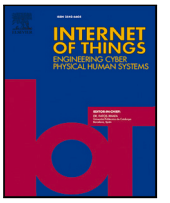
Internet of Things 24 (2023) 100959

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

Industrial data monetization: A blockchain-based industrial IoT data trading system

Mahmoud Abbasi [a,*], Javier Prieto [a], Amin Shahraki [d], Juan M. Corchado [a,b,c]

[a] BISITE Research Group, University of Salamanca, Edificio Multiusos I+D+i, Calle Espejo s/n, 37007 Salamanca, Spain
[b] AIR Institute, IoT Digital Innovation Hub, Salamanca, Spain
[c] Department of Electronics, Information and Communication, Faculty of Engineering, Osaka Institute of Technology, Osaka 535-8585, Japan
[d] Department of Informatics, University of Oslo, Oslo, Norway

ARTICLE INFO

Keywords:
Blockchain
IPFS
Ethereum
Smart contracts

ABSTRACT

The Internet of Things (IoT) devices utilized in manufacturing industries produce vast volumes of valuable data that can revolutionize productivity and sustainability. While existing centralized data marketplaces facilitate data trading, they are often marred by trust issues, a single point of failure, and significant security and privacy vulnerabilities. Addressing these critical shortcomings, this paper introduces a blockchain-based industrial data trading system that not only ensures secure and transparent data trading but also offers advantages over conventional systems. Unlike traditional marketplaces, our proposed system emphasizes trustworthiness by mitigating third-party risks, enhancing data integrity through decentralized storage, and employing graph technology for efficient blockchain querying. Further, with integrated access control, our system elevates security standards. Preliminary evaluations reveal the system's potential to offer a more secure, transparent, auditable, and trustworthy data trading environment, distinctly outpacing the capabilities of current marketplaces.

1. Introduction

We are witnessing a significant evolution in the Industrial Internet of Things (IIoT) with the emergence and development of smart physical devices [1]. These devices in industrial environments collect an extensive and heterogeneous amount of data. The collected data can be utilized to make optimal decisions, troubleshoot, optimize parameter configuration, find performance bottlenecks, and detect anomalous behavior [2].

Despite these promising applications, industrial data could lead to several serious technical, technological, and security challenges, and it takes work to address them [3]. One of the first challenges facing IIoT is that the majority of data amassed from IIoT is unstructured, rendering it challenging to store and manage using conventional database systems. This poses concerns in terms of data retrieval, analytics, and integration with other systems. Moreover, the sheer volume of data can overwhelm IIoT devices, especially those with limited storage and computational capabilities. This is particularly concerning given the real-time demands of many industrial processes. In addition, with industries adopting varying data management techniques, there lack of a unified data management process. This disparity can cause compatibility issues, hampering data sharing and analytics across different platforms or sectors. Finally, for some industries, deriving tangible profit from the data remains a quandary. In contrast, others grapple with sourcing precise and actionable IIoT data for their specific needs [4].

* Corresponding author.
E-mail addresses: mahmoudabbasi@usal.es (M. Abbasi), javierp@usal.es (J. Prieto), am.shahraki@ieee.org (A. Shahraki), corchado@usal.es (J.M. Corchado).

https://doi.org/10.1016/j.iot.2023.100959
Received 10 April 2023; Received in revised form 26 September 2023; Accepted 27 September 2023
Available online 4 October 2023

Addressing the challenge of monetization and utility, the idea of an IIoT data marketplace emerges as a promising solution [5]. Such a marketplace can unlock the vast potential of industrial data, facilitating its optimal use to spawn new applications and amplify the value derived from data collections. In today's landscape, the systematic collection and access to IIoT data stand as critical differentiators for numerous industries, given the transformative potential of this data.

Nowadays, industrial IoT data marketplaces mainly depend on a centralized third party that acts as a mediator for the data exchange between the data provider and buyer, such as Xignite,[1] Factual,[2] BIG IoT,[3] and Windows Azure Data Marketplace.[4] These centralized solutions provide high storage capacity and computing power and facilitate organizations' use of resources and data sharing anytime and anywhere.

Despite the efficiency, there need to be more trust guarantees in the centralized data trading/sharing systems. In other words, particular security and privacy issues and concerns exist when storing and trading data through centralized third parties [6]. For example, IIoT data stored in a cloud service provider is vulnerable to data leakage. The consequences of data leakage are severe due to the high sensitivity of IIoT data. Moreover, industries depend on third-party cloud storage services to store a huge volume of data, which may run the risk of a single point of failure (SPOF). In addition, industries that want to join the demanding IIoT data marketplace should find an answer to their central question about the data sovereignty of traded data [7]. Last but not least, data transfer cost is another issue with centralized IIoT data trading. Due to intermediary fees, it is still expensive for industrial businesses to transfer an extensive volume of fine and coarse industrial IoT data in a real-time manner.

A solution to the problem of the centralized approach is to adopt an advanced decentralized IIoT data management and trading approach [8]. An example of the decentralized IoT marketplace is IDMoB [9]. A carefully designed decentralized marketplace for IIoT data is secure, and transparent and ensures excellent data integrity and immutability to establish trust for trading industrial data. Distributed ledger technology (DLT), also known as blockchain technology, particularly smart contracts (SCs), such as those that execute on the Ethereum blockchain, can satisfy most of these requirements [10]. These decentralized technologies can promote trustworthy IIoT data exchanges between data providers and buyers, transparent data usage auditing, and better traceability while conforming to general data protection regulation (GDPR) legislation. Blockchain technologies offer outstanding robustness against SPOF due to their decentralized peer-to-peer (P2P) nature. Thanks to applying cryptography techniques, blockchain systems can also preserve data privacy. Note that one needs to recognize the distinction between data integrity, which blockchain inherently offers, and data privacy, which requires additional considerations, especially in public blockchains. More specifically, to ensure data privacy on public blockchains, the following steps can be taken: (1) before writing data to the blockchain, it is encrypted using cryptographic algorithms, ensuring only parties with the appropriate decryption key can access the actual data, (2) another strategy is to store sensitive or large data off-chain, while only storing references or hashes of this data on the blockchain, (3) some blockchain platforms, like Hyperledger Fabric, allow for private channels where transactions are only visible to participants of the channel, and (4) while not a solution for public blockchains, some industries might opt for permissioned or consortium blockchains, where network participation and data visibility are limited to a select group, inherently offering more data privacy.

Across the growing diversity of data marketplaces and data sharing systems, especially IoT data, there is an urgent need to implement a decentralized IIoT data marketplace based on blockchain technology. Current IoT data sharing/exchange platforms based on blockchain put significant trust in a single mediatory agent, such as a broker, who provides all services for market security and trustworthiness (e.g., [11,12]). Another group of studies mainly focuses on access control policy to handle the shared data that can be accessed by the other participants (e.g., [13,14]). Moreover, even works that provide solutions for blockchain-based decentralized IoT data trading usually do not consider all the essential elements, such as metadata query by the buyer, data rating, and data pricing model (e.g., [15,16]).

In response, this paper presents a blockchain-based decentralized data marketplace that is secure, privacy-preserving, transparent, and provides a high degree of integrity and immutability to establish trust for trading IIoT data. Cryptographically signed transactions and the data provenance delivered by the Ethereum blockchain guarantee non-repudiation and accountability in the proposed method [17]. In addition, we utilize InterPlanetary File System (IPFS), an off-chain decentralized storage technology, to store and retrieve big IIoT data securely. Moreover, the proposed data marketplace's functionalities have been developed using Ethereum SCs that provide programmable logic [18]. Our proposed data marketplace uses graph[5] technology as an indexing protocol for querying the Ethereum blockchain network. Furthermore, we employ role-based access control (RBAC) as an identity management layer that provides the ability to enforce some restrictions. The key contributions of our work can be summarized as follows:

- We offer a blockchain-based decentralized IIoT data marketplace that enables companies to trade IIoT data securely. Our platform guarantees transparency, traceability, auditability, and trustworthiness.
- We enhance our data marketplace by pairing it with off-chain decentralized storage, addressing the vast amounts of data typical in industrial settings.
- We have crafted and implemented an Ethereum smart contract that offers a range of features, including registering or deleting IIoT devices, adding or removing data users, orchestrating data auctions, and placing bids.

[1] https://www.xignite.com/
[2] https://www.factual.com/
[3] https://iot-epi.eu/project/big-iot/
[4] https://datamarket.azure.com/
[5] https://thegraph.com/en/

- Our proposed IIoT data marketplace introduces an RBAC mechanism that clearly delineates permissions for various actions. Additionally, by employing graph technology as an indexing protocol, the system simplifies and streamlines data querying from blockchains.
- We have carried out simulations and assessments of our proposed trading system, illustrating its applicability in scenarios demanding core attributes like security, privacy, and cost-effectiveness. The code for this system is openly accessible on GitHub[6]

The remainder of the paper is structured as follows. Section 2 provides the necessary background to our work and its motivations. Section 3 describes the details related to the design of the proposed blockchain-based data marketplace. The implementation details of our solution are discussed in Section 3.3. Section 4 presents the simulation details and analysis of the data trading system. Finally, Section 7 concludes this study.

2. Background and motivation: industrial data marketplace

A data marketplace would act as a virtual trading system where the data providers (data producers, owner, or sellers) can sell their data, and data users (buyers or customers) can buy it [19]. In recent years, data marketplaces have gained considerable traction, driven by the increased recognition of the potential value locked in data. Notably, initiatives like the Ocean Protocol have emerged as front-runners in the domain of decentralized data marketplaces, offering frameworks that allow data to be shared without compromising the security or privacy of the information.[7] Ocean Protocol is a decentralized data exchange protocol that enables data providers to share data without losing control over it while allowing data consumers to access data without compromising privacy or security. The project is designed to unlock data for AI and broader use, addressing the issue of data silos and data hoarding. Ocean Protocol uses a tokenized service layer to ensure that data providers are compensated for their data. This means that providers can monetize their data without actually having to sell it outright. Furthermore, decentralized data hubs like Gaia-x represent another significant stride in the evolution of data exchange platforms.[8] Gaia-X is a European initiative aiming to develop a federated data infrastructure. The main goal of Gaia-X is to promote a transparent and user-centric digital ecosystem where data and services can be made available, collated, and shared in an environment of trust.

IIoT data marketplaces based on blockchain technology have been implemented. The IoTA platform is one of the most famous examples that was launched at the end of 2017 [20]. The main objective of IoTA is to break industrial data silos and make it easy to sell and buy data among different entities. To this end, IoTA uses masked authenticated messaging (MAM) channels as a data communication protocol to allow data users to pay for the provided data through micropayments of IOTA tokens. dClimate is another example of a blockchain-based IIoT data marketplace [21]. dClimate is the first decentralized platform for climate-related models, forecasts, and data. This platform allows the participants to sell different forecasted data/models for various climatic variables. dClimate includes four essential layers that realize a decentralized system to search, deliver, store, and monetize high-quality climate data. These layers include: (1) governance, (2) oracle, (3) blockchain and data storage, and (4) marketplace. There are other examples of industrial data marketplaces, such as LemoChain,[9] the DX Network,[10] Longenesis,[11] and ElectriCChain.[12]

Establishing an IIoT data marketplace can provide all the players in the system with considerable advantages, including technical, economic, and user-facing benefits [9]. More specifically, different participants can enjoy the benefits of an IIoT data marketplace in *four* different ways: (1) artificial intelligence (AI)/machine learning(ML) service providers are able to access a massive amount of pre-collected data that they were unable to access before; (2) data users (buyers) are able to use the provided data to build novel services and products; (3) industries, AI/ML service providers, and other beneficiaries obtain economic benefits in direct and indirect ways; (4) industries do not have to run and maintain cloud-based services for their data as the data marketplace delivers essential data analytics services over blockchain.

As previously mentioned, an IIoT data marketplace can be exploited for novel applications/services and added economic value. In the following, we present a list of potential data monetization pathways in industrial IoT.

- ***Plant optimization***: Data related to unexpected events, such as machine failures or malfunctions, are rare. Nevertheless, ML and deep learning (DL) models require enormous amounts of training data about these events in order to make accurate predictions. Having the same data from different equipment and processes improves predictive maintenance through learning-based methods, raising equipment lifespan and product quality and preventing equipment failure. A potential solution is buying industrial data or even pre-trained models from other providers with the same or similar data/tasks through external monetization.
- ***Monitoring and controlling processes***: Sharing data related to products and operations, e.g., the location of products or the manufacturing process's status, can help to eliminate defects and immediately respond to disruptions and unusual variability in supply and value chains, such as poor-quality products. A primary example is end-to-end visibility in supply chain monitoring.

[6] https://github.com/Mahmoud-Abbasi-svg/Industrial-data-monetization
[7] https://oceanprotocol.com/
[8] https://gaia-x.eu/
[9] https://www.lemochain.com/
[10] https://dx.network/
[11] https://longenesis.com/
[12] http://www.electricchain.org/

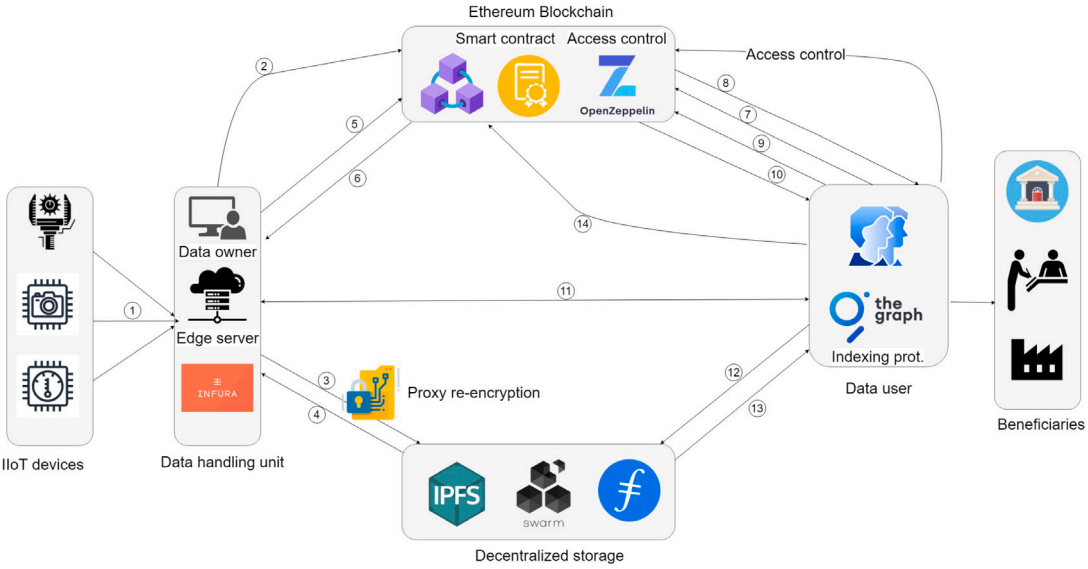


Fig. 1. System diagram of the blockchain-based IIoT data trading.

