Blockchain-Enabled Data Collection and Sharing for Industrial IoT With Deep Reinforcement Learning

Chi Harold Liu, *Senior Member, IEEE*, Qiuxia Lin, and Shilin Wen

***Abstract*—With the rapid development of smart mobile terminals (MTs), various industrial Internet of things (IIoT) applications can fully leverage them to collect and share data for providing certain services. However, two key challenges still remain. One is how to achieve high-quality data collection with limited MT energy resource and sensing range. Another is how to ensure security when sharing and exchanging data among MTs, to prevent possible device failure, network communication failure, malicious users or attackers, etc. To this end, we propose a blockchain-enabled efficient data collection and secure sharing scheme combining Ethereum blockchain and deep reinforcement learning (DRL) to create a reliable and safe environment. In this scheme, DRL is used to achieve the maximum amount of collected data, and the blockchain technology is used to ensure security and reliability of data sharing. Extensive simulation results demonstrate that the proposed scheme can provide higher security level and stronger resistance to attack than a traditional database based data sharing scheme for different levels/types of attacks.**

***Index Terms*—Blockchain, crowdsourcing, deep reinforcement learning (DRL), energy efficiency, industrial Internet of thing, secure data sharing.**

I. Introduction

RAPID development of smart portable mobile terminals (MTs, e.g., smartphones, UAVs), which are equipped with rich set of sensors (e.g., camera, gyroscope, and GPS), has facilitated a new type of data collection method for industrial Internet of thing (IIoT), namely mobile crowdsensing (MCS). A typical MCS system consists of a cloud-based server and a collection of MTs [1]. The MCS server will launch a set of sensing tasks and select a set of MTs around the corresponding target area to execute these tasks.

MCS has been studied recently for various smart city related applications. For example, Wan *et al.* [2] proposed an MCS-based technology to achieve dynamic route choices for drivers wishing to avoid congestion. Pu *et al.* [3] proposed a comprehensive system model called "Crowslet," which is a novel self-organized mobile crowdsourcing paradigm and defines task, worker arrival and worker ability models. Furthermore, the use of MCS represents a benefit for IIoT. The introduced advantages include: mobile and scalable measures are provided; new areas can be monitored without the need for additional dedicated devices to be installed; subjective assessments can be easily and cost-effectively collected; human wisdom can be straightforwardly integrated into machine intelligence; information and decision-making processes can be shared among the whole industrial community [4]. MCS enables the collection of impressive amounts of data and the opportunity of analyzing them to perform more advanced processes and applications. Typical applications include asset utilization (e.g., asset tracking, environmental monitoring, fault detection, and predictive maintenance), quality control in manufacturing (e.g., real-time optimization, advanced analytics), supply chain management (e.g., real-time monitoring, logistic tracking, route planning, quality checking), product monitoring, and workplace safety (e.g., personal and environmental monitoring), smart grid [5]–[7], etc. As a particular example, Shu *et al.* [8] overviewed the existing IIoT solution for petrochemical plants and proposed that human-as-sensors, such as specialty clothes, smart-helmets, and smartphones equipped with gravity, GPS, and proximity sensors, can sense, upload, and share data to contribute to industrial sensing system by MCS that leads to a new way of collaborative data collection in large-scale petrochemical plants. And the industrial intelligent sensing ecosystem in the petrochemical plants can also provide security and surveillance service which can increase workplace safety and improve factory productivity.

Nevertheless, how to use smart MTs to collect high-quality data and perform secure data sharing among them has been rarely studied, especially when MTs may be intelligent agents that needs autonomous navigation without human interventions. On the one hand, achieving high-quality data collection is very important for MTs with limited MT energy resource and sensing range. On the other hand, these collected data are more

Manuscript received August 28, 2018; revised October 29, 2018 and December 3, 2018; accepted December 16, 2018. Date of publication December 28, 2018; date of current version June 12, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1003701, and in part by the National Natural Science Foundation of China under Grant 61772072. Paper no. TII-18-2220. *(Corresponding author: Chi Harold Liu.)*

C. H. Liu is with the Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. and also with the Department of Computer and Information Security, Sejong University, Seoul 143-747, South Korea (e-mail: liuchi02@gmail.com).

Q. Lin and S Wen are with the Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: linqiuxia 1995@126.com; 3120185530@bit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2018.2890203

meaningful only if they are shared whose values are mined after. Since the purpose of data sharing is that each MT can acquire the data from other MTs, which brings benefits to their collaboration and cooperation, and as a return to complete the assigned tasks in a better way. This primarily motivates our study in this paper. Specifically, our goal is to design an efficient scheme, which can achieve the maximum amount of collected data while ensuring secure data sharing among MTs in a fully distributed manner. Our contribution is three fold.

1) We propose a joint framework that integrate energy-efficient data collection and secure data sharing among MTs enabled by blockchain and DRL particularly for IIoT. Specifically, we designed a distributed DRL based approach for each MT to move to a location for data collection, while achieving the maximum data collection ratio and geographic fairness in the long run. Then, we use Ethereum to ensure data security and reliability when MTs share data, since Ethereum can effectively maintain a tamper-proof ledger shared by the participating MTs without the need of a trusted third central organization.
2) We analyze the security of our proposed framework under different malicious attacks including eclipse attack, majority attack, terminal device failure, etc. In view of the distributed characteristics and the adopted cryptology mechanism of blockchain in design, our framework can deal with most attacks compared with traditional database approach, to protect the data security.
3) We design and conduct extensive simulations and the results demonstrate that our proposed framework outperforms database storage and querying by higher reliability, security, and stronger resistance to attack in data collection and sharing.

The rest of this paper is organized as follows. Section II introduces the related work. Section III describes the system model. Section IV presents our proposed solution. Section V gives the security analysis of our framework when considering different malicious attacks. Section VI shows our simulation results. Finally, we conclude the paper in Section VII.

II. Related Work

In this section, we mainly describe the overview of related work from four areas: security for IIoT, MCS, blockchain, and DRL.

1) Security for IIoT: In [9], the authors introduced a number of research challenges and opportunities such as using cryptography and other techniques to ensure security and privacy in IIoT application and services. Karati *et al.* [10] proposed a lightweight certificateless signature scheme to ensure data authenticity in IIoT systems. Shrouf *et al.* [11] presented a reference architecture for IoT-based smart factories and defined main characteristics of such factories with a focus on the sustainability perspectives. They also proposed an approach for energy management in smart factories based on the IoT paradigm.

2) MCS: In [12]–[14], the authors comprehensively surveyed the related research efforts on participant selection and incentive allocation for MCS. For example, Wan *et al.* [2] proposed an MCS-based technology to achieve dynamic route choices for drivers wishing to avoid congestion. Pu *et al.* [3] proposed a comprehensive system model called "Crowslet," which is a novel self-organized mobile crowdsourcing paradigm and defines task, worker arrival and worker ability models. Moreover, Yang *et al.* [15] considered two system models including the crowd-sourcer-centric and the user-centric and designed corresponding mechanisms to allocate incentives to participants. Cheung *et al.* [16] proposed a two-stage approach considering the interactions between service provider and users and the simulation results showed the approach can obtain high gain in the user payoff over three benchmark heuristic schemes. Xiao *et al.* [17] proposed two algorithms called AOTA and LOTA in order to solve the makespan sensitive task assignment problems for MCS mobile social networks. In [18]–[23], the authors did a series of research works on MCS from quality-aware, energy efficient participant selection, and its applications to distributed event detection perspectives.

