Received February 25, 2020, accepted April 20, 2020, date of publication April 29, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991267

TIDES: A Trust-Aware IoT Data Economic System With Blockchain-Enabled Multi-Access Edge Computing

I-HSUN CHUANG, (Member, IEEE), SHIH-HAO HUANG, (Student Member, IEEE), WEI-CHU CHAO, (Student Member, IEEE), JEN-SHENG TSAI, AND YAU-HWANG KUO
Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan

Corresponding author: Yau-Hwang Kuo (kuoyh@ismp.csie.ncku.edu.tw)

This work was supported by the Ministry of Science and Technology (MOST), Taiwan, under Grant 106-2221-E-006-244-MY3, Grant 107-2221-E-006-164-MY2, and Grant 108-2321-B-006-008.

ABSTRACT To make good use of valuable Internet of Things (IoT) data assets, this paper proposes a trust-aware IoT data economic system (TIDES) with complete IoT data pricing, trading and protection functions. To ensure reliable and automatic data trading, the entire trading process is automatically performed by smart contracts on a hierarchical blockchain. Moreover, we develop several sophisticated methods to ensure the efficiency and service quality of TIDES. First, a complete evaluation model that takes the data trading profile and reputation into consideration is proposed for both suppliers and demanders to assess the trustworthiness of their trading partners. Second, a client-centric data value evaluation model and a game-theory-based pricing model are used to promote win-win transactions in which the demanders obtain higher quality data at an acceptable price and the suppliers receive higher profits. Third, a dispute arbitration model is invoked to detect suspicious trading and refund these payments automatically. TIDES further utilizes a multi-access edge computing (MEC) architecture to alleviate the huge burdens of IoT devices from blockchain operations, reduce the trading latency, and help mobile devices to trade IoT data. The simulation results have shown the advantages of TIDES in terms of trading time, storage overhead, data trading profit, quality data trading, pricing efficiency, and reliability on data asset management and trading.

INDEX TERMS Blockchain, data economy, data trading, edge computing, Internet of Things (IoT).

I. INTRODUCTION

With the emergence of Internet of Things (IoT), various IoT devices are widely deployed to realize a wide variety of smart cyber-physical systems (CPSs), such as intelligent transportation systems and smart grids, where data are the most valuable assets. However, a large IoT deployment for only one silo application is not easy to sustain. Additionally, complex data-intensive applications usually need data from multiple sources; therefore, IoT service providers tend to purchase data from multiple sources. Thus, IoT data trading is becoming active, and the establishment of trustworthy and reliable IoT data economic systems has become a necessity.

Such systems must be able to efficiently manage the life-cycle of IoT data, which typically consists of stages: data collection, data analytics, data pricing, data trading, and data protection [1]. During data collection and data analytics, massive amounts of heterogeneous data are collected and applied to generate useful knowledge, and the results are called data services. However, these two stages often suffer from the problems of inefficient data access and changeable data quality. To valuate the provisioning of data services, deciding how to charge for the services occurs in the data pricing stage. However, in a highly dynamic market, deriving an immutable pricing strategy for diverse IoT applications is practically infeasible. In the data trading stage, the demanders decide their target data and the suppliers ship their commodity. However, an untrusted supplier might provide counterfeits, and on the other hand, a deadbeat could bilk, which leads to unreliable transactions. In addition, because IoT data are usually sensitive and valuable, cybersecurity for data storage and data delivery is ensured at the data protection stage.

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Merlino.

Finally, to sustain applicable IoT services, all operations in these five stages are required to perform automatically [2] and efficiently and to easily support mobile devices.

A solid IoT data economic system, including the proposed trust-aware IoT data economic system (TIDES), should be designed to fulfill some critical requirements. The most important requirement is reliability. For this requirement, the system should provide stable trading services and further prevent trading and its records from manipulating and dominating. The confidentiality of data storage and data delivery also needs to be protected. Finally, when one side in a transaction is malicious, the system should be able to rapidly address this situation. However, traditional trading systems usually provide services from centralized servers that are vulnerable to single points of failure, which might lead to system unreliability. Intelligence is required to guarantee service quality so that the system can smartly help both demanders and suppliers to make decisions based on their expectations. According to the hypothesis of rational people [3], suppliers tend to maximize their profits, and demanders want to obtain the most satisfactory commodity at an acceptable price. Generally, a higher price increases profits but reduces the purchase intention which is also related to commodity quality. The existence of competitors also influences the probability of transactions. Thus, for suppliers, providing a pricing strategy that considers the variable market situation to maximize profits is a necessity. For demanders, the first thing they need to do before purchasing a data commodity is to assess its value, which is related to data quality. Automation is an inevitable requirement for trading, and the massive IoT data necessitates an algorithmic trading mechanism [4] without manual intervention. Efficiency is also highly demanded; thus, latency and resource utilization need to be taken into consideration. The time for a data transaction includes both the trading time and the access time. The volume of IoT data always leads to heavy burdens on data access, while a complicated data trading mechanism takes a very long trading time. Moreover, as most IoT data are queried locally, not all data need to be delivered globally. To this end, edge computing is a promising solution that benefits not only system performance but also network resource utilization. Finally, the requirement for mobility management cannot be ignored. As both data suppliers and data demanders can be either mobile users or IoT devices, they might need to create transactions when moving.

There have been several blockchain-based solutions for distributed data trading [5]–[8]. A publish/subscribe framework [5] was proposed by Hashemi *et al.* Roman and Stefano proposed a trusted data marketplace architecture by evaluating user credit [6], and both the demander and the supplier have to sign a smart contract to manage their operations by manual intervention. Filecoin [7], similar to Bitcoin [8], is a data storage service that applies the concept of proof-of-retrievability. It encourages nodes to contribute their storage as much as possible. Although these distributed data trading systems alleviate the problems of centralized systems, they neither provide IoT data pricing strategies nor handle trading disputes, and hence fail to fulfill the intelligence and reliability requirements as indicated above.

In our previous work [9], the reliable IoT data economic system (RIDES) was proposed to provide IoT data as a foundation trading service. This paper further presents TIDES, which carries out the holistic functions of data pricing, trading and protection. This system also provides secure and tamper-proof data trading channels with a hierarchical blockchain that fulfills the reliability requirement. Owing to the decentralization property of blockchain technology, users can trade fairly without being dominated. The secure design of TIDES for IoT data lifecycle management also ensures reliable data storage and delivery. To ensure trading reliability, an integrated evaluation model that takes the data trading profile and reputation into account is proposed for both suppliers and demanders to assess the trustworthiness of their trading partners and further reduce the trading failure rate. Even if the demander has unfortunately created a transaction with a malicious supplier, its payment can be refunded automatically through a dispute arbitration procedure. To fulfill the intelligence requirement, on the supplier side, we propose an intelligent, game-theoretical pricing model in accordance with the pricing strategies of competitors to determine an appropriate price. This pricing model gives the suppliers notably higher profits than other models. On the demander side, a client-centric data value evaluation model is used to evaluate the data value and determine which data commodity is worth purchasing. With this model, the demanders can obtain satisfactory data commodities at acceptable prices. To address the automation requirement, all trading procedures in TIDES are algorithmic and performed automatically by exploiting the smart contract technique. Furthermore, TIDES applies the multi-access edge computing (MEC) paradigm [12] with a hierarchical blockchain. With this efficient data trading architecture, the burden on IoT devices can be alleviated and the mobility issue can be solved. Therefore, TIDES also fulfills the requirements of efficiency and mobility. Then, thorough simulations are conducted. The simulation results demonstrate that the system performance and storage overheads of TIDES are better than those of existing systems. The increased profit, higher quality data trading and better pricing efficiency brought by the proposed intelligent pricing model and the improved trading reliability due to the trustworthiness evaluation also show the advantage of TIDES. To the best of our knowledge, TIDES is the first and most complete system proposed to efficiently provide IoT data commodities with a reliable, intelligent, and automatic trading environment.

The paper is organized as follows. Section II provides background information related to this work. Section III describes the details of the system architecture, and Section IV presents the proposed methods for reliability and trust-awareness design. The system flows of TIDES are described in Section V. Then, the simulation results for TIDES are shown in Section VI. Finally, the concluding remarks for this paper are given in Section VII, and the abbreviations used in this paper are summarized in the appendix.

II. BACKGROUND

A. BLOCKCHAIN TECHNOLOGY AND SMART CONTRACT

Blockchain is an emerging technology whose immutability allows it to reliably store data. Bitcoin [8] is the first successful blockchain-based cryptocurrency, and Ethereum [10] further improves blockchain as a programmable distributed trusted infrastructure. The core technique making Ethereum programmable is the smart contract, which is a self-executing data structure described in high-level languages, such as Python, compiled into bytecode, and then executed by all nodes equipped with the Ethereum Virtual Machine. A smart contract can apply conditional expressions to describe a deterministic environment. When a condition described in a smart contract is satisfied, an activation transaction will be deployed to the blockchain network to activate the smart contract. Then, the corresponding trading specified in the smart contract will be performed. In an Ethereum smart contract, the Contract ID and application binary interface (ABI) [11] with bytecode are the most important information. The former indicates where the smart contract is, and the latter describes the content of smart contracts for human reading. In addition, some trading information, such as data URLs, can also be stored in a smart contract to detail the trading.

B. MULTI-ACCESS EDGE COMPUTING (MEC)

In the past, cloud computing was widely applied to all types of services. However, the volume of IoT data has congested cloud servers leading to the denial of service. The high network latency between cloud servers and IoT devices also makes cloud computing technology inapplicable in high-speed IoT applications. Therefore, the MEC architecture has been proposed to provide additional resources, such as computing power and storage space, at the edge side, which can fulfill the low-latency requirement of IoT services while significantly alleviating the cloud server loads [13]. As MEC services are usually provided by network service providers, the roaming problem of mobile devices can also be addressed.

C. DATA PRICING MODEL

The data pricing models can be categorized into two types [1]. The economic-based pricing model evaluates the price of data according to economic principles, such as costs and margins. The game-theory-based pricing model considers competitor behaviors before pricing.

1) ECONOMIC-BASED PRICING MODELS

- Cost model [14]: This model considers only the commodity costs and profit margins. The simplicity of this model is a remarkable advantage. However, the lack of external information leading to market misjudgment also makes it ineffective in data trading [15].
- Consumer perceived model [16]: The suppliers using this model take the demander's feedback into account. In general, this model is more suitable for data pricing than the cost model. However, ignoring supply conditions makes it difficult to derive the most appropriate price.
- Supply and demand model [17], [18]: In this kind of model, the relationship between suppliers and demanders is the most important factor for pricing. In [17], a linear supply function and a linear demand function are formulated to derive the price. In [18], a pricing system, called MGA, is proposed by formulating the willingness of suppliers and demanders as two natural logarithmic functions. Although the intersection of supplier and demander functions implies the price that has the best probability to reach a deal, the neglect of market situations, such as the competition between suppliers, leads to inappropriate pricing.

2) GAME-THEORY-BASED PRICING MODELS

- Non-cooperative game model [19], [20]: In [19], suppliers do not need to negotiate with each other. Instead, suppliers derive their prices by referring to the pricing strategies announced by other suppliers. This model is applicable in IoT environments but might result in a price war. Exploiting a centralized broker, the authors of [20] proposed a pricing method that takes all demanders and suppliers into account to derive appropriate prices without price cutting. However, since numerous transactions for IoT data will be created dynamically, a centralized pricing system, which usually becomes a performance bottleneck, is unpractical for IoT data trading.
- Stackelberg game model [21]–[23]: In this kind of model, each supplier announces its price after the major supplier does. In the IoT data trading environment, however, the determination of the major supplier is difficult and inefficient, which makes this model impractical.
- Bargaining game model [24]: In this model, suppliers and demanders negotiate to reach an agreement. Nevertheless, the negotiation process is usually time-consuming and resource-wasting. Thus, this model is not suitable in the IoT data trading scenario.

