Cluster Computing
https://doi.org/10.1007/s10586-022-03745-6

A blockchain-based IoT data marketplace

Michael Sober[1,4] · Giulia Scaffino[2,4] · Stefan Schulte[1,4] · Salil S. Kanhere[3]

Received: 17 August 2022 / Revised: 17 August 2022 / Accepted: 3 September 2022

Abstract
The (IoT) is growing steadily, and so is the number of data that is generated by (IoT) devices. This makes it difficult to find and leverage relevant data (and data sources) without a data marketplace. Such a marketplace provides a platform to enable different parties, e.g., sensor operators and service providers, to trade their data. Today, most data marketplaces are based on centralized solutions, which may become a single point of failure and come with expensive infrastructure, trust problems, and privacy issues. Therefore, we propose the application of blockchain technology to implement a data marketplace for the IoT. Within the proposed marketplace, smart contracts are used to implement various functionalities and enforce the rules of the data exchange. The marketplace also includes a proxy, a broker, and (GUIs) to enable data trading. To show the applicability of the proposed data marketplace, we analyze the costs arising from the utilization of smart contracts.

Keywords Data marketplace · Blockchain · Smart contract

1 Introduction

The (IoT) is a worldwide network of interconnected objects which are uniquely addressable and communicate over standardized protocols [1]. As the number of (IoT) devices rises rapidly, there is also rapid growth of data that is generated and transmitted over the Internet by those devices.

Data marketplaces allow organizations to exchange (large amounts of) data for free, as is the case in open (government) data scenarios [2] or based on a cost model [3]. In many scenarios, data is very valuable, and thus interest in making economic benefit from it is high. Studies have shown that the potential annual value ranges hundreds of billions of US dollars in certain application areas [4].

Centralized approaches to creating a data marketplace come with different challenges concerning expensive infrastructure, trust problems, privacy issues, and single points of failure [5]. The owner of a centralized data marketplace acts as a central authority, which is in charge of the rules of the system and can change it arbitrarily, only considering its own interests. Additionally, transparency is also limited to the owner of the data marketplace. Further, to cope with the huge amounts of data, an individual or an organization would need to build an enormous infrastructure. Concerning (IoT) data, one particular issue with centralized approaches is that (IoT) devices lack the resources to maintain multiple connections to other parties, e.g., because of energy constraints [6]. One possible way to cope with these problems would be to explore solutions that follow a decentralized approach.

In recent years, blockchain technology gained popularity, especially through its usage as the technical backbone for cryptocurrencies like Bitcoin [7]. A blockchain is a linked list of blocks where the links are hash pointers [8]. Blockchains provide properties such as public verifiability,

✉ Michael Sober
michael.sober@tuhh.de

Giulia Scaffino
giulia.scaffino@tuwien.ac.at

Stefan Schulte
stefan.schulte@tuhh.de

Salil S. Kanhere
salil.kanhere@unsw.edu.au

[1] TU Hamburg, Hamburg, Germany

[2] TU Wien, Vienna, Austria

[3] UNSW, Sydney, Australia

[4] Christian Doppler Laboratory for Blockchain Technologies for the Internet of Things, Hamburg, Germany

Published online: 22 September 2022

transparency, integrity, and redundancy [9]. The second generation of blockchain technology offers not only the possibility to exchange tokens but also to execute smart contracts. Such contracts are scripts that get stored on the blockchain. A smart contract can be invoked by sending a transaction to it, which results in the automatic and independent execution of the script by every node in the network [10]. For a decentralized (IoT) data marketplace, smart contracts could provide the means to enable trustless data trading. A blockchain could then store the history of all trades that occurred to make data exchanges verifiable [5].

Already existing concepts and solutions to apply blockchain technologies for data exchange in the IoT focus for the most part on blockchain-based data trading, which is only one part of a data marketplace. But even works that propose solutions for decentralized data marketplaces usually do not include all the important key elements that are necessary (see Sect. 3). In addition to data trading, solutions also have to provide additional functionality, including user management, device management, product management, negotiation, discovery, selection, routing, settlement, rating, monitoring, and web access.

Therefore, to explore solutions that follow a decentralized approach, we conceptualize and implement a marketplace for trading (IoT) data. For this, we investigate if the utilization of (P2P) and blockchain technology can provide the necessary technological functionalities for IoT data marketplaces. Our solution not only facilitates the exchange of data but also provides the above-mentioned additional functionality to make the data marketplace applicable in real-world settings.

The remainder of this paper is organized as follows: In Sect. 2, we introduce some prerequisite information for a blockchain-based (IoT) data marketplace. Afterward, we discuss related work in Sect. 3. After that, we define the requirements for a blockchain-based IoT data marketplace in Sect. 4 and present the design of the data marketplace in Sect. 5. We follow up with an explanation of the implementation details in Sect. 6 and evaluate the costs associated with the use of smart contracts in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Background

In this section, we introduce some background information about the main technological building blocks of the presented work, namely blockchain technology (see Sect. 2.1) and data marketplaces (see Sect. 2.2).

2.1 Blockchain technology

The first cryptocurrency, Bitcoin, was introduced by Satoshi Nakamoto [7] and represents the seminal application of blockchain technology. The author envisioned an electronic payment system without the need for trusted third parties, whereby cryptographic primitives such as cryptographic hash functions and digital signatures provide the means to achieve this goal.

Most cryptocurrencies use a blockchain as their underlying data structure to record transactions in an append-only and tamper-evident log. A blockchain is a linked list that utilizes hash pointers (see Fig. 1). A hash pointer is not only a pointer to the location of the stored data, but also includes the data's hash value. This offers the advantage that besides knowing the location, we can also verify the data. If an adversary changes the information of one element, it must change all hash pointers of the following elements [8], which allows storing transactions (and also more generic data) in a tamper-proof way.

Subsequent works refer to blockchains not only as a simple data structure that can be stored on a single machine but rather as an append-only distributed ledger of transactions managed by a (P2P) network. Every node in the system is responsible for verifying transactions and storing them in its copy of the distributed ledger. Blockchain systems can differ in numerous ways from each other [11]. One can differentiate between permissionless, permissioned, and partially permissioned blockchains. In a permissionless system, every node is allowed to join the network. The opposite is a permissioned system, where a node has to get permission to join the system. There are also hybrid systems, which allow every node to join the network, but only certain nodes are allowed to take part in the consensus mechanism. Further, a major distinction is made in how the system selects the participants, which are allowed to verify and propose the next block, that should be included in the blockchain. The most popular mechanism is (PoW), introduced in [7], requiring the participants to solve a cryptographic hash puzzle.

In second-generation blockchains, smart contracts add additional value. Smart contracts are not bound to blockchains and have first been mentioned in [12] defining a

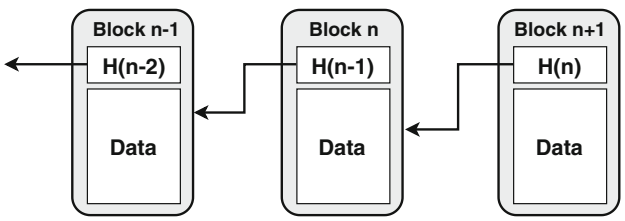

Fig. 1 A linked list of blocks utilizing hash pointers

smart contract as "a computerized transaction protocol that executes the terms of a contract". This offers the possibility to reduce the risk introduced by trusted intermediaries and minimize errors, whether they were made intentionally or accidentally. In the area of blockchain technology and cryptocurrencies, smart contracts are pieces of code that get stored on the blockchain and are executed by multiple or rather every node of the underlying (P2P) network in a deterministic manner [8]. In general, a smart contract is stored on the blockchain and has a unique address and a state. The unique address is used to invoke the functions of a smart contract, which either reads or manipulates the state of the blockchain.

Every participant of the network can view the smart contract since it is stored directly on the blockchain. This promotes confidence and trust in the provided smart contract since the code can be viewed by everyone, e.g., to check for malicious behavior. In general, smart contracts are tamper-proof as well, i.e., once deployed to a blockchain, a smart contract's code cannot be altered or deleted. Among other things, this has to be considered when developing decentralized applications [10, 13].

2.2 Data marketplaces

In general, a data marketplace acts as a platform for trading data between one or more entities (see Fig. 2). Concerning (IoT) data, marketplaces provide data producers with the opportunity to make data accessible to arbitrary data consumers. Thus, data producers can obtain a higher economic benefit. There is great potential behind this idea since statistics show that the number of connected (IoT) devices is expected to rise from 42.62 billion in 2022 to 75.44 billion worldwide by 2025 [14].

Apart from organizations, also individuals can benefit from data marketplaces by personally offering their data for sale or by receiving access to premium services or compensation from organizations to share their data. One application area would be in the field of fitness and health data. Smartwatch manufacturers may ask their users if they can share their fitness or health data collected through their smartwatches and offer some form of compensation in exchange [15]. The machine-to-machine economy is also becoming increasingly interesting, with various devices automatically providing data and services in exchange for money. Data marketplaces would provide additional means to enable a machine-to-machine economy by offering a platform for machines to exchange their data with each other for money [16, 17]. Smart cables paying for electricity provided by smart sockets [18] or vehicles paying for fuel at the gas station [19] provide further examples where data marketplaces may help the data producers to get a benefit by providing their data.

Fig. 2 Data marketplace

There are already many different proposals for the realization of data marketplaces, including data marketplaces specialized in specific data, e.g., (IoT) data marketplaces (see Sect. 3). We can further distinguish between data marketplaces based on a centralized architecture [6] or a decentralized architecture [5]. Data marketplaces can be open to anyone to participate or be restricted to only certain organizations and individuals. In this work, we focus on creating a decentralized (IoT) data marketplace based on blockchain technology. This allows us to trade data without having to trust a third party, i.e., a particular data marketplace operator. Furthermore, decentralized data marketplaces, which use a public distributed ledger, provide transparency for all transactions, making it easier to verify data exchanges in retrospect.

3 Related work

Research on data marketplaces has led to several different concepts and solutions. The related work includes traditional data trading, including the (SaaS) model and various types of centralized (IoT) data marketplaces (see Sect. 3.1). Furthermore, blockchain-based data trading concepts have also been proposed (see Sect. 3.2), covering blockchain-based data payment protocols and solutions for decentralized (IoT) data marketplaces.

3.1 Traditional data trading

Zaslavsky et al. propose the (SaaS) model, whereby sensor data should be provided for a fee [4]. The authors identify three main participants within this model: the owners of the sensors, mediators for publishing and managing sensors, and the data consumers. Notably, the (SaaS) model does not constitute a data marketplace but can be seen as the underlying principle of data marketplaces, bringing together different parties to exchange data for monetary value.

The authors of [20] conducted an initial survey on data marketplaces in 2013. The authors define a data marketplace as "a platform on which anybody (or at least a great number of potentially registered clients) can upload and maintain data sets". The authors examined data providers based on twelve dimensions: Type, time frame, domain, data origin, pricing model, data access, data output,

language, target audience, trustworthiness, size of vendor, and maturity. Each of these dimensions captures different properties of data marketplaces to differentiate between various data vendors. In the following discussions, we will focus on a specific type, namely (IoT) data marketplaces.