3) Blockchain: The blockchain concept has attracted wide attention [24]. It is essentially a distributed ledger database, which is a series of data blocks generated by cryptography. A complete blockchain system contains many technologies such as a peer-to-peer network, distributed ledger, consensus mechanism, and smart contracts. Due to the design characteristics of blockchain, the research community at present have studied and applied various blockchain technologies in many fields such as IoT, economics, medicine, and so on. For example, Karlsson *et al.* [25] presented a partition-tolerant blockchain called Vegvisir in power-constrained IoT environments with limited network connectivity, and Vegvisir can provide a shared and tamper-proof data storage service. Li *et al.* [26] proposed a called energy blockchain based on consortium blockchain to achieve secure energy trading in IIoT and designed a credit-based payment scheme. Chen *et al.* [27] did a pioneering work that proposed a privacy-protected and intercloud data fusing platform, which is needed to the demand for data mining and analytic activities in IoT. Kshetri [28] provided a detailed analysis and description of blockchain's roles in tracking the sources of insecurity in supply chains related to IoT devices. And this column also discussed and evaluated initiatives of organizations, interorganizational networks, and industries on the frontlines of blockchain. Miller [29] introduced that in the industrial market, the two key technologies of blockchain and IoT can not only improve efficiencies, transparency, and visibility, but also provide new business opportunities and address regulatory requirements. In addition, the combination of the two technologies can bring business value to the industrial sector.

4) DRL: DRL combines the traditional reinforcement learning (RL) and deep learning, which has made itself huge success in computer games, decision making systems, and Internet applications. The first well-known algorithm was DQN [30] for Atari games, as a variant of Q-learning, which uses a deep neural network (DNN) as the function approximator. Recently, DDPG [31] was proposed as a kind of the actor-critic method that was designed for scenarios with exponentially large, continuous action space. Other variations to improve their performance are still the center of research worldwide.

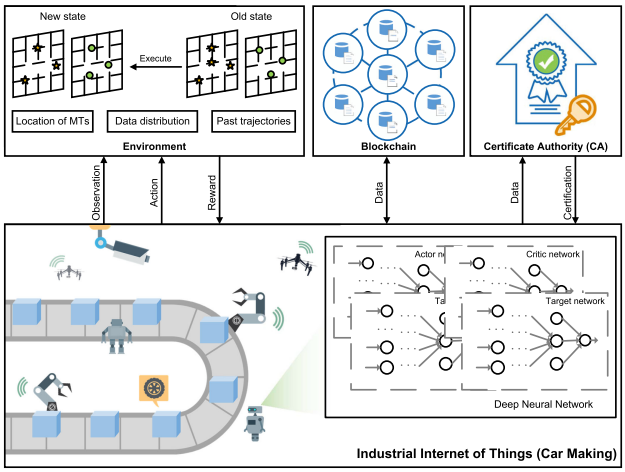


Fig. 1. Proposed overall system framework.

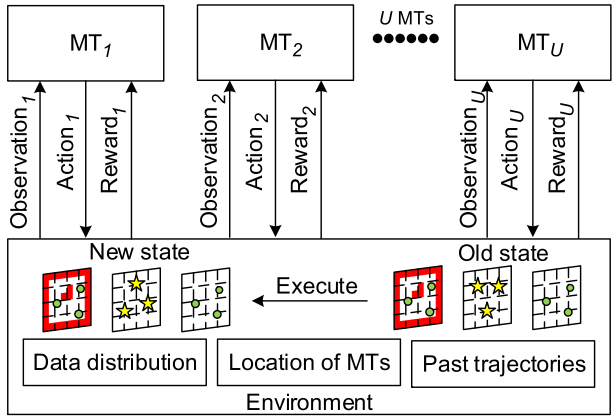


Fig. 2. Overall system flow of DRL-based data collection.

III. System Model

In this section, we introduce our system model that combines both blockchain and DRL for efficient data collection and secure sharing, as shown in Fig. 1.

A. IIoT for Energy-Efficient Data Collection

Without loss of generality, we consider a manufactory plant as a two-dimensional target area with several point-of-interests (PoIs) as sensing points and obstacles that MTs cannot go beyond. PoIs are defined as $\mathcal{K} \triangleq \{k = 1, 2, \ldots, K\}$, and each of them is equipped with certain amount of data to be collected by MTs. We also consider the existence of obstacles, such as buildings and engineering work. And we define obstacles as $\mathcal{C} \triangleq \{c = 1, 2, \ldots, C\}$. As shown in Fig. 1, M MTs as a set $\mathcal{M} \triangleq \{m = 1, 2, \ldots, M\}$ are deployed in the factory, that can be any smart device with mobility, communication, and computing capabilities, like human-carried smartphones, autonomous ground vehicles, and even low-level flying UAVs.

Since MTs are constrained by limited battery power, we consider them moving around in the area for fixed time slots to collect data and return back to the origin for charging as one round. In this way, we define the total number of collection rounds as R, where a set $\mathcal{R} \triangleq \{r = 1, 2, \ldots, R\}$, and each round contains T timeslots, where $t = 1, 2, \ldots, T$. If any PoI in $\mathcal{K}$ is within the sensing range of an MT, its data are considered to be sensed and collected. Note that in order to facilitate better data mining and analysis, the collected data should be evenly distributed from most of PoIs, but collecting them will consume energy. Thus, our goal is to maximize the total amount of collected data and geographical fairness, while reducing energy consumption.

B. Blockchain Network for Secure Data Sharing

We use Ethereum as the required data storage service and build a private blockchain, which includes $\mathcal{N} \triangleq \{n = 1, 2, \ldots, N\}$ Ethereum nodes for storing and sharing data. We then classify them into the following two categories.

1) Mining Nodes: They are used to verify data sharing transactions and compile them into blocks. They need to consistently use machine computing resources to solve computing problems and submit blocks to the blockchain network.

2) Nonmining Nodes: Since the nonmining node is only responsible for receiving and broadcasting data sharing transaction requests, it does not need the same amount of resources if compared to a mining one.

Note that each Ethereum node keeps a complete, validated copy of the blockchain and service MTs according to smart contracts. We assume that each MT has valid connection with the blockchain network and after each round r, MTs upload their collected data from PoIs to the blockchain network. Therefore, at round r, the total amount of data storage requests S made by MTs is denoted by $S = M * r$.

C. Certificate Authority (CA)

Each MT registers to the CA and obtains its public and private keys $(pk_m, sk_m), \forall m \in \mathcal{M}$, to become a legal terminal identity. To ensure that collected data sent to the blockchain network is valid and cannot be forged, MT needs first to encrypt data with its private key and then send it to the CA, which will check whether the data come from a legitimate MT, and if it is real. After verification, the signature of CA and encrypted data will return back to MT and can be sent to the blockchain as storage requests by MT.

Besides, we also assume that there are malicious MTs in the system, which are able to launch a certain level of attack in each round, we set the attack level as $\alpha = 1$ or $\alpha = 2$. Without loss of generality, we let level-2 attack be more dangerous to the system than level-1 attack.