III. TRUST-AWARE IOT DATA ECONOMIC SYSTEM

Fig. 1 and Fig. 2 show the functional model and system architecture of TIDES based on the MEC infrastructure where the data suppliers and data demanders could be human-operated devices or autonomous IoT devices within each radio access network (RAN) area. An IoT device or user equipment may play the roles of both supplier and demander. For resource-constrained data suppliers and demanders, IoT gateways can be the intermediaries between them and TIDES. As shown in Fig. 2, TIDES contains the functional entities of data repositories (DRs), an information broadcasting station (IBS), data trading agencies (DTAs), a data trading blockchain (DTB), and a transaction dispute arbiter (TDA) to constitute a complete ecosystem of the IoT data economy. DRs implement a decentralized data storage scheme that can adapt to different requirements for data privacy, access speed,

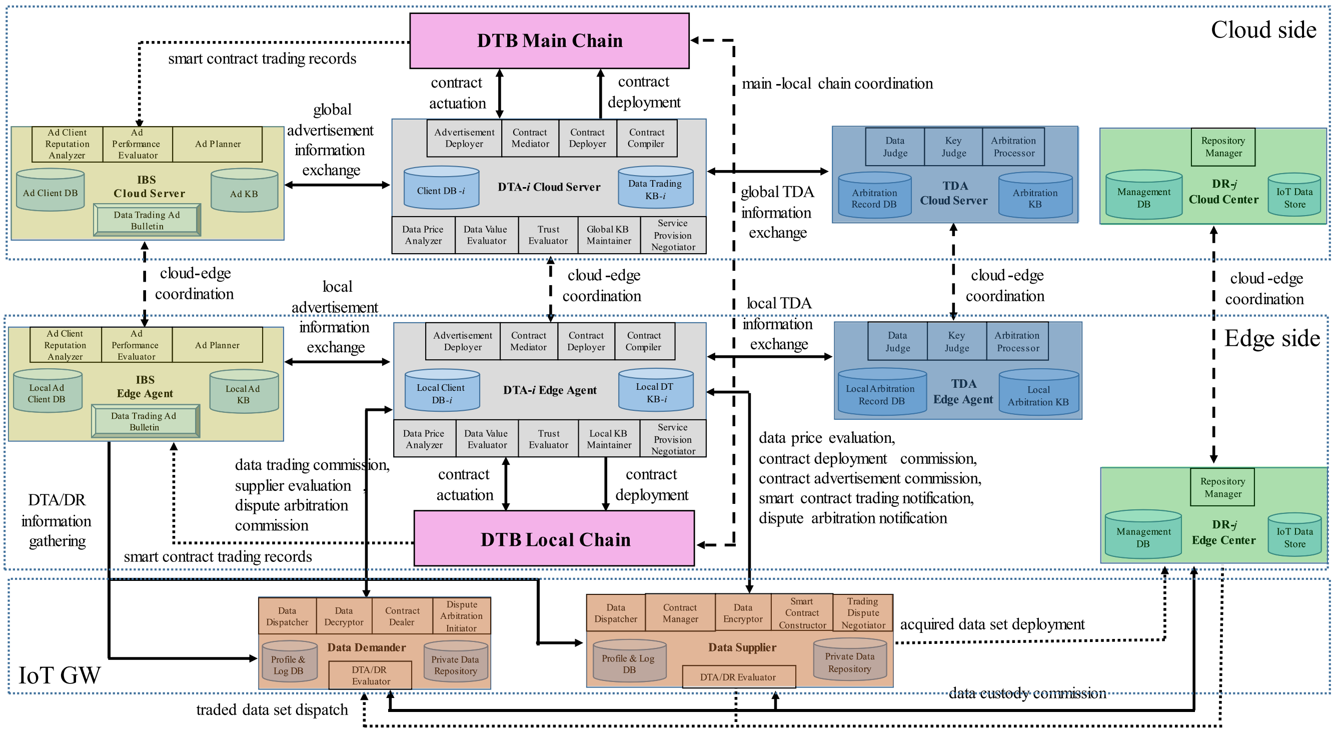

Cloud side
Edge side
IoT GW
DTB Main Chain
DTB Local Chain
smart contract trading records
contract actuation
contract deployment
main -local chain coordination
global advertisement information exchange
global TDA information exchange
cloud-edge coordination
local advertisement information exchange
local TDA information exchange
IBS Cloud Server
IBS Edge Agent
DTA-i Cloud Server
DTA-i Edge Agent
TDA Cloud Server
TDA Edge Agent
DR-j Cloud Center
DR-j Edge Center
Data Demander
Data Supplier
data trading commission, supplier evaluation dispute arbitration commission
data price evaluation, contract deployment commission, contract advertisement commission, smart contract trading notification, dispute arbitration notification
DTA/DR information gathering
acquired data set deployment
data custody commission
traded data set dispatch

FIGURE 1. The functional model of TIDES.

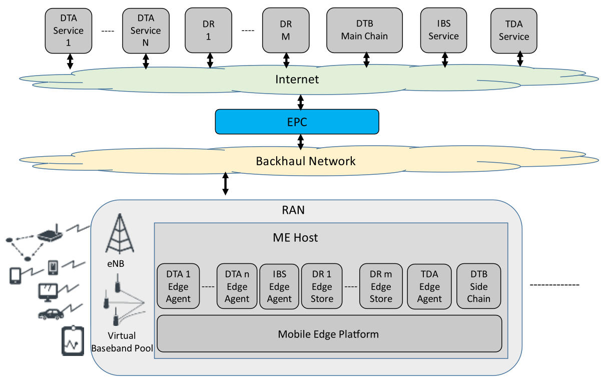


FIGURE 2. The MEC-based system architecture of TIDES.

and storage capacity. DTAs allow data trading stakeholders to draw smart contracts and accomplish each data transaction. IBS is commissioned to advertise the smart contracts announced by data suppliers, the service specification of each DR entity, and the service specification of each DTA. TDA is authorized to arbitrate the disputes of IoT data transactions. These four types of functional entities are for public cloud services with avatars on ME hosts. A cloud-edge coordination mechanism is designed for each cloud site and its edge avatars to fully exploit the advantages of MEC. DTB is implemented on the Ethereum platform to provide a secure and tamper-proof financial flow for data trading. It also adopts a hierarchical structure that consists of a main chain over the Internet and multiple local chains on ME hosts. In TIDES, IBS, TDA and DTB are public services, while DR and DTA are contract-based services.

Based on the algorithmic trading mechanism [4], TIDES realizes a trust-aware algorithmic data trading scenario. On the supply side, when a supplier acquires a new IoT data asset, it encrypts the asset, and then deposits the encrypted asset in a commissioned DR. In addition, the supplier prices the data asset and draws a smart contract to sell this asset by consulting a DTA. The contract is then deployed to DTB through DTA and announced on IBS. On the demand side, every demander smartly chooses what it wants by referring to the advertisements announced on IBS. Then, it starts to purchase the chosen data commodity according to the corresponding smart contact in DTB and accomplishes the payment to the assigned account. After the payment has been transferred to the data supplier, the purchased data commodity and its corresponding key can be delivered to the demander securely. Next, the demander verifies whether the received data are satisfactory. If any argument occurs, it can request the TDA to arbitrate, and obtain the refund once this request is approved. The details of the operational flows will be introduced in Section IV.

From the viewpoint of commercial business, TIDES can have multiple DTAs with their own service know-how. As depicted in Fig. 1, each specific DTA deploys its edge agents on ME hosts in different RANs to serve nearby suppliers and demanders. These edge agents allow IoT devices to make local data trades, and each of them deploys smart contracts into the local chain of DTB on the same ME host under the commissions from data suppliers. To provide its data asset for global usage, the supplier designates its smart contract to be deployed to the main chain of DTB. Then, the commissioned DTA edge agent transfers this smart contract deployment task to its cloud server. Similarly, one demander delegates a DTA edge agent to purchase a data commodity from the local chain of DTB in the same RAN. If the data commodity is supplied by a cloud DR or an edge DR in another RAN, then the delegated edge agent will transfer the task to its DTA cloud server to activate the target smart contract from the main chain of DTB. Purchasing a data commodity from another RAN requires more collaboration between the DTA cloud server and the corresponding edge agents.

As shown in Fig. 1 and Fig. 2, the local interactions of functional entities on the same ME host can achieve most of the TIDES functions for the data suppliers and demanders in the same RAN, which can make data trading more efficient and even more reliable. The IBS edge agents and TDA edge agents have similar edge-cloud collaboration scenarios to that of DTA. Each IBS edge agent advertises the smart contracts and the DR edge centers for local clients in the same RAN, while the IBS cloud server collects advertisements for global clients. Each TDA edge agent serves the dispute arbitration requests from demanders in the same RAN. When receiving an arbitration request to the data supplier in another RAN, the TDA edge agent will transfer this task to the TDA cloud server to coordinate the arbitration procedure between the corresponding edge agents.

As other functional entities, TIDES also implements a data repository with a hierarchical structure, which consists of DR cloud centers and their edge agents. Different DRs have different service strategies. For privacy and security considerations, the traded data commodity may be stored in the private storage of a data supplier or an IoT gateway. In this case, the traded data set will be directly delivered from the supplier to the demander.

The structure of the smart contract for IoT data trading in TIDES, which is composed of contract ID, contract ABI with bytecode, and trading information, is shown in Fig. 3(a). The former two parts follow the rules defined in Ethereum. The last part consists of four fields. The DR field details the ID of DR and the location where the data asset is stored. The DTA field names the ID of the commissioned DTA. The trading field describes the traded data. The optional field enumerates additional information about the trade, such as the supplier's public key. When the data supplier receives a payment for a smart contract that it announced, it provides the DR URL and data wrapped key for this transaction in the smart contract. Then, the demander can obtain the traded data from the DR and the data wrapped key from blockchain. In addition, Fig. 3(b) shows the information described in a smart contract advertisement.

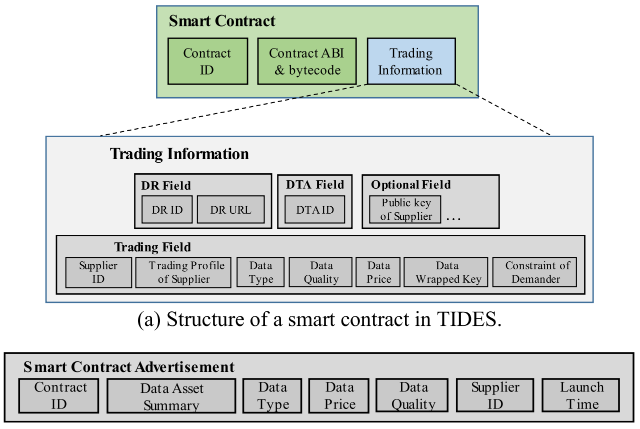


FIGURE 3. The smart contract and smart contract advertisement in TIDES.

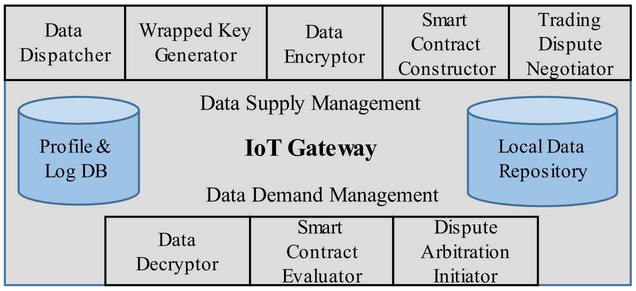


FIGURE 4. Functional structure of the IoT gateway.