Mišura and Žagar propose a model for a centralized (IoT) data marketplace [6]. The authors mention that the design of existing solutions of data marketplaces on the Internet like the deprecated Microsoft Azure Data Market [21] and Salesforce's data.com are not well-suited to implement a marketplace for (IoT) data. These marketplaces only offer collections of data that have been previously acquired. Instead, (IoT) data marketplaces require a special design taking into account the characteristics of data generated by (IoT) devices. In the proposed solution, (IoT) devices and data consumers register themselves at the marketplace through a Web interface. Sensor measurements of devices are saved in a database. Devices never communicate directly with consumers to ensure low battery usage. For consumers to retrieve the data, a query mechanism is used. Devices that satisfy the query are returned by the system and can be accepted by the consumer. While this solution is based on a centralized approach, it gives important insights into design issues for (IoT) data marketplaces.

In [3], Cao et al. discuss problems similar to those identified by [6]. The authors develop a new design for a data marketplace especially to handle real-time human sensing data that encourage data providers to publish their data. The requirements include effective near real-time distribution of data, push and pull communication, data contract management, monitoring quality, and an (API) for providers and consumers. The result is an architecture consisting of several services. Data providers use the management services to offer their data on the data marketplace. Further, a data discovery service provides the means for data consumers to find specific offers. The data is delivered from producers to consumers through databuses. Additionally, these databuses are monitored by services for data quality analysis to ensure credibility and a payment service.

The authors of [22] also propose a centralized marketplace platform for the (IoT), called (I3). The platform provides mechanisms including a user interface for each user group, a registry, a recommendation system, a rating system, mechanisms for billing and payments, real-time data routing, historical data access, authorization, metering, privacy, different data formats, and (API). The registry stores important information about all participants of the data marketplaces as well as the offered data streams. The important functionality of matching buyers to data streams is facilitated by a recommendation system. It provides the buyers with a reliable way to identify and find the data which they need. Trust is built by using a rating system, where buyers and sellers can rate each other. Billings and payments can be realized in various ways, whereby the system utilizes data access and metering mechanisms to determine the exact price based on the data which got traded. Real-time data routing is realized through a data routing middleware, e.g., message brokers, where the devices publish generated data on specific topics, and authorized data consumers can subscribe to these topics. In some situations, data consumers may not need access to the data in real-time but rather historical data, which is achieved by providing a data repository for historical data.

Looking at traditional data trading approaches, we can identify several disadvantages of centralized data marketplaces. The data marketplace represents a single point of failure since the users are fully dependent on central components. In the event of a failure, the users would no longer be able to use the data marketplace, which can be very problematic for applications that heavily rely on the availability of certain data. Centralized data marketplaces also come with trust and privacy issues since users have to deal with a centralized authority that is capable of manipulating the market and accessing the data. These reasons, among other things, motivate the exploration of the application of blockchain technology in this research area.

3.2 Blockchain-based data trading

3.2.1 Blockchain-based data payment protocols

Radhakrishnan and Krishnamachari [23] provided some seminal work on applying blockchains to data marketplaces, however, with a focus on the actual payment for data. For this, the authors describe a (SDPP) to realize a micropayment system for (IoT) data streams. The protocol is based on distributed ledger technology, and the reference implementation uses IOTA both as a cryptocurrency and distributed ledger. The protocol consists of three different components, including a data channel, a payment channel, and a record medium. The buyer initiates the connection with a hello message, to which the seller is going to reply with a menu that consists of the offers. The buyer can now make an order. After that, the buyer uses the public key of the seller to exchange a symmetric encryption key, which is used for further communication. Consequently, the data flow starts, and the buyer pays the seller based on the granularity defined in the menu. Every order, invoice, and payment is recorded on the distributed ledger.

(SPS) is a protocol proposed by Zhao et al. [24] to enable reliable data distribution in the environment of (CPS). The architecture differs from traditional pub-sub solutions in that middleware is not required due to the use

of blockchain technology. The authors create blockchain-based fair payments with the additional use of a reputation system. Within the (SPS) model, data providers advertise on the blockchain which topics they offer. Data consumers announce their interest in certain topics and make a deposit. The provider uses the blockchain system as a communication medium, whereby the data to be transmitted is stored in encrypted form on the blockchain. Afterward, the stored data can be retrieved and decrypted by the consumer. The reputation system ensures that only publishers with a reputation score over a specific threshold can publish a transaction. While the model ensures confidentiality, verifiability, anonymity, no trust, fairness, and reputation, the throughput and storage space of Bitcoin or Ethereum can become a significant problem since the data is stored on-chain.

In [25], the authors propose a protocol that enables fair data trading based on Bitcoin transactions. The authors make use of private key-locked transactions by providing a new Bitcoin opcode and exploiting an (ECDSA) vulnerability. The protocol can be divided into three different phases: the data correctness proof, the signature commitment, and the private key exchange. Beginning with the data correctness proof, the buyer receives encrypted chunks of the requested data. The data must ensure a condition that the buyer stated in the request. Based on a cut and choose protocol, the buyer selects a random subset from which the seller has to provide a correctness proof based on the condition. With this evidence, the buyer can be sure that the data is correct with a certain probability. Subsequently, the signature commitment phase can be carried out. The buyer requests a signature from the seller, including a specific nonce. The seller generates a signature using its secret key and the received nonce and transmits it to the buyer, which can verify the signature by using the public key of the seller. In the last part of the protocol, the buyer broadcasts a private key-locked transaction, including the payment for the data. After that, the seller can receive the funds by providing a signature with the same nonce as in the preceding phase, which is then used by the buyer to restore the private key and decrypt the data.

The previously discussed protocols take advantage of the blockchain, but their focus is on the actual payment process rather than providing a full-fledged data marketplace. Consequently, we will also look at different solutions for decentralized data marketplaces, which may use similar protocols to implement data trading.

3.3 Blockchain-based data marketplaces

Wörner and Von Bomhard [26] propose a system where sensors can offer their measurements in exchange for electronic cash. The authors propose a decentralized infrastructure that allows the exchange of data on a worldwide data market. The system consists of a sensor client, a requester client, and a sensor repository. The sensor client is the provider of the data and has to respond to payments by creating and publishing transactions with the bought data on the blockchain. A requester client wants to consume data and pays the consumer by issuing Bitcoin transactions. The sensors which provide the data are registered in the sensor repository to enable requesters to find sensors. Entries of the sensor registry contain basic information about the offered product and additional metadata. However, the system faces scalability issues since the data is stored on the blockchain, and the transmission is limited by the throughput of the blockchain.

Missier et al. [27] come up with an architecture for (IoT) traffic metering and contract compliance. The architecture is built on top of the (IoT) brokered data infrastructure, whereby the authors assume that communication is mediated through an infrastructure that consists of message brokers to exchange data between (IoT) devices. Buyers and sellers agree upon specific topics, which will be accessible to the buyers for a certain timeframe. Contract enforcement and settlement are realized through traffic cubes, wherein all or partial data flows are recorded. Producers and consumers only construct unilateral traffic cubes since the broker decouples the communication between producer and consumer, whereby the broker has the whole view of all data flows. The cubes are published through transactions on the blockchain to ensure transparency and then checked for consistency for the settlement. The design focuses mainly on decentralized metering of (IoT) data and trustless contract settlement. However, the solution does not provide a platform to discover and select certain products. Further, the incentive why a third party should operate a broker is missing.

Building upon the previous work by Missier et al. [27], Bajoudah et al. [28] create a new design for an (IoT) data marketplace that does not require reports of the participants. Instead, the participants exchange data receipts at regular intervals. First, a producer and consumer create a trade agreement to define the conditions for a trade, whereby both parties need to sign it. Then, the consumer can subscribe to and receive a data stream. After receiving a batch of data, the consumer sends a data receipt to the smart contract to confirm it. If the consumer acknowledges the receipt of the data, the provider continues with the transmission. Otherwise, the provider terminates the trade agreement. Again, the authors mainly focus on data trading and do not consider other necessary functionality to make the data marketplace applicable in the real world.

The authors of [5] propose a three-tier framework for a dynamic and decentralized (IoT) data marketplace. The proposed framework consists of three major participants,

including data providers, data consumers, and brokers. Providers and consumers perform the same role as in most other works about data marketplaces, with the provider supplying the data from (IoT) devices and the consumer buying the published data of interest. The role of the broker is inherited from a highly-resourced device. Its task is to mediate the data trading process by providing registration, discovery, and selection functionality. The broker is responsible for matching buyers with sellers and is encouraged by a fee-based system. Data trading is based on the usage of smart contracts, whereby the participants can negotiate with each other before proceeding with the contract. Further, the concept also incorporates a reputation framework, encrypted data transmissions, and a settlement solution that checks the quality of the trade. While this concept provides very extensive functionalities, it has not been implemented. Partially, we apply the fundamental considerations by [5] in the work at hand.

In [29], the authors propose another (IoT) data marketplace and introduce a mechanism for energy-aware demand selection and allocation. The data marketplace includes an agreement framework, a pricing model, and a rating system implemented as smart contracts on the Ethereum platform. The mechanism for demand selection and allocation enables sellers to optimize their revenue by matching buyer's demands with the seller's (IoT) devices concerning its energy, quality, and allocation constraints. Unfortunately, there is no real distinction between devices and products, and the solution does not provide a user interface.

Ramachandran et al. [30] present a decentralized data marketplace using blockchain technology to facilitate trust and transparency. Product information is stored using the (IPFS), which is a (DFS). The metadata is saved on the blockchain, and the buyer makes use of both the blockchain and the (DFS) to find the desired product. The implementation uses the previously mentioned (SDPP), and after each trade, buyers and sellers can rate each other. Smart contracts on the Ethereum blockchain are used for the registration of the products as well as the rating. The authors make an effort to consider other functionalities that are needed by a data marketplace besides data trading. Yet, the presented solution does not offer negotiations or a user interface.

Another data marketplace based on blockchain technology is proposed by Banerjee and Ruj [31]. The authors describe a blockchain-enabled data marketplace that fulfills fairness, efficiency, security, privacy, and adherence to regulations. In the design of the data marketplace, the blockchain is used as a trusted third party, which ensures these properties. The design of the data marketplace ensures fairness by using a FairSwap protocol, where buyer and seller agree over a certain condition expressed as a boolean circuit. A judge contract is responsible for managing the funds until the buyer agrees to the trade or raises a valid complaint. Transparency, security, and privacy are given through the usage of the distributed public ledger, whereby the data is encrypted and can be verified using zero-knowledge proofs or proofs of misbehavior. Different regulations for data are also described by a predicate to verify if the regulations are followed. The authors also provide suggestions to efficiently define and verify the condition and regulation predicates but mention it as a difficult hurdle to overcome. However, they do not mention how to discover other participants or data products. Furthermore, no other features like price negotiation are offered since the authors have focused on data trading.

In [15], the authors introduce an (IoT) collectability marketplace model to trade collectability of (IoT) devices, whereby collectability refers to the capability of an (IoT) device to collect and transmit real-time data to a specific destination. The model consists of four conceptual layers: the sensor owner, sensing provider, blockchain/marketplace, and data consumer layers. The main responsibility of the blockchain and marketplace layer is to record all transactions and exchanges and provide a reputation system by monitoring the service quality level. However, the marketplace is missing an incentive system to encourage honest behavior of the participants.