IV. Proposed Solution

A. Multiagent DRL Based Distributed Data Collection by MTs

Since our scenario is a fully distributed and continuous multiagent data collection environment, traditional policy gradient based methods of DRL cannot meet our requirement. For example, the deep Q-network (DQN [30]) can only work well in a limited action space which is discrete, discontinuous, and nondistributed, thus it is not suitable for our application scenario. We, therefore, propose a new solution here. As shown in Fig. 2, each

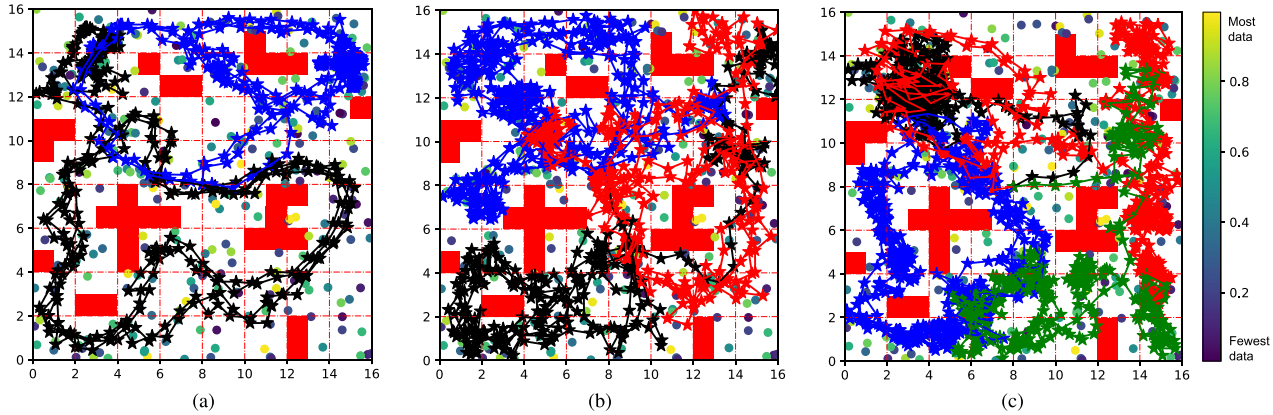


Fig. 3. MT trajectories in a plant/factory (stars for MTs, red blocks for obstacles, and dots for PoIs). (a) Two MTs. (b) Three MTs. (c) Four MTs.

MT m generates an observation $\boldsymbol{o}_t^m = (x_t^m, y_t^m, e_t^m)$, which is a part of state $\boldsymbol{s}_t$ at each timeslot t, and gives it and action a_t^m to the environment, then obtains a reward r_t^m from the environment. Environment consists a set of states, which are data distribution $\mathcal{S}_1$, location of MTs $\mathcal{S}_2$, and past trajectories $\mathcal{S}_3$ and that will give detailed explanation later. After the execution of actions, the environment would change from old state $\boldsymbol{s}_t$ to new state $\boldsymbol{s}_{t+1}$. To the best of our knowledge, state, action, and reward are three basic components for DRL. Given a state and a set of possible actions to choose from, the goal is to find a policy that maximizes the accumulated reward. In our system, state, action, and reward are defined as follows.

1) State Space: State, which is a description of the environment, is denoted as $\mathcal{S} = \{(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)\}$ as three channels. We assume that the simulated environment is a map of $E_x \times E_y$ size, and covered $\mathcal{K}$ PoIs and several red obstacles. And $\mathcal{S}_1$ represents the locations of PoIs and obstacles, therefore, can be defined as $\mathcal{S}_1 = \{(x^k, y^k), (x^c, y^c)\}_{k\in\mathcal{K}, c\in\mathcal{C}}$, where $x^k, x^c \in [0, E_x]$ and $y^k, y^c \in [0, E_y]$; $S_2 = \{(x_t^m, y_t^m, e_t^m)\}_{m\in\mathcal{M}}$, where x_t^m, y_t^m are the coordinates of MT m, and $e_t^m \in [0, 1]$ refers to the remaining energy percentage of m at timeslot t; $\mathcal{S}_3$ is sensing times $h_t(k) \in [0, T]$ for a PoI k until timeslot t, therefore, $h_{t+1}(k) = h_t(k) + 1$ if a PoI k is sensed and its data are collected at timeslot $t+1$.

2) Action Space: Moving direction θ_t^m and distance l_t^m constitute the action set $\mathcal{A} = \{(\theta_t^m, l_t^m)_{m\in\mathcal{M}} | \theta_t^m \in [0, 2\pi), l_t^m \in [0, l_{\max}]\}$, where $l_{\max}$ is the maximum distance that an MT can move in a timeslot.

3) Reward: All three parts, namely data collection amount b_t^m, achieved geographical fairness ω_t and energy consumption $\phi(b_t^m, l_t^m)$ contribute to the reward. We assume the energy consumption model $\phi(b_t^m, l_t^m)$ with two unit weights α and κ, to take into account the power consumption by both collecting data and moving, which is given as $\phi(b_t^m, l_t^m) = \alpha b_t^m + \kappa l_t^m$, where α and κ denote the energy consumption per data collected, and per distance traveled, respectively. The degree of geographical fairness among PoIs (in terms of remaining data amount) is denoted by ω_t as

$$\omega_t = \frac{\left(\sum_{k=1}^{K} h_t(k)\right)^2}{K \sum_{k=1}^{K} h_t(k)^2}. \tag{1}$$

In this way, we can compute a reward formulation r_t^m as

$$r_t^m = \frac{\omega_t b_t^m}{\phi\left(b_t^m, l_t^m\right)} \qquad \forall m \in \mathcal{M} \tag{2}$$

as the energy-efficiency for achieving data collection and fairness.

Each MT is implemented by four DNNs that serves as actor network $\pi^m(\boldsymbol{o}_t|\boldsymbol{\theta}^{\pi^m})$, critic network $Q^m(\boldsymbol{s}_t, \boldsymbol{a}_t|\boldsymbol{\theta}^{Q^m})$ with randomly initialized weights $\boldsymbol{\theta}^{\pi^m}$, $\boldsymbol{\theta}^{Q^m}$ and their two target networks with parameters $\boldsymbol{\theta}^{Q'^m} := \boldsymbol{\theta}^{Q^m}$, $\boldsymbol{\theta}^{\pi'^m} := \boldsymbol{\theta}^{\pi^m}$. In each collection round, we initialize environment and obtain the initial state $\boldsymbol{s}_1 \in \mathcal{S}$. Then, the data collection task that lasts for T timeslots is started.