IV. SECURITY, PRICING, AND TRUST-AWARENESS MODELS IN TIDES

A. SECURITY DESIGN FOR IoT DATA LIFECYCLE MANAGEMENT

TIDES employs multiple data protection mechanisms at the infrastructure layer and the data layer.

1) SECURITY DESIGN AT THE INFRASTRUCTURE LAYER

In TIDES, all functional entities interact through secure channels constructed by means of the security architecture of 5G MEC and the network slicing scheme to ensure communication security. Blockchain technology also guarantees the integrity and temporal consistency of all transactions to sustain reliable data trading.

2) SECURITY DESIGN AT THE DATA LAYER

The security design at the data layer intends to prevent IoT data assets from being disclosed to unauthorized users or devices, and this design is mainly implemented in IoT devices, user equipment, and IoT gateways. The functional components of stand-alone data suppliers and demanders are depicted in Fig. 1, while the functional structure of the IoT gateway that helps resource-constrained data suppliers and demanders to accomplish their data trading operations is illustrated in Fig. 4.

TIDES employs a series of encryption schemes to prevent data assets from unauthorized access while ensuring the originator identity. Whenever a supplier acquires a new data set, this device or its IoT gateway invokes the data encryptor to encrypt the data set with a random key, Key_r, as in (1).

$$Data_e = E_s\left(Data_p, Key_r\right), \tag{1}$$

where $E_s(x, y)$ indicates a symmetric encryption operation to encrypt the data x with the key y, while $Data_p$ represents the plain data set and $Data_e$ is the encrypted one. The $Data_e$ will be stored in a data repository, and the Key_r is held by the supplier. Then, whenever the data supplier receives the payment notification for an approved data transaction, it invokes the wrapped key generator to generate the data wrapped key, Key_w, of Key_r by (2).

$$Key_w = E_a\left(E_a\left(Key_r, Key^d_{pub}\right), Key^s_{pri}\right), \tag{2}$$

where $E_a(x, y)$ represents the asymmetric encryption operation to encrypt the data x with key y, Key^d_{pub} is the public key of the demander retrieved from the payment notification, and Key^s_{pri} is the private key of the supplier.

After receiving Key_w, the demander invokes its data decryptor to derive Key_r by (3).

$$Key_r = D_a\left(D_a\left(Key_w, Key^s_{pub}\right), Key^d_{pri}\right), \tag{3}$$

where $D_a(x, y)$ indicates the asymmetric decryption operation to decrypt the encrypted data x with key y, Key^d_{pri} is the private key of the demander, and Key^s_{pub} is the public key of the supplier retrieved from the smart contact. With Key_r, $Data_e$ can be decrypted via symmetric decryption, $D_s(x, y)$, where x is the encrypted data and y is the key for obtaining the original data set $Data_p$ by (4).

$$Data_p = D_s\left(Data_e, Key_r\right). \tag{4}$$

B. TRUSTWORTHINESS EVALUATION MODELS

Trust is a key factor of satisfactory IoT data trading. Thus, TIDES proposes trustworthiness evaluation mechanisms for suppliers and demanders to assess the trustworthiness of their trading partners.

For user d, the trustworthiness $Trust^d_s$ of a potential trading partner s can be derived in (5)

$$Trust^d_s = \beta \cdot OT_s + (1-\beta) \cdot ST^d_s, \quad 0 \leq \beta \leq 1, \tag{5}$$

where OT_s indicates the objective trustworthiness of s, ST^d_s is the subjective trustworthiness of s for d, and β is a weighting factor. First, user d will consult its DTA, whose trust evaluator provides the trustworthiness evaluation service for data trading stakeholders. All involved information for the trustworthiness evaluation should be tamper-proof and accessible. Therefore, the trust evaluator refers to only the trading records and account balance on DTB and the arbitration records on TDA when deriving OT_s. Because it is more sophisticated for a trading partner with more transactions, the trust evaluator considers the number of transactions that has been completed by s, $Trans_s$, as a parameter. In addition, a trading partner with more arbitration records means it is not reliable. Hence, the trust evaluator retrieves the arbitration records Ret_s of s from TDA as a parameter. Moreover, a wealthy supplier usually has less motivation to maliciously provide incorrect data and has a greater capability to pay compensation, whereas a wealthy demander usually has less possibility of fraud in data transactions. Hence, the trust evaluator also takes the account balance AB_s of its trading partner into account. As a result, OT_s can be defined as a linear combination of the three parameters in (6),

$$OT_s = \varepsilon_1\left(\frac{Trans_s}{Tran_{all}}\right) + \varepsilon_2(\frac{Ret_s}{Trans_s}) + \varepsilon_3\left(\frac{AB_s}{AB_{all}}\right), \tag{6}$$

where $Tran_{all}$ and AB_{all} indicates the total number of transactions and amount of crypto-currency in TIDES, respectively, and ε_1, ε_2, ε_3 are the weighting factors such that $\varepsilon_2 + \varepsilon_2 + \varepsilon_3 = 1$.

To assess ST^d_s, the trust evaluator considers the social relationship between s and d as a key factor because generally, a trading partner with more common friends is more trustworthy [25], [26]. In other words, s will be more worthy of d's trust if it has more common trading partners with d. Thus, ST^d_s is defined as the proportion of common trading partners between s and d with respect to the number of all trading partners of d in the trading records of TIDES.

$$ST^d_s = \frac{|TP_d \cap TP_s|}{|TP_d|}, \tag{7}$$

Here, $|TP_d|$ and $|TP_s|$ respectively denote the number of trading partners of d and s in their data trading records, and $|TP_d \cap TP_s|$ is the number of common trading partners between d and s. The information of trading partnership can be discovered in DTB.

C. DATA VALUE EVALUATION AND PRICING MODELS

Accurately evaluating data values and appropriately deciding data prices are two important issues for demanders and suppliers to achieve successful data trading. Generally, suppliers would like to obtain the most profit, which might decrease the purchase intention [14], and demanders want to buy the data commodity with better quality at an acceptable price. Thus, for demanders, TIDES proposes the client-centric data value evaluation model to assess data values and further determine what to buy based on data values and data prices. To ensure a data trading reliability, the trustworthiness of the data supplier will also be taken into account. For suppliers, TIDES proposes an intelligent pricing model based on an enhanced game-theory method.

1) CLIENT-CENTRIC DATA VALUE EVALUATION MODEL FOR DEMANDERS

The market value of a data asset is usually related to its quality satisfaction (QS) degree with respect to the requirements of the demander. The QS of a data asset, DA, with respect to the quality requirements of demander d, QS_d^{DA}, is given by (8):

$$QS_d^{DA} = \min_{k \in QA} \left\{ \frac{q_s^{DA}(k)}{rq_d^{DA}(k)} \right\}, \tag{8}$$

where QA denotes a set of quality attributes such as precision, granularity and integrity for a data asset, $q_s^{DA}(k)$ is the quality value of attribute k of DA, announced by its supplier s in a smart contract, and $rq_d^{DA}(k)$ is the required quality value of d on k for DA. According to the evaluation result of QS_d^{DA}, the value of DA, v_d^{DA}, for demander d can be derived as given by (9).

$$v_d^{DA} = \begin{cases} p_M^{DA} \cdot Trust_s^d, & \text{if } QS_d^{DA} \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where p_M^{DA} is the average selling price of data commodities of the same type and quality as DA in TIDES. Since all trading information is recorded, the selling prices of all data sets can be inquired in data trading KB, and p_M^{DA} can be regarded as the objective value of DA. For a demander, the value of a data set is also influenced by the trustworthiness of its data supplier. Thus, as in (9), $Trust_s^d$ which is the trustworthiness of s for d, is also taken into account. Note that QS_d^{DA} must be greater than one because otherwise, DA would be disqualified for the demander. As concluded in [18] and [27], demanders intend to purchase the data commodity with higher C-P ratio. Thus, TIDES estimates the C-P ratio if d purchases DA as follows,

$$CP_d^{DA} = \begin{cases} \frac{v_d^{DA}}{P_s^{DA}}, & \text{if } P_s^{DA} < PL_d^{DA} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where P_s^{DA} is the price of DA set by s, and PL_d^{DA} is the acceptable price of DA set by d. After PL_d^{DA} is determined, it will be delivered by d to its DTA agent for further usage. For simplification, it is supposed that d will assign the same acceptable price to the data commodities with the same type and quality.

In TIDES, a demander first defines the acceptable price for the data asset that it wants to purchase before performing the client-centric data value evaluation model. Then, it consults the data value evaluator of its DTA agent to compute the C-P ratio of each candidate data commodity. The evaluator will refer to the value of p_M^{DA} from the data trading KB and the value of $Trust_s^d$ from the client DB. These two values are maintained by the KB maintainer which continuously analyzes the trading records and market feedback information in TIDES. Then, the demander can decide whether it purchases DA or not by referring to CP_d^{DA}.

2) GAME-THEORY-BASED PRICING MODEL FOR SUPPLIER

To construct a smart contract for selling an IoT data asset, DA_s, the supplier s consults with its DTA. Its smart contract constructor will ask for a suggestion from the DTA's data price analyzer about the price of DA_s, and then the analyzer will refer to the data trading KB to respond this inquiry. In addition, the DTA's KB maintainer employs the following game-theory-based pricing model to update the data trading KB by referring to the historical information of data trading over TIDES.

For a supplier, both the demand and supply statuses and the bids from competitors significantly influence the probability of successful trading. Thus, an appropriate pricing strategy, such as the game-theory-based pricing model, should take the market situation into account. However, as mentioned in Section II, both the Stackelberg game model and the bargaining game model are impractical in IoT data trading. Traditional non-cooperative game models are more efficient but usually lead to a price war. Hence, by combining the economic-based pricing concept and the non-cooperative game model, TIDES proposes an enhanced game-theory-based pricing model that not only considers the market situation but also avoids low price competition. The proposed pricing model is introduced in the following section.

Similar to the cost model, the price of any data commodity offered by the supplier, s, can be formulated into a general form as described in (11):

$$P_s^{DA} = cost_s^{DA} \cdot \left(1 + m_s^{DA}\right), \quad m_s^{DA} > 0 \tag{11}$$

where $cost_s^{DA}$ denotes the cost spent to acquire and maintain a data set DA, and m_s^{DA} is the margin that s would like to obtain from selling DA. The main goal of TIDES is to find an m_s^{DA} that makes the decided selling price of DA generate the most profit for s. However, P_s^{DA} considers only the expectation of the data supplier. Referring to the consumer perceived model, our pricing model also takes the willingness of demanders into account. Since the intention that a demander purchases a commodity is related to its C-P ratio, the supplier should estimate the C-P ratio for its data commodity to be sold in the market by (12):

$$CP_{est}^{DA} = \frac{P_M^{DA}}{P_s^{DA}} \tag{12}$$

where p_M^{DA} can be obtained from the data trading KB. Then, based on [18], our pricing model proposes an exponential-based S function as in (13) for the supplier s to estimate the purchase intention of demanders to buy DA:

$$PI_s^{DA} = \left(1 + e^{-\left(\frac{CP_{est}^{DA} - \beta}{\gamma}\right)}\right)^{-1} \tag{13}$$

where γ and β are two pre-defined system parameters.

Then, the probability that a demander decides to purchase DA can be derived as in (14) by referring to the purchase intention of all data commodities, which have the same type as DA, from other suppliers,

$$Pr_s^{DA} = PI_s^{DA} \Big/ \sum_{k=1}^{n} PI_k^{DA} \tag{14}$$

where PI_k^{DA} denotes the purchase intention to buy a data commodity whose type is the same as that of DA and generated by supplier k, and n is the number of this kind of supplier.