This process starts with procuring materials from suppliers and ends when the customer receives the final product. In this case, direct data exchange (or sharing) and external data monetization (or trading) are necessary to guarantee the success of the process. More specifically, data exchange between involved parties through external and internal data monetization helps suppliers and producers optimize their operations.

- ***Provenance verification***: Many clients are increasingly focusing on verifying the provenance of selected products. If the manufacturers could obtain data through internal or external data marketplace/sharing platforms which describe the product life cycle, they could provide clear evidence to their customers when needed.

To conclude the above discussion, IIoT data marketplaces are influential because they motivate industries to earn income by trading the internal industrial data they produce through regular operations and processes. Indeed, the IIoT data marketplaces benefit both sellers and buyers since they allow fair data exchange, enabling a low-cost, transparent form of data trading between interested participants.

3. System design

This section describes the design of our decentralized blockchain-based system for IIoT data trading. Fig. 1 shows the diagram and the components of the system. In our design, the IIoT devices interact indirectly with the smart contract via the data owner and a data handling unit. The interested entities can also access on-chain information, listen to event logs, or interact with the smart contract. In the proposed solution, the decentralized storage system can be used to upload big industrial data. In the remaining parts of this section, we describe each system component in more detail.

3.1. System components

As is clear from Fig. 1, our proposed system consists of different components and entities, including IIoT devices, data handling unit, data owner, Ethereum blockchain, smart contract, decentralized storage, access control module, data user, indexing protocol, encryption scheme, and beneficiaries.

3.1.1. IIoT devices

IIoT devices are one of the system's critical components because they collect the data that are presented to potential trading participants. Such devices may include, but are not limited to, the following:

- Environmental sensors: measure parameters like temperature, humidity, air quality, and pressure. Their high precision makes them invaluable in settings like manufacturing, where even a small environmental fluctuation can impact the product quality.
- Connected industrial machines: these are machinery equipped with sensors and connectivity modules. They can provide real-time diagnostics, monitor wear and tear, and sometimes adjust their operations based on the data they gather.
- Production line monitoring systems: track the progress of goods on a production line, ensuring that delays, defects, or deviations are immediately identified and addressed.

- Cameras: utilized for security, quality control, and even predictive maintenance. Advanced cameras, coupled with AI algorithms, can spot defects in products or predict when a machine is likely to fail based on visual cues.

These devices may also be able to handle and carry out remote commands, preliminary data processing, and data transmission. This allows system operators or automated systems to adjust settings, initiate processes, or even shut down a device if necessary. Preliminary data processing at the source can help in data reduction, filtering out noise, and sending only pertinent data chunks for further analysis. These devices are usually resource-restricted. More specifically, given their design for specific tasks, they often lack the computational power and storage capabilities of more generalized computing systems. This constraint is why they typically transmit their sensed data to the data handling unit that can process, analyze, and store vast amounts of data.

Due to its volume and granularity, the data captured by the IIoT devices are uploaded to the off-chain decentralized storage. This ensures efficient data access and reduces the bloat of the blockchain. However, crucially, metadata (like device ID, timestamp of data capture) and hashes of this data are stored on-chain. This on-chain record ensures data integrity and provenance, allowing potential trading participants to verify the authenticity of the data they access.

These industrial IoT devices can communicate with the data handling unit via various communication technologies depending on their design and operational environment:

- Wired protocols: such as Ethernet, Modbus, and Profinet. These are often preferred in stable environments where devices remain stationary and require high data transfer rates.
- Wireless protocols: including Wi-Fi, Zigbee, LoRaWAN, and NB-IoT. These offer flexibility and are ideal for devices spread across large areas or in hard-to-reach locations.

Using the data handling unit in our solution is a solution to fill the gap between the limited capabilities of these devices and the programmable logic of the Ethereum smart contract.

3.1.2. Data handling unit

The data handling unit acts as an intermediary, bridging the fundamental disparity between the raw data streams from IIoT devices and the sophisticated processing logic inherent to blockchain systems like Ethereum. With IIoT devices being limited in their computational and storage abilities, the data handling unit ensures seamless data flow and processing, enabling optimal system operation. The handling data unit consists of other entities, including data owner, edge server, and blockchain node. As you will see, the data owner holds ownership and control over the data emanating from the IIoT devices. Regarding the edge server, if often located physically closer to the IIoT devices, the edge server facilitates quicker processing times and reduces latency. Its primary role is to provide computational resources closer to the data source, enabling real-time analytics, preliminary data processing, and acting as a buffer before the data reaches the blockchain or centralized servers. In addition, the blockchain node is critical to the functioning of any blockchain-based solution, a node ensures that the system remains decentralized, transparent, and secure. It can communicate with the blockchain, verifying transactions, and storing blocks. By running an Ethereum node, the DHU ensures the data's integrity and security. For ease of deployment and consistent access to the Ethereum blockchain, our system leverages Infura,[13] a renowned infrastructure provider. In addition, the DHU, specifically the data owner, is in charge of transferring the IIoT data to the decentralized storage. In other words, the DHU, led by the data owner, moves IIoT data into decentralized storage solutions, specifically IPFS. IPFS is chosen for its resilience, decentralized nature, and capability to handle vast amounts of data. But before any upload, the data is subjected to rigorous preprocessing, ensuring that any anomalies, missing values, or inconsistencies are addressed, guaranteeing the recipients receive only top-tier data. Finally, a unique feature of our system is its ability to let the data buyer rate the received data. This not only instills trust in the system but also acts as a feedback loop for the data owner. Repeated low ratings can prompt an internal review to pinpoint issues, ensuring continuous improvement in data quality and delivery.

3.1.3. Blockchain

Blockchain is a central component of the proposed data marketplace. We use Ethereum as an open-source blockchain platform. Unlike traditional blockchain platforms, Ethereum offers the distinctive feature of programmable logic through smart contracts. These self-executing contracts have terms of the agreement between buyer and seller being directly written into code lines. They are developed using languages like Solidity and then deployed onto the Ethereum blockchain. When certain conditions in these contracts are met, they autonomously perform predefined actions, such as transferring funds or validating a data trade. In our solution, every participant, be it a data owner, data user, or intermediary, possesses a unique Ethereum address. This address serves multiple purposes:

- Identity verification: ensures genuine interactions and filters out malicious or fake users.
- Transaction record: every transaction, whether it is a bid placement, payment processing, or even simple data access requests, is linked with this address and subsequently recorded on the Ethereum blockchain.
- Accountability: the indisputable and transparent nature of blockchain entries guarantees every action's traceability. Hence, any operation executed by a participant is forever etched on the blockchain, preventing any form of repudiation or deceit.

One of the prime features of our platform is the on-chain auction. Here's how it operates:

[13] https://www.infura.io/

- Initiation: a data owner, looking to sell a particular dataset, initiates an auction via the deployed smart contract. The auction specifies details such as starting bid, data description, and auction duration.
- Bidding: interested data users or buyers can place bids during the auction's active period. Each bid, backed by Ethereum's cryptocurrency (like Ether), gets registered on the blockchain.
- Closure: once the auction duration elapses, the smart contract autonomously determines the highest bidder and transfers the dataset's access rights to them.
- Feedback: post data access, users can rate the quality and relevance of the dataset. These ratings, also stored on the blockchain, provide transparent feedback and aid in the platform's credibility.

3.1.4. Data owner

In the trading system, we assume there is a set of data owners, also called data producers, providers, or sellers, with one or more sets of industrial data. They wield valuable sets of industrial data, insights culled from an array of sensors, machines, and other IIoT devices. As the primary sources of data, they play a pivotal role in populating the marketplace and ensuring its vibrancy. Before making data available to potential buyers, the data owner should take two important steps:

- Preprocessing: the data owner undertakes crucial preprocessing activities. This includes cleaning the data, addressing missing values, normalizing if necessary, and ensuring the data conforms to certain quality benchmarks. Ensuring the data's accuracy and integrity is paramount, as it directly impacts the data's value on the marketplace.
- Encryption: considering the sensitive nature of industrial data, security cannot be overlooked. Thus, before the data makes its way to the IPFS, it undergoes robust encryption. This not only preserves data confidentiality but also ensures that only authorized buyers can decrypt and access the data after purchase.

Moreover, the data owner manages the requests by interacting with the Ethereum smart contract and doing the following tasks:

- Initiates data listings: through the smart contract, a data owner can list their data sets, specifying key details like dataset description, price, access conditions, and duration of availability.
- Manages buyer requests: when a potential buyer expresses interest, the smart contract notifies the data owner, allowing them to review the request, verify the credentials of the buyer if necessary, and either accept or reject the proposal.

In addition, in the vast expanse of the data marketplace, not all data sets are equal. Some might have stringent access requirements due to their sensitivity. To cater to this, the data owner leverages an access control module, which lets them stipulate who can access the data, under what conditions, and for how long. Depending on the nature of the data, its intended audience, and the owner's preferences, various access control policies can be defined. This ensures a granular control over data dissemination.

3.1.5. Data user

Data users, commonly known as buyers, represent the demand side of the IIoT data marketplace. They seek specific datasets for various purposes — research, industry analysis, trend prediction, and more. Their participation ensures a dynamic and competitive auction environment, driving the data's value. Considering the proposed Ethereum-backed platform, when a data owner lists IIoT data for sale, it triggers an auction where data users can actively participate. To place a bid, a data user interacts with the smart contract, sending a transaction that specifies the amount they are willing to pay. Data users can track ongoing auctions, view highest bids, and adjust their bids accordingly to increase their chances of acquisition. In the proposed ecosystem, each data user has a unique Ethereum address which serves multiple purposes, including identity verification and access control. More specifically, this address acts as an identifier, ensuring that only genuine users participate in auctions. Furthermore, the Ethereum address plays a pivotal role in access control. Data owners can set policies specifying which addresses (or buyers) can interact with certain smart contract functions. This provides an added layer of security and customization. Post-auction, if a data user emerges as the highest bidder, they earn the right to the desired dataset. Indeed, the winning bidder receives a unique hash from the data owner. Using this hash, they can access and download the dataset from the decentralized IPFS server. After analyzing the data, users can rate the dataset based on its quality, relevance, and accuracy. This feedback mechanism ensures transparency and encourages data owners to maintain high data standards. Data users are driven by the need for accurate, timely, and specific IIoT datasets. Whether it is for academic research, industrial optimization, predictive modeling, or any other application, the quality and relevance of the data they acquire is paramount. Their active participation and feedback shape the marketplace, guiding data owners on the kinds of datasets that are in demand and their expected quality.

3.1.6. Smart contract

The smart contract is the backbone of the proposed IIoT data trading system. Deployed on the Ethereum blockchain, it encapsulates the predefined logic, rules, and interactions for data trading in a tamper-proof manner. This ensures all participants adhere to the defined rules, guaranteeing fairness, transparency, and security. The smart contract provides two main groups of functionalities: the data owner-related and the data user-related functionalities.

- Data owner-related functions

  Device registration: allows data owners to register new IIoT devices, ensuring they are recognized and authenticated within the system.

User onboarding: data owners can add new potential buyers or data users to the marketplace, thereby broadening the trading audience.

Data advertising: enables owners to announce the availability of new datasets, providing metadata like the data's origin, timestamp, and a brief description.

Auction initiation: data owners can kickstart a bidding process for a specific dataset, setting parameters like starting bid, auction duration, and any reserved prices.

- Data user-related functions

Bidding: users can propose an amount they are willing to pay for a particular dataset.

Payment processing: once an auction is finalized, the winning bidder can securely process payments through the smart contract.

Data retrieval: post-payment, users can access the data download function to retrieve the purchased dataset.

Feedback mechanism: after analyzing the acquired data, users have the option to rate its quality, thereby providing feedback to the owner and informing future potential buyers.

In addition, the smart contract is in charge of imposing access control policies. This means that certain functions, like device registration or auction initiation, are only accessible by authenticated data owners, only authenticated data users can bid, pay, and retrieve data, and no unauthorized tampering and keeps the integrity of the marketplace intact. Figs. 2 and 3 showcase the sequence diagrams, detailing the step-by-step interactions between the smart contract and the involved entities for each function. It gives a visual representation of how every transaction is processed, highlighting the flow of data and control. All interactions with the smart contract are recorded on the Ethereum blockchain, ensuring every transaction is traceable. This transparency fosters trust among participants, knowing that the system operates as promised without any hidden manipulations.

3.1.7. Decentralized storage

Industrial environments generate a prodigious volume of data. Traditional centralized storage solutions are neither efficient nor secure enough to meet the requirements of an IIoT data marketplace, especially considering the demands of scalability, accessibility, and decentralization.

Decentralized storage solutions, like IPFS, Swarm, and Filecoin, offer a new paradigm. They spread the data across a network of computers, ensuring redundancy, fault tolerance, and efficient access. These features make such solutions ideal for IIoT data trading systems, where data integrity, availability, and security are paramount. Among the various decentralized storage options, we selected the IPFS for our data marketplace system due to [22]:

- Content addressing capability: every piece of data stored on IPFS is associated with a unique hash, derived from the content itself. This ensures that even the slightest change in data will result in a different hash, providing an implicit guarantee of data integrity.
- High performance: IPFS's unique data retrieval mechanism, based on content rather than location, ensures quicker data access. Its peer-to-peer nature also ensures that data can be fetched from the nearest node, reducing latency.