For distributed training process, each MT owns a private buffer B_m, which contains abundant state transition samples $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$. We sample H groups of transitions as minibatches where each group contains h samples from each MT's buffer. For each MT m, actor target network will give a target action $\boldsymbol{a}_{t+1}^m$ with given observations $\boldsymbol{o}_{t+1}^m$ from a minibatch. Then, critic network Q^m is updated through minimizing a loss function $L(\theta^{Q^m})$, as

$$L(\theta^{Q^m}) = \mathbb{E}[(y_t^m - Q^m(\boldsymbol{s}_t, \boldsymbol{a}_{t+1}^1, \ldots, \boldsymbol{a}_{t+1}^M | \theta^{Q^m}))^2] \tag{3}$$

$$y_t^m = \gamma Q'^m(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}^1, \ldots, \boldsymbol{a}_{t+1}^M | \boldsymbol{\theta}^{Q'^m}) + \gamma r_{t+1}^m \tag{4}$$

while we updated actor network, using the gradient as

$$\nabla_{\theta^{\pi^m}} J \approx \mathbb{E}[\nabla_{\theta^{\pi^m}} \pi^m(\boldsymbol{o}|\theta^{\pi^m})|_{\boldsymbol{o}=\boldsymbol{o}_t} \nabla_a Q^m(\boldsymbol{s}, \boldsymbol{a}_{t+1}^1, \ldots, \boldsymbol{a}_{t+1}^M | \theta^{Q^m})|_{\boldsymbol{a}_{t+1}^m = \pi^m(\boldsymbol{o}_{t+1}^m)}]. \tag{5}$$

Note that target networks are two copies of the actor π^m and critic Q^m networks, but with different weights update rules. Specifically, after updating weights of networks π^m and Q^m, the weights of target networks, $\boldsymbol{\theta}^{Q'^m}$ and $\boldsymbol{\theta}^{\pi'^m}$, are then slowly updated with the original networks weights and update factor τ to improve the stability of learning, as

$$\boldsymbol{\theta}^{Q'^m} := \tau \boldsymbol{\theta}^{Q^m} + (1-\tau)\boldsymbol{\theta}^{Q'^m} \tag{6}$$

$$\boldsymbol{\theta}^{\pi'^m} := \tau \boldsymbol{\theta}^{\pi^m} + (1-\tau)\boldsymbol{\theta}^{\pi'^m}. \tag{7}$$

After adequate training, all the parameters in four DNNs are optimized for data collection task.

B. Blockchain-Enabled Secure Data Sharing Among MTs

After each data collection round, we assume that the outputs of DRL based data collection approach can be storage requests and query requests, which would be the inputs of the blockchain based storage system. Specifically, each MT have a valid connection with blockchain, therefore, MT can encrypt collected data with its private key and send it with signature to the blockchain as storage request. Furthermore, for data sharing, MTs can send query requests to the blockchain and obtain the returned data.

We aim to design a decentralized, secure, and reliable storage scheme that can achieve consensus in a trustless environment and ultimately guarantee that data records cannot be tampered with. Compared with Hyperledger, Ethereum has a more powerful ecosystem and makes the development of smart contracts and business logic very simple. Besides, Hyperledger is designed for storing confidential data and known as private consortium-blockchain, while Ethereum is a public blockchain for any kind of applications, which, therefore, is more suitable with our scenario. Most importantly, our simulation environment is compiled by Python language and Ethereum already provided Python interface for interacting with the blockchain, Hyperledger official, however, does not offer mature Python interface. To sum up, the Ethereum blockchain can meet our requirements, and we, therefore, use it to implement the blockchain network. We explain the whole process of secure data sharing among MTs by using pseudocode in Algorithm 1.

During the initialization process, we prepared the configuration file for genesis block, in which we define our independent blockchain network ID, set the system mining difficulty as 0x400, gasLimit as 0xFFFFFF, and other parameters. After, we create multiple Ethereum nodes on our private chain. Each launched Ethereum node will open a Javascript Console, where the built-in objects can perform certain operations, like querying blocks and transactions, mining, etc. Then, the Solidity language is used to write a smart contract, which includes initialization of the MT accounts and the storage and query function of its collected data. We deploy the smart contract on the blockchain, where the contract content is visible to all Ethereum nodes and each node can interact with the smart contract. We require the storage system to be able to facilitate data sharing. That is, any MT with access to blockchain can view the entire data records without restriction. Due to the open and transparent nature of Ethereum, it enables developers to build creative blockchain applications. However, we hope that data stored on blockchain cannot be stored arbitrarily, because it will cause malicious MTs to store fake and forged data. As far as we know, Ethereum cannot monitor data quality. Therefore, we set up a CA to review the authenticity and data ownership and manage the public keys of all MTs.

As shown in Algorithm 1, to be more specific, we first run Gen(1^n) to generate public and private keys pair (pk^m, sk^m) for each MT, where 1^n is a security parameter (Line 5). And CA would store MT ms id and public key (m, pk^m) in a list (Line 6). As we said in the data collection (see Section III-A), in our experiment, the terminal will collect data for R rounds, and there are T timeslots in each round. MT moves and collects data b_t^m in the timeslot of each round, and we think b_t^m accumulates. After the end of a round r, that is, after every T timeslots, the MT m will send its data, as

$$u_r^m = \sum_{t=1}^{T} b_t^m \tag{8}$$

and signature $\text{Sign}_{sk^m}(H(u_r^m))$ to CA (Lines 15–17). CA can decrypt it and judge whether it is the MT ms data according to the result $\text{Vrfy}_{pk^m}(u_r^m, \text{Sign}_{sk^m}(H(u_r^m)))$. If the result is 1, the request is sent by MT m, otherwise not. In addition, after decryption, it is also necessary to compare u_r^m with the data on the terminal itself to avoid fake data. Therefore, if $\text{Vrfy}_{pk^m}(u_r^m, \text{Sign}_{sk^m}(H(u_r^m)) = 1$ and $u_r^m = \text{collection}[m]$, then CA will add its own signature to the request and return it to the MT m (Lines 18–23). MT can package the request $(u_r^m, \text{Sign}_{sk^{CA}}(u_r^{m\prime}), pk^m)$, then send it to blockchain.

Algorithm 1: Blockchain-Enabled Secure Data Sharing Among MTs.

1: Initialize private blockchain, and setup N Ethereum nodes;
2: Deploy smart contract on the blockchain;
3: Blockchain starts mining;
4: **for** MT m in $\mathcal{M}$ **do**
5: Run $Gen(1^n)$ to obtain (pk^m, sk^m);
6: CA stores identity (m, pk^m);
7: **end for**
8: **for** Round $r = 1, 2, \ldots, R$ **do**
9: Initialize environment, receive initial state $\boldsymbol{s}_0$;
10: **for** timeslot $t = 1, 2, \ldots, T$ **do**
11: **for** MT m in $\mathcal{M}$ **do**
12: Collect data (see Section III-A);
13: **end for**
14: **end for**
15: $u_r^m := \sum_{t=1}^{T} b_t^m$;
16: Hash collected data $H(u_r^m)$;
17: Send $(u_r^m, Sign_{sk^m}(H(u_r^m)))$ to CA;
18: **if** $Vrfy_{pk^m}(u_r^m || Sign_{sk^m}(H(u_r^m)) == 1$ **then**
19: **if** $u_r^m == collection[m]$ **then**
20: $u_r^{m\prime} := Sign_{sk^m}(H(u_r^m))$;
21: Return $(u_r^m, Sign_{sk^{CA}}(u_r^{m\prime}))$ to MT m;
22: **end if**
23: **end if**
24: **if** MT m receives signature from CA **then**
25: Send transaction request$(u_r^m, Sign_{sk^{CA}}(u_r^{m\prime}), pk^m)$ to blockchain;
26: **if** Verify signature and identity is true **then**
27: Upload transaction to blockchain and wait for confirmation;
28: **end if**
29: **end if**
30: **end for**

After nonmining nodes receive transaction requests from MT, blockchain will verify the signature on the request and decide whether to submit the transaction to the blockchain network based on the verification results (Lines 26–28). The submitted transaction will be mined and written in a new block by mining nodes.

