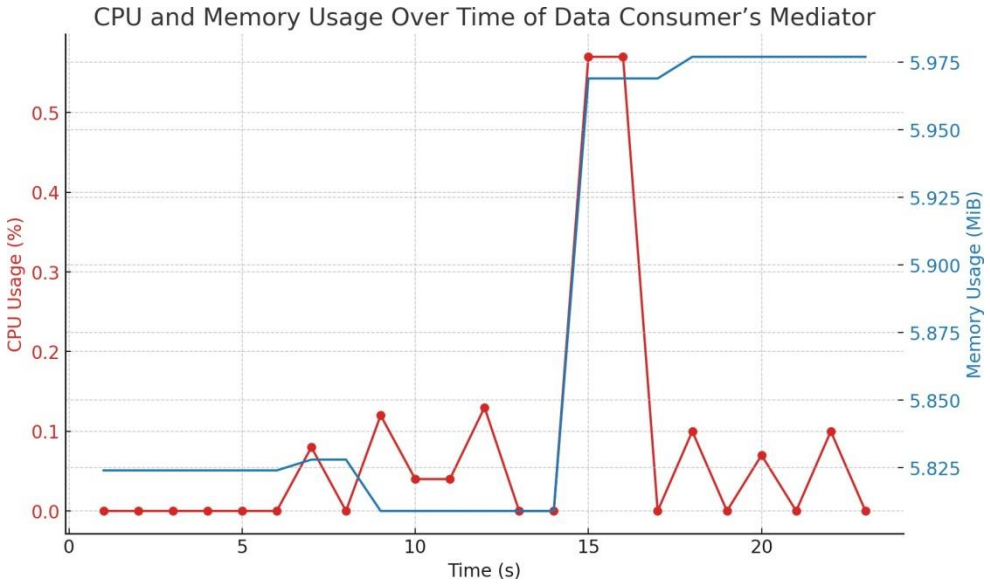


Figure 7. CPU and Memory Usage of Data Consumer's Mediator

Figures 6 and 7 illustrate the CPU and memory usage trends for the Data Provider's and Data Consumer's Mediator nodes, respectively. In these graphs, the x-axis represents the time elapsed since the Mediator nodes were started, while the red line indicates CPU usage and the blue line indicates memory usage. At certain points, the CPU usage shows a slight increase, which can be attributed to the initiation of a Data Request. This process involves several consecutive operations, including data upload, UUID encryption, and the transmission of the encrypted UUID to the blockchain. Similarly, a minor increase in memory usage is observed around the same time, likely due to internal caching mechanisms. Overall, both CPU and memory consumption remained minimal throughout the experiment, demonstrating the lightweight nature of the Mediator. This performance indicates its feasibility for deployment on IoT devices.

5.2.3. Costs for executing and deploying functions on the blockchain

To measure the gas costs associated with function execution and contract deployment on the blockchain, the Hardhat development environment was used in conjunction with the `hardhat-gas-reporter` plugin. This plugin allows for extended functionality in gas estimation, enabling the calculation of gas consumption during test code execution and smart contract deployment. Additionally, Coin Market Cap's API was used to retrieve the latest Ethereum price and average gas price, allowing for a USD conversion of the gas costs.

Table 4 presents the results of gas cost estimation obtained by running `npx hardhat test`. As of 14:22 UTC on August 20, 2024, the gas price was approximately 3 gwei, and the price of 1 ETH was $2636.05 USD. The gas consumption and corresponding costs were measured for both the deployment of the entire system and the execution of individual methods.

According to the table, the average gas cost incurred by the market owner for deploying the system was 3,467,190 gas, which is equivalent to $32.06 USD. Among the methods executed by system participants, the `deploy Merchandise` method was the most expensive, with an average gas consumption of 1,302,679 gas. This high cost is attributed to the fact that the method handles the deployment of a contract internally.

Table 4. Gas Reporter Output

Methods			
Contract	**Method**	**Average gas (gwei)**	**Average USD**
IoTMarket	deployMerchandise	1302679	10.30
Merchandise	emitUpload	29701	0.23
	purchase	118337	0.94
	verify	45075	0.36
	withdraw	34499	0.27
PubKey	registerKey	317934	2.51
Deployments			
Contract		**Average gas(gwei)**	**Average USD**
IoTMarket		2030357	16.06
Merchandise		1371714	10.85
PubKey		651119	5.15

In contrast, other methods such as `emitUpload` and `purchase` consumed significantly less gas, costing approximately $0.23 USD and $0.94 USD per execution, respectively. The `verify` method, which is responsible for ensuring data integrity, consumed an average of 45,075 gas, corresponding to a cost of $0.36 USD. Despite being a crucial part of maintaining the reliability of transactions, the cost of this method is relatively low.

These results demonstrate that while gas costs are inherent to blockchain operations, the overall costs associated with system deployment and function execution are reasonable, allowing the system to operate efficiently within the blockchain environment.