IPFS accepts the data, storing it across its distributed network of nodes. Once the storage process concludes, IPFS generates a Content Identifier (CID) or a unique hash for the stored data, which acts as its address within the IPFS ecosystem. This CID is essential for any subsequent data retrieval operation.

Despite the advantages of decentralized storage, there are inherent security concerns since anyone with the CID can potentially retrieve the data. To counteract this, we apply robust encryption to the data before uploading it to IPFS. This ensures that even if someone manages to access the data, they will not be able to decipher its content without the correct decryption key. The encryption process also means that the integrity of the data is maintained. Any unauthorized alterations would be detectable.

3.1.8. Proxy re-encryption scheme

The IIoT data marketplace faces unique challenges in terms of data security and accessibility. As buyers need access to specific data, a simple encryption scheme might be cumbersome, especially when the data owner does not want to share the decryption key directly.

Traditional encryption systems have limitations when it comes to granular access control and sharing encrypted data. Proxy re-encryption is an advanced cryptographic method that enables secure data sharing without revealing private keys or re-encrypting data in a traditional manner [23]. Proxy re-encryption works on the principle of transforming ciphertexts. It changes an encryption under the data owner's public key (delegator) into another encryption under the data user's public key (delegatee). This transformation is achieved without ever decrypting the data in the process. In the context of our trading system:

- Delegator (data owner): holds the original encrypted data and has the authority to delegate access rights.
- Delegatee (data user): wishes to access the encrypted data, requiring a transformation of the encryption keys.

When a data user (delegatee) expresses interest in accessing a piece of encrypted IIoT data, the request is channeled through the proxy. The proxy subsequently communicates with the data owner, relaying the data access request. Upon receiving a request, the data owner evaluates it. If they agree to share the data with the data user, they empower the proxy to generate a new re-encryption key. This key allows the transformation of the original ciphertext (encrypted with the data owner's public key) into a new ciphertext that can be decrypted using the data user's private key.

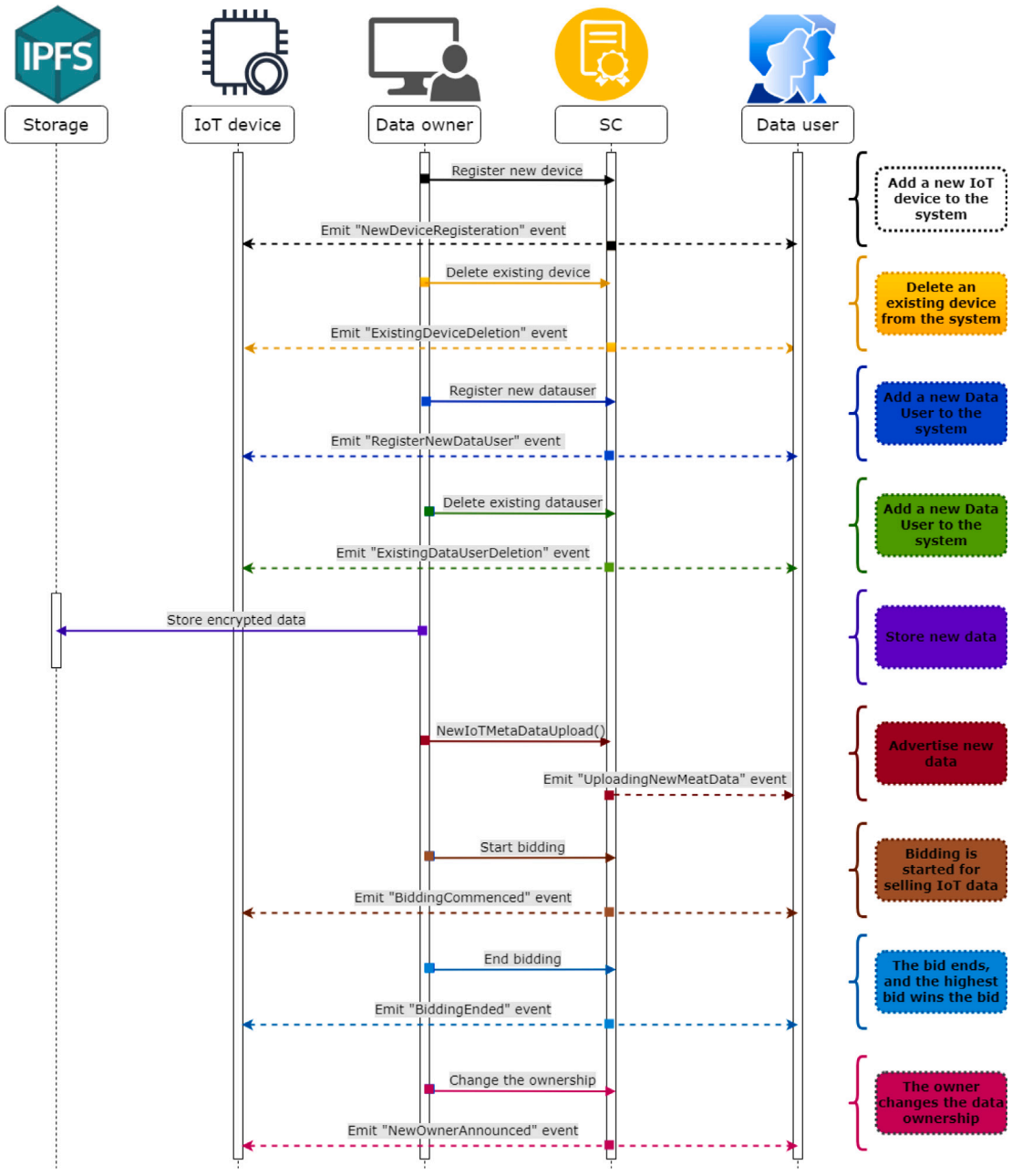


Fig. 2. The data owner sequence diagram.

It is vital to note that the proxy holds significant power in this model but is designed in a way that it never gets access to the actual decrypted data. Its role is strictly to facilitate the transformation of encryption, ensuring that access rights are preserved and unauthorized access is thwarted. Fig. 4 shows a simple description of the proxy re-encryption process and the interactions between the main actors. Using the proxy re-encryption scheme, our system ensures that unpaid or unauthorized users cannot access the encrypted data on IPFS. Furthermore, it simplifies the process of sharing encrypted IIoT data among multiple users without compromising security. Finally, this scheme minimizes computational overhead by avoiding the need to decrypt and then re-encrypt data for each new user.

3.1.9. Access control

In the realm of digital data systems, access control is a pivotal mechanism to determine who or what can view or utilize resources in a computing environment. Its importance in the IIoT data marketplace cannot be understated as it establishes the boundaries

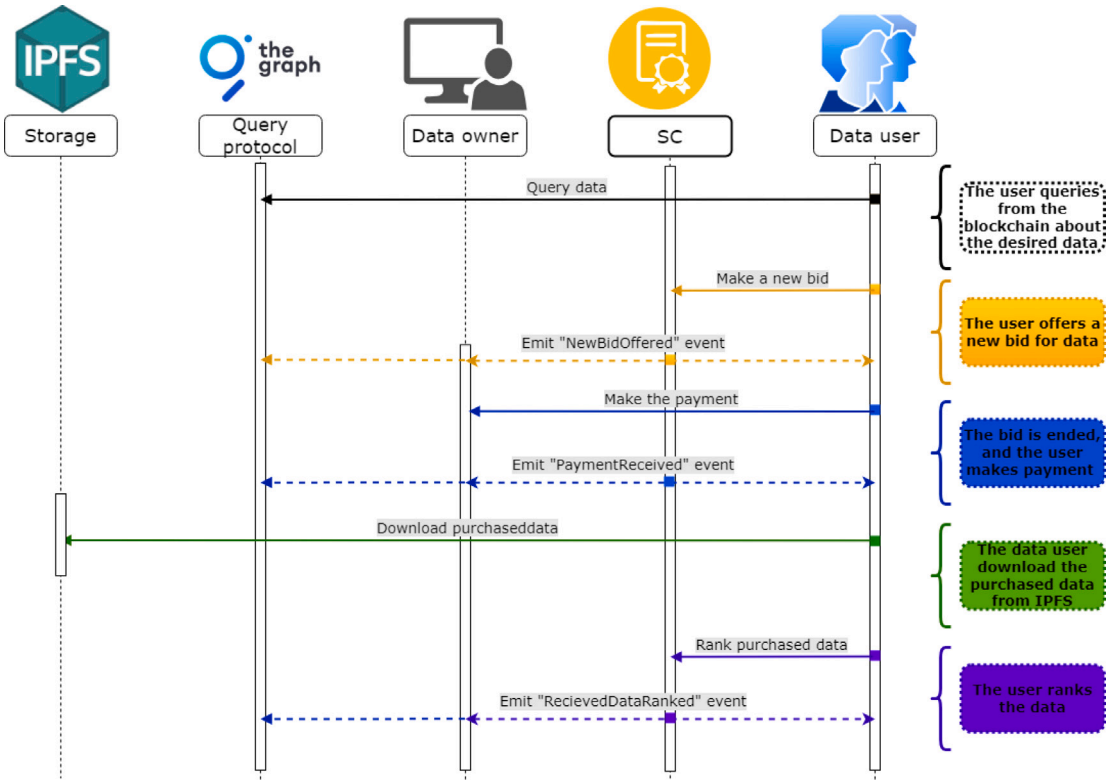


Fig. 3. The data user sequence diagram.

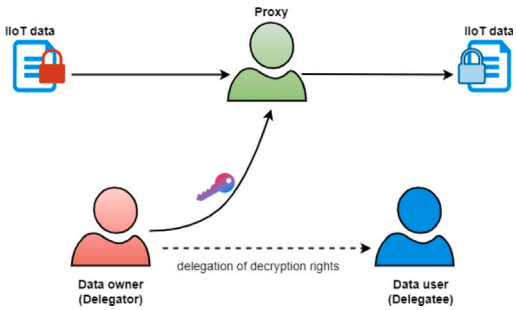


Fig. 4. The interactions in the proxy re-encryption technique.

between data owners, sellers, and users. Many different access control techniques have been presented in the literature to control users' access to information or physical facilities, such as attribute-based access control (ABAC) [24], rule-based access control (RBAC) [25], and RBAC [26]. The ABAC controls access based on attributes associated with users and resources. Decisions are typically based on a combination of attributes using a set of predefined policies. The RBAC is an access decision based on the execution of some predefined rules using attributes of users, resources, and environmental conditions, while the RBAC focuses on defining and assigning roles to users. Users gain access based on the roles assigned to them, thereby simplifying permissions management.

As mentioned in the Introduction section, our data trading system uses the role-based access control (RBAC) mechanism as an identity layer. This layer provides a variety of control capabilities. For example, by acting as an identity layer, RBAC provides a structured way to allocate permissions. Roles are defined based on the responsibilities within the system, enabling easier and more scalable permission management. In addition, using RBAC, the data owner can have granular control over who interacts with the provided data. For instance, they can place constraints on potential purchasers or they can validate the role of a potential buyer to confirm its legitimacy. And finally, the access control system is not just protective of the sellers. Buyers too can employ it to verify

the authenticity of a data seller, ensuring the credibility of the source. To convert theory into practical application, our system harnesses the power of the OpenZeppelin access control framework.[14] OpenZeppelin stands out for several reasons:

- Open-source: this ensures transparency and a large community backing, which means consistent updates and improvements.
- Tailored for RBAC: the framework is specially designed to support role-based access control, making it a suitable fit for our system.
- Flexibility: OpenZeppelin allows for the easy definition and management of roles, making the integration of access controls more streamlined.

Within the proposed system, two primary roles dominate the ecosystem: (1) ‘‘DATA USER’’, represents the clients or buyers in the data marketplace, actively looking to acquire IIoT data, and (2) ‘‘DATA SELLER’’, encompasses data owners, providers, or producers who introduce and offer data sets in the marketplace.

Through the integration of the RBAC mechanism via the OpenZeppelin framework, our data trading system enforces a robust and dynamic access control system, ensuring data integrity, security, and efficient role-based operations.

3.1.10. Indexing protocol

Decentralization, while providing numerous benefits in terms of transparency and security, poses unique challenges when it comes to the discovery and management of data. Within the vast landscape of a blockchain-based data trading system, the capability to swiftly locate and categorize data becomes vital for efficient operations. Data fragmentation and data discovery are two main challenges of decentralization. Regarding the former, as every transaction is a new entry on the blockchain, data is dispersed rather than being centralized in a single location. This causes data to be scattered throughout the blockchain. In the latter case, with the sheer volume of data entries, it is challenging for data users to identify the specific data sets they require, particularly when those sets are distributed across multiple blocks.

For a data marketplace to function efficiently, two primary objectives need to be met: (1) ease of data discovery: data users should be able to quickly discover and access the desired IIoT datasets, (2) metadata access: data users need to easily obtain an overview or metadata of the available datasets, aiding them in making informed decisions. Recognizing these requirements, our system adopts graph technology as its indexing backbone. The benefits are multifold:

- Structured representation: graphs naturally represent relationships, making them ideal for representing the connections between data owners, users, and the datasets themselves.
- Dynamic querying: graph technology, in conjunction with blockchain, offers dynamic querying capabilities. This means that data users can actively search and retrieve specific metadata associated with IIoT datasets.
- Reduction in redundancy: given that the blockchain inherently has data redundancy due to its distributed nature, an efficient graph-based indexing protocol can help in navigating through redundant data, ensuring users access the most relevant and recent data sets.
- Enhanced user experience: by providing a structured representation of data and relationships, combined with dynamic querying capabilities, users can enjoy a more intuitive and efficient data discovery process, reducing the learning curve often associated with decentralized platforms.

The Ethereum blockchain stores data in a series of interconnected blocks. By modeling these blocks and their contained transactions as a graph, our system can create an abstract representation of the entire blockchain. This representation allows for: (1) dynamic data retrieval: data users can utilize graph query languages (such as Cypher for Neo4j) to dynamically search and retrieve specific datasets or transaction records, and (2) real-time updates: as new blocks are added to the Ethereum blockchain, the graph database can be updated in real-time to reflect these changes.