Another decentralized marketplace aiming at a machine-to-machine economy for smart cities is proposed in [16]. The marketplace utilizes the IOTA tangle to enable consumers to buy and monitor different data streams. The architecture consists of two main applications: a Google Chrome extension and a program to collect and send the data. The former enables users to interact with the marketplace and buy the data streams. The latter is used to collect sensor data and send it to consumers using (MAM) (a second layer communication protocol). As a result, the performance also depends on the IOTA tangle. Further, the marketplace does not incorporate a rating system.

Zan et al. [17] also designed a decentralized data marketplace for the machine-to-machine economy. Similar to the work in [16], the authors also make use of the IOTA tangle and (MAM) to transfer the data. Additionally, a registration contract on the Ethereum platform is used to maintain a lookup table for the different participants, and product information is stored on the (IPFS). To buy a data product, a consumer has to pay a subscription fee to the purchase contract, which will add it to the consumer list. Further, this approach also makes use of brokers to handle the data trading process and publish the data streams. The marketplace also offers a voting-based refund policy to protect the data providers and consumers from misuse. Again, the performance of the data transfer depends on the IOTA tangle, and the solution does not include a rating system.

A different design of a blockchain-enabled data-sharing platform is described by Sharma et al. [32]. In contrast to other approaches, the authors base the architecture on microservices for better extensibility through modularization and strong system boundaries. Basic services implement the functional requirements, whereas extended services implement non-functional requirements. The data marketplace enables the sale of datasets, which are stored in the (IPFS) and offered through the data marketplace. A smart contract verifies the payment and the hash of the dataset and provides the consumer access to the dataset. Through the extended services, the users can also use other storage options, automatically generate metadata or decide to check the quality of the data. However, it is not possible to trade real-time streaming data.

The authors of [33] propose an (IoT) data marketplace based on blockchain technology in which devices upload datasets to Swarm (a decentralized file transfer system). Consumers can query the smart contract to search for data from specific sensors and pay for it through the usage of payment channels. (IoT) data uploaded to Swarm is encrypted with a symmetric key to prevent unauthorized users from retrieving the data. After purchasing a dataset, the parties exchange the symmetric key, and the consumer can retrieve the data via the specified filehandle. Further, a voting system enables participants to rank the data sources. Unfortunately, this solution also does not consider the sale of real-time streaming data.

In [34], the authors examine the usage of access control policies that are stored and managed on the blockchain to execute access requests for an (IoT) data marketplace. In the course of this, the authors propose a data marketplace and two different data sharing schemes. One of the data sharing schemes is based on (ACL), while the other is based on prefix decryption keys. The data is stored in the cloud which is responsible for the enforcement of the access policy. Basic data trading functions to create and verify metadata and to create and accept offers are implemented through smart contracts. While this work offers two interesting data sharing schemes, it is also not suitable for real-time streaming data.

Badreddine et al. [35] propose a real-time (IoT) data-sharing framework by combining the (MQTT) protocol with Ethereum smart contracts. The smart contracts automatically calculate the bills based on the proposed monetization approach, which are paid for by the subscribers with the native cryptocurrency. Further, the authors present three blockchain-based traceability solutions to record publish and subscribe operations. These solutions differ in the level of traceability, whereby the lowest level enables only unreliable verification, and the highest level incurs very high gas costs. The framework mainly relies on tracking data and does not apply a rating mechanism. Further, it does not provide a mechanism to search for specific products.

In [36], the authors present a privacy-aware data marketplace that provides data privacy, output verifiability, and atomicity of payments. The model of the data marketplaces consists of data generators, data brokers, and data consumers. Hereby, data brokers pay data generators for their data, compute functions over the data and sell it to data consumers. By using multi-client functional encryption, the data marketplace ensures that data generators are not able to learn any information about the transmitted data. Further, with the application of (ZKP), the data marketplace ensures that data generators can not alter the data, and consumers can verify the received result. While this work provides an interesting solution to provide the above-mentioned properties, it does not allow dynamic changes in the set of data generators and focuses on the privacy aspect of data marketplaces.

Meijers et al. [37] propose another blockchain-based IoT data marketplace that uses an optimized data trading mechanism to lower the costs by reducing the number of transactions. Consumers and producers agree off-chain on the conditions of trade. A consumer has to return the provider a receipt to acknowledge that it received the data. The provider can provide a receipt to a smart contract that escrows the funds for the trade. Additionally, the data marketplace applies a credit system where producers grant consumers credit to reduce the costs. This credit allows the producer to send data for a longer period without the consumer having to make a new deposit. As the authors already mentioned, the work's main goal is to present a blockchain-based IoT data trading mechanism. Other functionalities would still need to be added to make it a full-fledged data marketplace.

As it can be seen, several different data marketplace solutions have been proposed. However, only very few consider a broad range of functionality that would make the data marketplace applicable in real-world scenarios. Most of these solutions focus too much on data trading and neglect other important aspects. Although this is the main element of a data marketplace, it is not possible to operate a data marketplace that solely offers data trading. Nevertheless, these works offer approaches that are helpful for the development of a decentralized data marketplace and, therefore, should be considered more closely. However, the consideration of other functionality (see Sect. 4) is very important because only the interplay of these enables a data marketplace. A marketplace has to offer user management, device management, product management, negotiation, discovery, selection, routing, settlement, rating, monitoring, and web access. Furthermore, when designing the data marketplace, particular attention must be paid to the fact that the (IoT) has different characteristics than the Internet,

which must be reflected in the design. Therefore, we create a blockchain-based (IoT) data marketplace that incorporates a wide range of functionalities and takes the shortcomings of already existing solutions into account.

4 Requirements

In this section, we identify the requirements to be fulfilled by a blockchain-based IoT data marketplace. We start with a description of the different user classes (see Sect. 4.1) and the operating environment (see Sect. 4.2). Further, we define the design and implementation rules in Sect. 4.3. After that, we discuss the essential system features with their corresponding functional requirements (see Sect. 4.4), followed by the non-functional requirements (see Sect. 4.5) of the data marketplace.

4.1 User Classes and Characteristics

The user classes define the different participants of the data marketplace and their inherent characteristics. These user classes have already been mentioned in other works about decentralized data marketplaces [5, 27] and include data providers, consumers, and brokers. Providers and consumers are sometimes also called buyers and sellers and are identified in nearly every process that involves the trading of data, either in the form of concrete devices [18, 19] or abstract entities [23]:

Provider A provider is an owner of (IoT) devices, i.e., data sources. Owners of (IoT) devices could be individuals, small organizations, or large companies which use the data marketplace to incorporate their devices in the data marketplace to sell data generated by these devices. The main incentive of the provider to offer its data on the data marketplace is to increase profits by providing data that is otherwise uncollected or unused.

Consumer A consumer is an owner of devices that are capable of buying and receiving data on the data marketplace, i.e., data sinks. Same as with providers, the owners of these devices could be individuals, small organizations, or large companies which use the data marketplace to purchase specific data. The consumer is interested in finding data that can be used to improve quality, enhance user experience, or maximize the revenue of their products. Notably, a particular consumer could also be a data provider in a different application and vice versa.

Broker A broker is the owner of software and hardware infrastructure, which uses the data marketplace to offer its services as a middleman for data trading. The broker is interested in making a profit by providing resources to facilitate the transmission of bought data. A broker strives to facilitate as many data trades as possible to get a reward for its services, while producers and consumers can freely choose which broker they want to use. Brokers are thus easily interchangeable and should strive to provide a good quality of service and behave honestly.

4.2 Operating Environment

Before starting with the design and the implementation, it is important to determine in which environment the data marketplace is going to be operated. Otherwise, it might turn out in the end that the design is not suitable or the implementation is not appropriate for the environment and therefore can not be used. To be sure that the individual components of the (IoT) data marketplace can be operated without problems, great importance should be given to the backing technologies of (IoT) devices.

Further, the data marketplace uses a smart contract and decentralized application platform to implement specific functionalities. As a result, the data marketplace depends on this external system. It must be considered that future changes to the smart contract platform will have a direct impact on the data marketplace. The security, performance, and scalability of the platform will influence the operation of the data marketplace. The following requirements and dependency regarding the environment have to be considered:

$OE-1$ The data marketplace shall consider current technologies used by (IoT) devices.

$OE-2$ The data marketplace shall take into account the available resources of (IoT) devices.

$OE-3$ The used smart contract and decentralized application platform shall be reliable with good support and tools.

$OE-4$ The operation of the data marketplace depends on the used smart contract and decentralized application platform regarding security, performance and scalability.

4.3 Design and Implementation Constraints

This work aims to find out how far blockchain technology is suitable for the implementation of a decentralized (IoT)

data marketplace. The characteristics of the (IoT), as well as the advantages and disadvantages of traditional and blockchain-based data marketplaces, have already been discussed in Sects. 2 and 3 and provide the reasoning behind the exploration of blockchain technology for this purpose. We want to use blockchain technology not only as a payment system but also to realize more complex functionalities via the execution of smart contracts. For this purpose, the following design and implementation constraints can be stated:

$CO-1$ The architecture of the (IoT) data marketplace shall follow a decentralized approach by exploring blockchain technology.

$CO-2$ The data marketplace shall use smart contracts to implement data trading.

4.4 System features

The system features describe the main features of the data marketplace and their functional requirements. For the data marketplace, essential system features and their corresponding functional requirements are derived by looking at current solutions for (centralized and decentralized) data marketplaces.

4.4.1 User management

A provider, consumer, or broker needs to be registered to become part of the data marketplace [6, 31]. Since we are using a smart contract platform, respectively blockchain technology, to implement the data marketplace, each participant of the data marketplace must have an account for the chosen smart contract platform. The marketplace may also need more information (e.g., personal information) about the participant, which also requires a user profile.

4.4.2 Device management

Further, providers must be able to manage their devices that are the actual sources of the data to be traded on the data marketplace. Each provider can have multiple devices through which it generates different data. With this, the provider can then proceed to create various products, i.e., data streams offered by the registered devices. The distinction between products and devices makes sense with the assumption that devices can offer several different products, and not every consumer is interested in the whole batch of information generated by a single device. In addition, consumers get more detailed information about the device generating the data, as this can be added by the provider when the device is created. It also allows consumers to search for products generated by a particular device.

4.4.3 Product management

Product management enables providers to take care of their products offered on the data marketplace. A provider can create several products which can be bought by consumers, whereby every product is generated by a specific device. Also, a provider might want to update the interval, i.e., the frequency in which new data is generated because the mode of operation of the device has changed, or to adjust the price to respond to market fluctuations. If the provider is no longer willing to continue to offer the product or the device is no longer operated, it should be possible to delete the product from the data marketplace. The removal of products that are no longer offered reduces unnecessary interactions for potential consumers, which have to deal with outdated information.

4.4.4 Negotiation

While most existing data marketplaces focus on fixed pricing models, e.g. [23, 27], we also want to facilitate price negotiations between data producers and consumers. The opportunity to negotiate with each other makes it possible for the participants to represent their interests and thus identify better deals between the negotiating parties. For this, the buyer and seller should be able to directly communicate and engage in a negotiation process to agree on an offer.