V. Security Analysis

Although the security in blockchain has been continuously enhanced, attackers still have various ways in order to hack blockchain [32]. In this section, we summarize some potential attacks in the proposed system and provide solutions.

A. Transaction Forgery by MTs

Given that the data stored on the blockchain can be largely complete, consistent, and correct, an attacker will pay a heavy price if he/she wants to tamper with its data. Thus, it is more likely to attack during the process of sending and receiving data requests. He/she can intercept transaction proposals sent by legitimate user to the blockchain and has a high probability of success. After reading the encrypted transaction messages multiple times, the attacker will try to impersonate the legitimate user to send unreal transaction proposals to the blockchain network. In our scenario, each MT collects data from PoIs and sends storage requests as transaction messages to the blockchain to store data, the attacker may intercept the request and change the amount of collected data to be fictional or wrong. It is even possible that the attacker MT itself is an authenticated device in the blockchain network, but it attempts to inflate the amount of collected data, so it provides false data when sending a storage request.

We believe that it is difficult or extremely unlikely for an attacker MT to calculate the private key of CA, so it cannot forge the confirmation message of CA. After the attacker sends the forged storage request to the CA, the latter will confirm whether the message comes from a real device or not. If so, then check whether the MT actually owns the amount of data mentioned in the request. If all true, CA will add the center signature to the legitimate transaction and return it to MT. MT then sends the storage transaction to the blockchain network, which will also verify whether it is the real signature of the CA when receiving the transaction proposal.

B. Eclipse Attack by Network Users

The Eclipse Attack is a network-level attack on the blockchain, where the attacker basically controls the peer-to-peer network [33]. Since each Ethereum node on the network relies on connections to other nodes to obtain a complete view of the network, in an Eclipse Attack, the attacker only needs to control the node of target victim. In this way, an attacker can prevent victim from receiving complete information about the rest of the network. In our scenario, an attacker can use an Eclipse Attack to prevent Ethereum nonmining node from receiving storage and query requests to the blockchain, while making it impossible to invoke other events in the smart contract, affecting the normal interaction between devices and the blockchain network. This type of attack requires only two malicious nodes to isolate and influence other nodes. For this vulnerability, Ethereum has released an updated version of software, making the number of malicious nodes needed to carry out such an attack from two to thousands [34].

C. Vulnerability Attack by Network Users

Ethereum is an open source public blockchain platform with smart contract functionality. All users on the blockchain can see the smart contract deployed on the blockchain. When a smart contract has a critical vulnerability, it is very easy for attackers to exploit. The more powerful the contract, the more complex it is, and the more likely it is to have a logical loophole. Here, we can assume that an attacker who may utilize vulnerabilities in the smart contract to tamper with the data stored on the blockchain, or even if a more serious vulnerability is available, which may cause the entire collected data on the blockchain to be stolen and emptied.

When designing the smart contract, we have avoided recursive calling vulnerability [35], timestamp dependence [36], arithmetic problem [37], return value problem [38], and completed code audits and security tests. We have no reason to believe that other vulnerabilities of equal destructive power will not appear.

D. Majority Attack by Network Users

When more than half of the users in the blockchain network become malicious users and reach a consensus with the attacker, or the attacker controls more than half of the computing power in the network, we can confirm that the blockchain network is not secure [39]. The attacker can take advantage of the computing power to tamper with the records on the blockchain. The newly generated chain can belong to him/her completely, and it may not even contain any block that was mined by other miners. Because the longest chain is always considered to be the best credibility, the attacker can reverse the issued transaction, thus achieving double spending problem [40]. The attacker can also block confirmation of other transactions. In addition, the attacker can prevent other blocks from being dug out.

Considering that our blockchain network is set up as a private chain, and the mining nodes are owned and controlled by the system. To some degree, they are honest and they follow the rules. We can guarantee that more than half of the computing power will not be obtained by the attacker, so that such attack is avoided as much as possible.

E. MT Device Failure

In our scenario where multiple MTs can be quite heterogeneous while collecting and sharing data distributedly, the most critical problem is device consistency. That is, if we specify a series of operations for a given set of devices that is guaranteed by a certain protocol, every device will agree on the final processing result. However, with the long-term use of these devices, some may have software or hardware problems which may prevent these devices from continuing to function normally, such

as device downtime, network service dropping, network communication failure, etc. These problems are generally divided into two types. One is that only data are delayed or lost. The other is even serious, because in the network there may exist malicious users that send false data to other users and this is specifically described as "The Byzantine Generals Problem" [41]. In other words, if an attacker obtains some information about those devices which have failed in our case, it can pretend to be a malfunctioning MT as an authorized or legal MT, and then falsely report the actual amount of data, which may cause data inconsistency or data insecurity as a result. In order to address these possible problems, certain consensus algorithms in a blockchain such as Paxos [42] and PBFT [43] have been adopted in our proposal to mitigate or overcome them. In this way, we can effectively solve the MT device failure problem.

To sum up, as for secure data storage, blockchain can largely prevent our system from data inconsistency, and certain attacks can be avoided by enough security tests. Besides, to ensure data transmission security, our system have cryptology mechanism and CA, as a third party, which can distribute secret keys to make sure the data ownership and validate data to review the authenticity.

VI. Performance Evaluation

A. Simulation Setting

We implemented the solution for DRL-based data collection by MTs in Section IV-A and sending data storage requests back-and-forth by Python, and this environment was configured on the machine where the blockchain and the compared database method are located. We deployed a private blockchain network for blockchain enabled data sharing solution in Section IV-B, based on Geth 1.7.2 (Go Ethereum), which is built on the Ubuntu 16.04 LTS with Intel Core 3.40 GHz i7-6700 CPU and 16 GB of RAM, where the Ethereum nodes were all running on the same machine. Our private blockchain includes three Ethereum nodes, one of which receives storage and query requests from MTs and two additional mining nodes whose mining thread is set as 2. We simulate two kinds of attacks, denial-of-service (DoS) attack and distributed denial-of-service (DDoS) attack. Specifically, they can carry out different levels of resource consumption attacks by sending requests to the port 3306 of database server or the two nodes in the blockchain (one is mining node and the other is the one receiving transaction).

1) DoS Attack: We simulate malicious network users that can frequently send the requests for querying blockchain the latest block information, or querying the total number of records in the table to MySQL, resulting in slow speed for legitimate storage transactions. We implement two different levels of DoS attack, with 20 concurrent processes as level-1 and 50 concurrent processes as level-2.

2) DDoS Attack: We simulate other malicious network users outside the system that can launch DDoS attacks. The attacks were executed in a machine provided with Windows 10 system, Intel Core i7-8750 @2.2 GHz, 8 GB RAM, and a machine with Windows 10 system, Intel Core i7-6700 CPU @3.40 GHz, 16 GB RAM. Consistent with DoS attacks, we implemented two levels of attack. The number of processes used for lower attack (level-1) is 2, and the number of processes used for higher attack (level-2) is 5.

B. Baseline and Comparing Metrics

For DRL-based distributed data collection module, we compare our solution with Random algorithm, that is: each MT m randomly selects a moving direction $\theta_t^m \in [0, 2\pi)$ and a distance $l_t^m \in [0, l_{\max}]$ without any strategy at any timeslot t.