It can be estimated by the data trading KB by referring to the historic pricing records. In addition, because TIDES tends to provide demanders with higher quality data, a calibrating function that combines both data quality and purchase probability is further provided in (15),

$$CPr_s^{DA} = \frac{PI_s^{DA}/Q_s^{DA}}{\sum_{k=1}^{n} PI_k^{DA}/Q_k^{DA}} \tag{15}$$

where Q_i^{DA} denotes the minimal value of all quality attributes in a data commodity whose type is the same as that of DA and supplied by i. Then, the supplier s can derive the expected profit according to the calibrated probability as in (16),

$$AEP_s^{DA} = cost_s^{DA} \cdot m_s^{DA} \cdot CPr_s^{DA} \tag{16}$$

Additionally, the margin that can create the most profit for s by selling DA, called $m_{s,max}^{DA}$, can be derived as follows,

$$m_{s,max}^{DA} = \underset{CSW_y \in Y^*}{\operatorname{argmax}} AEP_s^{DA} = \underset{CSW_y \in Y^*}{\operatorname{argmax}} cost_s^{DA} \cdot m_s^{DA} \cdot CPr_s^{DA} \tag{17}$$

Moreover, as malicious demanders could return commodities for no reason, suppliers might suffer inestimable losses. To prevent this type of badmouth attack, suppliers can set a constraint in smart contracts to exclude untrustworthy demanders.

D. DISPUTE ARBITRATION MODEL FOR DATA TRADING

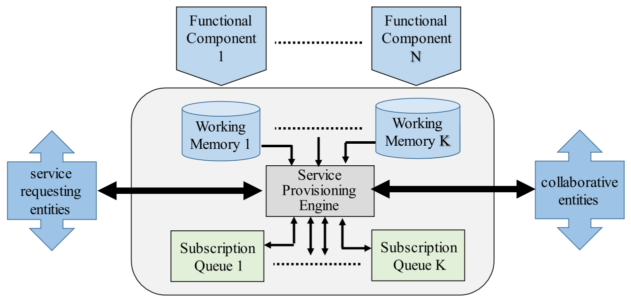
Disputes are always issued by the demanders. Three types of disputes might occur in a data transaction: get_wrong_key, miss_key, and get_wrong_data. The TDA entities contain two functional modules, key judge and data judge, to arbitrate these disputes.

1) KEY JUDGMENT MODEL

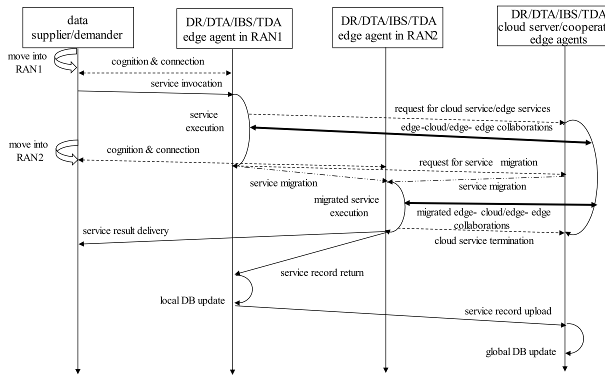
After the demanders deliver their payments, they will wait for the data wrapped key and use it to generate the decryption key. However, the suppliers might fail to deliver the key or deliver a wrong key that incurs a dispute. In the miss_key dispute, if the demanders apply the blockchain payment mode to deliver their payments, the data wrapped key should be stored on blockchain. Thus, it is easy for TDA to determine the miss_key situation and then perform the refund procedure by sending a smart contract. In the get_wrong_key dispute, the demanders need to transmit the received decryption key to TDA. Then, TDA can discover whether the data wrapped key is correct and the decryption key can decrypt the data. Otherwise, TDA will activate the refund procedure.

2) DATA JUDGMENT MODEL

It is also possible that the received data commodity is not what the demander wants. In such a case, the demander can transmit this commodity to TDA, which will conduct the arbitration based on the information in IBS. If the argument is approved, the refund procedure will then be performed.

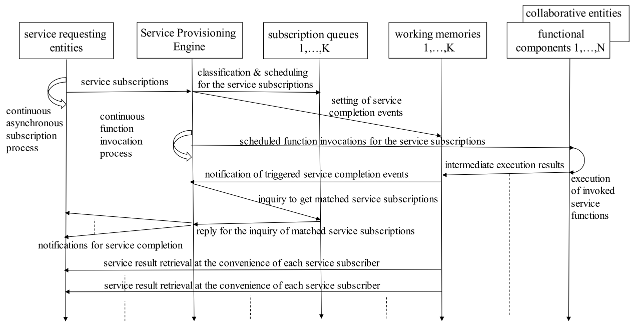


(a) The structure of the service provisioning mechanism.



data supplier/demander
DR/DTA/IBS/TDA edge agent in RAN1
DR/DTA/IBS/TDA edge agent in RAN2
DR/DTA/IBS/TDA cloud server/cooperative edge agents
move into RAN1
cognition & connection
service invocation
service execution
request for cloud service/edge services
edge-cloud/edge- edge collaborations
move into RAN2
cognition & connection
request for service migration
service migration
service migration
migrated service execution
migrated edge- cloud/edge- edge collaborations
service result delivery
cloud service termination
service record return
local DB update
service record upload
global DB update

(b) The subscription-based service provisioning protocol.



service requesting entities
Service Provisioning Engine
subscription queues 1,…,K
working memories 1,…,K
collaborative entities
functional components 1,…,N
service subscriptions
classification & scheduling for the service subscriptions
setting of service completion events
continuous asynchronous subscription process
continuous function invocation process
scheduled function invocations for the service subscriptions
intermediate execution results
notification of triggered service completion events
execution of invoked service functions
inquiry to get matched service subscriptions
reply for the inquiry of matched service subscriptions
notifications for service completion
service result retrieval at the convenience of each service subscriber
service result retrieval at the convenience of each service subscriber

(c) The service handover and coordination protocol.

FIGURE 5. The service provisioning mechanism of each functional entity in TIDES.

V. OPERATIONAL FLOWS AND PROTOCOLS OF TIDES

A. SERVICE PROVISIONING, HANDOVER AND COORDINATION SCHEME OF TIDES

TIDES adopts the service-oriented architecture (SOA). As shown in Fig. 5(a), all entities employ a subscription-based service provisioning mechanism to realize various data trading services in a mobile environment. The service provisioning engine (SPE) is used to implement the protocols illustrated in Figs. 5(b) and (c). Each subscription queue records the requests issued from other entities for a specific service function until the subscribed services are completed or aborted, and it has a corresponding work memory to store the intermediate results generated by the functional components invoked to implement the subscribed services. These functional components may further collaborate with other functional entities to complete their missions.

The service provisioning protocol is illustrated in Fig. 5(b). For a functional entity, all the potential collaboration entities can raise their service requests to its SPE at any time. Whenever a new request is subscribed, the SPE classifies this subscription, places it into the corresponding queue, and sets the service completion event to be matched with the intermediate service execution results stored in the working memory. Subscriptions with the same requirement specification will be clustered as an aggregated subscription to deliver the service result concurrently. The SPE persistently schedules and invokes the responsible functional components to implement the subscribed services. The invoked functional components will place their execution results in the corresponding working memory. Whenever a service completion event is triggered by the content of working memory, the SPE will be notified and then collect the satisfied subscriptions from the corresponding queue. The original service requesting entities of these subscriptions will be notified to retrieve the service results from the working memory at its convenience. Obviously, according to the protocol of Fig. 5(b), the SPE can support asynchronous and multi-threaded service provisioning.

In TIDES, functional entities are deployed in both cloud centers and ME hosts, but data suppliers, demanders, or their gateways may be mobile devices. Thus, the mobility issue must be solved. Fig. 5(c) shows the service handover and coordination protocol executed by SPE to successfully deliver each service to the requesting device even if it has moved into another RAN. This protocol is the underlying mechanism of Fig. 5(b), which is automatically executed to support service handover based on the operational structure of the ETSI MEC. When service handover occurs, the SPE also needs to maintain the continued operation for the collaborations between its host edge agent and the other edge agents of different functional entities or the cloud server of the same functional entity. As shown in Fig. 5(c), the locations of mobile entities in TIDES are tracked. When a mobile device was detected moving from one RAN into another RAN, the SPE of each functional entity that provides service to this mobile device will be notified to initiate a service migration procedure. The ongoing service task and the associated context information will be transferred from the original edge agent to the current edge agent of the same functional service. The migration must include those ongoing cross-entity collaborations. To maintain consistency, the service record is also sent back to the initial service-providing edge agent.

All the following workflows are based on the service provisioning mechanism and protocols in Fig. 5.

B. DTA AND DR COMMISSION PROCESSES

When a data supplier, the demander or IoT gateway is deployed in TIDES, it will sign commission contracts with one or more DTAs and one or more DRs for future data trading needs. Fig. 6 illustrates the appointment processes as follows.

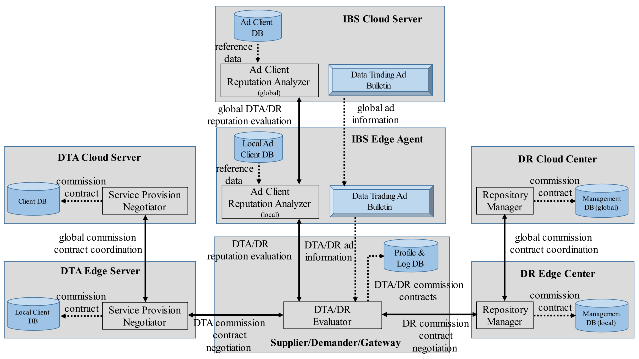


FIGURE 6. The DTA/DR commission process.

DTA Commission Process:

Step 1: The DTA/DR evaluator of the consignor (the supplier, demander or IoT gateway) surveys the DTAs that advertise their services on the bulletin of the IBS edge agent, IBS^e, in the same RAN area as the consignor.

Step 2: According to the survey result, the consignor's DTA/DR evaluator selects some qualified candidate DTAs that their service specifications best match the consignor's requirements for the service charge, service level, security policy, etc.

Step 3: The consignor's DTA/DR evaluator collects the reputation information of the selected candidates from the IBS^e's ad client reputation analyzer, excludes inappropriate candidates, ranks the remaining candidates, and finally selects one or more top-ranked candidates to negotiate the commission contract for data trading. The IBS^e can independently provide the reputation information of candidate DTAs in local data trading. If the consigner intends to proceed with global data trading, then the IBS^e will ask for the support from the cloud server, IBS^c, to provide the reputation information of candidate DTAs on global data trading.

Step 4: For each candidate DTA selected at Step 3, the consignor's DTA/DR evaluator negotiates with the service provision negotiator of this DTA's edge agent deployed in the same RAN as the consignor. However, the edge negotiator will work with the same DTA's cloud negotiator to achieve a final contract agreement.