4.4.5 Settlement

A trade is concluded with the settlement. Both parties have to agree on the payment and the conditions of the trade. The seller would like to charge the buyer for the transferred data, and the buyer must be able to pay the seller. Additionally, the protection of the participants engaging in a data trade is an important issue that needs to be considered. The participants may fail to abide by the conditions of the trade, be it intentional or unintentional false behavior.

4.4.6 Routing and transmission

Another essential part of a data marketplace is how the data is routed and transmitted. The buyer must be able to receive the purchased product. In the literature, there are different approaches on how to realize the communication between buyers and sellers. Some solutions rely on direct communication between the participants, e.g. [5], while others, e.g. [27], rely on brokers to forward the data to the

buyer. Without a reasonable exchange of data, no trade between contractors can be handled.

4.4.7 Rating

A rating or reputation system is extremely common in today's data marketplaces [5, 27, 30]. It enables participants of the data marketplace to assess the reliability of other participants and the quality of the offered products. Further, it encourages honest behavior since participants want to keep their rating as high as possible to be appealing to potential trading partners. Participants with a good rating may be very hesitant to interact with others that have a bad rating.

4.4.8 Monitoring

Monitoring provides a data marketplace with the ability to track certain metrics, which can be used to compare and rate its participants and products. The most frequent metric tracked by current solutions of data marketplaces is the number of data transfers between users [5]. The monitoring ability lays out the basis for the protection of providers and consumers since it can be determined whether both parties agree on the quality of the trade by comparing the recorded metrics, and if this is not the case, a dispute can be lodged.

4.4.9 Discovery and selection

The discovery and selection of products is an important part for every consumer. Consumers need to find the right products that satisfy their needs and select them for purchase. Consumers should be able to query the data marketplace, which then matches the products to the query and returns the positive matches to the consumer. Therefore, every product should provide the necessary metadata to facilitate this process. This metadata provides more precise information about the product, enabling a targeted search to satisfy the customer's needs.

4.4.10 Web access

Finally, the participants shall also be able to interact with the data marketplace through user-friendly interfaces. While there are many data marketplaces, many of them neglect the necessity of a user-friendly interface to make the data marketplace applicable. It provides users with the ability to browse the data marketplace and make use of its different features. For example, [6] provide access to the data marketplace through the usage of WS-* or (REST) Web services, which enable the registration of devices and consumers.

Summing up the discussion, we identify the following functional requirements:

- $FR-1$ The user shall be able to manage a profile on the data marketplace.
- $FR-2$ The user shall be able to manage (IoT) devices on the data marketplace.
- $FR-3$ The user shall be able to manage data products on the data marketplace.
- $FR-4$ The user shall be able to negotiate the terms of a trade.
- $FR-5$ The user shall be able to settle a trade.
- $FR-6$ The data marketplace shall be able to transmit and route the product from the provider to the consumer.
- $FR-7$ The data marketplace shall be able to calculate and store the ratings of every user.
- $FR-8$ The data marketplace shall be able to track and store certain metrics for every trade.
- $FR-9$ The user shall be able to discover products by providing a search query.
- $FR-10$ The data marketplace shall provide a Web interface for its various functionalities.

4.5 Non-functional requirements

Apart from the functional requirements discussed in Sect. 4.4, it is also necessary to define the non-functional requirements of the data marketplace: Transparency, privacy, and security are major aspects that have to be considered within the design of a data marketplace [31]. A transparent data marketplace prevents the unlawful exchange of data by providing a public log that records the transmission of ownership of data traded through the system. Additional details of the data trade are captured and used later on to make the participants accountable for their actions. Furthermore, secure transmissions improve the privacy and security of the system since third parties that are not involved in the specific data exchange are not able to access the data. [5] also mention the use of symmetric encryption to ensure the integrity of the data.

The performance of the data marketplace must be taken into account so that the solution can also be used in a realistic scenario. Among other things, attention must be paid to how the data is transmitted. If the data exchange takes place via a blockchain, the transmission speed is bound to the transaction speed of the used blockchain. For example, [24] use a blockchain for the (SPS) architecture, whereby publishers send events to the blockchain, and subscribers receive notifications from the blockchain. Furthermore, many works also deal with fairness and, therefore, the credibility of the system [5, 6, 27]. Fairness

is ensured through the use of a rating system or tracking of all data transfers.

Summing up the discussion in this subsection, we identify the following non-functional requirements to be considered in the design of our blockchain-based data marketplace:

NFR − 1 The performance of data transmissions shall not be restricted by the smart contract platform.

NFR − 2 The data marketplace shall encrypt the data transmission to ensure privacy.

NFR − 3 The data marketplace shall ensure fairness between users.

5 Design

After defining the requirements for a blockchain-based (IoT) data marketplace, we proceed with the design. We explain the architecture by discussing the overall architecture and follow up with a detailed explanation of the system design.

5.1 Overview

The design of the data marketplace is based on a three-tier architecture (see Fig. 3), whereby each tier is assigned to a specific user group (i.e., providers, consumers, and brokers, as discussed in Sect. 4.1). The components are independent of each other and operate in different environments and on different platforms. This contributes to better scalability, fault tolerance, flexibility, interchangeability, and loose coupling of the components, which ensures good interchangeability.

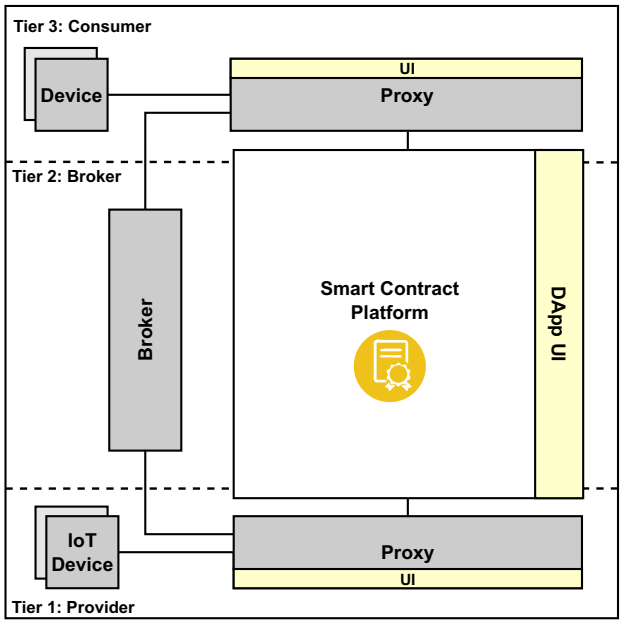

Fig. 3 Conceptual three-tier architecture for the blockchain-based (IoT) data marketplace

The first tier defines the components for providers. It consists of the (IoT) devices from the different providers which are generating the data that should ultimately be sold and purchased. Further, the first tier includes the proxies, which act as the link between the first tier and the second tier. The proxies enable the communication between (IoT) devices, the (DApp) on the smart contract platform, and brokers. Furthermore, a (GUI) allows the provider to interact with a proxy and the (DApp).

The operators of brokers and their corresponding components are represented in the second tier of the data marketplace. Further, the second tier includes the smart contract platform with its peer-to-peer network. The second tier facilitates the data trading process between providers and consumers by providing the infrastructure and (DApp) for the data marketplace.

The third tier includes the components for data consumers. This layer forms the counterpart to the first layer, whereby devices from consumers want to purchase data. It also includes the proxies acting as the link between the second tier and the third tier. Same as the provider, the consumer uses a (GUI) to interact with the proxies and the (DApp).

5.1.1 Proxy

A proxy enables providers and consumers to integrate their (IoT) devices with very few computational resources in the decentralized (IoT) data marketplace. A provider respectively consumer can operate one or more proxies, depending on how far its (IoT) devices are distributed. This possibility ensures that (IoT) devices that are geographically not close enough to each other do not need to use the same proxy, which would result in inefficient communication. The most important task of a proxy is to represent the underlying (IoT) devices in all matters viable, so they have to fulfill as few resource-intensive tasks as possible. (IoT) devices usually have quite limited computational or energy resources [38]. This lack of resources makes it very difficult to implement various functionalities necessary for participation in the data marketplace.

(IoT) devices would have to carry out a high number of cryptographic operations. Providers and consumers need to encrypt the data to ensure privacy, confidentiality, and integrity of the data. It is particularly problematic if the (IoT) devices need to maintain multiple encrypted connections simultaneously, whereby multiple connections alone are already stressful enough. Further, some

interactions with the smart contract platform require the (IoT) device to sign a transaction.

If the (IoT) devices have enough resources, it would also be possible to communicate directly with the brokers and the smart contract platform, i.e., no proxies are necessary. However, a proxy also brings benefits in terms of security. Instead of allowing direct access to the (IoT) devices, the proxy facilitates increased security by hiding the infrastructure behind the proxy. Therefore, it is not possible to access the (IoT) devices directly from outside the network.

5.1.2 Broker

A broker facilitates the data trading process and is responsible for highly resource-intensive tasks. Further, brokers act as mediators between producers and consumers if they have a dispute during a trade. Brokers have all the details about the traffic and, therefore, can resolve a dispute.

The device on which a broker is operated should have enough computational resources. An incentive mechanism ensures that a broker is rewarded for providing its resources since more computing power also means higher costs. Therefore, not only the provider is paid for the sale of its data but also the broker who has made its computing power available. This reward gives owners of hardware infrastructures an incentive to run a broker.

Since a proxy is operated directly by the provider or consumer, it is also advantageous to relocate the heaviest and most resource-intensive tasks to a broker. These tasks include the management of numerous concurrent connections and product discovery. The more participants the data marketplace has, the more connections have to be managed by (IoT) devices or the proxy, whereby the number of connections can become very high. A broker acts as a middleman between providers and consumers and forwards the messages to the right consumer. A broker also prevents message loss if a consumer is offline or can not receive the messages.

Also, the discovery of products listed on the decentralized (IoT) data marketplace is particularly complex, as the number of them can become very large. Consumers need to find products that meet their requirements. Checking these criteria must be done for each product listed on the data marketplace, which makes it a very resource-intensive task and is therefore also taken over by the broker.

Monitoring the data transfer is also an important responsibility of a broker. The recorded metrics enable a broker to determine if the provider or consumer behaves honestly. A broker knows both what data the provider has sent and which data the consumer has received. This assessment works only as long as at least two parties behave honestly. This approach is not enough to ensure fairness if a provider or a consumer cooperates with a broker to cheat the other party. However, brokers are encouraged to behave honestly to maintain their reputation.

5.1.3 IoT device

The (IoT) device is a context-aware device that can store and process data, communicate with other entities over standardized protocols, and is uniquely addressable. (IoT) devices can be smart devices, sensors, or similar objects, which among other things, can record various physical or chemical properties of their surroundings.

However, these devices mostly do not have a lot of computational or energy resources and storage capacity, which has to be considered in the design of a blockchain-based (IoT) data marketplace. Therefore, as pointed out above, the (IoT) devices "offload" certain functionalities to brokers and proxies. The (IoT) devices use a proxy to use the functionalities of the data marketplace and communicate through the proxy to invoke functions of smart contracts or transmit or receive data from a broker to buy or sell products on the data marketplace.