One of the strengths of graph databases is their ability to traverse large sets of data efficiently. Traditional relational databases may require complex JOIN operations to gather related data, which can be time-consuming. In contrast, graph databases can quickly navigate relationships, making them more performant for interconnected data like that on a blockchain. Moreover, given the distributed and ever-growing nature of the blockchain, it is vital for the indexing protocol to be both maintainable and scalable. Graph technology can efficiently accommodate and process new data without significant performance degradation. Furthermore, advanced graph databases support clustering, ensuring high availability and horizontal scaling.

3.1.11. Beneficiaries

The holistic impact of IIoT data stretches beyond immediate stakeholders and is of vital importance to the broader technological ecosystem. One of the most notable beneficiaries of the IIoT data marketplace is the AI and ML service providers. AI and ML service providers have become an increasingly essential part of the industrial landscape due to their capability to unlock deeper insights from vast amounts of data. By integrating IIoT data into their frameworks, these providers can:

- Feature enhancement: enhance the features used in their machine learning models. A richer dataset offers more data points and variables for algorithms to analyze, thus making predictions and insights more accurate.

[14] https://docs.openzeppelin.com/contracts/2.x/access-control

- Algorithm training: improve the training of their algorithms. Real-world data from IIoT devices provides a comprehensive, practical dataset for training purposes. This, in turn, ensures models are well-calibrated to handle real-world scenarios.

Specialized companies that offer ML training datasets stand as another direct beneficiary. Their role is pivotal as they act as middlemen, curating and refining raw IIoT data. These firms conduct processes like data labeling, where raw data is tagged with meaningful labels that can be utilized by ML algorithms. For instance, temperature readings from an IIoT device in a factory might be labeled as 'normal', 'elevated', or 'critical'. Furthermore, they can use techniques like data augmentation to increase the volume of data, ensuring that ML models have a diverse set of data to learn from. This might include creating synthetic readings based on real-world data or manipulating existing data to reflect various scenarios.

Beyond the immediate use in AI and ML models, IIoT data also enables a slew of other solutions, including:

- Data-driven decisions: with ML algorithms that are trained on vast, diverse datasets, businesses can make decisions that are anchored in concrete data, enhancing efficiency and accuracy. For instance, by analyzing IIoT data, a company might determine optimal operating temperatures for machinery or identify patterns that suggest impending equipment failure.
- System control: integrating ML algorithms with IIoT data allows for more sophisticated automation. Systems can self-regulate based on the continuous flow of data, adjusting parameters in real-time to ensure optimal performance.
- Predictive maintenance: one of the standout applications is in predictive maintenance. Instead of following traditional schedules, maintenance can be done based on data-driven predictions. If IIoT sensors detect unusual vibrations or temperatures in machinery, ML algorithms can predict potential failures and trigger maintenance processes before a costly breakdown occurs.

3.2. IIoT data trading system

In our IIoT data trading system, the data owner first sets up the system and publishes the smart contract on the blockchain. The deployed contract acts as a trusted mediator between the data owner and the data user. To explain the different functionalities of the proposed IIoT data trading system, the corresponding description of each step is provided as follows (see Fig. 1):

(1) : IIoT devices, acting as the primary data sources, are strategically positioned across various industrial setups. These include connected sensors (e.g., temperature sensors) and actuators (e.g., opening a valve or adjusting machinery parts based on sensor data), machines (e.g., CNC machines, and conveyor belts), robots (e.g., welding, assembly, or material handling), monitoring areas (e.g., cameras or specific sensors), etc.

Once the data is collected, it follows some structured steps to reach the data handling unit, including analog to digital signals conversion and data aggregation. For the transmission, IIoT devices use various communication protocols depending on the setup and requirements. Common protocols include Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Open Platform Communications-Unified Architecture (OPC-UA). In addition, given the critical nature of industrial data, security is paramount. Encryption methods, secure data transmission techniques, and other cybersecurity measures are implemented at this stage to ensure data integrity and confidentiality.

(2) : Once transmitted, the data is received by the centralized data handling unit, often a high-capacity server or a cluster of servers. This unit is able to do different operations such as data buffering, initial data processing operations such as normalization, cleaning (removing anomalies or outliers), and segmentation (dividing data into meaningful chunks), preparing it for further in-depth analysis or storage. Moreover, the data handling unit serves as a bridge between the traditional IIoT setup and the blockchain. Indeed, in the data handling unit, the data owner uses the Infura API to gain access to the Ethereum blockchain and deploy the smart contract. Infura provides a scalable, reliable, and secure gateway to Ethereum networks, allowing users to run a node without the inherent overhead of directly maintaining one. This makes it a popular choice for applications needing reliable access to the Ethereum blockchain. Before any interaction with the Ethereum blockchain, the data handling unit establishes a session using the Infura API. This involves authenticating the data handling unit and obtaining tokens or session keys for secure communication. The data handling unit, in tandem with Infura, keeps a track of all blockchain interactions, ensuring traceability and aiding in potential troubleshooting.

(3) : Given the wide variety of IIoT devices and the volume of data they generate, the data owner performs vital preprocessing steps on the data as we mentioned. This step ensures accuracy and consistency of data. After the preprocessing, the proxy re-encryption scheme encrypts the data before uploading them to the IPFS network. Proxy re-encryption offers a way to re-encrypt data, enabling the data owner to securely share their encrypted data without revealing their private keys. Before any encryption occurs, the data owner sets up the re-encryption scheme by choosing a cryptographic key pair (public and private keys). The preprocessed IIoT data is encrypted using the data owner's private key, ensuring that only someone with the corresponding public key (or a delegated key through the proxy re-encryption process) can decrypt it. If the data owner decides to share data with another user, they generate a re-encryption key. This key allows a proxy to transform the ciphertext meant for the data owner into one that the delegatee (data user) can decrypt using their private key, without ever having the data decrypted in the intermediate stage. As mentioned, once the data is encrypted, the data owner prepares to upload it to IPFS.

(4) : The IPFS network returns the CID or the hash of the uploaded data to the data owner. The CID is a unique label used in IPFS to identify a piece of content stored within its network. The CID system ensures immutability. If the content were to change in any way, the CID would also change. This is especially important in the data trading system, guaranteeing the data's integrity and ensuring that the content retrieved corresponds precisely to what was originally uploaded. Once the IIoT data is encrypted and ready for sharing, the data owner interacts with IPFS using its API. During this interaction, the encrypted data is split into blocks, and

each block is hashed. The hashes then form a Merkle DAG structure, with the root node being the CID. After successful uploading, IPFS provides the data owner with the CID instantly. This acts as a receipt of successful data upload and will be the main point of reference for any future data access or sharing.

(5) : The data owner pushes the CID with other metadata (e.g., a description of the data and timestamp) on the blockchain (on-chain) for subsequent query and access by the potential buyers. This can be done by making a blockchain transaction. In other words, in the blockchain, transactions represent a change of state. When the data owner includes the CID and its metadata, a new transaction is created, and upon being confirmed, it is added to a block. Once this transaction is added to the blockchain, the information becomes immutable. This means the CID and associated metadata cannot be altered, ensuring that any future access or reference remains consistent. While the actual IIoT data resides off-chain on IPFS, the blockchain stores metadata. This allows for a lightweight yet informative reference that potential buyers can use to assess the data's relevance without retrieving the entire dataset. With the CID and metadata on the blockchain, potential buyers can perform on-chain queries to discover datasets of interest. It is worth to be mentioned that when the data owner submits the CID and metadata as a transaction, the blockchain network validates it. The blockchain's consensus mechanism ensures that only valid transactions get added to the blockchain, preserving the network's integrity.

(6) : When the transaction is added to the blockchain, it returns the transaction hash that can be used to look up the given transaction. In other words, every blockchain transaction has a unique identifier called a transaction hash or txid. It is generated using a cryptographic hash function, which takes the transaction details as input and produces a fixed-size string of characters. The txid is distinct for each transaction. Even a tiny change in the transaction details will produce an entirely different txid. As mentioned, the txid serves as a digital fingerprint for the transaction. Users can input the txid into a blockchain explorer to view the transaction's details, including its status, input, output, and more. For the data owner and potential data buyers, the txid offers a way to verify the transaction's success. They can ensure that the CID and associated metadata were correctly registered on the blockchain.

(7) : Equipped with graph technology, the data user can query the Ethereum blockchain to find appropriate and relevant information about the available IIoT data for sale. More specifically, the Graph offers a decentralized solution to the challenge of indexing blockchain data. It allows for efficient and performant querying, providing a remedy to the cumbersome process of extracting detailed insights from blockchain data. The Graph uses indexed "subgraphs" that can be queried using a standard GraphQL API. While there exists a hosted version of this service, its decentralized protocol retains the same capabilities, ensuring data integrity and decentralization. At the heart of the Graph's operations are subgraph descriptions or the subgraph manifest. These descriptions define the relevant smart contracts for a given subgraph, highlight the events within those contracts to monitor, and detail how to map event data to the data that the Graph will index in its database. Once a subgraph manifest is deployed, it governs how the Graph interacts with Ethereum transactions. This relationship ensures that data is indexed accurately, efficiently, and in a manner that is queryable by end users. By leveraging the Graph, our IIoT data trading system can efficiently and effectively query detailed data from the Ethereum blockchain, streamlining processes and ensuring users access to precise and comprehensive data insights

(8) : After getting the results of the query, the data user decides whether to make a bid to buy data or not. The Graph's query results are structured responses, typically provided in JSON format. These results encompass not just basic data but also a myriad of detailed insights sourced directly from the Ethereum blockchain. By structuring this data, The Graph ensures users can easily parse, understand, and use the results. The data user can take advantage of the query results to make a decision about the buying the data or not. In fact, the potential buyer can use the Graph's results to compare various IIoT data sets available on the blockchain. By overlaying metadata and understanding the nuances of each data set, buyers can more effectively discern which data sets are most relevant to their needs. Before placing a bid, data users will consider their budget, the perceived value of the data, and the potential return on investment. The comprehensive information from the Graph ensures that this financial decision is well-informed.

(9) : When the decision has been made, the data user has to participate in an auction by placing a bid and competing with other potential participants to buy the data. The auction for the IIoT data is managed through the dedicated smart contract on the Ethereum blockchain. This contract holds the rules of the auction, such as start/end times, minimum bid, and any other relevant parameters. All bids placed are transparently recorded on the blockchain. This ensures a transparent, tamper-proof, and verifiable process where all participants can verify the authenticity of bids and the integrity of the auction. The auction format used in our system is a dynamic pricing auction involve competitive bidding, where participants place incremental bids until the auction concludes.

(10) : At the end of the auction process, the data owner closes the auction and informs the participant who wins the auction. In other words, the smart contract processes the recorded bids to identify the highest bid, which determines the auction winner. Given the immutable nature of blockchain, the contract transparently and reliably identifies the winner without external interference. In rare instances where two bids of the same amount are recorded at the exact same block time, the smart contract can use predetermined logic (e.g., the first recorded bid) to break the tie. Once the winner is determined, the smart contract emits a blockchain event. This event typically contains details about the winning bid and the associated bidder. Blockchain events are efficient means to notify external actors about contract state changes.

(11) : This step involves completing the payment process between the data owner and the winning participant. The smart contract, which governs the auction, automatically verifies the receipt of the payment. It ensures that the exact winning bid amount has been transferred to the contract's address (or a specified escrow). Once the payment is confirmed, it is recorded on the blockchain, ensuring a permanent and tamper-proof record of the transaction. This gives both the data owner and the bidder confidence in the transparency and integrity of the process. Upon successful payment verification, the smart contract automatically trigger the release of the CID, allowing the winning bidder to access the IIoT data from IPFS.

(12) : The data user can request the encrypted data from the IPFS network by providing the related CID. The data user, using a node or an interface, connects to the IPFS network. This can be achieved through local IPFS nodes or using public gateways. By providing the CID, the data user is essentially asking the network: "who has the content corresponding to this identifier?" The IPFS network, then, finds the nodes that hold the content and initiates the transfer. IPFS uses a mechanism called BitSwap to determine the best peers to fetch the data from, optimizing for speed and reducing latency. As the data is retrieved, its content is hashed and compared against the CID to ensure that the data has not been tampered with during transit. The IIoT data on IPFS remains encrypted, ensuring that even if someone intercepts or accesses the data without authorization, they will not be able to derive meaningful information from it. After retrieval, the encrypted data might be temporarily stored on the data user's local storage for further processing or decryption.

(13) : The encrypted data is fetched and decrypted by the data user. Toward this end, the decryption key, which might have been securely shared post-auction or through another secure channel, must be retrieved by the data user. Once decrypted, a checksum or hash of the data might be compared to a previously known value to ensure data integrity and to confirm that the data has not been altered since its encryption. After decrypting, the data should be stored in a secure location, especially if it contains sensitive IoT insights or information. Proper access controls should be in place.

3.3. Implementation details and smart contract design

The details of each function and the algorithm used in our data trading system are described in this section. The Ethereum smart contract is programmed in Solidity using the web-based Remix IDE [27]. Besides, we use the Ganache-Truffle Suite [28] as a personal Ethereum blockchain to deploy and test the smart contract. The Truffle Suite is a popular development framework for Ethereum. It offers a suite of tools designed to help developers create smart contracts, write tests, deploy to various networks, and interact with smart contracts. Ganache is a part of this suite. With Truffle, one can write smart contracts, compile, migrate/deploy them, and also write tests for them. Truffle provides a command-line tool for various tasks such as compiling, deploying, and testing. On the other hand, Ganache is a personal Ethereum blockchain. It allows to run tests, execute commands, and inspect the state while controlling how the chain operates. Essentially, it is a blockchain emulator. It is worth mentioning that Ganache does not use a traditional consensus mechanism such as Proof of Work (PoW) or Proof of Stake (PoS) which is common in public Ethereum networks. Instead, it operates in a way that is designed for development and testing purposes. In Ganache, the consensus process operates as follows:

- Instant mining: Ganache mines a block instantly whenever there is a transaction. This behavior contrasts with the Ethereum mainnet or testnets where there are block intervals. The instant mining feature in Ganache helps developers quickly test and see results without waiting for the usual block times.
- Zero difficulty: blocks in Ganache do not have any difficulty associated with them. In real-world Ethereum networks, miners solve cryptographic puzzles to mine a block, and this difficulty adjusts to ensure block time remains roughly consistent. Ganache bypasses this completely to provide developers with a streamlined experience.
- Deterministic: for repeatability, Ganache provides a deterministic experience. This means that if you start Ganache with the same seed phrase, it will produce the same addresses with the same private keys and the same initial balance. This deterministic behavior can be very helpful for debugging and repeated testing.
- Control over time: Ganache also gives developers the ability to manipulate time, such as jumping forward in time to simulate conditions like a contract's time lock.