Step 5: If one or more contract agreements are achieved, the consignor's DTA/DR evaluator signs commission contracts with the target DTAs, and then terminates this procedure. Otherwise, if there is no agreement achieved, the consignor's DTA/DR evaluator decides to go back to Step 1 or terminate this procedure. At the consignor side, all signed contracts are stored in the profile and log DB. At the consignee side, the client DB of DTA cloud server will store all commission contracts, whereas the local client DB of

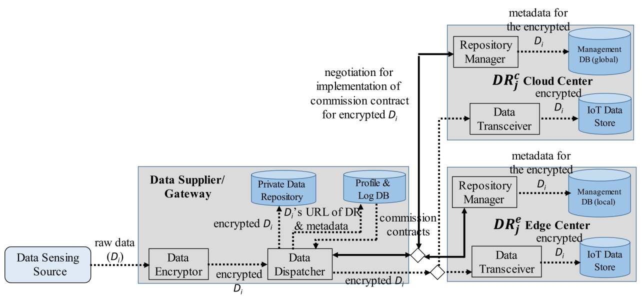


FIGURE 7. The data collection and deposit process.

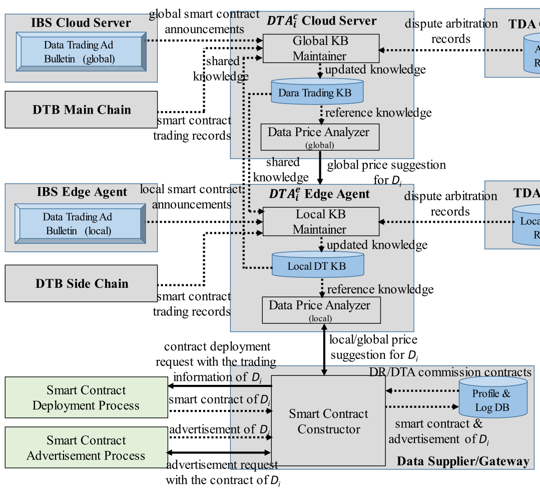

IBS Cloud Server
Data Trading Ad Bulletin (global)
global smart contract announcements
DTA_i^c Cloud Server
Global KB Maintainer
Data Trading KB
Data Price Analyzer (global)
dispute arbitration records
TDA Cloud Server
Arbitration Record DB
DTB Main Chain
smart contract trading records
shared knowledge
updated knowledge
reference knowledge
global price suggestion for D_i
IBS Edge Agent
Data Trading Ad Bulletin (local)
local smart contract announcements
DTA_i^e Edge Agent
Local KB Maintainer
Local DT KB
Data Price Analyzer (local)
TDA Edge Agent
Local Arbitration Record DB
DTB Side Chain
local/global price suggestion for D_i
contract deployment request with the trading information of D_i
Smart Contract Deployment Process
Smart Contract Advertisement Process
smart contract of D_i
advertisement of D_i
advertisement request with the contract of D_i
Smart Contract Constructor
DR/DTA commission contracts
Profile & Log DB
smart contract & advertisement of D_i
Data Supplier/Gateway

FIGURE 8. The smart contract construction process.

DTA edge agent stores only the commission contracts of which the consignors are in the same RAN.

The process of commissioning DRs is similar to the process of commissioning DTAs, while the consignor's DTA/DR evaluator will negotiate with the repository manager of DRs.

C. DATA COLLECTION AND DEPOSIT PROCESS

Fig. 7 illustrates the data collection and deposit process of a data host (a data supplier or its gateway) in TIDES.

Step 1: The data encryptor encrypts the acquired data set D_i according to (1).

Step 2: The data dispatcher decides the storage scheme for the encryptedD_i, based on the host's data management policy. If the private storage scheme is adopted due to high privacy concern, go to Step 3. On the other hand, if the third-party storage scheme is adopted to promote data aggregation and trading, go to Step 4.

Step 3: (Private storage scheme) The data dispatcher deposits the encrypted D_i in the private data repository, and DR_s, logs the URL of DR_s in the profile and log DB and then terminates this process.

Step 4: The data dispatcher further considers the sales policy for D_i. If the local sales policy is adopted, go to Step 5. Otherwise, go to Step 6.

Step 5: (DR assignment for edge storage scheme) The data dispatcher negotiates with those DR^e edge centers that have signed a commission contract with the host, in the sequence of their ranks until the destination DR^e is decided or all candidate $DR^e s$ are run out. If no DR^e can provide storage service for the encrypted D_i, the host may have the following alternative decisions: (a) change to the private storage scheme and go back to Step 3, (b) re-invoke the DR commission process to delegate new DR and then repeat this step, (c) change to the cloud storage scheme and go to Step 6.

Each candidate DR_j^e performs the following actions.

Step 5a: The data dispatcher inquires the DR_j^e's repository manager of the service availability. If the repository manager admits this service request, go to Step 5b. Otherwise, go back to Step 5 to determine the next candidate DR^e.

Step 5b: The data dispatcher submits the profile of D_i to the local management DB of DR_j^e. A copy of the profile will also be saved in the global management DB of the same DR's cloud center (DR_j^c).

Step 5c: The data dispatcher delivers the encrypted D_i to the IoT data store of DR_j^e, logs the URL of DR_j^e in the profile and log DB, and then terminates this process.

Step 6: (DR assignment for the cloud storage scheme) The assignment process is similar to Step 5, but those commissioned $\mathrm{DR^c}$ cloud centers participate in negotiation. If no $\mathrm{DR^c}$ can provide storage services for $\mathrm{D_i}$, then the host may have the following alternative decisions: (a) change to the edge storage scheme and go back to Step 5, (b) change to the private storage scheme and go back to Step 3, (c) re-invoke the DR commission process to delegate a new DR and then repeat this step. During the negotiation, the candidate DR's edge center in the same RAN as the data host will play the role of the service proxy between the host and the candidate cloud DR.

Each candidate DR_j^c performs the same actions on the cloud side as the DR_j^e in Steps 5*a*-*c* and invokes the repository manager of DR_j^e as a proxy.

D. SMART CONTRACT CONSTRUCTION PROCESS

Fig. 8 illustrates the process by which the supplier's smart contract constructor executes to establish the smart contract, as shown in Fig. 3, for data set D_i.

Step 1: (Activate DTA_i) Negotiate with the commissioned DTAs in the sequence of their ranks until the destination DTA, say DTA_i, is decided or all candidate DTAs are run out. If a DTA_i is available, activate it and then go to the next step. Otherwise, if no DTA is available, the supplier either waits until some DTA is available (this process is temporally pended) or drafts the smart contract for D_i by itself (this process is terminated).

Step 2: Employ the DTA_i^e's data price analyzer to obtain the reference price of D_i according to the aforementioned data pricing model. If the local sales policy for D_i is adopted, the reference price is estimated with the parameter values of the data pricing model from the DTA_i^e's data trading KB. However, if the global sales policy for D_i is adopted, then the inquiry for the reference price is forwarded to the DTA_i^c's data price analyzer which estimates the reference price of D_i by referring to the DTA_i^c's data trading KB.

Step 3: Based on the reference price of D_i, decide the final price of D_i to be announced in the contract according to the supplier's sales strategies and experiences that are accumulated in the profile and log DB.

Step 4: Fills the values of all fields in the smart contract of D_i based on the information recorded in the supplier's profile and log DB and the execution results achieved at previous steps.

Step 5: Activate the smart contract deployment process to compile the complete smart contract of D_i.

Step 6: (optional) Activate the smart contract advertising process for the smart contract of D_i.

Step 7: Store the smart contract and advertisement of D_i into the supplier's profile and log DB and then terminate this process.

The DTA_i^e's data trading KB is continuously updated by its local KB maintainer based on the analytical results of the smart contract announcements on the IBS^e, the data trading records in the DTB local chain (DTB^e), and the local dispute arbitration records. The maintainer also refers to the knowledge of interest from the DTA_i^c's data trading KB to fine tune the local KB. DTA_i^c applies the same mechanism for maintaining its global data trading KB by referring to the smart contract announcements on IBS^c, the data trading records in the DTB main chain (DTB^c), and the global dispute arbitration record.

E. SMART CONTRACT DEPLOYMENT PROCESS

Fig. 9 shows the details of this process. It is a subprocess of Fig. 8. To initiate this process, the supplier's smart contract constructor sends a contract deployment request with the trading information of D_i to the commissioned DTA_i^e.

Step 1: DTA_i^e reviews the supplier's request; if a global contract deployment is requested, go to Step 6. Otherwise, go to the next step.

Step 2: The DTA_i^e's contract compiler generates the bytecode and ABI according to the trading information of D_i, and sends them to the contract deployer.

Step 3: DTB^e responds the contract deployer with a contract ID.

Step 4: The contract deployer deploys the bytecode of the D_i's local smart contract to DTB^e.

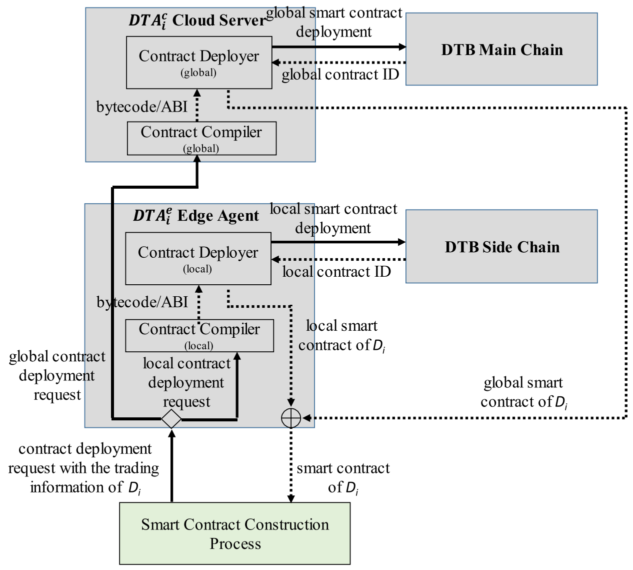

DTA_i^c Cloud Server
global smart contract deployment
Contract Deployer (global)
global contract ID
DTB Main Chain
bytecode/ABI
Contract Compiler (global)
DTA_i^e Edge Agent
local smart contract deployment
Contract Deployer (local)
local contract ID
DTB Side Chain
bytecode/ABI
Contract Compiler (local)
local smart contract of D_i
global contract deployment request
local contract deployment request
global smart contract of D_i
contract deployment request with the trading information of D_i
smart contract of D_i
Smart Contract Construction Process

FIGURE 9. The smart contract deployment process.

Step 5: The contract deployer sends the complete local smart contract of D_i back to the supplier's smart contract constructor, and then terminates this process.

Step 6: The trading information of D_i is sent to the DTA_i^c's contract compiler through DTA_i^e. Then, the following actions used to generate the global smart contract of D_i are the same as those described in Steps 2~5. However, the smart contract of D_i is deployed in DTB^e.

F. SMART CONTRACT ADVERTISING PROCESS

Fig. 10 shows the details of this process. It is a subprocess of Fig. 8. Initially, the supplier's smart contract constructor sends an advertisement request with the smart contract of D_i to the commissioned DTA_i^e.

Step 1: DTA_i^e reviews the supplier's request; if a global advertisement is requested, go to Step 6. Otherwise, go to the next step.

Step 2: The DTA_i^e's advertisement deployer sends an advertisement request with the smart contract of D_i to the IBS^e's ad planner.

Step 3: The IBS^e's ad planner evaluates the D_i's smart contract and creates an advertisement for D_i by referring to the ad KB which is maintained by the ad performance evaluator according to the analytical results for the trading records of the smart contracts in DTB^e.

Step 4: The IBS^e's ad planner gets approval for the advertisement of D_i from the DTA_i^e's advertisement deployer.

Step 5: The IBS^e's ad planner announces the advertisement of D_i on the ad bulletin and logs it in the ad client DB. Then, this process is terminated.