And we use the following three metrics to measure their performances.

1) Data collection ratio σ_D: calculated as a ratio between the final amount of collected data over initial generated data.
2) Energy consumption ratio σ_E: calculated as a ratio between the final consumed amount of energy for all MTs and their initial amount of energy.
3) Geographical fairness ω_t: calculated according to 1). If the data collection result is fairer among all PoIs, the value of ω_t will be more close to 1.

As for Blockchain-enabled secure data sharing module, we implemented MySQL database version 5.7.22 on a Windows 10 server with Intel(R) Core(TM) i7-4790 @ 3.60 GHz and 12 GB RAM, as baseline to compare with our proposal. After each round, all MTs will send the collected data to MySQL, and the latter stores them if a legitimate request. Then, we created a table with three columns (round_id, terminal_id, data_amount), and made round_id and terminal_id columns primary keys. And we can use MySQL COUNT(*) function to query the total number of datasets. We use the following two metrics to measure the performance of our solution.

1) Immediate query failure ratio λ: Every ten rounds, we query the database or blockchain for total amount of currently available data ΔQ, and then calculate the ratio of unavailable data whose data transaction requests may be blocked/delayed or canceled, as: $\lambda = 1 - \Delta Q/S$, where $S = M * r$, and $r = 1, 2, \ldots, R$.
2) Query successful ratio ρ: calculated as a ratio between the final amount of inserted data and total storage transaction requests, when blockchain or database does not process any request and MTs stop sending requests, which can be written as: $\rho = \Delta I/S$, where $S = M * R$.

Overall, the above-mentioned two ratios calculate the requested processing results under different system operational states. Specifically, immediate query failure ratio λ is calculated every ten collection rounds, when the system is still active since it still receives continuous data requests and processes them. On the contrary, query successful ratio ρ is calculated when the system operation finishes, i.e., MTs no longer move around and the storage system does not process any request further. To this end, we can conclude that if a transaction request fails when calculating ρ, it should also be blocked or canceled which was counted into λ, otherwise not true the other way around. Also, λ reflects the transient response behavior, while ρ reflects the robustness of the system.

TABLE I
IMPACT OF NO. OF MTS ON DATA COLLECTION RATIO, ENERGY USAGE RATIO, AND GEOGRAPHIC FAIRNESS

No. of MTs		2	3	4
Data collection ratio σ_D	Random	0.403	0.434	0.489
	Ours	0.836	0.895	0.844
Energy usage ratio σ_E	Random	0.163	0.171	0.186
	Ours	0.299	0.419	0.433
Geographic fairness ω_t	Random	0.287	0.305	0.344
	Ours	0.632	0.661	0.628

C. Illustrative MT Moving Trajectories

We next show moving trajectories for two, three, and four MTs in Fig. 5. As shown in Fig. 5(a), two MTs learned to mainly move around in half of the area, responsible for their data collection, which can potentially maximize their reward. Since each MT has limited sensing and collection capability, one single sense is not enough to collect all the data of that PoI, both MTs successfully learned to circulate around a small area until data is all collected from a particular PoI. Meanwhile, in order to obtain a relatively fairer data collection (as defined in the reward function), these two MTs do not have a fixed route to circulate the area, but they have rather made good adaptations slightly deviated from the circulating route, to cover other PoIs. For example, the black MT even goes to the upper left corner for collecting some corner case data.

With the increase of number of MTs, we observe a much finer trajectories of each MT (i.e., smaller moving step size and collection area). For example, in Fig. 5(b), three MTs first sensed central part of all PoIs by circulating around and then go to other places. Furthermore, from Fig. 5(c), we see that each MT takes responsibility to sense a local region because enough MTs are deployed and they have learned to collaborate but not to go farther places of other's area. Comparing Fig. 5(a) and (c), we find that the moving length for two MTs is obviously larger on average. This is because that in order to keep a fairer data collection, two MTs must sense PoIs, which are rarely sensed and thus hopping back and forth cannot be avoided. Finally, we see that all MTs successfully avoid obstacles and never go beyond the border.

D. Results and Discussions

We first conducted some experiments to compare the performance of Random solution and our solution in the data collection ratio σ_D, energy consumption ratio σ_E and geographically fairness ω_t, as illustrated in Table I.

1) We can see that our solution significantly outperforms Random solution. For instance, when the MT number is 2, the data collection ratio of Random approach is 0.403, while that of ours is 0.836. Additionally, our method can increase geographically fairness ratio by 34.5% compared to Random solution. Similar increases can be found when MT number is 3 or 4. Although the energy consumption ratio is higher than Random, the other two metrics show large increases, for the reason that our solution consumes more energy due to the data collection costs.
2) We further find that with the increase of MT numbers both our method and Random method substantially show increasing trend on three metrics. It is because that more MTs can collect more data, therefore, more energy would be consumed, and higher fairness can be achieved. Although in our solution, there is a slightly decrease in data collection ratio and geographically fairness when the MT number increases to 4 compared to three MTs, our method still outperforms Random method with overwhelming advantages.

We next present the impact when changing the number of MTs and attack levels on the immediate query failure ratio under DoS attacks, as shown in Fig. 5. We observe the following.

1) Regardless of database or blockchain, with the increase of the number of storage transactions sent by MTs, the proportion of data that cannot be queried in the total data decreases. That is, DoS attacks have a similar effect on both blockchain and database, blocking a certain number of data storage requests.
2) When changing the number of MTs or adjusting the attack level, the maximum immediate query failure ratio λ for our approach is no more than 0.3, while 0.7 for database. Blockchain approach achieves generally lower λ than that of database approach. This is because MySQL uses a client-server architecture, but no other nodes to backup data, which results in all data requests being processed by one machine. While blockchain is a kind of distributed database, even if a DoS attack is injected on one node, other nodes can still work normally, and the availability of the whole system will not be greatly affected.
3) In Fig. 5(b), when level-1 attack is assumed, $\lambda = 0.1$ for our approach, and 0.5 for database approach. When we increase the attack to level-2, λ rises to 0.2 and 0.63, respectively. This is because that higher attack severity, the number of blocked requests will inevitably increase. Therefore, the failure ratio increases given the same number of data requests.
4) When the number of MTs increases from 2 to 4, both approaches have a significant change between the above-mentioned three figures. For instance, comparing Fig. 5(a) and (c), with level-1 attack, λ for blockchain decreases from 0.2 to 0.05, and for database decreases from 0.6 to 0.45, when transaction number is 20. The similar trend can be observed under attack level-2. This is because that more MTs means more transaction requests, and thus, the lower query failure ratio is expected.

We then compare the performance of database and blockchain in terms of the query successful ratio, as shown in Fig. 4. We observe the following.