6. DISCUSSION

In this study, we proposed a decentralized data integration platform utilizing blockchain technology and developed a proof-of-concept (PoC) prototype for evaluation. The system employs data integrity verification by comparing hash values embedded in metadata and retries the process until accurate data is obtained. Furthermore, by leveraging smart contracts to automate the validation process and manage cryptocurrency withdrawals for Data Providers, the system minimizes the potential for malicious behavior from both Data Providers and Data Consumers, thus ensuring a reliable and secure transaction flow.

The experimental results demonstrate that the Mediator component is lightweight, confirming that it can operate efficiently on relatively high-performance IoT devices such as Raspberry Pi. Additionally, our analysis of gas costs associated with system operations revealed that the overall costs remain within a feasible range for practical use. Although deploying metadata incurs higher costs due to the nature of smart contract interactions, the system can still be operated at a reasonable expense.

However, the current implementation is limited to handling only active Data Requests, thereby constraining the system’s functionality. Furthermore, the evaluation did not include tests to assess the system's performance under high loads, where multiple Data Requests are made simultaneously. This leaves open the question of how the system handles scalability under such conditions. Additionally, the experiment was conducted within a virtualized network environment on a single PC. Future work will need to include tests conducted on real IoT devices in a more realistic operational environment to better assess the system’s viability in practice.

Regarding the metadata uploaded to the blockchain, future iterations of the system may require the embedding of richer data to enhance search ability and improve overall efficiency. However, this would likely increase deployment costs, and careful consideration must be given to the trade-off between the volume of data and the associated costs.

Looking ahead, future experiments will focus on simulating real-world scenarios and evaluating the system's performance under large-scale Data Request conditions. The insights gained from these tests will inform further improvements to the proposed system, ensuring its effectiveness and robustness in practical deployments.

7. CONCLUSION

This paper proposes a decentralized demand-supply matching system leveraging blockchain technology for the distribution of IoT device data. The research goes beyond the typical scope of data distribution platforms by systematically organizing the requirements for the entire matching system, including the users and IoT devices, and presenting a system concept designed to meet these requirements. To demonstrate the feasibility of the proposed concept, a proof-of-concept implementation was developed and evaluated.

The evaluation focused on three key aspects: the binary size of the Mediator program, its CPU and memory usage during operation, and the gas costs associated with blockchain function execution and deployment. The results showed that the system operates efficiently within practical limits, demonstrating that the proposed concept is viable for real-world deployment.

In addition, this study examined several prior works on decentralized data integration platforms. While many of these studies focus on privacy, data integrity, and confidentiality, few have placed sufficient emphasis on usability. By comparing these works with the proposed system, this paper highlights the unique contributions of this research, specifically its focus on user- centric design and comprehensive system requirements, setting it apart from existing studies.

For future work, we recognize that the metadata uploaded to the blockchain plays a crucial role in stimulating data distribution. We aim to explore more efficient ways to structure and manage metadata, focusing on improving both search ability and cost-effectiveness. By refining the design of metadata, we hope to further enhance the functionality and scalability of decentralized data platforms, making them even more accessible and practical for widespread use.

REFERENCES

[1] Jiang, Shubo & Hu, Shuang. (2023). Research on Smart Agriculture Big Data Platform Based on the Internet of Things. Proceedings of the 2023 4th International Conference on Big Data and Social Sciences (ICBDSS 2023), 365–374. 10.2991/978-94-6463-276-7_40.

[2] Kasrin, Nasr & Benabbas, Aboubakr & Elmamooz, Golnaz & Nicklas, Daniela & Steuer, Simon & Sünkel, Michael. (2021). Data-sharing markets for integrating IoT data processing functionalities. CCF Transactions on Pervasive Computing and Interaction. 3. 10.1007/s42486- 020-00054-y.

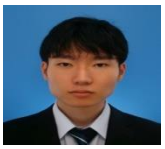
[3] Ul Hassan, Muneeb & Rehmani, Mubashir Husain & Chen, Jinjun. (2019). Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions. Future Generation Computer Systems. 97. 10.1016/j.future.2019.02.060.

[4] Gugueoth. Phd, Vinay & Safavat, Sunitha & Shetty, Sachin & Rawat, Danda B. (2023). A review of IoT security and privacy using decentralized blockchain techniques. Computer Science Review. 50. 100585. 10.1016/j.cosrev.2023.100585.