Moreover, to interact with Ethereum, we also use the Web3.py [29] tool, particularly for connecting to IPFS. In our implementation, we leverage Infura IPFS as a decentralized storage infrastructure to store the big IIoT data. For cryptographic purposes, we use PyCryptodome and hashlib Python package. Finally, we utilize online Graph technology to create Subgraph to query the blockchain.

The IIoT data trading smart contract provides the following functionalities:

NewDeviceRegisteration(): The data owner, i.e., the smart contract creator, is the only entity that can execute this function. In other words, the data owner can use this function to register a new IIoT device into the system to capture and preprocess its data for future sales. When the data owner calls this function, it is checked whether the given device was registered before or not. If the answer is no, then the new device is added to the existing device list. This list is a Solidity mapping variable type that works as a hash table that stores data in key–value pairs. In our implementation, the key is the device's MAC address, and the value is true/false, indicating the device's registration status. When a new device is added, an event is emitted to store the log information on the blockchain. In this case, the information includes *sender* and *MAC address* (see Algorithm 1).

In Solidity, modifiers are a robust way to run checks before executing function bodies. For the `NewDeviceRegistration` function, we' use a modifier to ensure only the data owner can execute the function. More specifically, the function is set to public to be callable, but with the `onlyDataOwner` modifier, only the data owner can successfully execute it.

ExistingDeviceDeletion(): This function caters to scenarios where there is a need to deregister or remove an IIoT device from the blockchain record. This could be driven by various reasons, for example, because of privacy concerns, the lack of quality data, or competitive advantages. The ownable feature of Openzeppelin is used to grant access to the data owner to be the only entity that can execute this function. The function checks if the device (specified by macAddress) is registered in the `deviceExists` mapping. If not, the function will revert with a message "Device does not exist!". If the device exists, its status in the `deviceExists` mapping is set to false, indicating it is no longer active or registered. Moreover, an event `ExistingDeviceDeletion` is emitted, recording the deletion action on the blockchain. Algorithm 2 shows the details of this function.

Algorithm 1 New Device Registeration

```
Input: sender's address, string memory macAddress
Output: bool
if NewDeviceMacAddress exists then
    return false;
else
    deviceExists[macAddress] = true;
    emit NewDeviceRegisteration(Sender, macAddress);
    return true;
end if
```

Algorithm 2 Delete an existing device

```
Input: string memory macAddress
Output: bool
if NewDeviceMacAddress not exists then
    return false;
else
    deviceExists[macAddress] = false;
    emit ExistingDeviceDeletion(macAddress);
    return true;
end if
```

NewDataUserRegisteration(): The `NewDataUserRegistration` function serves the purpose of allowing a potential data user to express their interest in the IIoT data trading system. By registering, they share their details, specifically their address and the type of data they are interested in. like previous, the function uses the `onlyDataOwner` modifier, ensuring only the data owner can register new users. The function checks if the user, specified by the address `_NewDataUser`, is already registered in the `RegisteredDataUsers` mapping. If they are, the function will revert with a message "User already registered!". The desired data type shared by the user is stored in `UserDesiredData` mapping against their address. In addition, The OpenZeppelin's `_setupRole` function is used to grant the new user the `DATA_USER` role. This function is part of the `AccessControl` feature of OpenZeppelin and helps in managing dynamic roles. You would have defined the `DATA_USER` role earlier in your smart contract (see Algorithm 3).

Algorithm 3 Register New DataUser

```
Input: address NewDataUser, string memory Desireddata
Output: bool
if NewDeviceMacAddress not exists then
    RegisteredDataUsers[NewDataUser] = true;
    GrantingRoles(DATA USER, NewDataUser);
    emit NewDataUserRegisteration(NewDataUser, Desireddata);
    return true;
else
    return false;
end if
```

RemoveExistingDataUSer(): The `RemoveExistingDataUser()` function is designed to purge a data user from the trading system. Such a function is crucial for maintaining the integrity and trust of the system. If a data user's activity becomes suspicious, or if they simply request their own removal, the data owner needs a reliable way to execute this without disrupting the system. Only the data owner or the original creator of the contract has the authority to invoke this function. This ensures that random users or even registered users cannot maliciously remove others from the trading system. The access control is implemented using Ethereum's `msg.sender` property, which allows the function to check who is trying to execute it. In this function, the event `ExistingDataUserDeletion` is emitted, which logs the Ethereum address of the user that was removed. This provides transparency and a historical trace. As it is clear from the Algorithm 4, the function is fairly straightforward. Before attempting to remove, the function checks if `ExistingDataUser` is registered. This is done by looking up the `RegisteredDataUsers` mapping. If the user is found in the mapping, the associated boolean value is set to false, effectively marking the user as inactive or non-registered. Using the `RevokingRoles()` function, the user's role of `DATA_USER` is taken away. This function is a part of the broader role-based access control system implemented in the smart contract. Finally, the function then returns true to indicate the successful removal of the user.

NewMetaDataUpload(): The `NewMetaDataUpload()` function facilitates the addition of new metadata for IIoT data onto the blockchain. Given that data trading heavily depends on the accuracy and availability of data, metadata serves as a beacon for

Algorithm 4 Remove Existing Data User

```
Input: address ExistingDataUser
Output: bool
if ExistingDataUser exists then
    RegisteredDataUsers[ExistingDataUser] = false;
    RevokingRoles(DATA USER, ExistingDataUser);
    emit ExistingDataUserDeletion(ExistingDataUser);
    return true;
else
    return false;
end if
```

potential buyers, illuminating the nature and content of the data that is up for grabs in an auction. Metadata essentially acts as a 'preview' or 'description' of the actual data, helping prospective buyers determine the value and relevance of the IIoT data. It typically consists of IPFS hash, data description, counter, and timestamp. The counter is essentially a unique ID for each data piece, making it easier to reference, bid upon, or perform any other operations on it.

A critical point is the transition of the contract state to `availableForSale` upon successful metadata upload. This state change is pivotal as it signifies that the associated data is ready to be auctioned off. The state acts as a control mechanism to coordinate auction processes and ensure data integrity. Before any operation, the function checks if the provided description is not empty and if the length of the IPFS hash provided is 46 characters (standard length for an IPFS CID v0). Then, the counter (`fileCount`), is incremented. This ensures that every new data entry has a unique ID. The newly provided metadata is stored in the files mapping using `fileCount` as the key. After that, the contract's state is updated to `contractState.availableForSale`. And finally, the event `NewIoTDataPushed` is emitted, providing a log of the newly added metadata. This is vital for external observers or applications wanting to track new data additions (see Algorithm 5).

Algorithm 5 New MetaData Upload

```
Input: string memory ipfsHash, uint256 timestamp, string memory description
Output: bool success
if description.length > 0 and ipfsHash == 46 then
    fileCount++;
    files[fileCount] = File(fileCount, ipfsHash, timestamp, description);
    state == contractState.availableForSale;
    emit NewIoTDataPushed (fileCount, ipfsHash, timestamp, description,);
    return true;
else
    return false;
end if
```

StartAnAuction(): The `StartAnAuction()` function is pivotal in the data trading process, offering potential customers the opportunity to bid on available IIoT data. It is essentially the bridge connecting data owners and prospective data users. Auctions are a transparent mechanism for data valuation, allowing market dynamics to determine the price based on demand and supply. The function first checks if the contract state is `availableForSale`, ensuring that an auction can indeed be initiated for the given data (see Algorithm 6). The `UnitPrice` is initialized to the `minBid` value, setting a baseline for the auction. Moreover, the msg.sender is set as the initial Winner. This is typically the data owner. Indeed, it can serve as a placeholder until genuine bids are placed. In the next line, the AuctionDuration is set to the provided duration, determining the time window in which bids can be placed. The contract's state transitions from `availableForSale` to `AuctionInProgress` ensures that processes and functions pertinent to an ongoing auction can now be activated. At the end, the function emits the `BiddingCommenced` event that notifying potential bidders and observers about the auction's start. It broadcasts the file count, description, timestamp, auction duration, and minimum bid.

EndAuction(): The `EndAuction()` function marks the culmination of the auctioning process, determining the winning bidder for the IIoT data on sale. It ensures that the auction is concluded in a systematic and transparent manner, following the rules set by the data owner and the smart contract. The function verifies that the contract's current state is `AuctionInProgress` to ensure an auction is ongoing and can be concluded. It also checks if the current blockchain time (block.timestamp) exceeds the sum of the auction's starting `TimeStamp` and its allowed `AuctionDuration`. This confirms that the auction duration has indeed elapsed and it is time to conclude the auction. If the verification checks pass, the contract state is updated to `AuctionEnded`, marking the official end of the bidding process. As evident from Algorithm 7, the `BiddingEndedAnnounce` event is emitted. This event serves as a formal and transparent notification to all auction participants and external systems, indicating the auction's conclusion.

DataTransfering(): The `DataTransfering()` function oversees the secure transfer of IIoT data to the auction's winner post-payment. This crucial function ensures that only legitimate and eligible entities gain access to the valuable data and also facilitates the decryption process for the buyer. The function first verifies the contract's current state is set to `PaymentCompletedByWinner`,

Algorithm 6 Start Auction

```
Input: uint256 fileCount, string memory description, uint duration, uint minBid
Output: bool
if state == availableForSale; then
    UnitPrice = minBid;                                   ▷ initialize device current price
    Winner = msg.sender;                                  ▷ current owner
    AuctionDuration = duration;
    TimeStamp = block.timestamp;
    state = AuctionInProgress;
    emit BiddingCommenced(fileCount, description, TimeStamp, duration, minBid);
    return true;
else
    return false;
end if
```

Algorithm 7 End Auction

```
Input: uint256 AuctionDuration, uint256 TimeStamp
Output: bool success
if state == AuctionInProgress AND block.timestamp ≤ (AuctionDuration + TimeStamp) then
    state = AuctionEnded;
    emit BiddingEndedAnnounce();
else
    return false;
end if
```

ensuring that the payment has been successfully processed. It then checks the role of the requesting account to ensure it is designated as `DATA_USER`. This double validation prevents unauthorized or malicious data access and fosters trust within the platform. Once validations pass, the function facilitates data transfer, i.e., the data owner provides the metadata associated with the purchased data. Key elements of this metadata are the IPFS hash and the counter, with the latter serving as a unique index for the IIoT data. Considering the IIoT data is encrypted before its upload to IPFS using the proxy-encryption network, the data owner generates a decryption key tailored for the buyer. This allows the buyer to download and subsequently decrypt the acquired data, making it accessible for their use.

QueryData(): The QueryData() function introduces an elegant way for data users to efficiently search and locate IIoT datasets and their respective owners within the data trading system. The crux of this solution is its reliance on Graph technology, which seamlessly extends the querying capabilities of traditional blockchain structures. By embracing Graph technology, our trading system transforms the conventional blockchain data structure into indexed, easily-queryable segments known as subgraphs. This aids in the swift location of datasets or data owners. A subgraph, once built, can be approached using standard GraphQL queries. This subgraph specifies which portions of data the graph indexes from the Ethereum mainnet and dictates the storage format. The subgraph's manifest is represented as a YAML file. This manifest is fundamental to the subgraph's operations, determining which Solidity events to monitor and establishing the mapping from event logs to the entities stored within the subgraph. Within the manifest, two pivotal components are referenced. The source, which denotes the address of the smart contract in question. The contract Application Binary Interface (ABI), a crucial interface that allows two binary program modules to communicate. Since we built the smart contract on our own, we obtained the ABI file through Remix IDE.

MakeABid(): The `MakeABid()` function is a crucial aspect of our data trading system, which allows registered data users to participate in an ongoing data auction. It enforces multiple conditions to ensure that the bidding process is both fair and transparent. Its primary objective is to facilitate competitive bidding, update the highest bid, and identify the winning bidder at any given time during the auction. The function first checks if the Ethereum address of the caller (bidding party) is present in the `RegisteredDataUsers` mapping. This ensures that only genuine, registered users are permitted to participate in the bidding. Moreover, the bidding is constrained within the bounds of the auction's predetermined duration. The function checks the current blockchain timestamp (block.timestamp) against the sum of `AuctionDuration` and `TimeStamp`. This condition ensures that the auction has not concluded yet. In addition, the function ascertains that no other user is presently bidding (`state != aBuyerIsBidding`). If the aforementioned conditions are met, the state of the contract is set to `aBuyerIsBidding`, signifying that a bid is currently in progress. The function evaluates the provided `bidAmount` against `UnitPrice`, which is a reference to the minimum acceptable bid or the last highest bid. If the new `bidAmount` exceeds `UnitPrice`, then, UnitPrice is updated to the new bid value and the `Winner` variable is updated to the Ethereum address of the current highest bidder. The function emits an event (`NewBidOffered`) to notify other participants and external systems about the latest bid. This event provides details such as fileCount, description, and the updated UnitPrice.

Algorithm 8 Make a Bid

Input: uint256 fileCount, string memory description, uint256 bidAmount

```
if (RegisteredDataUsers[msg.sender] == true) AND (block.timestamp < (AuctionDuration + TimeStamp)) AND (state != aBuyerIsBidding) AND (state == availableForSale or state == EndAuction ) then
    state = aBuyerIsBidding;
    if bidAmount > UnitPrice then
        UnitPrice = bidAmount;
        Winner = msg.sender;
        emit NewBidOffered(fileCount, description, UnitPrice);
    end if
else
    return false;
end if
```

MakePayment(): The `MakePayment()` function is pivotal to the proposed data marketplace as it manages the monetary exchange, ensuring that the highest bidder, who has rightfully won the auction, can pay for and access the data. This function streamlines the payment process, adding layers of validation to ensure both security and transparency. The function should verify that the Ethereum address initiating the payment (`msg.sender`) matches the winner address. This check ensures that only the auction winner is permitted to proceed with the payment. Before the transaction proceeds, the winner has the opportunity to confirm the authenticity of the data owner (seller). This is accomplished by validating that the seller's designated role in the system is `DATA_SELLER`. Such a check offers an added layer of trust to the system. The winner submits the winning bid amount in Ether. Utilizing Ethereum's built-in functions for transferring Ether can achieve this. Upon successful payment, the winner gains access to the auctioned data. This is facilitated using the metadata, most importantly, the IPFS hash. With this hash, the winner can fetch the respective data from the IPFS network. Once the payment process concludes, an event is emitted to document the transaction on the blockchain. This event ensures transparency by notifying all network nodes that the payment has been successfully completed by the winner.