5.1.4 Graphical user interfaces

The users of the data marketplace can interact with the (DApp) through a user-friendly interface. The (DApp) UI furnishes the possibility to use the various functionalities provided by the smart contracts of the data marketplace. Through the (DApp) UI, users register themselves to be able to participate in the data marketplace. Furthermore, the (DApp) UI offers users the possibility to browse the data marketplace and view the different users, devices, and products. Further, users can interact with the (DApp) UI to manage their profiles and devices. The proxy has an additional (GUI), which allows a user to configure the different wallets for their (IoT) devices.

5.1.5 Smart contracts

The decentralized (IoT) data marketplace uses different smart contracts deployed on a smart contract and decentralized application platform to realize specific functionalities and enforce the rules of the data marketplace.

To participate in the system, users have to register themselves and provide personal information for their user profile. The user profile enables the possibility to get the contact details of other users and the necessary information to generate invoices. The *user contract* is responsible for managing the user profiles storage and keeps the personal information about individuals, small organizations, and large companies on the data marketplace. This smart contract provides the means to create new user profiles, and it

gives users the ability to update their profiles to keep their personal information up-to-date. Additionally, this contract makes it possible to retrieve the profile of specific users. If a user no longer wishes to use the system, existing profiles can also be deleted. Since blockchains do not support the deletion of data, we will take a closer look at this process later (see Sect. 6.1).

Another smart contract is responsible for device management (*device contract*). Users need to be able to manage the devices that generate the different data products to be offered on the marketplace. The information provided about the devices gives other users the possibility to get more detailed information about the origin of the product and the reliability of the device. Through the usage of the device contract, a user can add new devices. Furthermore, the smart contract allows a user to update and delete devices. It is important that devices can be found on the data marketplace, which is also accomplished through the usage of the device contract.

As with users and devices, another smart contract is needed to manage the data products offered by providers. The *product contract* manages all products of the data marketplace and gives more information about the products being offered for sale. Like the smart contracts which have already been discussed before, it also offers the possibility to carry out create, read, update and delete operations.

Further, a *broker contract* provides the possibility to register instances of brokers on the decentralized (IoT) data marketplace and gives further information about them. It is necessary to communicate with a broker to use its functionalities which is only possible if the broker can be found by providers and consumers. Therefore the broker contract enables the discovery of different brokers based on certain criteria, such as their location.

Another smart contract is used to realize a rating system to rate devices and brokers. The *rating contract* is responsible for the calculation and storage of the rating. The rating system provides participants with the possibility to obtain an assessment of the reliability of other participants.

The data trading process is also realized through the usage of smart contracts. The *negotiation contract* provides the participants with the possibility to start negotiations with each other. It is responsible for managing the *bidding contracts* whereby a bidding contract enables providers and consumers to exchange bids and creates a transparent view of the negotiation process. Both parties can see how an agreement or no agreement was reached. Furthermore, the *trading contract* is responsible for managing all trades. Each trade is assigned a *settlement contract*, which acts as a conditional escrow and facilitates the settlement process. Further information on the implemented smart contracts is provided in Sect. 6.1.

5.2 System design

After explaining the architecture and the various components that are part of the blockchain-based (IoT) data marketplace, we now elucidate the actions taken by each component and how they interact with each other.

5.2.1 Entity management

Initially, a user creates a profile on the data marketplace. Managing the user information requires only very few interactions with the user contract. The (GUI) provides the user with the possibility to send transactions to the user contract to invoke the create, find, update and delete operations offered by the smart contract. For this purpose, the user must have an account on the smart contract and decentralized application platform, which can be used by the (GUI) to interact with the smart contract. Further, the user contract ensures that users can only manage their profile that is associated with their private key. The deletion does not completely remove the data but rather only removes it from the current state of the smart contract as it stays part of the blockchain's history.

Once a user has created a profile, it is possible to use all functionalities of the device contract. The device contract offers the functionalities to create, read, update and delete devices. For this, too, only a transaction has to be sent to the smart contract, as is made possible by the (GUI). Further, devices must also be registered on the proxy since the proxy must be able to authenticate the various devices to send transactions to the smart contract platform on their behalf. The proxy manages a separate key pair for each device, which can only be used if the device is successfully authenticated. If a device can carry out the necessary operations itself and has direct access to the smart contract platform, the detour via the proxy does not have to take place.

Each user that wants to sell data can use the product contract to list new data products on the blockchain-based (IoT) data marketplace. The created products can also be updated, deleted, and searched, and the smart contract ensures that a user can only manage its products. To use these product management functionalities of the data marketplace, users also only have to send a corresponding transaction to the smart contract platform, which then carries out the respective operation.

5.2.2 Discovery and selection

The discovery of products (see Fig. 4) is a very resource-intensive task facilitated by a broker since the broker is usually operated on a highly resourced device. To search

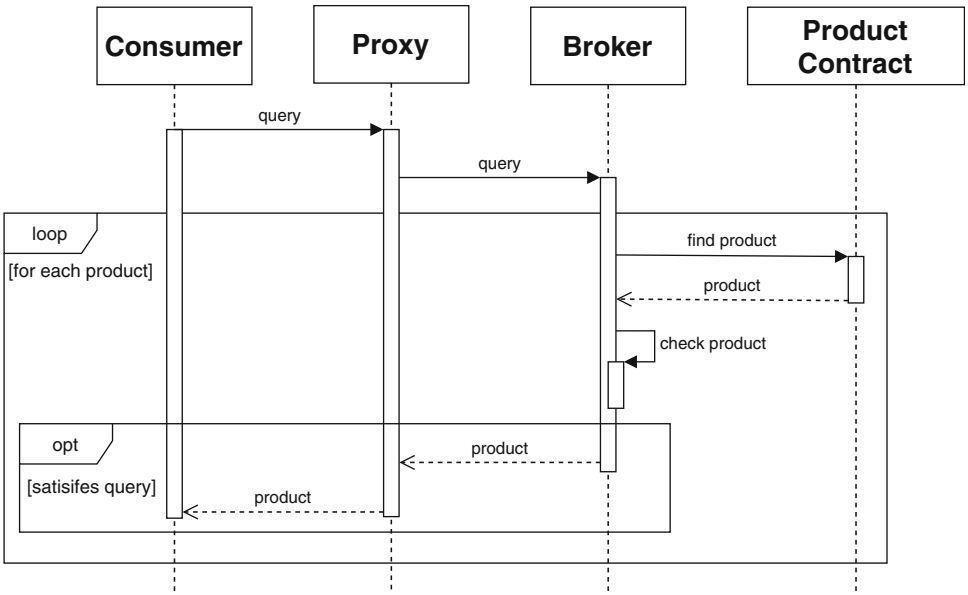

Fig. 4 The process of searching for a product on the data marketplace

for a product, the consumer sends a search query to the proxy, which is responsible for forwarding this request to a broker. Since several brokers can be considered, the consumer can also use the proxy to search for a suitable broker. The consumer can decide which broker is most suitable based on several criteria, e.g., the geographical location of the broker to reduce latency. The search process for a broker is the same as for a product, only that it is carried out directly by the proxy.

After the broker has received the search query of a proxy, it uses the product contract to iterate over all products listed on the data marketplace. Here, the broker examines each product according to the desired properties specified in the query. After the broker has checked whether the product satisfies the query, it sends the product back in response to the proxy, which forwards the result to the consumer. The consumer can use this result to select a product that meets his previously defined properties.

5.2.3 Negotiation

The negotiation process is always initiated by the consumer who wishes to buy a particular product. For this purpose, the consumer sends a transaction to the negotiation contract to request a negotiation with the provider of the product. The smart contract ensures that only the provider of the product can respond to the request, which can either accept or reject the request. If the provider accepts the request, a new bidding contract will be created.

After the creation of the bidding contract, both parties will be able to use this contract to negotiate the terms of the trade. The consumer uses the smart contract to submit the first bid. The provider is informed about the bid and has two ways to respond. Should the provider be satisfied with the bid, it can accept the offer directly. If there is any further need for negotiation, the provider can submit a counter-bid, via which the consumer will also be informed. The consumer now has the same options as the provider to respond to the counter-bid. Either the consumer accepts the counter-bid or creates a new counter-bid. This process can be repeated until the two parties have agreed on the terms of the data trade. As soon as an offer is accepted, the provider takes over the creation of the trade through the trading contract. Here, all conditions of the trade are determined and who is involved in it. Creating the trade also generates a new settlement contract which is used for the settlement of the trade (see Sect. 5.2.5). After the creation of the trade, both provider and consumer are notified so that they can continue with the data trading process.

If the consumer wants to buy the product directly, no negotiation is necessary. In this case, the trading contract offers the possibility to create a trading request for a specific product, which can be accepted or declined by the provider. If the provider accepts the request, a trade will be created with the price specified in the product contract. The advantage here is that no additional fees arise from a negotiation.

5.2.4 Routing and transmission

After the consumer has paid for the product, the provider receives a notification, and the data transfer can start. To encrypt the data, the provider needs to get the consumer's public key from the device contract. After that, the provider sends the data and the key to the proxy. The data is encrypted with the key and sent to the responsible broker.

After the broker has confirmed that it has received the data, the provider increments its counter that records the number of transmitted data packets. The broker also manages a counter, which indicates how much data has been received. After paying for the product, the consumer starts to pull the data from the broker. The data is also routed through the proxy, which decrypts the data and sends it to the consumer. After receiving the data, the consumer also increases its counter. This process is repeated until the trade expires, which is determined at the time of purchase. If the consumer desires an extension, it must renegotiate with the provider.

5.2.5 Settlement

The settlement process is started before the data is exchanged. The process starts with the payment of the product by the consumer. The reason for this is to protect the provider from sending the data to a consumer who can not pay for the product. On the other hand, the consumer is also protected against malicious providers since the settlement contract acts as a conditional escrow, which ensures that the funds are only transferred if certain conditions are met. After the data transmission, the provider and consumer send a transaction to the settlement contract to settle the trade by disclosing their counters. These counters are used by the settlement contract as a basis for deciding if a dispute should be lodged after receiving the last counter. If the counters match, the funds are transferred to the involved parties. Otherwise, the broker will be informed of a dispute, which will then react to it by sending a transaction with its counter to the settlement contract, which will dissolve the dispute and then transfer the money.

5.2.6 Rating

The data marketplace uses a rating system to reward honesty and penalize misbehaving participants. The rating process is initiated by a settlement contract which calls the rating contract. Hereby, the rating contract ensures that a settlement contract can only rate the participants involved in the corresponding trade. After the contract has ensured that the settlement contract is authorized to carry out a rating, it asks the device contract for the current rating of the provider or consumer. The rating is calculated and then set by using the device contract.

6 Implementation

In this section, we examine how the blockchain-based (IoT) data marketplace is implemented. For this, we discuss the used technologies as well as the implementation of the individual components, which is available as open-source software on GitHub.[1]

6.1 Smart contracts

The smart contracts are implemented for the Ethereum smart contract and decentralized application platform using Solidity. We selected Ethereum since it is these days the most-applied second generation blockchain, and a rich set of tools is available for it. A disadvantage is that Ethereum entails very high costs for the execution of smart contracts. However, the proposed solution is portable to other public blockchains as long as they provide the necessary smart contract capabilities.