[5] Shrimali, Bela & B.Patel, Hiren. (2021). Blockchain State-of-the-Art: Architecture, Use Cases, Consensus, Challenges and Opportunities. Journal of King Saud University - Computer and Information Sciences. 34. 10.1016/j.jksuci.2021.08.005.
[6] Buterin, V. (2013). *Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform*.
[7] Jain, Akshita & Arora, Sherif & Shukla, Yashashwita & Patil, T & Sawant-Patil, S & Patil, Tulshihar. (2022). Proof of Stake with Casper the Friendly Finality Gadget Protocol for Fair Validation Consensus in Ethereum.
[8] Wu, Kaidong & Ma, Yun & Huang, Gang & Liu, Xuanzhe. (2019). A first look at blockchain- based decentralized applications. Software: Practice and Experience. 51. 10.1002/spe.2751.
[9] Stamatakis, Dimitrios & Kogias, Dimitrios & Papadopoulos, Pericles & Karkazis, Panagiotis & Leligou, Helen. (2024). Blockchain-Powered Gaming: Bridging Entertainment with Serious Game Objectives. Computers. 13. 14. 10.3390/computers13010014.
[10] Chen, Yan & Bellavitis, Cristiano. (2019). Decentralized Finance: Blockchain Technology and the Quest for an Open Financial System. SSRN Electronic Journal. 10.2139/ssrn.3418557.
[11] Wohrer, Maximilian & Zdun, Uwe. (2018). Smart contracts: security patterns in the ethereum ecosystem and solidity. 2-8. 10.1109/IWBOSE.2018.8327565.
[12] Zheng, Jiawei & Dong, Xuewen & Zhang, Tao & Chen, Junfeng & Tong, Wei & Yang, Xiaozhou. (2018). MicrothingsChain: Edge Computing and Decentralized IoT Architecture Based on Blockchain for Cross-Domain Data Shareing. 350-355. 10.1109/NANA.2018.8648780.
[13] V. Koutsos, D. Papadopoulos, D. Chatzopoulos, S. Tarkoma and P. Hui, "Agora: A Privacy- Aware Data Marketplace," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 6, pp. 3728-3740, 1 Nov.-Dec. 2022, doi: 10.1109/TDSC.2021.3105099.
[14] Alsharif, Ahmad & Nabil, Mahmoud. (2020). A Blockchain-based Medical Data Marketplace with Trustless Fair Exchange and Access Control. 1-6. 10.1109/GLOBECOM42002.2020.9348192.
[15] Liu, Hui & Tai, WeiPeng & Wang, Yaofei & Wang, Shenling. (2022). A blockchain-based spatial data trading framework. EURASIP Journal on Wireless Communications and Networking. 2022. 10.1186/s13638-022-02130-6.
[16] Nguyen, Bao & Lydia, Laxmi & Elhoseny, Mohamed & Pustokhina, Irina & Pustokhin, Denis & Selim, Mahmoud & Nhu, Nguyen & Shankar, Kripa. (2020). Privacy Preserving Blockchain Technique to Achieve Secure and Reliable Sharing of IoT Data. Computers, Materials & Continua. 65. 87-107. 10.32604/cmc.2020.011599.
[17] Conoscenti, Marco & Vetro, Antonio & De Martin, Juan Carlos. (2017). Peer to Peer for Privacy and Decentralization in the Internet of Things. 288-290. 10.1109/ICSE-C.2017.60.
[18] Chanson, Mathieu & Bogner, Andreas & Bilgeri, Dominik & Fleisch, Elgar & Wortmann, Felix. (2019). Blockchain for the IoT: Privacy-Preserving Protection of Sensor Data. Journal of the Association for Information Systems. 10.3929/ethz-b-000331556.

AUTHORS

Kenta Kawai is currently in the final year of his master's program at the Graduate School of Informatics, Nagoya University, where he also completed his bachelor's degree. His research interests include the integration of IoT devices with blockchain technology, focusing on innovative solutions for secure and decentralized data distribution.



Yuxiao Wu is currently doing his doctoral course in Computing and Software Systems at Nagoya University, Aichi, Japan. He received the B.S. degree in information security from Nanjing University of Post and Telecommunication, Nanjing, China, and the M.S. degree in information science from Nara Institute of Science and Technology, Nara, Japan in 2021, and 2024, respectively. His research interests include privacy preservation in blockchain-based applications using zero- knowledge proof and smart contract-based e-voting systems.

Yutaka Matsubara is an Associate Professor at the Graduate School of Informatics, Nagoya University. He received his Ph.D. degree in Information Science from Nagoya University in 2011. From 2009 to 2018, he was a Researcher, and then an Assistant Professor at the Center of Embedded Computing Systems(NCES), Nagoya university. His research interests include real-time operating systems, real-time scheduling theory, and system safety and security for embedded systems. He is a member of IEEE, IPSJ, IEICE and JSAE.

Hiroaki Takada is a professor at Institutes of Innovation for Future Society, Nagoya University. He is also a professor and the Executive Director of the Center for Embedded Computing Systems (NCES), the Graduate School of Informatics, Nagoya University. He received his Ph.D. degree in Information Science from University of Tokyo in 1996. He was a Research Associate at University of Tokyo from 1989 to 1997, and was a Lecturer and then an Associate Professor at Toyohashi University of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and automotive embedded system. He is a fellow of IPSJ and JSSST, and is a member of IEEE, ACM, IEICE, and JSAE.