1) With the increase of transaction requests, query successful ratio ρ by database approach increases, while blockchain approach always keeps 100% success. That is, DoS attack does have a certain influence on the MySQL

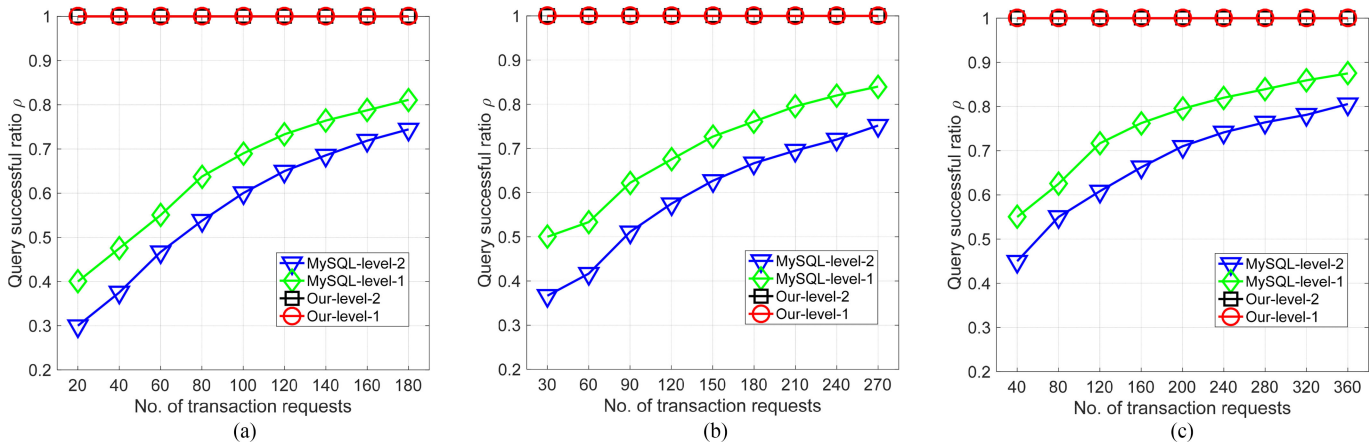


Fig. 4. Impact of DoS attack and number of MTs on query successful ratio. (a) Two MTs. (b) Three MTs. (c) Four MTs.

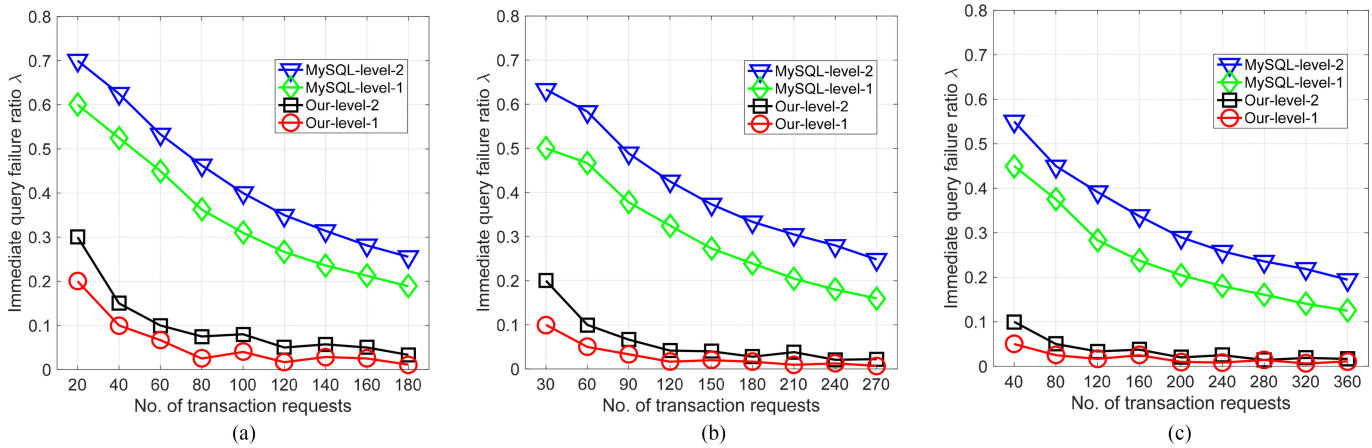


Fig. 5. Impact of DoS attack severity and number of MTs on immediate query failure ratio. (a) Two MTs. (b) Three MTs. (c) Four MTs.

TABLE II
RESULTS OF DDOS ATTACK

No. of MTs		2					3					4				
No. of trans. requests		20	60	100	140	180	30	90	150	210	270	40	120	200	280	360
λ	MySQL-level-2	0.600	0.483	0.370	0.286	0.222	0.500	0.422	0.333	0.252	0.196	0.400	0.325	0.255	0.193	0.150
	MySQL-level-1	0.500	0.333	0.22	0.157	0.122	0.400	0.311	0.207	0.148	0.115	0.300	0.225	0.175	0.129	0.100
	Ours-level-2	0.200	0.100	0.030	0.014	0.022	0.167	0.067	0.033	0.024	0.011	0.100	0.017	0.010	0.011	0.003
	Ours-level-1	0.100	0.067	0.020	0	0	0.067	0.022	0.02	0.014	0	0.05	0	0.005	0.007	0
ρ	MySQL-level-2	0.400	0.517	0.630	0.714	0.778	0.500	0.578	0.667	0.748	0.804	0.600	0.675	0.745	0.807	0.850
	MySQL-level-1	0.500	0.667	0.780	0.843	0.878	0.600	0.689	0.793	0.852	0.885	0.700	0.775	0.825	0.871	0.900
	Ours-level-2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Ours-level-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

database, but as the number of transaction requests increases, the proportion of stored data affected by this attack decreases, resulting in a slow increase of ρ. On the other hand, the blockchain is less affected by its multinode structure.

2) The number of MTs has no impact on the blockchain, and more MTs, query successful ratio by database approach slightly increase. This is because that under the same level of attack, more MTs means more storage requests sent to the database. However, the blocked ones are also fixed, thus ρ is slightly increasing.

Finally, we deploy DDoS attacks on both database and blockchain, while changing the attack severity and number of MTs. We show both λ and ρ, in Table II, and the observations are very similar to DoS attack. It is clear that both attacks can consume system resources to cut off service from the Internet, therefore, MTs cannot connect storage system and their requests would be blocked or lost. Hence, DoS and DDoS attacks roughly follow the same result trend. Also, in our experiment, we can see that the result of DDoS attack is 0.1% worse than the DoS attack. One possible reason is that the severity of DoS attack designed in our experiment has almost reached the upper bound of

the load capability of the storage system. Therefore, the DDoS attack cannot decrease further.

VII. Conclusion

In this paper, we proposed an joint framework for both efficient data collection and secure data sharing scheme combining Ethereum blockchain and DRL for MCS enabled IIoT scenarios. We proposed a novel, fully distributed DRL scheme that help each MT to sense nearby PoIs to achieve maximum data collection amount, geographic fairness, and minimum energy consumption. Then, blockchain is used to share data among MTs to pertain their security levels. We conducted simulation experiments to evaluate the effectiveness of the proposed scheme and the experimental results showed that compared with a traditional data storage and sharing scheme based on MySQL database, our scheme can provide higher security, reliability, and stronger resistance to some malicious attacks (DoS, DDoS, etc.) for data sharing.

In the future, we plan to do further research in the following two directions, including 1) how to extend the existing DRL framework to closely into every blockchain node to make each node more intelligent, which can guarantee the security of data sharing more effectively, and 2) how to design novel deep models for each MT to execute multiple tasks simultaneously, e.g., not only collecting data but also providing other services such as communication service as a mobile communication base station.

References

[1] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surveys*, vol. 48, no. 1, 2015, Art. no. 7.

[2] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, p. 88, 2016.