6.1.1 Data storage

The main task of the user, device, broker, and product contract is to provide (CRUD) operations for the respective entities that are to be saved via the smart contracts. But also, the negotiation and trading contract must at least provide create and read operations. Solidity provides two data structures for storing collections. The first option would be to save the data in an array of fixed or dynamic lengths. However, the data can only be accessed through the index, which is sometimes not sufficient. The other option would be to save the data in a mapping, which enables random access through a key.

A mapping offers the opportunity to efficiently read specific entries from the smart contract, which is essential functionality for the data marketplace. However, this advantage also comes with the disadvantage that Solidity does not offer the possibility to iterate over the keys and entries of mappings, and there is no possibility to determine the number of entries in a mapping. To compensate for this, we store an index for the mapping, which is a dynamic array that stores all keys of the mapping separately. The number of entries in the array can be easily determined, and thus also the number of entries in the mapping. It is also possible to iterate over arrays, which indirectly makes it possible to iterate over the mapping since the keys of the mapping that are stored in the array can be used to directly access the entries of the mapping. The combination of a mapping and an array, therefore, offers the basic requirements for managing the data.

[1] https://github.com/soberm/data_marketplace

Creating a new user, device, broker, or any other entity that needs to be persisted through a smart contract requires the smart contract only to store the entity in the mapping with its ID as the key and add the key to the index. Furthermore, the necessary references to other entities have to be set by calling the corresponding function of the smart contract, which manages the other entity. For example, when creating a device, the ID of the device must be added to the user, which is saved via the user contract. Smart contracts provide the necessary functions to set these relationships between entities. When updating an entity, the smart contract first performs a look-up with the key and then updates the entity. With *findById*, *findByIndex*, and *count* functions, a smart contract provides the necessary interface to retrieve specific entries or iterate over all entries stored in the mapping of the smart contract. To iterate over all entries, the caller gets the size of the index array and retrieves each entry one by one by providing the index. Deleting entries is only possible for users, brokers, devices, and products, whereby only a soft delete is performed by setting a deleted flag. Furthermore, delete operations are cascaded, e.g., deleting a user means that the relevant brokers, devices, and products are also deleted.

6.1.2 Authentication and authorization

Authentication and authorization is an important factor that was taken into account when implementing the smart contracts, as these are public and accessible to everyone. Therefore, for all functions of the smart contracts that change the state, the sender's address is first used to check whether the account is authorized to carry out this action, or it is ensured directly via the code that an account can only manage its data. It was also necessary to consider how the smart contracts are linked to each other dynamically and during deployment. The addresses of settlement contracts, for example, must also be saved in the rating contract so that it can ensure that only these contracts can change the rating. It is not possible to replace the implementations of the smart contracts after the deployment because the addresses can only be set once in the corresponding contracts. This has the advantage that no central authority can swap the implementation in its favor after the deployment, but it is also not possible to carry out an update.

6.1.3 Data trading

In Sect. 5, we have already discussed smart contract-based data trading. Now it is time to take a closer look at the implementation of the individual components which are necessary for this.

The negotiation contract stores all negotiation requests and negotiations. To request a negotiation, the consumer only needs to call the smart contract's request negotiation function, specifying the ID of the desired data product. In contrast, the provider can use the accept negotiation function of the smart contract to start the negotiation process. When the function is called, a negotiation is saved on the blockchain, and a new bidding contract is deployed. The negotiation object specifies the consumer, the bidding contract, and the subject of the negotiation, i.e., the product.

After that, the consumer can proceed with creating a bid through the previously deployed bidding contract and call the make bid function stating the bid, which includes the price as well as the start and end time of the data stream. The same is also possible for the provider, so that offers can be exchanged. It further provides functions to accept and cancel the bidding process, which ensures that no new actions are possible apart from retrieving data from the smart contract. The smart contract regulates that both parties can only alternately call functions of the smart contract. The last saved bid of a bidding contract in which the accept function was called thus represents the offer that has been agreed upon.

Purchase without negotiation takes place directly via the trading contract by calling the request trading function, whereby the consumer specifies which product it wants to buy, the broker that facilitates the trade, and the start and end time of the data stream. The request is then stored on the blockchain, whereby the consumer automatically accepts the specified price by the provider.

Subsequently, the provider has two options to create a trade. For this, it can either call the accept trading request function or create function of the trading contract. In the first variant, the provider must specify the trading request which should be accepted, whereby the trading request stores the conditions of the trade. In the second variant, the provider specifies the negotiation and the broker, whereby the conditions agreed on in the bidding contract are used for the trade. In both cases, the smart contract enforces that a trade can only be created with the conditions that both partners have agreed on. When a trade is created, it is saved in the trading contract and stores the conditions of the trade and the address of the settlement contract.

The settlement contract provides the consumer with a deposit function to send Ether to the contract and ensures that a correspondingly high amount is paid to cover the costs of the trade. Then, both provider and consumer can start exchanging data. After the transfer has been completed, both provider and consumer call the settle trade function of the contract and provide their counters (see Sect. 5.2.4). When the function is first called, only the caller's counter is saved, and the second call compares the

two counters and continues with the settlement. If both counters match, the counter is used to calculate how much money is paid out to the provider, broker, and consumer. Then, the funds are transferred to the corresponding parties, and the trade is considered settled.

However, since some problems can arise, such as that the provider and consumer disagree or do not continue the process, certain exceptions must be considered. If the counters do not match, a dispute is created, which the broker must resolve. For this, the smart contract offers the settle dispute function, which can only be used by the broker after a dispute has occurred, whereby the broker determines the value of the counter. Since it cannot be ruled out that one of the two parties does not call the settle trade function, a resolve timeout function has been added to the settlement contract. If one of the parties does not call the settle trade function up to a certain point in time (e.g., 24 hours after the actual end of the trade), the party that behaved as expected can withdraw the funds from the contract. If the settle trade function is not called from either party, the funds are sent back to the consumer.

At the end of the settlement process, the settlement contract calls the rating contract to evaluate the involved parties. The rating process is kept very simple since the focus is more on ensuring that only the settlement contract is authorized to rate a user, as has already been discussed in Sect. 6.1.2. Nevertheless, the contract provides two functions for the settlement contract to rate the involved parties of a trade. One calculates and sets the rating in the case of a successful settlement and the other one if a dispute occurred. In the first case, the user's rating increases, whereby in the second case, it decreases. A user's rating is represented by a numerical value between zero and 100, with 100 being the best and zero the worst. In future work, more complex calculations of the rating could be considered, but currently, Solidity does not support fixed-point numbers, which makes complex calculations difficult.

6.2 Services

Both the proxy and the broker were implemented in the Go[2] programming language. The proxy and the broker communicate through the usage of (gRPC). Further, the services use the Go Ethereum client for the Ethereum (RPC) (API). A Go binding generator was used that generates the corresponding counterparts of the smart contracts in Go to avoid a lot of boilerplate code.

6.2.1 Proxy

The proxy implements multiple (gRPC) services. These services include one service for each smart contract, which enables communication with the smart contracts deployed on the blockchain using a proxy account with its corresponding wallet. The accounts with their wallets can be managed by an administrator through the account service and the wallet service. Furthermore, the proxy provides a discovery service through which brokers and products can be searched. An authentication service enables the users of the proxy to authenticate themselves, and a message service is responsible for the encrypted transmission of the traded data.

The data of the accounts and wallets are stored in an SQLite database. An account consists of a username, password, and role (admin or user), whereby bcrypt is used to only store the hash value of the password in the database. A wallet can be added to each account, with one wallet consisting of the Ethereum address, public key, passphrase, and the path to the key file. The key files are stored in a separate folder on disk, whereby every file is encrypted with the passphrase specified for the wallet.

After accounts have been created for the users, they can authenticate themselves using their username and password and receive a (JWT) in exchange. In addition to the standard claims, the user ID and the role of the user are also encoded in the token. All services except the authentication service require that a valid (JWT) is present in the authorization header of every request.

Calling smart contracts with the help of contract services works for every smart contract according to the same principle (see Fig. 5). The provider or consumer issues a unary call to the corresponding contract service with the (JWT) included in the header. The proxy selects the appropriate wallet based on the provided token and unlocks the key file. After that, a transaction signer is created, which is passed to the contract bindings that use it to create the transaction. After sending the transaction to the smart contract, the account gets locked again. This enables the unencrypted key to be kept in memory only as long as necessary to create the transaction. Events can also be listened to via the various contract services and are streamed to the caller.

The discovery service provides the user with the possibility to search for brokers and products by providing a search query (see Fig. 4). Brokers can be filtered by name and location and products by data type, frequency, and cost. The blockchain is scanned for brokers directly by the proxy, whereas a product query is forwarded to a broker that was specified by the user in the search query. In both cases, the filtered results are streamed to the caller.

[2] https://golang.org/

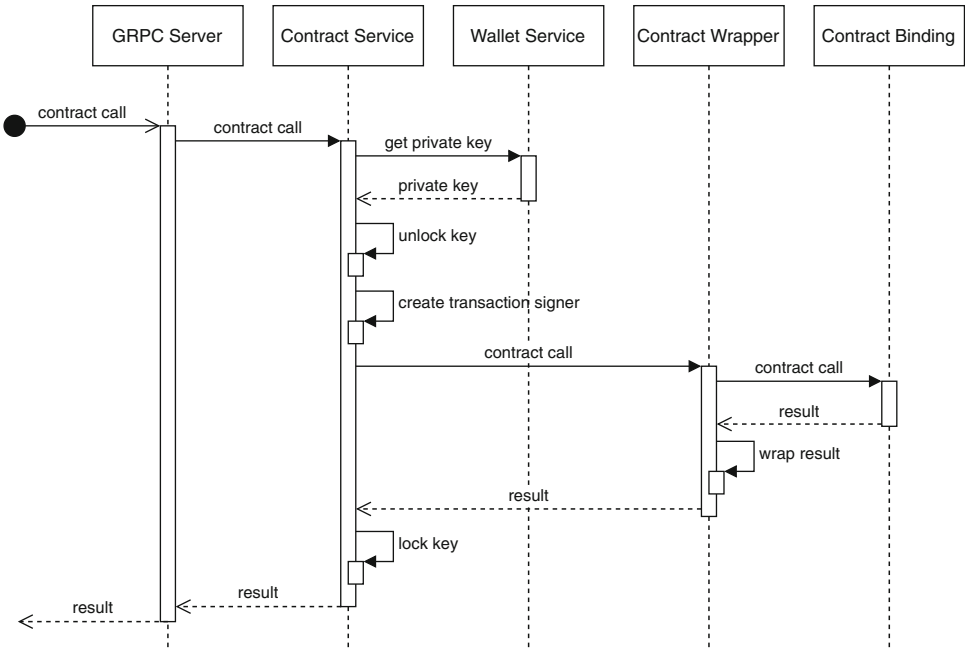

Fig. 5 Sequence of interactions during a contract call through the proxy

Messages that are to be transferred from providers to consumers are encrypted using the crypto message service and forwarded to the specified broker. The public or private key of the respective Ethereum account is used to encrypt and decrypt the data using the (ECIES). This means that no additional key pair has to be created and no key exchange has to take place, since the public key of the consumer can be read directly from the device contract. The messages are pushed unencrypted to the proxy, which uses the public key of the consumer to encrypt the payload. The consumer pulls the message through a proxy, which again uses the token to determine the appropriate key file and uses the account's private key to decrypt the payload.