DataRating(): The `DataRating()` function is instrumental in maintaining a trustworthy data trading ecosystem. By allowing data buyers to rate the data they purchase, the platform fosters trust and transparency. This mechanism not only holds data sellers accountable for the quality of data they provide but also guides potential future buyers in their decision-making. The data rating system is vital to providing fair conditions in the data marketplace.

Our data trading system considers a post-purchase feedback scheme where a data user can rate the purchased data. Following this scheme, future potential buyers can select only high-quality providers. Specifically, the data users have five options for giving feedback, including 1=very poor, 2=poor, 3=fair, 4=good, and 5=excellent. These ranks can be assigned to a data owner that has provided a specific IIoT dataset. To distinguish between different data owners/datasets, the rating system uses the Ethereum address of the owners and the counter as a unique ID. This can be done by executing the *DataRating()* function. The function accepts the Ethereum address, counter, and rank as the arguments. Moreover, it emits an event to notify all the Ethereum listeners of the result.

4. Deployment and testing

In this section, we test and validate the details of the IIoT data trading smart contract. The evaluation of performance of our data trading system was conducted on a laptop with an AMD Ryzen CPU clocked at 3.20 GHz, 16 GB of RAM, and running Windows 11. The smart contract consists of different functions tested for correct execution and expected results. As mentioned, our implementation and testing have been accomplished using the web-based Remix IDE. The Ethereum addresses of the participants have been assigned at the execution time.

4.1. Registration and deletion of an IIoT device

The smart contract's initiation and deployment process is intrinsically tied to the data owner. Upon deployment, the constructor function is triggered, which performs multiple crucial actions. One of its primary roles is to assign the designation of ''`DATA SELLER`'' to the data owner. This role does not just act as a label, but it grants specific privileges within the contract, ensuring that the data owner has exclusive rights to manage and control the data on sale. This provision is essential to maintain the integrity and authenticity of the data being transacted within the platform.

Beyond this role assignment, the constructor function is also tasked with the generation of a real-time event. This event, when emitted, serves as a broadcast signal to all entities involved in the IIoT trading platform, notifying them of the contract's successful initiation. This real-time notification ensures that all participants are promptly informed and can adapt their actions accordingly. However, the data owner's capabilities don not end with the initiation. The smart contract provisions allow the data owner to add or register a new IIoT device to the system. This registration is crucial as each device can produce unique datasets, and incorporating them into the trading platform increases the diversity and range of data on offer. The process involves specifying the device's unique characteristics, its data generation capabilities, and any other pertinent metadata that potential buyers might find valuable. The result of the successful execution of the functions is shown in Figs. 5 and 6.

Fig. 5. Logs showing a successful IIoT device registration.

Fig. 6. Logs showing a successful IIoT device deletion.

4.2. Registration and deletion of a data user

In our data trading system, the data users play the role of the customers that compete to buy IIoT datasets through an auction. Primarily, data users gravitate toward the data trading system with an aim to harness these datasets to enhance their operational decisions. Whether it is for streamlining manufacturing processes, optimizing supply chains, or forecasting market demands, the acquired data serves as a cornerstone for data-driven insights. Moreover, the high-quality datasets available empower these entities to elevate the quality of their offerings, ensuring they can provide superior products and services to their consumers. In essence, the data becomes a pivotal tool for them, guiding improvements and innovative strategies. Registering a new data user is more than just adding a new entity. It is about onboarding a new participant, ensuring they understand the nuances of the platform, and facilitating their seamless integration into the existing ecosystem. This entails capturing their unique specifications, preferences, and granting them the necessary privileges to actively engage in the auctioning process. Figs. 7 and 8 clearly indicating the processes of adding and deleting a data user. By showcasing these operations, all participants in the platform can be assured of the system's integrity, with clear, verifiable records of each action taken.

Fig. 7. Logs showing a successful data user registration.

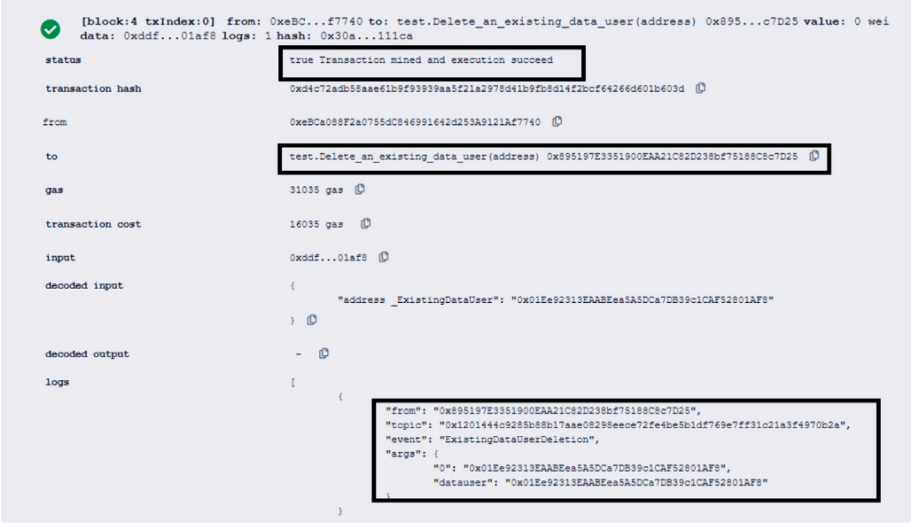


Fig. 8. Logs showing a successful data user deletion.

4.3. Uploading metadata to the blockchain

By uploading metadata onto the blockchain, the data owner effectively provides a snapshot of the data they intend to sell. This metadata encapsulates crucial attributes, such as timestamp, size, format, and a brief description. Essentially, it acts as a window through which potential data buyers can glean insights into the dataset without accessing its entirety.

In the blockchain context, metadata takes on an even more significant role. Given the decentralized and immutable nature of blockchain, once metadata is uploaded, it stands as a permanent record that is tamper-proof and easily verifiable. This ensures that potential data users can trust the authenticity and accuracy of the metadata, which, in turn, builds confidence in the data's underlying quality and relevance.

The permission to upload metadata is exclusively reserved for the data owner. This restriction is essential to maintain the integrity of the trading system. By limiting this privilege, the proposed systems ensures that only genuine, vetted data sets make their way onto the marketplace, minimizing the risk of spam or malicious data. Moreover, by entrusting the data owner with this task, the system also guarantees that the metadata represents the most accurate and up-to-date depiction of the data on offer. This functionality of the smart contract has been tested successfully, and the result is shown in the log, as can be seen in Fig. 9.

Fig. 9. Logs showing successful metadata uploading.

Fig. 10. Logs showing a successful starting of the auction.

4.4. IIoT data auction starting

The process of data auctioning serves as a linchpin in the IIoT data trading system. By providing a competitive landscape, auctions enable data owners to potentially maximize their returns while ensuring that data users acquire datasets that align with their perceived value. The mechanism begins when a data owner decides to monetize their IIoT data. To initiate an auction, the data owner does not merely list the data; they set critical parameters that guide the entire auction process. Two primary parameters are initial price and auction duration. For data users, these parameters provide valuable insights. The initial price offers them a ballpark figure, guiding their bidding strategy. It gives them a sense of what the data might be worth and how much they might need to outbid competitors. The auction duration, on the other hand, provides them with a timeline, helping them pace their bids and strategize accordingly. To keep it simple and focus on its functionality, we show the execution result of a simple auction commencement in Fig. 10.

Fig. 11. Logs showing a data user placing a bid.

4.5. Making a bid for the data

With the integration of blockchain technology, data users have an avenue to explore available datasets. They can delve deep into the repository, querying specific attributes or broad categories, to unearth datasets that resonate with their objectives. This transparency ensures that potential bidders are never in the dark about the opportunities at their disposal. Once a dataset catches the eye, the data users can make a bid. The bidders take into account various factors, from the dataset's relevance to its perceived value. As long as the auction continues and the duration has not ended, the registered data users can place their bids for the data. As it clear from the figure, our testing environment captured a specific instance of a bidding process. As showcased in Fig. 11, a registered user, recognizing the value and potential of the ''IIoT data'' with file ID ''1'', decided to make a substantial bid of 270 Ether. The system, validating the bid against the auction's parameters and current highest bid, deemed it acceptable.

4.6. End of the auction, the selection of the winner and making the payment

The responsibility of concluding the auction rests on the data owner's shoulders. It is more than just identifying the highest bid. It is about verifying its legitimacy, and ensuring adherence to auction terms. Once a winner is chosen, the next step is the payment process. The winning bidder, now transitioning into the role of a buyer, needs to ensure that their financial commitment matches their bidding enthusiasm. The blockchain's transparent and secure environment facilitates this, ensuring that the payment process is seamless, trustworthy, and irrefutable. Fig. 12 indicates the auction winner making the payment successfully and becoming the new owner of the data.

4.7. The rating of the received data

After receiving the data, the data user can rate it based on its quality as part of the rating process. The result of this rating process would then be available for future potential data users to use. The result of the successful execution of the *DataRating()* function is shown in Fig. 13. The buyer rates the received data in this figure as ''1=very poor''. Potential future buyers, while navigating the data marketplace, can harness these insights, making more informed decisions and setting calibrated expectations.

5. Performance evaluation and discussion

This section discusses the assessment of the proposed data trading system, its strengths and weaknesses, and how it differs from the other existing methods.

Fig. 12. Logs showing a successful payment by the winner.

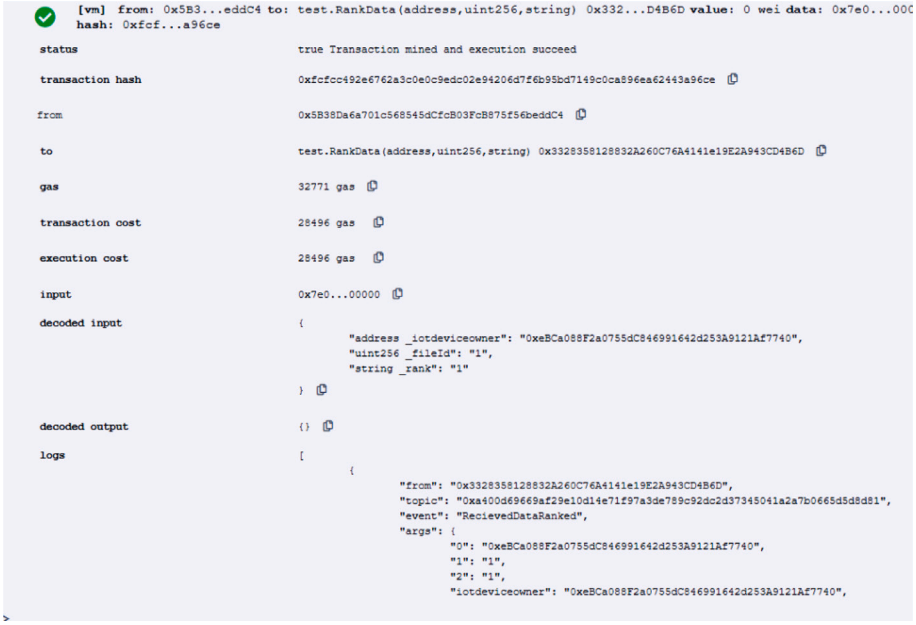


Fig. 13. Logs showing a successful rating process by the data user.

5.1. Security and privacy analysis

We leverage several technologies in the proposed data trading system, including the Ethereum blockchain, IPFS, proxy re-encryption, rule-based access control, and smart contract. Compared to the traditional centralized data trading systems, our system realizes more benefits in terms of security and privacy. These benefits are described in greater detail in the following section.

5.1.1. No third party

Many existing industrial IIoT data trading systems operate under a centralized architecture, using third parties like Dawex [30]. Such centralized systems often host the trading platform, the data storage, and the transaction management all under one authority. This arrangement several security and privacy challenges, including:

- Data leakage: central storage hubs are enticing targets for hackers, as breaching them provides access to a vast amount of valuable data.
- Accidental threats: system crashes or inadvertent data deletion can occur due to various reasons, jeopardizing the stored data's integrity.
- Abuse by third parties: the hosting party may have motives or incentives that are not aligned with the data owners', leading to potential misuse of data.
- Single points of failure: a centralized architecture means that any failure, be it technical, financial, or organizational, impacts the entire system.
- Privacy threats: even if the third party is well-intentioned, hosting sensitive data in a centralized location may inadvertently expose it to surveillance or unauthorized access.

In contrast, our system pivots away from this centralized model. Key technical aspects include:

- Decentralized storage with IPFS: instead of a singular storage hub, data is distributed across nodes in the IPFS network. This means no single entity holds the entirety of the data, making massive data breaches nearly impossible. Also, IPFS's content-addressable nature ensures that the data retrieved is the exact version that was stored, validating its integrity.
- Role of the smart contract: the smart contract on the Ethereum blockchain automates and enforces the terms of the IIoT data trade. Because it is immutable once deployed and operate without intermediaries, it ensures trustworthiness in transactions. Its transparent nature ensures all parties can validate the contract terms and their execution.
- User empowerment through fine-grained access control: the proposed data trading system provides data owners with tools to grant permissions at granular levels. This might be achieved through cryptographic keys authentication, allowing owners to dictate who can access their data and under what conditions.
- Reliability through redundancy: the decentralized nature of our system means that data is often replicated across multiple nodes. This ensures that even if some nodes fail or go offline, the data remains accessible.
- Authenticity checks: given the immutable nature of blockchain, once data is recorded, it cannot be altered without leaving a trace. This provides a strong assurance of the data's authenticity.