Step 6: The D_i's smart contract is sent to the DTA_i^c's advertisement deployer through DTA_i^e. Then, the following actions to generate the global advertisement of D_i are

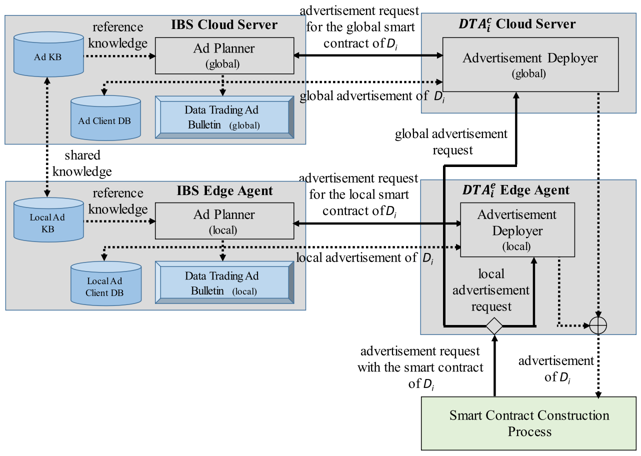

IBS Cloud Server
Ad KB
reference knowledge
Ad Planner (global)
Ad Client DB
Data Trading Ad Bulletin (global)
advertisement request for the global smart contract of D_i
global advertisement of D_i
DTA_i^c Cloud Server
Advertisement Deployer (global)
global advertisement request
shared knowledge
IBS Edge Agent
Local Ad KB
Ad Planner (local)
Local Ad Client DB
Data Trading Ad Bulletin (local)
advertisement request for the local smart contract of D_i
local advertisement of D_i
DTA_i^e Edge Agent
Advertisement Deployer (local)
local advertisement request
advertisement request with the smart contract of D_i
advertisement of D_i
Smart Contract Construction Process

FIGURE 10. The smart contract advertising process.

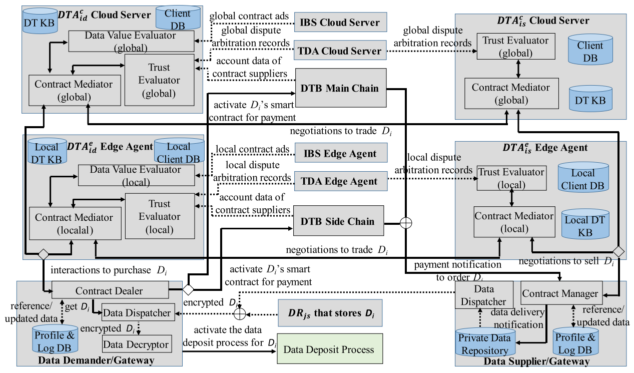

DTA_{id}^c Cloud Server
DT KB
Client DB
Data Value Evaluator (global)
Trust Evaluator (global)
Contract Mediator (global)
global contract ads
global dispute arbitration records
account data of contract suppliers
IBS Cloud Server
TDA Cloud Server
DTB Main Chain
activate D_i's smart contract for payment
negotiations to trade D_i
DTA_{is}^c Cloud Server
Trust Evaluator (global)
Contract Mediator (global)
DTA_{id}^e Edge Agent
Local DT KB
Local Client DB
Data Value Evaluator (local)
Trust Evaluator (local)
Contract Mediator (localal)
local contract ads
local dispute arbitration records
account data of contract suppliers
IBS Edge Agent
TDA Edge Agent
DTB Side Chain
DTA_{is}^e Edge Agent
Trust Evaluator (local)
Contract Mediator (local)
Local DT KB
interactions to purchase D_i
activate D_i's smart contract for payment
payment notification to order D_i
negotiations to sell D_i
Contract Dealer
Data Dispatcher
Data Decryptor
Profile & Log DB
reference/ updated data
get D_i
encrypted D_i
Data Demander/Gateway
DR_{is} that stores D_i
activate the data deposit process for D_i
Data Deposit Process
Data Dispatcher
Contract Manager
data delivery notification
Private Data Repository
Profile & Log DB
Data Supplier/Gateway

FIGURE 11. The trust-aware data trading process.

the same as those at Steps 2∼5. However, the advertisement of D_i is announced on IBS^c's the ad bulletin.

G. TRUST-AWARE DATA TRADING PROCESS

Fig. 11 illustrates the entire data trading flow over TIDES, which can be divided into the qualified contract sifting subprocess and the trading approval, payment and data delivery subprocess. When a demander is going to purchase data set D_i, it activates a DTA_{id} by a procedure similar to Step 1 of the smart contract construction process, and then the two subprocesses are executed as follows.

1) QUALIFIED CONTRACT SIFTING

Step 1: The demander's contract dealer sends the data purchase requirements for D_i to the activated DTA_{id}^e.

Step 2: The DTA_{id}^e's contract mediator reviews the demander's requirements; if global trading is requested, then go to Step 6. Otherwise, go to the next step.

Step 3: The DTA_{id}^e's data value evaluator collects the data contract advertisements that meet the demander's requirements from the local IBS^e, and evaluates the collected data trading contracts based on the data value evaluation model proposed in Section IV.C. The evaluation results are then sent to the contract mediator as a reference for deciding the qualified data contracts to proceed with data transactions.

Step 4: The DTA_{id}^e's trust evaluator employs the model proposed in Section IV.B to estimate the suppliers' trustworthiness of the contracts collected at Step 3. The evaluation results are then sent to the contract mediator as a reference for deciding the qualified data contracts to proceed with data transaction.

Step 5: According to the evaluation results from Steps 3 and 4, the DTA_{id}^e's contract mediator abandons the contracts with untrustworthy suppliers, and then ranks the remaining contracts according to their valuations. If at least one qualified contract exists, then go to the trading approval subprocess. Otherwise, the contract mediator notifies the demander's contract dealer for no qualified data contract, and goes to Step 9.

Step 6,7,8: Respectively similar to Steps 3, 4 and 5, but executed by the cloud-side entities and referring to cloud-side information.

Step 9: The demander's contract dealer makes one of the following two decisions:
(1) Abort the data purchase request and terminate the data trading process.
(2) Change the data purchase policy from local trading to global trading and go to Step 6 (this decision is considered only when the previous step is Step 5).

2) TRADING APPROVAL, PAYMENT AND DATA DELIVERY

The following procedure describes the operational flow for local data trading. Global data trading has a similar operational flow. This subprocess is executed for each of the qualified contracts according to their ranks decided at the previous subprocess until the data purchase request is approved by a supplier or declined by all suppliers of the qualified contracts.

Step 1: The DTA_{id}^e's contract mediator obtains the contact window DTA_{is}^e from the advertisement of the target contract.

Step 2: The DTA_{id}^e's contract mediator notifies the DTA_{is}^e's contract mediator for the purchase request for D_i.

Step 3: The DTA_{is}^e's trust evaluator employs the evaluation model proposed in Section IV.B to estimate the demander's trustworthiness on data trading.

Step 4: The DTA_{is}^e's contract mediator notifies the supplier for the evaluation result obtained in Step 3.

Step 5: According to the evaluation result, the supplier's contract manager approves or declines the request, and then notifies DTA_{is}^e of the decision.

Step 6: The DTA_{is}^{e}'s contract mediator notifies DTA_{id}^{e} of the supplier's decision. Then, the DTA_{id}^{e}'s contract mediator forwards this decision to the demander.

Step 7: If the request is declined, the demander terminates this process; otherwise, it executes one of the following two actions:
(1) If the blockchain payment mode is adopted, the demander's contract dealer activates the corresponding smart contract in DTB^{e} to pay for purchasing D_i. Then, DTB^{e} notifies the supplier with a payment completion message.
(2) If the off-chain payment mode is adopted, then the demander's contract dealer contacts the supplier's contract manager to complete the payment directly.

Step 8: After receiving the payment completion message, the supplier's contract manager generates the data wrapped key of D_i according to (2). Then, if the blockchain payment mode is adopted, the data wrapped key will be updated on blockchain by smart contract; otherwise, the data wrapped key will be sent to the demander through the contract mediators of DTA_{is}^{e} and DTA_{id}^{e}. At the same time, the private or third-party DR that stores D_i is instructed under the supplier to deliver the encrypted D_i to the demander.

Step 9: The demander's data decryptor decrypts the received D_i by applying (3) and (4) with the supplier's public key announced in the dealt smart contract.

Step 10: The demander's contract dealer judges whether the decrypted D_i is the desired ones. If yes, this process is terminated. Otherwise, the demander's dispute arbitration negotiator will apply for the dispute arbitration process through TDA^{e} to refund the payment.

H. DISPUTE ARBITRATION PROCESS

Fig. 12 illustrates the details of this process as a subprocess of the trust-aware data trading process. The following procedure describes only the operational flow for local data trading. Global data trading has a similar operational flow.

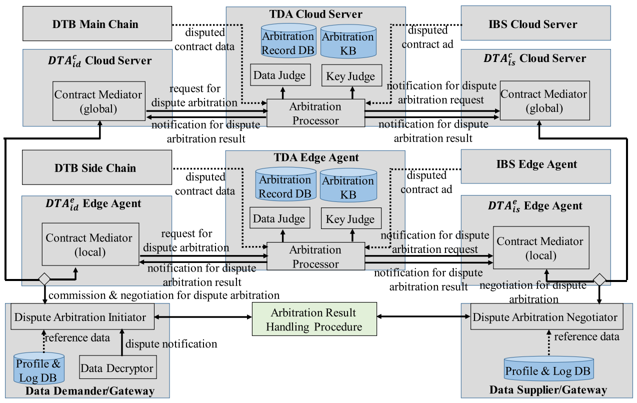
Step 1: The demander's dispute arbitration initiator asks its DTA_{id}^{e} to file an arbitration request for the trading of D_i.

Step 2: The DTA_{id}^{e}'s contract mediator files an arbitration request to TDA^{e} with related evidence.

Step 3: The TDA^{e}'s arbitration processor notifies the supplier of D_i of the arbitration request through DTA_{is}^{e}, and collects the other evidence from IBS^{e} and DTB^{e}.

Step 4: The TDA^{e}'s arbitration processor activates the key judge and data judge and gives the collected evidence.

Step 5: The TDA^{e}'s key judge and data judge employ the judgment models proposed in Section IV.D to arbitrate the dispute, and then reply to the arbitration processor.



DTB Main Chain
disputed contract data
TDA Cloud Server
Arbitration Record DB
Arbitration KB
disputed contract ad
IBS Cloud Server
DTA_{id}^{c} Cloud Server
Contract Mediator (global)
request for dispute arbitration
Data Judge
Key Judge
notification for dispute arbitration request
DTA_{is}^{c} Cloud Server
Contract Mediator (global)
Arbitration Processor
notification for dispute arbitration result
notification for dispute arbitration result
DTB Side Chain
disputed contract data
TDA Edge Agent
Arbitration Record DB
Arbitration KB
disputed contract ad
IBS Edge Agent
DTA_{id}^{e} Edge Agent
Contract Mediator (local)
request for dispute arbitration
Data Judge
Key Judge
notification for dispute arbitration request
DTA_{is}^{e} Edge Agent
Contract Mediator (local)
Arbitration Processor
notification for dispute arbitration result
notification for dispute arbitration result
negotiation for dispute arbitration
commission & negotiation for dispute arbitration
Dispute Arbitration Initiator
Arbitration Result Handling Procedure
Dispute Arbitration Negotiator
reference data
dispute notification
reference data
Profile & Log DB
Data Decryptor
Profile & Log DB
Data Demander/Gateway
Data Supplier/Gateway

FIGURE 12. The dispute arbitration process.

Step 6: According to the received judgment results, the TDA^{e}'s arbitration processor makes the final arbitration for the dispute, records the result in the arbitration result DB, and notifies the demander and supplier through DTA_{id}^{e} and DTA_{is}^{e}.