6.2.2 Broker

A broker implements three services and makes two of them available via (gRPC). These three services include the message service, the dispute service, and the discovery service. A broker also has its own Ethereum account so that smart contracts can be called, which is necessary for dispute resolution.

The main task of a broker is fulfilled by the message service, which accepts and counts messages and puts them in a queue where they can then be pulled by the consumer. This enables the consumer to be offline while the provider sends the messages, but only a certain number of messages are stored in the queue to limit memory usage, and a broker also does not provide persistent storage. Furthermore, the prototype does not provide any means of authentication, which enables anyone to consume the messages from a broker. In future work, a broker should be expanded to include an authentication mechanism, which ensures that only the consumer who bought the data can pull the messages from the server. For example, token-based authentication could also be implemented here, whereby the user does not provide a username and password to receive a token but has to prove that it has the private key of a certain Ethereum account.

Like the proxy, a broker takes over the task of providing different products by entering a search query. As has already been mentioned in Sect. 6.2.1 the search query enables filtering by data type, frequency and cost. A broker scans all products saved in the product contract and streams the filtered results to the caller, which in most cases is a proxy that forwards the results to the consumer.

The last service is responsible for dispute resolution. If providers and consumers cannot agree during the settlement process, a dispute event is triggered on the blockchain. A broker listens for these events and sends a transaction to the corresponding settlement contract to resolve the dispute by providing the message count recorded by the message service.

6.3 Graphical user interfaces

Using Angular,[3] a TypeScript-based Web application framework, two independent (GUI) were implemented. Furthermore, both user interfaces were created using material design.[4] Angular applications can be composed of multiple modules, whereby each module serves its purpose,

[3] https://angular.io/

[4] https://material.io/design/

which contributes to interchangeability and maintainability. For this reason, the applications are broken down into several modules. One of the two user interfaces is responsible for the administration of the proxy, whereas the second user interface allows the user to communicate with the smart contracts. In combination, both interfaces allow the user to perform all of the tasks necessary to use the data marketplace.

6.4 Proxy UI

The user interface of the proxy was kept very simple, only containing the necessary functions to enable the administrator to create new accounts with wallets for devices that should communicate with the proxy. Two modules provide the necessary functionalities for this.

As mentioned in Sect. 6.2.1, the proxy provides multiple services that can be called via (gRPC). To enable the communication between the (GUI) and the proxy, (gRPC)-Web is used to generate the client service stubs and create the client code. At the moment, it is unfortunately not possible to use (gRPC)-Web in the browser without further ado, which requires the use of a proxy that accepts HTTP/1.1 requests and translates them into HTTP/2 requests. For this, we configured an Envoy proxy, which has good support for (gRPC) as well as HTTP/2 and is mentioned in the documentation of (gRPC).

To use the proxy, the administrator has to authenticate itself. For this purpose, the authentication module provides the user with a login component, which is used to send the credentials to the authentication service of the proxy. If the authentication is successful, an access token is made available, which will be sent to the proxy with every future request.

The account module provides different components to view the accounts managed by the proxy and create new accounts. When creating an account, the administrator specifies the user name, password of the account, and the passphrase of the wallet. The (GUI) then sends the requests to the account and wallet service of the proxy. For the time being, more functionalities such as updating and deleting accounts have not been implemented. Furthermore, the administrator lacks the options to access the wallets of the different accounts, which should be considered in future work.

6.5 DApp UI

The (GUI) for the decentralized application covers two major tasks. It enables the user to view the data stored on the blockchain and manage their data. The user can view all user profiles, devices, brokers, products, and trades. All data marked as deleted is filtered out by the application. If a user has saved their user profile, devices, brokers, or products on the blockchain, they can view, edit or delete them using the respective management view.

To be able to interact with the smart contracts, the (GUI) must be able to send transactions to the Ethereum network. For this purpose, we use MetaMask,[5] which is a browser extension that provides wallet functionalities and connects the (GUI) with the Ethereum network. It further gives the user the possibility to review the transaction before it is sent to the Ethereum network by showing transaction details like the amount, gas price, and gas limit.

As with the user interface for the proxy, this application is also divided into several modules: a user module, device module, broker module, product module, and a trade module. Each module provides different components to view and manage the respective entity. Truffle contracts provide an abstraction layer that facilitates the interaction with the smart contracts by shadowing the complexities of the JavaScript Ethereum (API).

7 Evaluation

For the evaluation, we focus on the costs which arise from the use of the implemented smart contracts. In addition to the costs, we also discuss how smart contracts are suitable for implementing a blockchain-based (IoT) data marketplace. Therefore, we consider the limitations and possible weaknesses of smart contracts observed during the development of the presented data marketplace.

Ethereum uses the concept of gas to protect the platform from misuse. All computations require the payment of gas fees to prevent network abuse, infinite loops, or other actions which waste resources. Each computational step requires a certain amount of gas. To not only describe the costs in the form of gas, we can also specify the value in a specific fiat currency. All that is required is the used gas, the gas price used for the transactions, and the current exchange rate from (ETH) to the desired currency. As of August 16, 2022, the price for one Ether is $1,852.54, and we are assuming a gas price of 20 gWei.

7.1 Deployment costs

The first costs arise when deploying the various smart contracts for the blockchain-based (IoT) data marketplace. However, it should be noted that these costs only have to be paid once by the operator so that the data marketplace can be put into operation. An exception to this is the deployment of settlement and bidding contracts since these are dynamically deployed when a new negotiation or trade

[5] https://metamask.io/

is created. By adding up the costs from all contract deployments (see Table 1), we get 16,088,194 gas or $596.08 for the deployment of all smart contracts. There may be a lot of costs associated with deployment, but it must be considered that, contrary to a centralized approach, there are no running costs for infrastructure and operations.

7.2 Costs of administrative functions

After the costs for the deployment of the smart contracts have been clarified, it is necessary to examine the recurring costs that arise from calling up the various functions of the smart contracts. However, functions that do not change the state of the blockchain can be excluded from these considerations since these are carried out by a local node, and therefore there are no costs for the caller. This means that all participants in the network can read out all the data that is stored by the smart contracts of the data marketplace without having to pay transaction fees.

In contrast, all operations that change the state of the blockchain consume gas. This includes, among other things, all administrative functions, such as creating, updating, and deleting users, devices, brokers, and products. By making sure that no loops are used during the implementation, the costs for these operations (see Table 2) remain constant even if the number of saved entries increases. Nevertheless, they are not the same for all entities. One reason for this is that not every entry requires the same storage space. However, it must be noted that the costs for delete operations are only constant if they are not cascaded. With the deletion of an entity that triggers the deletion of another entity, the costs increase with the number of entities to which it references.

7.3 Data trading costs

In addition to the costs incurred for the administration of the user profile, brokers, devices, and products, the costs that occur in data trading are particularly interesting. For this, we look at the costs of the individual functions that are carried out in the course of the data trading process (see Table 3). The costs of these operations remain constant except for the acceptance of negotiation and trading requests. These depend on the number of requests that the provider has not yet processed because if such a request is accepted, all pending requests must be iterated over to remove it.

First, we look at the costs involved in trading without negotiation. Here, we get a value of 1,117,702 gas or $41.41 for the provider and 531,031 gas or $19.68 for the consumer. These costs are calculated by summing up the costs of the individual operations.

If a negotiation takes place before the actual trade, the costs for this must also be considered. In our investigation, we restrict ourselves to an exchange of bids, whereby only the consumer makes a bid that is accepted directly by the provider, which is the shortest possible negotiation process. Furthermore, no trading requests are required, and the trade is created by the consumer directly, specifying the negotiation.

Summing up the costs for the provider and consumer, we get 507,756 gas or $18.81 for the consumer and 1,882,180 gas or $69.74 for the provider. When comparing these costs to the costs that arise without negotiation, it can be seen that these are 4.38% lower for the consumer and 68.39% higher for the provider. The reason for this is that a trading request requires more storage space than a negotiation request and therefore causes higher costs for the consumer. However, this does not apply to the provider who bears a considerably larger part of the costs than the consumer due to the deployment of an additional bidding contract.

7.4 Critical discussion

After we have discussed the costs of using smart contracts, we want to take a closer look at how smart contracts are suitable for implementing a decentralized data marketplace for the (IoT). For this, we consider not only the advantages of smart contracts and blockchain technology but also the disadvantages that were mainly perceived during the implementation.

Among other things, we use smart contracts to store various entities on the blockchain. In this respect, the blockchain acts as a distributed database that ensures transparency, integrity, and verifiability of the data. In particular, the fact that transactions are either carried out completely or not at all, the integrity and consistency of the data which is stored by the smart contracts can be well guaranteed. However, there are disadvantages compared to traditional databases, which can be noticed when implementing and using smart contracts.

A disadvantage of smart contracts compared to traditional databases is that it is more difficult to query the data. In contrast to smart contracts, traditional databases provide efficient query mechanisms for data retrieval. To query the data stored through the various smart contracts, we have to iterate over the entire storage of a smart contract and filter out the data that is of interest. It is possible to implement certain filter methods in the smart contract, but this causes additional costs.

During the implementation, we also had to cut back on the complexity of certain functionalities. A limitation of smart contracts is, for example, that they can only have a certain size. This limitation can be a problem, especially

Table 1 Smart contract deployment costs

Smart contract	Gas	Ether	USD
Migrations	190982	0.00381964	7.08
UserContract	1468608	0.02937216	54.41
DeviceContract	3938622	0.07877244	145.93
ProductContract	3054637	0.06109274	113.18
BrokerContract	1939980	0.0387996	71.88
NegotiationContract	1793263	0.03586526	66.44
TradingContract	3264283	0.06528566	120.94
RatingContract	437819	0.00875638	16.22

Table 3 Data trading costs

Operation	Gas	Ether	USD
requestNegotiation	213587	0.00427174	7.91
acceptNegotiationRequest	589352	0.01178704	21.84
makeBid	89915	0.0017983	3.33
accept	28944	0.00057888	1.07
requestTrading	326777	0.00653554	12.11
acceptTradingRequest	1051960	0.0210392	38.98
create	1198142	0.02396284	44.39
deposit	23303	0.00046606	0.86
settleTrade	65742	0.00131484	2.44
	180951	0.00361902	6.70

with smart contracts that have to provide a variety of functions. While it makes sense to keep the smart contracts as small as possible to decrease the gas fees, it is sometimes desirable to have certain functionalities encapsulated in one smart contract. In our case, the device contract provides a variety of functions related to devices that cannot be outsourced to another smart contract. We had to be particularly careful here, as the maximum size allowed was almost exceeded.

Furthermore, smart contracts programmed in Solidity do not offer support for floating-point numbers, which are essential for more complex calculations. In our case, we did not use a complex calculation method for the rating system and replaced it with a much simpler one, which is easy to implement with integers. Support for fixed-point numbers is planned for the future, where the presentation of the number is precisely defined.

In addition, the costs of using smart contracts are also a weakness. The users of the data marketplace have to decide for themselves which ratio between purchase price and data trading costs is suitable for them and when it is no longer worthwhile. However, other approaches could be followed, which cause fewer costs. The negotiation process could take place completely off-chain, but the advantages of blockchains (e.g., transparency and verifiability) would be lost. Furthermore, the administrative costs could be reduced by using a (DFS) instead of the actual blockchain to store certain data.