[3] L. Pu, X. Chen, J. Xu, and X. Fu, "Crowdlet: Optimal worker recruitment for self-organized mobile crowdsourcing," *IEEE INFOCOM*, San Francisco, CA, pp. 1–9, 2016.

[4] V. Pilloni, "How data will transform industrial processes: Crowdsensing, crowdsourcing and big data as pillars of industry 4.0," *Future Internet*, vol. 10, no. 3, pp. 1–14, 2018.

[5] Y. Zhang, R. Yu, M. Nekovee, Y. Liu, S. Xie, and S. Gjessing, "Cognitive machine-to-machine communications: Visions and potentials for the smart grid," *IEEE Netw. Mag.*, vol. 26, no. 3, pp. 6–13, Apr. 2012.

[6] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar, "Dependable demand response management in the smart grid: A stackelberg game approach," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 120–132, Mar. 2013.

[7] Y. Zhang, R. Yu, W. Yao, S. Xie, Y. Xiao, and M. Guizani, "Home m2m networks: Architectures, standards, and QOS improvement," *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 44–52, Apr. 2011.

[8] L. Shu, M. Mukherjee, M. Pecht, N. Crespi, and S. N. Han, "Challenges and research issues of data management in IoT for large-scale petrochemical plants," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2509–2523, Sep. 2018.

[9] K. K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial internet-of-things: Research challenges and opportunities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3567–3569, Aug. 2018.

[10] A. Karati, S. H. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3701–3711, Aug. 2018.

[11] F. Shrouf, J. Ordieres, and G. Miragliotta, "Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, 2015, pp. 697–701.

[12] X. Zhang *et al.*, "Incentives for mobile crowd sensing: A survey," *IEEE Commun. Surveys Tut.*, vol. 18, no. 1, pp. 54–67, First Quarter 2016.

[13] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij, "A survey of incentive techniques for mobile crowd sensing," *IEEE Internet Things J.*, vol. 2, no. 5, pp. 370–380, Oct. 2015.

[14] I. J. Vergara-Laurens, L. G. Jaimes, and M. A. Labrador, "Privacy-preserving mechanisms for crowdsensing: Survey and research challenges," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 855–869, Aug. 2017.

[15] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1732–1744, Jul. 2016.

[16] M. H. Cheung, F. Hou, and J. Huang, "Delay-sensitive mobile crowdsensing: Algorithm design and economics," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2761–2774, Dec. 2018.

[17] M. Xiao, J. Wu, L. Huang, R. Cheng, and Y. Wang, "Online task assignment for crowdsensing in predictable mobile social networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 8, pp. 2306–2320, Aug. 2017.

[18] B. Zhang, Z. Song, C. H. Liu, J. Ma, and W. Wang, "An event-driven QoI-aware participatory sensing framework with energy and budget constraints," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, pp. 42:1–42:19, Apr. 2015.

[19] C. H. Liu, B. Zhang, X. Su, J. Ma, W. Wang, and K. K. Leung, "Energy-aware participant selection for smartphone-enabled mobile crowd sensing," *IEEE Syst. J.*, vol. 11, no. 3, pp. 1435–1446, Sep. 2017.

[20] C. H. Liu, J. Zhao, H. Zhang, S. Guo, K. K. Leung, and J. Crowcroft, "Energy-efficient event detection by participatory sensing under budget constraints," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2490–2501, Dec. 2017.

[21] C. H. Liu, J. Fan, P. Hui, J. Wu, and K. K. Leung, "Toward QoI and energy efficiency in participatory crowdsourcing," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4684–4700, Oct. 2015.

[22] B. Zhang, C. H. Liu, J. Tang, Z. Xu, J. Ma, and W. Wang, "Learning-based energy-efficient data collection by unmanned vehicles in smart cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1666–1676, Apr. 2018.

[23] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[24] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2017, pp. 557–564.

[25] K. Karlsson *et al.*, "Vegvisir: A partition-tolerant blockchain for the internet-of-things," in *Proc. Int. Conf. Distributed Comput. Syst.*, 2018, pp. 1150–1158.

[26] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 3690–3700, Aug. 2017.

[27] W. Chen, M. Ma, Y. Ye, Z. Zheng, and Y. Zhou, "IoT service based on jointcloud blockchain: The case study of smart traveling," in *Proc. IEEE Symp. Service Oriented Syst. Eng.*, Mar. 2018, pp. 216–221.

[28] N. Kshetri, "Can blockchain strengthen the Internet of Things?" *IT Professional*, vol. 19, no. 4, pp. 68–72, 2017.

[29] D. Miller, "Blockchain and the Internet of Things in the industrial sector," *IT Professional*, vol. 20, no. 3, pp. 15–18, 2018.

[30] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," in *Proc. NIPS Deep Learn. Workshop*, 2013.

[31] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016.

[32] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comp. Syst.*, 2017.

[33] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network." in *Proc. USENIX Sec. Symp.*, 2015, pp. 129–144.

[34] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, no. 10, pp. 533–546, 2016.

[35] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on Ethereum's peer-to-peer network," *IACR Cryptol.*, vol. 2018, p. 236, 2018.

[36] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM CCS*, 2016, pp. 254–269.

[37] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in *Proc. Int Conf. Financial Cryptography Data Sec.*, 2016, pp. 79–94.

[38] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (SOK)," *Principles Sec. Trust*, pp. 164–186, 2017.
[39] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—A systematic review," *PloS One*, vol. 11, no. 10, 2016, Art. no. e0163477.
[40] D. Bradbury, "The problem with bitcoin," *Comp. Fraud Sec.*, vol. 2013, no. 11, pp. 5–8, 2013.
[41] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
[42] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.
[43] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

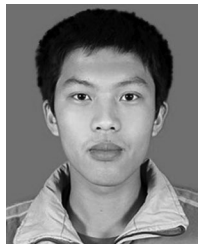
Chi Harold Liu (SM'15) received the Ph.D. degree in electrical and electronic engineering from Imperial College, London, U.K., and the B.Eng. degree in electronic and information engineering from Tsinghua University, Beijing, China.

He is currently a Full Professor and a Vice Dean with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. Before moving to academia, he joined IBM Research—China as a Staff Researcher and Project Manager, after working as a Postdoctoral Researcher with Deutsche Telekom Laboratories, Germany. He has authored or coauthored more than 90 prestigious conference and journal papers and owned more than 14 EU/U.S./UK/China patents. His current research interests include the Internet-of-things (IoT), big data, and deep learning.

Dr. Liu is the IEEE ICC 2020 Symposium Chair on Next Generation Networking, the Area Editor for *KSII Transactions on Internet and Information Systems*. He was the recipient of the IBM First Plateau Invention Achievement Award in 2012.

Qiuxia Lin is currently working toward the M.Eng. degree in computer science engineering under the supervision of Prof. Chi Harold Liu at the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China.

Her research interests include applying novel deep learning techniques like deep reinforcement learning and transfer learning on mobile computing, as well as blockchain.



Shilin Wen is currently working toward the Ph.D. degree in computer science and engineering under the supervision of Prof. Chi Harold Liu at the School of Computer Science and Technology at Beijing Institute of Technology, Beijing, China.

His research interests include applying novel deep learning techniques like deep reinforcement learning on mobile computing, big data system optimization, as well as blockchain.