Step 7: After receiving the arbitration result, the demander and the supplier must proceed to the arbitration result handling procedure as follows:
(1) If the dispute does not stand, the trading of D_i is reserved.
(2) If the miss_key or get_wrong_key dispute stands, the demander can ask the supplier to resend the data wrapped key or refund the payment.
(3) If the get_wrong_data dispute stands, the supplier refunds the payment.

VI. SIMULATION

A. SIMULATION SETTINGS

TIDES has been implemented in blade servers equipped with Intel Xeon E5-2620 2.1 GHz processors with 32 cores and 4 cores to simulate the operational environments of the cloud servers and MEC hosts, respectively. In all simulations, both the main chain and local chain of DTB are implemented by Ethereum technology; the former is executed in cloud servers and the latter in MEC hosts. The simulation environment is composed of 10 RAN areas, and each area contains ten ME hosts and several IoT gateways. Each area generates 50 local transactions per second for local data, and only ten percent of these local data will be further queried for global usage by default. Referring to the setting of the Ethereum public chain [28], each transaction requires approximately 320 bytes of storage, and each block contains 150 transactions at most. Additionally, *Geth1.6* is utilized to implement the Ethereum client. Twei is applied as the trading unit. Twei is a denomination of ether, ETH, which is the crypto-coin applied on the Ethereum network, and 1 Twei $= 10^{-6}$ Eth.

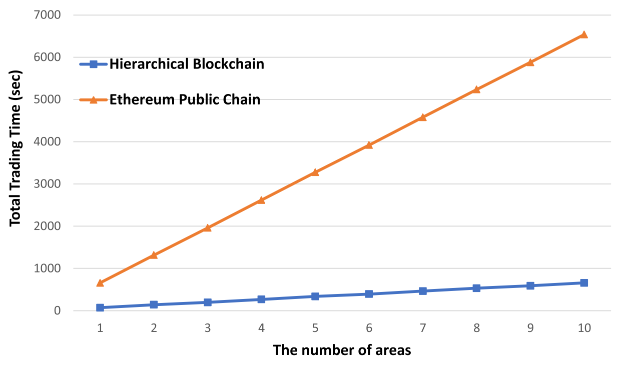


FIGURE 13. Comparison of total trading time under different numbers of areas.

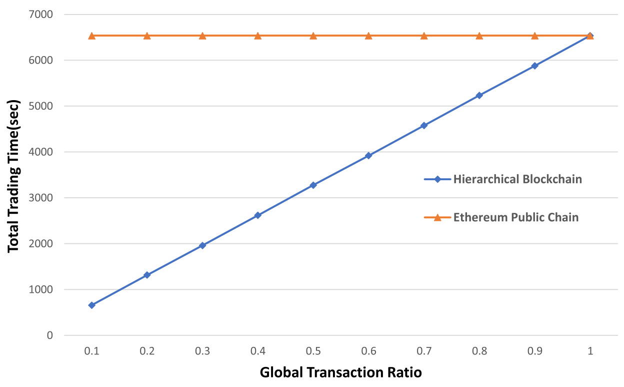


FIGURE 14. Comparison of total trading time under different global transaction ratios.

Two types of simulations are performed in the paper, including the system performance simulation and the trading efficiency simulation. The system performance simulation shows the improvement of TIDES in terms of trading time and storage costs. The trading efficiency simulation demonstrates that suppliers in TIDES can make higher profit. In the meantime, the data quality of purchased data and pricing efficiency can also be improved. Additionally, when the environment is malicious, TIDES can make more successful transactions than other pricing models because TIDES considers the trustworthiness of both demanders and suppliers.

B. SIMULATION FOR SYSTEM PERFORMANCE

In the following simulation, assume that each area acquires data to trade lasting for 140 seconds, respectively. Different from the Ethereum public chain, TIDES applies a hierarchical blockchain in which only the global transactions need to be carried out through the main chain and delivered around the world.

1) TRADING TIME

Generally, trading time is the most important criterion for performance evaluation. A smaller trading time means that the demanders can obtain their desired data commodities more quickly and that more transactions can be made during the same time period. The trading time includes the time to perform PoW operation and the time to deliver block to all full nodes. As estimated in [28], the Ethereum public chain spends 2 seconds on the PoW operation and 12 seconds on global block delivery. Thus, in the system performance simulation, each global transaction takes approximately 14 seconds. For a local transaction, 2 seconds are also assumed to be spent in the PoW operation and 1 second is spent on local block delivery.

The total trading time indicates that the time to complete all global transactions, which can be regarded as the capability of a trading system. As shown in Fig. 13, compared to the Ethereum public chain that spreads all transactions to all full nodes around the world even though most data transactions are done locally, the total trading time of the TIDES hierarchical blockchain is much smaller. It can be discovered that

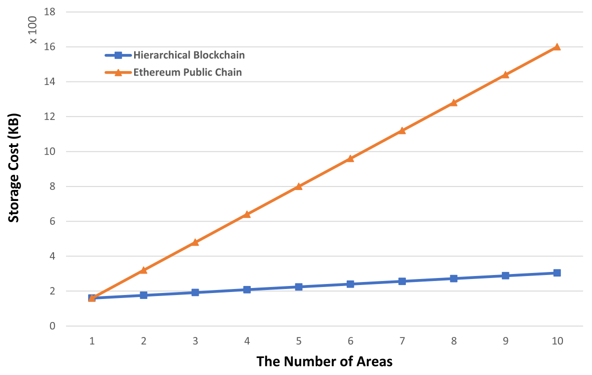


FIGURE 15. Comparison of storage costs for different numbers of areas.

the Ethereum public chain can easily become a performance bottleneck because too many transactions are required to be stored in global chain. Moreover, as the number of RAN areas increases, more transactions are made, which increases the difference between the hierarchical blockchain and the Ethereum public chain.

Fig. 14 shows the total trading time with respect to different global transaction ratios from 10% to 100%. Although the total trading time of the TIDES hierarchical blockchain grows and becomes close to the Ethereum public chain when the global transaction ratio increases, it is still much less than that of the Ethereum public chain in most cases. In practice, most transactions are made locally.

2) STORAGE COST

For an IoT data trading system, the storage cost is mainly spent in recording the IoT data transactions. Since most data transactions are made locally, it is a waste to log all transactions in a global chain. As shown in Fig. 15, the total storage cost of the TIDES hierarchical blockchain is always much lower than that of the Ethereum public chain. The enormous storage cost will severely increase the maintenance expense, which might lead to system monopolization. As a result, the reliability of the data trading system will be

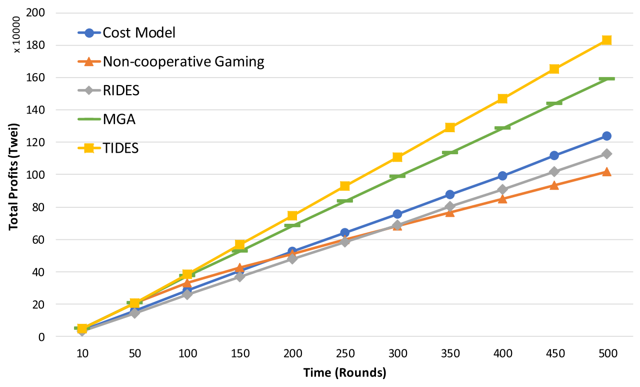


FIGURE 16. Comparison of total profit under different pricing models.

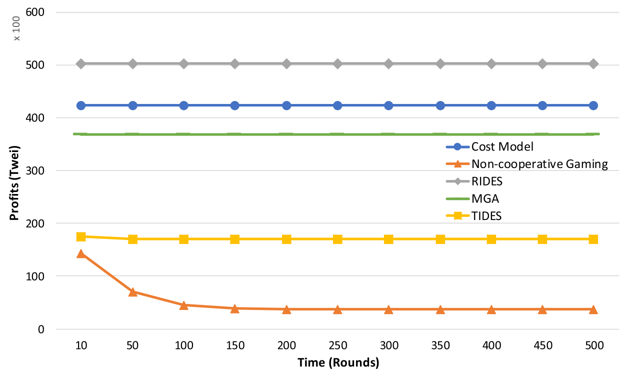


FIGURE 17. Comparison of average profit in a single transaction.

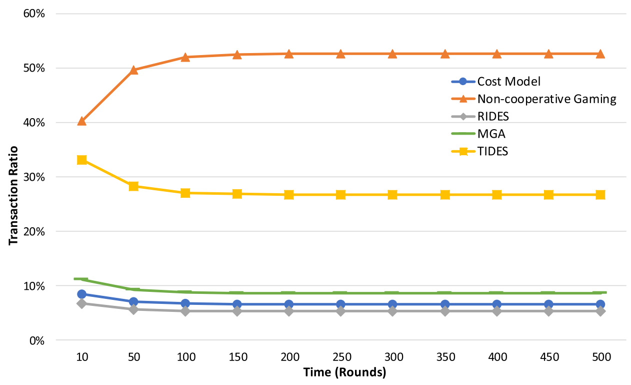


FIGURE 18. The transaction ratio of different pricing models.

compromised. Thus, the proposed hierarchical blockchain is more applicable as an IoT data trading system.

C. SIMULATION FOR TRADING EFFICIENCY

This simulation evaluates the data trading efficiency of various pricing models under two different settings for the trading environment, including an ideal one and a malicious one. The former indicates that all participants are honest and the latter assumes that malicious suppliers exist. In the simulation, five suppliers with different pricing models are used to provide data commodities continuously. The data quality is randomly set as low, medium or high. The costs of data commodities are set between 500 and 50,000 based on their data quality. The initial margin is randomly set between 1.0 and 2.0. The acceptable price of the demanders is set between 1,000 and 150,000 based on the quality of their desired data. The purchase intention is assumed to be related to the C-P ratio of data commodities.

1) TRADING IN AN IDEAL ENVIRONMENT

In this simulation, all suppliers are assumed to be honest. The five suppliers respectively apply the TIDES's pricing model, the cost model [15] with a static margin, the RIDES's [9] and MGA's [18] pricing models which ignore the market situation, and the conventional non-cooperative game model [20] which refers to the information of the competitors. For the last pricing model, if the supplier fails to reach a deal in this round, it will decrease its margin compared with the margin of a successful competitor.

As shown in Fig. 16, TIDES always obtains the higher total profit because it takes the market situation into consideration. The conventional non-cooperative game model obtains the least profit as it continuously decreases its margin.

Fig. 17 compares the average profits in a single transaction under different pricing models. It is clear that the cost model assigns a static margin, and therefore, the average profit in a single transaction is also fixed. In addition, as RIDES and MGA do not consider other competitors, their average profits remain the same. TIDES has lower profits in a single transaction than the cost model, RIDES and MGA because it considers the market situation and adjusts its margin to undercut all competitors in a timely manner. Even so, TIDES still achieves the highest total profit. For the non-cooperative game model, the profit in a single transaction decreases steadily, which indicates that it falls into the price war.

As shown in Fig. 18, although more than half the transactions are performed by the non-cooperative game model, its total profit is not as high as that of TIDES. By contrast, TIDES performs the second most transactions and obtains the most profit.

Moreover, the pricing efficiency denotes the profit obtained from per pricing effort, which is achieved by dividing the profit by the time spent on pricing. As shown in Fig. 19, TIDES has the best pricing efficiency although it takes some effort to derive an appropriate price. The cost model has the second best pricing efficiency due to its simplicity. Although RIDES is slightly faster than MGA, the latter makes more profit and thus has better pricing efficiency than RIDES. The non-cooperative game model has worse performance and the worst profit leading to the worst pricing efficiency.