5.1.2. No single point of failure

Traditional centralized systems inherently possess single points of vulnerability. Be it servers, databases, or network junctions, any single failure can disrupt the whole system's functionality. In contrast, decentralized architectures, like our proposed system that are based on Ethereum and IPFS, are designed to operate without such singular vulnerabilities. In our system, the Ethereum blockchain is not just a data structure but a whole network of nodes collaboratively validating and recording transactions. Indeed, it adds robustness through distributed validation, immutable ledger, and automatic failover. The last one refers to the situation that if a node in the Ethereum network fails, the system does not collapse. Other nodes continue to operate, ensuring uninterrupted service.

In essence, the combined strengths of Ethereum and IPFS ensure that our data trading system remains resilient against both technical malfunctions and malicious attacks. The decentralized model distributes the risk and, in doing so, eliminates singular points of failure.

5.1.3. Access control

Access control is an essential layer in any secure data management system. Especially in the context of blockchain, where data is immutable once written, it becomes pivotal to regulate who has the authority to write or interact with the data in the first place. Based on the need to protect the IIoT data training system and to have various levels of authorization, we use RBAC to define fine granular permissions, including:

- Defining roles: roles such as "DATA USER" and "DATA SELLER" serve as categories of permissions. Each role encompasses a set of actions that its bearers can perform. For instance, a "DATA SELLER" might be permitted to list IIoT data for sale, or set prices, while a "DATA USER" might be authorized to purchase or query available data.
- Dynamic role management: the beauty of modern smart contract platforms, like Ethereum that has been used in the proposed system, is their ability to adapt. Users can create new roles or modify existing ones (e.g., "DATA USER") as the system evolves, ensuring that the access control remains both secure and flexible.

All in all, by leveraging both contract ownership and the RBAC control mechanism, the proposed data trading system ensures that data operations are not only transparent and auditable but also strictly regulated, bolstering the overall data security of the platform.

5.1.4. Data privacy

Regarding enforced privacy and security laws and regulations worldwide, such as the general data protection regulation (GDPR) in the European Union (EU), it is essential to protect privacy-sensitive data on our data trading system [31]. Blockchain, in its essence, is a transparent ledger. However, our system manages to strike a delicate balance between transparency and privacy. The decentralized nature of blockchain prevents any single entity from having unfettered access to all data. No central servers store data, reducing the risk of massive data breaches. In addition, Ethereum, while transparent, has certain privacy mechanisms, including

private and public keys, and pseudo-anonymity. Regarding the latter, Ethereum addresses (derived from public keys) don not directly reveal the identity of users. While transactions and balances are public, they are tied to addresses, not personal identities. This offers a degree of anonymity, though it is worth noting that advanced analytics can sometimes de-anonymize users.

The crux of ensuring data privacy on our platform is to minimize the touchpoints through which external entities can access data. More specifically, through the strategic use of blockchain, the system ensures that external entities cannot easily siphon off or log sensitive data. This safeguard is pivotal, especially when trading sensitive or proprietary data.

5.1.5. Smart contract security

Our data trading system relies heavily on the functionalities implemented through a smart contract. This smart contract code is inscribed onto the blockchain and is triggered automatically when predefined conditions are met. Given the irreversible nature of blockchain transactions, it is of paramount importance to prioritize smart contract security. Vulnerabilities or oversights can lead to costly implications and are often susceptible to malicious exploitation. Drawing insights from the comprehensive analysis presented in [32] and applying the STRIDE framework, smart contract-based security concerns, especially in the context of IoT use cases, can be clustered into: (1) inherently vulnerable particularities, (2) programming vulnerabilities, and (3) attacks against smart contracts. Building on this study, our goal is to proactively recognize, comprehend, and tackle any potential threats that might impact our system.

Delving deeper into our system's architecture, a critical vulnerability that often emerges in smart contract development is the inadvertent exposure of functions meant to be limited in access. This constitutes a major programming vulnerability. When such critical functions remain inadequately shielded, they become prime targets for malicious entities. This can result in substantial financial losses or other catastrophic consequences. To counteract this, we have embedded "require" statements within our smart contract code. These statements delineate essential prerequisites that need to be satisfied for a function to be executed. Think of them as security checkpoints, diligently ensuring the function operates only under legitimate conditions. Our adoption of "require" statements acts as a pivotal line of defense against unauthorized incursions.

Augmenting our security measures, we have assimilated specific function modifiers like the "onlyOwner" modifier. This particular modifier circumscribes the accessibility of crucial functions, allowing only the contract's creator to invoke them. Such an approach provides a fortified layer against potential external threats, ensuring that critical contract parameters remain uncompromised.

Further refining our security blueprint, we have implemented visibility modifiers. These modifiers delineate the range and extent of function accessibility, reducing the surface area vulnerable to attacks. For instance, functions marked as "private" remain strictly confined within the contract, ensuring their sanctity. "Internal" functions, though a bit more flexible, can only be summoned by related contracts. On the other hand, "public" and "external" functions are more accessible and can be invoked more broadly. Our strategic deployment of these visibility designations guarantees an optimal balance between the contract's functionality and its robust security architecture.

5.2. Comparison with the existing work in the literature

The following table summarizes the existing IIoT data trading systems in the literature and highlights their difference from our proposed solution. In our comparison, we consider a number of different factors, including blockchain platform, off-chain storage, data encryption, etc. As the table shows, our solution complements the previous work on IIoT data trading by utilizing IPFS for decentralized data storage, adopting access control policies, supporting data re-encryption, and implementing blockchain indexing protocol.

The work conducted in [33] focused on data sharing in smart cities based on blockchain technology. The paper's authors claimed that data security and privacy are preserved in the proposed system by using various channels in the network, where every channel includes a limited number of authorized organizations and processes a distinctive type of data. In addition, the authors used an access control mechanism to control the user's access within a channel. Tang et al. [4] proposed a data marketplace based on the blockchain that leverages side-contract mechanisms. The authors combined a group of data owners, buyers, and arbitrators to create a decentralized data trading system under a consortium. In their system, data dispute resolving was the responsibility of arbitrators. Moreover, the authors mentioned that the side-contracts concept could help to decrease the overhead and improve the operating efficiency of their system. Chi et al. [6] provided a blockchain-based data sharing platform for IIoT data. The main focus was on the platform's efficiency, security, and privacy. Toward this end, the authors used identity authentication and Hyperledger Fabric blockchain. Zhang et al. [34] developed an IIoT data trading scheme that improves latency and service quality more than its counterparts. The authors proposed a monitor-based usage control model to implement data usage policies, eliminating frequent interactions between data owners and users. Chenet al. [35] tried to meet two critical requirements of the data trading mechanisms, including the trust guarantee and the real-time limits. Toward this end, the authors presented a blockchain-based non-repudiation approach for IoT data trading. The proposed approach uses a divide-and-conquer strategy and two commitment techniques for efficient IoT data trading. The commitment techniques run in a two-round manner. The authors in [36] combined blockchain technology and non-fungible tokens (NFTs) for private data monetization and time-bound access. In this approach, data owners can upload encrypted data and mint it into NFTs. Then, data users can get access to the uploaded data by requesting a purchase or a license. Dixit et al. [15] proposed an IIoT data marketplace, named FAST DATA, based on blockchain. In the proposed solution, hyperledger fabric blockchain is responsible for hosting data streaming in a reliable and fault-tolerant way. Moreover, the data network layer is built on the VerneMQ broker technology. Badreddine et al. [12] combined message queuing telemetry transport (MQTT) as a transport layer protocol with blockchain to create an IoT data marketplace. Blockchain technology provides

monetization logic and trust for the proposed data marketplace through smart contracts deployment. In addition, the author used a bloom filter-based technique to verify data exchange efficiently. The solution provided by [37] focused on providing a decentralized marketplace for IoT data (e.g., sensor data) based on blockchain. More specifically, Ethereum smart contracts are used to implement different data exchange functionalities and enforce the related rules. Moreover, the authors designed a graphical user interface (GUI) that lets the provider interact with the related decentralized application (DApp). Li et al. [7] provided a framework for private and privacy-preserving private IoT data sharing. To this end, the authors took advantage of blockchain technology characteristics, such as privacy, security, and multi-sharing access control policy for the data owners. They also used the Monero technology and ring signature to share data privately with the data users. In [38] by Xue et al. the authors introduce a blockchain method for private and fair data trading. Using attribute-based credentials, encryption, and zero-knowledge proofs, the system allows buyers to specify data attributes on the blockchain. Sellers then demonstrate data availability in an encrypted form without revealing unnecessary attributes. The mechanism supports splitting data into blocks and omitting sensitive parts without compromising data verification. The scheme ensures seller identity concealment and untraceability of repeated transactions, and its efficacy is validated through formal proofs and simulations. In the study by González et al. [39], the authors introduce the Blockchain-based IoT Data Marketplace (BIDM). This platform aims to decentralize and securely facilitate data sharing between IoT data producers and consumers. BIDM allows IoT infrastructure owners to monetize their data, controlling its access and setting its price. Through their research, the team assesses the operational and scalability aspects of BIDM, highlighting both its potentials and the inherent challenges of using blockchain for data marketplaces. In [40] by Khezr et al. the authors highlight the immense potential of monetizing data from IoT edge devices. They propose a blockchain-based data trading system, enhanced by fog computing. The system leverages crowdsourced fog nodes on the edge network to gather data from IoT devices, emphasizing security and reliability. Performance evaluations using the Hyperledger blockchain assess transaction speeds, latency, and resource usage in varied scenarios. Miguel Pincheira et al. in [41] propose DeBlock, an innovative decentralized data-sharing architecture. Addressing the limitations of current centralized solutions, DeBlock combines public blockchain with distributed storage for enhanced transparency and scalability. Empirical testing showcased its superior performance in terms of gas consumption, latency, and scalability, especially when handling a large number of actors and datasets. Last but not least, in the study by Klaine et al. [42], the authors introduce a blockchain framework to address data leaks and the under-utilization of stored data. This platform allows data producers to store information externally while using blockchain to record transactions and manage access control. The data generator retains full ownership, with blockchain merely facilitating temporary access. The framework accommodates various data types and ensures data quality, timeliness, and uniqueness. The paper also explores potential applications of this system and highlights existing challenges (see Table 1).

5.3. Cost analysis

In the Ethereum network, the execution of every operation, be it a simple transaction or the deployment of a smart contract, incurs a cost. This cost is termed ‘‘gas’’, representing the computational efforts required to successfully process an operation. Users compensate for this computational effort by paying in Ether (ETH), Ethereum’s native cryptocurrency. Gas values are typically expressed in ‘gwei’, a smaller subunit of ETH, with 1 gwei equating to 10^{-9} ETH. To maintain a standardized point of reference, our cost estimation leans on the comprehensive guidelines detailed in the Ethereum yellow paper [10]. This seminal document serves as a touchstone, outlining the gas implications associated with various computational tasks. Table 2 shows the gas used for deploying and calling each function. Of all the operations, smart contract deployment invariably stands out as the most gas-intensive. This finding underscores the computational intricacy of initializing a smart contract on the Ethereum blockchain. For the scope of our experimental analysis, we capped the gas limit at 3,000,000 units to ensure the operations are bounded and to avoid exorbitant costs. Using the conversion ratio for gwei to ETH and subsequently converting ETH to its USD equivalent (based on an average rate of 1333.45 USD per ETH), we derived the tangible cost for every function. To illustrate, the function *NewDataUserRegisteration()* clocks in at an expense of 0.064 USD. A closer look at the functions reveals that *NewMetaDataUpload()* is one of the most expensive functions in the smart contract. This can be attributed to the presence of global mapping variables. In other words, inherently, mapping variables, especially global ones, demand significant storage on the blockchain. Per the Ethereum yellow paper’s insights, any operation that necessitates storing global variables is bound to be resource-intensive [10].

While our focus here is predominantly on the Ethereum network’s gas consumption, it is crucial to acknowledge that other operations could add to the overall cost. Specifically, operations associated with IPFS, encryption protocols, and indexing might introduce additional financial implications, which have not been encapsulated in this analysis.

6. IIoT data trading challenges

A few potential critical open issues for future research work on IIoT data trading systems are presented in this section.

6.1. Pricing models

In an IIoT data marketplace, the data pricing model for the delivered data products is a crucial issue. This is because the data owner or third-party intermediary negotiates the pricing model with the data users and manages the transactions. In other words, the data owners determine prices and terms, and the third party allows the users to access the data (under centralized settings). However, when we look at the literature, there needs to be more guidance on how data owners should set a price for their data, particularly in competitive cases. Despite this gap, some pricing models have been proposed, such as content-time pricing, free models, packaging models, pay-per-use models, flat-fee models, and two-part-tariff models. However, the mentioned models have their weaknesses, namely:

Table 1
Comparison of our proposed system with the existing solutions.

Ref.	Blockchain platform	Data type	Off-chain storage	Data encryption	Access control	Smart contracts	Blockchain indexing	Blockchain role
[33]	Hyperledger Fabric	Smart cities data	No	Yes	Yes	Yes	No	Hyperledger fabric used for data security and privacy preserving
[4]	Ethereum	Private data	Yes	Yes	No	Yes	No	For data trading operations, such as requesting data and paying
[6]	Hyperledger Fabric	IIoT data	No	No	Yes	Yes	No	To ensure data security and improve the performance of data sharing
[34]	Hyperledger Fabric	IIoT data	Yes	Yes	Yes	Yes	No	The hyperledger fabric smart contract is responsible for data storage, indexing, and providing services for other entities
[35]	Ethereum	IoT data	No	Yes	No	Yes	No	Ethereum smart contract is used to deal with disputes on-chain in real-time
[36]	Ethereum	Healthcare data	Yes	Yes	Yes	Yes	No	Ethereum smart contracts are used for private data sharing
[15]	Hyperledger Fabric	IIoT	No	Yes	No	Yes	No	Hyperledger Fabric is employed for the execution of smart contracts and transaction processing
[12]	Ethereum	IoT data	No	No	No	Yes	No	Ethereum blockchain for deploying smart contracts and consequently for events and access management
[37]	Ethereum	IoT data	No	No	No	Yes	No	Using Ethereum smart contracts for developing all the data marketplace functionalities
[7]	Hyperledger Fabric	IoT data	No	Yes	Yes	Yes	No	Hyperledger Fabric blockchain is used to guarantee the anonymity of both sides of transactions of data sharing
[38]	PoA-based Blockchain	Healthcare data	Yes	Yes	No	Yes	No	The blockchain serves as an intermediary and automation tool for the data trading process between buyers and sellers, ensuring both fairness and privacy.
[39]	Ethereum	IoT data	Yes	Yes	No	Yes	No	The authors utilize blockchain to provide a decentralized platform where data producers and consumers can interact.
[40]	Hyperledger	IoT data	Yes	Yes	No	Yes	No	Hyperledger serves as the underlying infrastructure for the proposed data trading system. It ensures that transactions between data providers and consumers are recorded immutably, providing trust in the trading process.