Another possibility would be to port the entire solution to a different blockchain. The smart contracts could be deployed on another Ethereum-based blockchain, such as the Binance Smart Chain, without major changes. With an exchange rate of $312,85 for 1 BNB and a gas price of 5 Gwei, the costs for data trading would drop from $41.41 to $1.75 for the provider and $19.68 to $0.83 for the consumer. Therefore, by changing the blockchain, a cost reduction of 96% can be achieved. However, it must be noted that other blockchains differ from Ethereum in certain aspects like the degree of decentralization, security, and performance.

8 Conclusion

Blockchains and smart contracts are promising technologies for the creation of a decentralized (IoT) data marketplace, as traditional data marketplaces come with different challenges, including expensive infrastructure, single point of failure, trust problems, and privacy issues.

Therefore, we designed and implemented a blockchain-based (IoT) data marketplace. The data marketplace is based on a three-tier architecture, and smart contracts provide various functionalities to enforce the rules of the data marketplace. A proxy gives providers and consumers

Table 2 Costs of create, update and delete operations

Smart contract	Create			Update			Delete		
	Gas	Ether	USD	Gas	Ether	USD	Gas	Ether	USD
User contract	178103	0.00356206	6.60	72994	0.00145988	2.71	45171	0.00090342	1.67
Device contract	252964	0.00505928	9.37	70690	0.0014138	2.62	46697	0.00093394	1.73
Product contract	300232	0.00600464	11.12	83446	0.00166892	3.09	49075	0.0009815	1.82
Broker contract	207946	0.00415892	7.71	55757	0.00111514	2.07	46377	0.00092754	1.72

the possibility to integrate (IoT) devices with little resources. A broker helps to facilitate the data trading process, takes over resource-intensive tasks, and is responsible for dispute resolution in case a provider and consumer cannot agree during the settlement process. We examined the costs arising from the usage of smart contracts and discussed the problems encountered during the implementation.

In our future work, amongst other things, we will investigate trustless dispute resolution, efficient product discovery, cost optimization, and more mature rating and negotiation mechanisms.

Author contributions All authors contributed to the conception and design of the solution. Data collection and analysis were performed by MS. SS and SK provided supervision, as well as reviewed, and edited this work. The first draft of the manuscript was written by MS and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development as well as the Christian Doppler Research Association.

Data availability All data generated or analysed during this study are included in this published article.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Janssen, M., Charalabidis, Y., Zuiderwijk, A.: Benefits, adoption barriers and myths of open data and open government. Inf. Syst. Manag. **29**(4), 258–268 (2012)
3. Cao, T.-D., Pham, T.-V., Vu, Q.-H., Truong, H.-L., Le, D.-H., Dustdar, S.: Marsa: a marketplace for realtime human sensing data. ACM Trans. Internet Technol. **16**(3), 1–21 (2016)
4. Zaslavsky, A., Perera, C., Georgakopoulos, D.: Sensing-as-a-service and big data. In: 2012 International Conference on Advances in Cloud Computing, pp. 21–29 (2012)
5. Gupta, P., Kanhere, S.S., Jurdak, R.: A decentralized IoT data marketplace. In: 3rd Symposium on Distributed Ledger Technology (SDLT) (2018)
6. Mišura, K., Žagar, M.: Data marketplace for internet of things. In: 2016 International Conference on Smart Systems and Technologies, pp. 255–260 (2016). https://doi.org/10.1109/SST.2016.7765669. IEEE
7. Nakamoto, S., et al.: Bitcoin: a peer-to-peer electronic cash system. Bitcoin Whitepaper. Working Paper (2008)
8. Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S.: Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press, Princeton (2016)
9. Wüst, K., Gervais, A.: Do you need a blockchain? In: 2018 Crypto Valley Conference on Blockchain Technology, pp. 45–54 (2018). IEEE
10. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)
11. Natoli, C., Yu, J., Gramoli, V., Esteves-Verissimo, P.: Deconstructing blockchains: a comprehensive survey on consensus, membership and structure (2019). arXiv:1908.08316
12. Szabo, N.: Smart contracts (1994). http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html. Accessed 1 Mar 2022
13. Brandstätter, T., Schulte, S., Cito, J., Borkowski, M.: Characterizing efficiency optimizations in solidity smart contracts. In: The 3rd IEEE International Conference on Blockchain, pp. 281–290 (2020). IEEE
14. IHS: Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. [Online; accessed 2022-03-01] (2016)
15. Nguyen, D.-D., Ali, M.I.: Enabling on-demand decentralized IoT collectability marketplace using blockchain and crowdsensing. In: 2019 Global IoT Summit, pp. 1–6 (2019). https://doi.org/10.1109/GIOTS.2019.8766346. IEEE
16. Musso, S., Perboli, G., Rosano, M., Manfredi, A.: A decentralized marketplace for M2M economy for smart cities. In: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 27–30 (2019). https://doi.org/10.1109/WETICE.2019.00014. IEEE
17. Wang, Z.-J., Lin, C.-H.V., Yuan, Y.-H., Huang, C.-C.J.: Decentralized data marketplace to enable trusted machine economy. In: 2019 IEEE Eurasia Conference on IOT, Communication and Engineering, pp. 246–250 (2019). https://doi.org/10.1109/ECICE47484.2019.8942729. IEEE
18. Lundqvist, T., De Blanche, A., Andersson, H.R.H.: Thing-to-thing electricity micro payments using blockchain technology. In: 2017 Global Internet of Things Summit, pp. 1–6 (2017). IEEE
19. Hanada, Y., Hsiao, L., Levis, P.: Smart contracts for machine-to-machine communication: possibilities and limitations. In: 2018 IEEE International Conference on Internet of Things and Intelligence System, pp. 130–136 (2018). IEEE
20. Schomm, F., Stahl, F., Vossen, G.: Marketplaces for data: an initial survey. ACM SIGMOD Rec. **42**(1), 15–26 (2013)
21. Ramel, D.: Microsoft closing Azure dataMarket (2016). https://adtmag.com/articles/2016/11/18/azure-datamarket-shutdown.aspx. Accessed 1 Mar 2022
22. Krishnamachari, B., Power, J., Cyrus, S., Kim, S.H.: IoT marketplace: a data and API market for IoT devices (2017). https://msbfile03.usc.edu/digitalmeasures/gerardpo/intellcont/USCIoTMarketplace_Jan152017-1.pdf. Accessed 1 Mar 2022

23. Radhakrishnan, R., Krishnamachari, B.: Streaming data payment protocol (SDPP) for the Internet of Things. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1679–1684 (2018). IEEE
24. Zhao, Y., Li, Y., Mu, Q., Yang, B., Yu, Y.: Secure pub-sub: blockchain-based fair payment with reputation for reliable cyber physical systems. IEEE Access **6**, 12295–12303 (2018)
25. Delgado-Segura, S., Pérez-Solà, C., Navarro-Arribas, G., Herrera-Joancomartí, J.: A fair protocol for data trading based on Bitcoin transactions. Future Gener. Comput. Syst. **107**, 832–840 (2020)
26. Wörner, D., von Bomhard, T.: When your sensor earns money: exchanging data for cash with Bitcoin. In: 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp. 295–298 (2014). ACM
27. Missier, P., Bajoudah, S., Capossele, A., Gaglione, A., Nati, M.: Mind my value: a decentralized infrastructure for fair and trusted IoT data trading. In: Seventh International Conference on the Internet of Things (2017). ACM. (**Article 15**)
28. Bajoudah, S., Dong, C., Missier, P.: Toward a decentralized, trust-less marketplace for brokered IoT data trading using blockchain. In: 2019 IEEE International Conference on Blockchain, pp. 339–346 (2019). https://doi.org/10.1109/Blockchain.2019.00053. IEEE
29. Gupta, P., Dedeoglu, V., Najeebullah, K., Kanhere, S.S., Jurdak, R.: Energy-aware demand selection and allocation for real-time IoT data trading. In: 2020 IEEE International Conference on Smart Computing, pp. 138–147 (2020). https://doi.org/10.1109/SMARTCOMP50058.2020.00038. IEEE
30. Ramachandran, G.S., Radhakrishnan, R., Krishnamachari, B.: Towards a decentralized data marketplace for smart cities. In: 2018 IEEE International Smart Cities Conference, pp. 1–8 (2018). IEEE
31. Banerjee, P., Ruj, S.: Blockchain enabled data marketplace—design and challenges (2018). arXiv:1811.11462
32. Sharma, P., Lawrenz, S., Rausch, A.: Towards trustworthy and independent data marketplaces. In: The 2nd International Conference on Blockchain Technology, pp. 39–45 (2020). https://doi.org/10.1145/3390566.3391687. ACM
33. Özyilmaz, K.R., Doğan, M., Yurdakul, A.: Idmob: Iot data marketplace on blockchain. In: 2018 Crypto Valley Conference on Blockchain Technology, pp. 11–19 (2018). IEEE
34. Truong, H.T.T., Almeida, M., Karame, G., Soriente, C.: Towards secure and decentralized sharing of IoT data. In: 2019 IEEE International Conference on Blockchain, pp. 176–183 (2019). https://doi.org/10.1109/Blockchain.2019.00031. IEEE
35. Badreddine, W., Zhang, K., Talhi, C.: Monetization using blockchains for IoT data marketplace. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency, pp. 1–9 (2020). https://doi.org/10.1109/ICBC48266.2020.9169424. IEEE
36. Koutsos, V., Papadopoulos, D., Chatzopoulos, D., Tarkoma, S., Hui, P.: Agora: a privacy-aware data marketplace. In: IEEE Transactions on Dependable and Secure Computing (2021). IEEE
37. Meijers, J., Putra, G.D., Kotsialou, G., Kanhere, S.S., Veneris, A.: Cost-effective blockchain-based iot data marketplaces with a credit invariant. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1–9 (2021). IEEE
38. Ojo, M.O., Giordano, S., Procissi, G., Seitanidis, I.N.: A review of low-end, middle-end, and high-end iot devices. IEEE Access **6**, 70528–70554 (2018). https://doi.org/10.1109/ACCESS.2018.2879615

Michael Sober received his master's degree in Software Engineering & Internet Computing from TU Wien in 2020. He is a research assistant and Ph.D. student at the Institute for Data Engineering at TU Hamburg and working on blockchain interoperability solutions in the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things (CDL-BOT).

Giulia Scaffino is a Ph.D. student at TU Wien. After receiving her M.Sc. degree in Nuclear Physics in 2019, she joined the Security and Privacy Group of the Institute of Logic and Computation at TU Wien. She also works on blockchain interoperability solutions contributing to the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things (CDL-BOT).

Stefan Schulte heads the Institute for Data Engineering at Hamburg University of Technology, Germany, and leads the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things (CDL-BOT). His research interests include the areas of data stream processing, the Internet of Things, and distributed ledger technologies. Findings from his research have been published in more than 100 refereed scholarly publications.

Salil S. Kanhere received his M.S. and Ph.D. degrees from Drexel University, Philadelphia, USA. He is a Professor in the School of Computer Science and Engineering at UNSW Sydney, Australia. His research interests include the Internet of Things, cyber-physical systems, blockchain, pervasive computing, cybersecurity, and applied machine learning.