In the simulation, although suppliers randomly generate data of three different qualities, demanders also randomly define a minimal quality limit. Hence, data of higher quality will be purchased more than that of lower quality. As shown in Fig. 20, demanders in TIDES purchase more high-quality data (55.4%) and less low-quality data (9.5%) than other

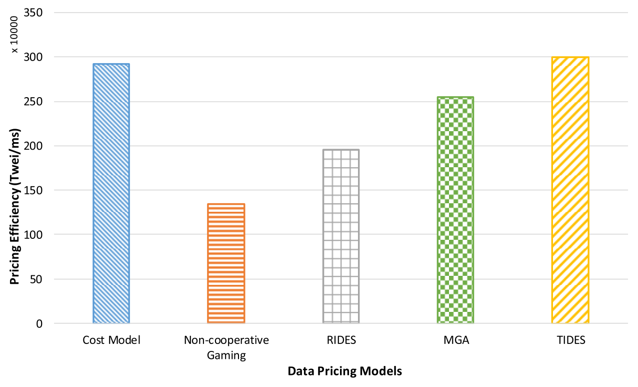


FIGURE 19. The pricing efficiency of different pricing models.

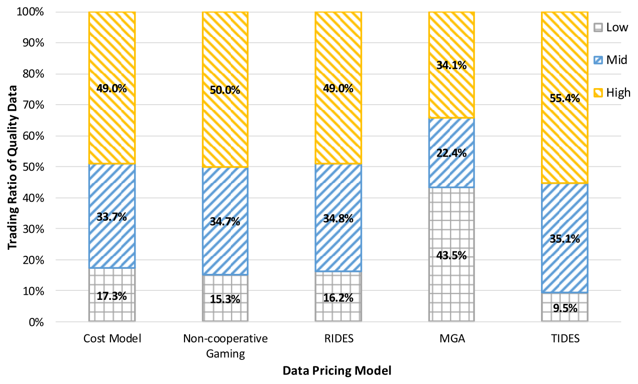


FIGURE 20. The trading ratio of various pricing models for different data qualities.

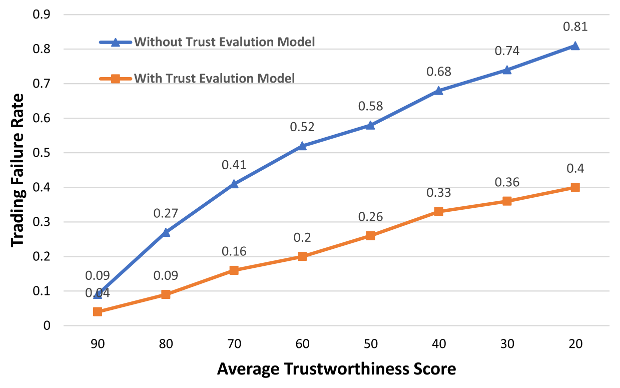


FIGURE 21. The trading failure rate with different average trustworthiness.

pricing models. Thus, it can be seen that TIDES provides not only the highest profits for suppliers but also better quality data for demanders; thus, the result is a win-win situation.

2) TRADING IN A MALICIOUS ENVIRONMENT

Generally, the demanders are eager to successfully obtain what they want. However, in a malicious environment, the suppliers might provide counterfeits, which leads to the failure of trade. To solve this problem, TIDES considers suppliers' trustworthiness to eliminate malicious suppliers. In this simulation, the trustworthiness of a supplier indicates the probability that this supplier is honest. As shown in Fig. 21, when the average trustworthiness of the suppliers is 90, which means that the trading environment is friendly, the trustworthiness evaluation model can decrease by 5% of the trading failure rate. As the average trustworthiness of the suppliers decreases, which indicates that the trading environment becomes unreliable, the trustworthiness evaluation model can reduce the trading failure rate by approximately 40%. Apparently, the trustworthiness evaluation mechanism significantly improves trading reliability. Therefore, the demanders in TIDES are more likely to obtain what they want.

TABLE 1. Acronyms and abbreviations.

Abbreviation	Description
ABI	Application Binary Interface
CPSs	Cyber-physical Systems
DA	Data Asset
DB	Database
DRs	Data Repositories
DTAs	Data Trading Agencies
DTB	Data Trading Blockchain
IBS	Information Broadcasting Station
IoT	Internet of Things
KB	Knowledge Base
MEC	Multi-access Edge Computing
PoW	Proof-of-Work
QS	Quality Satisfaction
RAN	Radio Access Network
SOA	Service-oriented Architecture
SPE	Service Provisioning Engine
TDA	Transaction Dispute Arbiter

VII. CONCLUSION

This paper proposes and further implements TIDES to provide users an automatic, reliable and intelligent IoT data trading environment. Further with blockchain technology and smart contracts, IoT data trading can be automatically performed. The trustworthiness evaluation model can eliminate malicious suppliers and demanders to ensure data transactions. In addition, the client-centric data value evaluation model and game-theory-based pricing model help the demanders to obtain higher quality data commodities at an acceptable price and the suppliers to obtain the highest profits. With the hierarchical blockchain, the system performance is increased, and the storage cost is greatly reduced. The MEC technology also makes TIEDS satisfy the mobility requirement. The simulation results have demonstrated that TIDES ensures the efficiency, intelligence and reliability of IoT data trading.

APPENDIX

Abbreviations in this paper are summarized in TABLE 1.

REFERENCES

[1] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, "A survey on big data market: Pricing, trading and protection," *IEEE Access*, vol. 6, pp. 15132–15154, 2018.

[2] J. Rose, O. Rehse, and B. Rober. (2012). The value of our digital identity. Liberty Global website. [Online]. Available: http://www.libertyglobal.com/PDF/public-policy/The-Value-of-Our-Digital-Identity.pdf

[3] G. Katona, "Rational behavior and economic behavior," *Psychol. Rev.*, vol. 60, pp. 307–318, Sep. 1953.

[4] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, "Algorithmic trading," *Computer*, vol. 44, pp. 61–69, Nov. 2011.

[5] S. H. Hashemi, F. Faghri, P. Rausch, and R. H. Campbell, "World of empowered IoT users," in *Proc. IEEE 1st Int. Conf. Internet-Things Design Implement. (IoTDI)*, Apr. 2016, pp. 13–24.

[6] D. Roman and G. Stefano, "Towards a reference architecture for trusted data marketplaces: The credit scoring perspective," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 95–101.

[7] Filecoin. *Filecoin: A Cryptocurrency Operated File Storage Network*. Accessed: Jul. 2017. [Online]. Available: https://lecoin.io/lecoin.pdf

[8] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Oct. 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] I.-H. Chuang, T.-C. Weng, J.-S. Tsai, M.-F. Horng, and Y.-H. Kuo, "A reliable IoT data economic system based on edge computing," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1–5.

[10] Ethereum White Paper. *A Next-Generation Smart Contract and Decentralized Application Platform*. Accessed: Jun. 2019. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[11] G. Wood. *Ethereum: A Secure Dencentralised Generalised Transaction Ledger*. Accessed: Oct. 2019. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[12] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[13] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[14] A. Roncoroni, "Commodity price models," ' in *Encyclopedia of Quantitative Finance*. Hoboken, NJ, USA: Wiley, 2010.

[15] E. F. Fama and K. R. French, "Commodity futures prices: Some evidence on forecast power, premiums, and the theory of storage," in *The World Scientific Handbook of Futures Markets*. Singapore: World Scientific, 2016, pp. 79–102.

[16] R. Harmon, H. Demirkan, B. Hefley, and N. Auseklis, "Pricing strategies for information technology services: A value-based approach," in *Proc. Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2009, pp. 1–10.

[17] T. F. Bresnahan, "The oligopoly solution concept is identified," *Econ. Lett.*, vol. 10, nos. 1–2, pp. 87–92, Jan. 1982.

[18] H. Oh, S. Park, G. M. Lee, H. Heo, and J. K. Choi, "Personal data trading scheme for data brokers in IoT data marketplaces," *IEEE Access*, vol. 7, pp. 40120–40132, 2019.

[19] J. Nash, "Non-cooperative games," *Ann. Math.*, vol. 54, no. 2, pp. 286–295, 1951.

[20] Z. Li, Z. Yang, and S. Xie, "Computing resource trading for edge-cloud-assisted Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3661–3669, Jun. 2019.

[21] X. Kang, R. Zhang, and M. Motani, "Price-based resource allocation for spectrum-sharing femtocell networks: A Stackelberg game approach," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 3, pp. 538–549, Apr. 2012.

[22] K. Liu, X. Qiu, W. Chen, X. Chen, and Z. Zheng, "Optimal pricing mechanism for data market in blockchain-enhanced Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9748–9761, Dec. 2019.

[23] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.

[24] Y. Mao, T. Cheng, H. Zhao, and N. Shen, "A strategic bargaining game for a spectrum sharing scheme in cognitive radio-based heterogeneous wireless sensor networks," *Sensors*, vol. 17, no. 12, p. 2737, 2017.

[25] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social Internet of Things (SIoT)—When social networks meet the Internet of Things: Concept, architecture and network characterization," *Comput. Netw.*, vol. 56, no. 16, pp. 3594–3608, Nov. 2012.

[26] S. Yang, U. Adeel, and J. A. McCann, "Selfish mules: Social profit maximization in sparse sensornets using rationally-selfish human relays," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 6, pp. 1124–1134, Jun. 2013.

[27] L. Rittenberg and T. Tregarthen, "Principles of economics," FlatWorld, Boston, MA, USA, Tech. Rep., 2014.

[28] Ethereum. *Clique PoA Protocol & Rinkeby PoA Testnet #225*. Accessed: Mar. 2017. [Online]. Available: https://github.com/ethereum/EIPs/issues/225

I-HSUN CHUANG (Member, IEEE) received the M.S. and Ph.D. degrees in computer science and information engineering from National Cheng Kung University, in 2006 and 2014, respectively. He is currently working as a Postdoctoral Fellow with National Cheng Kung University. His research interests include Blockchain, network security, wireless networks, and cognitive radio.

SHIH-HAO HUANG (Student Member, IEEE) received the B.S. degree in computer science and information engineering from the National University of Tainan, in 2017, and the M.S. degree in computer science and information engineering from National Cheng Kung University, in 2019. His research interests include information security and mobile networks.

WEI-CHU CHAO (Student Member, IEEE) received the B.S. degree in computer science and information engineering from the National Yunlin University of Science and Technology, in 2017, and the M.S. degree in computer science and information engineering from National Cheng Kung University, in 2019. His research interests include Blockchain and wireless networks.

JEN-SHENG TSAI received the Ph.D. degree in computer science and information engineering from National Cheng Kung University (NCKU), Tainan, Taiwan, in 2012. He is currently working as a Postdoctoral Fellow with the Center for Research of E-Life Digital Technology, NCKU. His research interests include multimedia security, edge computing, and intelligent computing.

YAU-HWANG KUO received the Ph.D. degree in electrical engineering from National Cheng Kung University (NCKU), Taiwan, in 1988. He is currently a Distinguished Professor with the Department of Computer Science and Information Engineering and the Director of the Center for Research of E-Life Digital Technology, NCKU. In his career, he is persistently active in the academia, education, and government policy planning of digital innovation and digital equality. He has served as the Executive Secretary of the Board of Science and Technology, an Executive Yuan (Cabinet), the Board Chairman of the Institute for Information Industry, the Director of the Engineering and Technology Promotion Center and the Director of the Computer Science Program in National Science Council, the Director of the Computer Center in the Ministry of Education, the President of the Taiwanese Artificial Intelligence Association, and the Dean of the College of Science, National Chengchi University, Taiwan. His research interests include mobile communication, the Internet of Things, intelligent computing, and context-aware computing.