(*continued on next page*)

Table 1 (*continued*).

Ref.	Blockchain platform	Data type	Off-chain storage	Data encryption	Access control	Smart contracts	Blockchain indexing	Blockchain role
[42]	Ethereum	Sensors data	Yes	Yes	Yes	Yes	No	The blockchain is used to transparently record buy and sell transactions between parties. This ensures that each transaction is immutable and can be verified by any participant.
[41]	Ethereum	N/A	Yes	Yes	No	Yes	No	In this study, blockchain, particularly through smart contracts, plays a central role in ensuring a transparent, scalable, and trustworthy mechanism for data sharing in a decentralized manner.
Our work	Ethereum	IIoT data	Yes	Yes	Yes	Yes	Yes	Ethereum smart contracts are used for the implementation all of the data trading functionalities and access control

Table 2
Gas cost fee of the smart contract functions.

Function	Gas used (Gwei)	Actual cost (ETH)	USD
Smart Contract Deployment	2,699,640	0.00269964	3.59
NewDeviceRegisteration()	70,577	0.000070577	0.094
ExistingDeviceDeletion()	36,196	0.000036196	0.048
NewDataUserRegisteration()	48,453	0.000048453	0.064
RemoveExistingDataUSer()	31,035	0.000031035	0.041
NewMetaDataUpload()	198,733	0.000198733	0.26
StartAnAuction()	115,616	0.000115616	0.15
EndAuction()	51,017	0.000051017	0.068
MakeABid()	49,966	0.000049966	0.066
MakePayment()	32,604	0.000032604	0.043
DataRating()	32,812	0.000032812	0.044
DataTransfering()	29,812	0.000029812	0.040

- They usually do not consider data quality-related issues.
- They are not standard pricing models.
- They are mainly data owner-driven and do not provide a clear picture of the data acquisition, cleaning, and packaging process for the data users.

By considering these issues, there is a need to develop a rigorous and fair pricing model for IIoT data marketplaces. In this context, one of the interesting studies is [43] by Miguel Pincheira et al. In this study, the authors provide a model to assess the infrastructure costs and benefits in applications-based blockchain. The proposed system is based on parameters that describe the application and can be simply recognized at any phase of the application life-cycle.

6.2. Lack of trust

As mentioned, the centralized data marketplace mainly depends on third parties, platforms, or project partners. In this scenario, the lack of trust in these partners regarding data sovereignty and integrity is challenging. In many cases, this unanswered issue may cause the creation of data silos and fragmented data in which data is collected and stored by a centralized business across two or multiple repositories. Usually, these data silos are not value-creating assets because external organizations cannot access them and are generally not created for interoperability.

To alleviate this issue, some solutions can be adopted. For example, creating decentralized infrastructures required for data collection and processing, developing data security and privacy platforms, and developing data marketplaces that satisfy data integrity and audit data-related requirements.

6.3. Blockchain-based data marketplace challenges

Decentralized data trading systems based on blockchain are a reasonable alternative to their centralized counterparts. However, our investigation found that cost and scalability are two critical challenges in blockchain-based solutions. Blockchain technologies, e.g., Ethereum, have to create a trade-off between speed, efficiency, and security. In other words, these technologies impose some limitations on the number of blocks that can chain to the ledger to increase security. However, these limitations can compromise the efficiency and speed of the network, causing delays and performance degradation.

Scalability is also an issue for blockchain-based solutions. The continuously increasing number of users and limited number of transactions per second has led to the blockchain scalability problem. Indeed, different factors can determine blockchain scalability, such as transaction fees, throughput, block size, response time, and networking. Over the recent years, some solutions have been proposed to solve or alleviate the different scalability challenges in the blockchain, including improved consensus mechanism, sharding, and nested blockchain. For example, the new consensus mechanism used in the upcoming upgrade to Ethereum, Ethereum 2.0 (ETH2), is supported with the sharding technique to decrease congestion and improve the network throughput.

7. Conclusion

In this paper, we have proposed a data trading system based on blockchain for industrial IoT data that is decentralized, secure, privacy-preserving, transparent, traceable, and auditable. A role-based access control mechanism is adopted to impose access regulations. Later, an indexing protocol based on graph technology is introduced to further ease the querying of blockchain data. In addition, we combined IPFS decentralized storage to deal with the large-size IIoT data issue. We developed an Ethereum smart contract with different functionalities, allowing the data owners to sell their data through a public auction to potential buyers. A proxy re-encryption scheme was used in the proposed data trading system for securely storing and sharing IIoT data. To test and validate the proposed data trading system functionalities, all the smart contract functions have been deployed and executed on the testnet. Further, the system's performance has been investigated through security and cost analysis, revealing that our blockchain-based data trading system offers enhanced decentralization, improved query efficiency via the graph technology-backed indexing protocol, and robust data security using the proxy re-encryption scheme. When compared with existing proposals, our system demonstrates a superior balance between security and efficiency, substantiated by the consistent performance metrics observed on the testnet.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: This work has been performed under the IoTalentum project within the framework of Marie Sklodowska-Curie Actions ITN-ETN (grant number 953442) which is funded by the European Union Horizon 2020 research and innovation program

Data availability

Data will be made available on request.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, IEEE Trans. Ind. Inform. 14 (11) (2018) 4724–4734.
[2] J.-Q. Li, F.R. Yu, G. Deng, C. Luo, Z. Ming, Q. Yan, Industrial internet: A survey on the enabling technologies, applications, and challenges, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1504–1526.
[3] C.-L. Chen, J. Yang, W.-J. Tsaur, W. Weng, C.-M. Wu, X. Wei, Enterprise data sharing with privacy-preserved based on hyperledger fabric blockchain in IIOT's application, Sensors 22 (3) (2022) 1146.
[4] H. Tang, Y. Qiao, F. Yang, B. Cai, R. Gao, dMOBAs: A data marketplace on blockchain with arbitration using side-contracts mechanism, Comput. Commun. 193 (2022) 10–22.
[5] L. Mikkelsen, K. Mortensen, H. Rasmussen, H.-P. Schwefel, T. Madsen, Realization and evaluation of marketplace functionalities using ethereum blockchain, in: 2018 International Conference on Internet of Things, Embedded Systems and Communications, IINTEC, IEEE, 2018, pp. 47–52.
[6] J. Chi, Y. Li, J. Huang, J. Liu, Y. Jin, C. Chen, T. Qiu, A secure and efficient data sharing scheme based on blockchain in industrial internet of things, J. Netw. Comput. Appl. 167 (2020) 102710.
[7] T. Li, H. Wang, D. He, J. Yu, Blockchain-based privacy-preserving and rewarding private data sharing for IoT, IEEE Internet Things J. (2022).
[8] X. Zheng, J. Lu, S. Sun, D. Kiritsis, Decentralized industrial IoT data management based on blockchain and IPFS, in: IFIP International Conference on Advances in Production Management Systems, Springer, 2020, pp. 222–229.
[9] K.R. Özyilmaz, M. Doğan, A. Yurdakul, IDMoB: IoT data marketplace on blockchain, in: 2018 Crypto Valley Conference on Blockchain Technology, CVCBT, IEEE, 2018, pp. 11–19.
[10] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Pap. 151 (2014) (2014) 1–32.
[11] J. Hu, M. Reed, N. Thomos, M.F. Al-Naday, K. Yang, A dynamic service trading in a DLT-assisted industrial IoT marketplace, IEEE Trans. Netw. Serv. Manag. (2022).
[12] W. Badreddine, K. Zhang, C. Talhi, Monetization using blockchains for IoT data marketplace, in: 2020 IEEE International Conference on Blockchain and Cryptocurrency, ICBC, IEEE, 2020, pp. 1–9.
[13] Y. Wang, Z. Su, N. Zhang, J. Chen, X. Sun, Z. Ye, Z. Zhou, SPDS: A secure and auditable private data sharing scheme for smart grid based on blockchain, IEEE Trans. Ind. Inform. 17 (11) (2020) 7688–7699.

[14] S. Qi, Y. Lu, Y. Zheng, Y. Li, X. Chen, CPDS: Enabling compressed and private data sharing for industrial internet of things over blockchain, IEEE Trans. Ind. Inform. 17 (4) (2020) 2376–2387.
[15] A. Dixit, A. Singh, Y. Rahulamathavan, M. Rajarajan, FAST DATA: A fair, secure and trusted decentralized IIoT data marketplace enabled by blockchain, IEEE Internet Things J. (2021).
[16] K.O.M. Salih, T.A. Rashid, D. Radovanovic, N. Bacanin, A comprehensive survey on the internet of things with the industrial marketplace, Sensors 22 (3) (2022) 730.
[17] M.A. Khan, K. Salah, IoT security: Review, blockchain solutions, and open challenges, Fut. Gener. Comput. Syst. 82 (2018) 395–411.
[18] L.W. Cong, Z. He, Blockchain disruption and smart contracts, Rev. Financ. Stud. 32 (5) (2019) 1754–1797.
[19] P. Sharma, S. Lawrenz, A. Rausch, Towards trustworthy and independent data marketplaces, in: Proceedings of the 2020 the 2nd International Conference on Blockchain Technology, 2020, pp. 39–45.
[20] I. Foundation, IOTA provides digital trust, enabling us to build A better world. [Online]. Available: https://www.iota.org/.
[21] dclimate community, The first decentralized network for climate Data. [Online]. Available: https://www.dclimate.net/.
[22] Y. Chen, H. Li, K. Li, J. Zhang, An improved P2P file system scheme based on IPFS and blockchain, in: 2017 IEEE International Conference on Big Data, Big Data, IEEE, 2017, pp. 2652–2657.
[23] S. Kim, I. Lee, IoT device security based on proxy re-encryption, J. Ambient Intell. Humaniz. Comput. 9 (4) (2018) 1267–1273.
[24] D. Servos, S.L. Osborn, Current research and open problems in attribute-based access control, ACM Comput. Surv. 49 (4) (2017) 1–45.
[25] A. Masoumzadeh, P. Narendran, P. Iyer, Towards a theory for semantics and expressiveness analysis of rule-based access control models, in: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies, 2021, pp. 33–43.
[26] J.P. Cruz, Y. Kaji, N. Yanai, RBAC-SC: Role-based access control using smart contract, Ieee Access 6 (2018) 12240–12251.
[27] REMIX PROJECT, Remix IDE. [Online]. Available: https://remix-project.org/.
[28] Ganache, Ganache one click blockchain. [Online]. Available: https://trufflesuite.com/ganache/.
[29] Web3.py, Web3.py is a Python library for interacting with ethereum. [Online]. Available: https://web3py.readthedocs.io/en/v5/index.html.
[30] Dawex, To monetize and acquire IoT data. [Online]. Available: https://www.dawex.com/en/monetization-data-iot/.
[31] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), first ed., in: A Practical Guide, vol. 10, (no. 3152676) Springer International Publishing, Cham, 2017, pp. 10–5555.
[32] K. Peng, M. Li, H. Huang, C. Wang, S. Wan, K.-K.R. Choo, Security challenges and opportunities for smart contracts in Internet of Things: A survey, IEEE Internet Things J. 8 (15) (2021) 12004–12020.
[33] I. Makhdoom, I. Zhou, M. Abolhasan, J. Lipman, W. Ni, PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities, Comput. Secur. 88 (2020) 101653.
[34] X. Zhang, X. Li, Y. Miao, X. Luo, Y. Wang, S. Ma, J. Weng, A data trading scheme with efficient data usage control for industrial IoT, IEEE Trans. Ind. Inform. 18 (7) (2021) 4456–4465.
[35] F. Chen, J. Wang, C. Jiang, T. Xiang, Y. Yang, Blockchain based non-repudiable IoT data trading: Simpler, faster, and cheaper, in: IEEE INFOCOM 2022-IEEE Conference on Computer Communications, IEEE, 2022, pp. 1958–1967.
[36] M. Madine, K. Salah, R. Jayaraman, A. Battah, H. Hasan, I. Yaqoob, Blockchain and NFTs for time-bound access and monetization of private data, IEEE Access 10 (2022) 94186–94202.
[37] M. Sober, G. Scaffino, S. Schulte, S.S. Kanhere, A blockchain-based IoT data marketplace, Cluster Comput. (2022) 1–23.
[38] L. Xue, J. Ni, D. Liu, X. Lin, X. Shen, Blockchain-based fair and fine-grained data trading with privacy preservation, IEEE Trans. Comput. (2023).
[39] V. González, L. Sánchez, J. Lanza, J.R. Santana, P. Sotres, A.E. García, On the use of blockchain to enable a highly scalable internet of things data marketplace, Internet Things 22 (2023) 100722.
[40] S. Khezr, A. Yassine, R. Benlamri, Towards a secure and dependable IoT data monetization using blockchain and fog computing, Cluster Comput. 26 (2) (2023) 1551–1564.
[41] M. Pincheira, E. Donini, M. Vecchio, S. Kanhere, A decentralized architecture for trusted dataset sharing using smart contracts and distributed storage, Sensors 22 (23) (2022) 9118.
[42] P.V. Klaine, H. Xu, L. Zhang, M. Imran, Z. Zhu, A privacy-preserving blockchain platform for a data marketplace, Distrib. Ledger Technol.: Res. Pract. 2 (1) (2023) 1–16.
[43] M. Pincheira, E. Donini, M. Vecchio, R. Giaffreda, Towards an infrastructure cost model for blockchain-based applications, in: International Congress on Blockchain and Applications, Springer, 2022, pp. 345–355.