Blockchain-Based Secure Computation Offloading in Vehicular Networks

Xiao Zheng, Mingchu Li, Yuanfang Chen, *Member, IEEE*, Jun Guo, Muhammad Alam, *Senior Member, IEEE*, and Weitong Hu

Abstract—**Vehicular ad hoc networks (VANETs) has become an important part of modern intelligent transportation systems (ITS). However, under the influence of malicious mobile vehicles, offloading vehicle tasks to the cloud server is threatened by security attacks. Edge cloud offloading (ECCO) has considered a promising approach to enable latency-sensitive VANET. How to solve the complex computation offloading of vehicles while ensuring the high security of the cloud server is an issue that needs urgent research. In this paper, we studied the safety and offloading of multi-vehicle ECCO system based on cloud blockchain. First, to achieve consensus in the vehicular environment, we propose a distributed hierarchical software-defined VANET (SDVs) framework to establish a security architecture. Secondly, to improve the security of offloading, we propose to use blockchain-based access control, which protects the cloud from illegal offloading actions. Finally, to solve the intensive computing problem of authorized vehicles, we determine task offloading via jointly optimizing offloading decisions, consensus mechanism decisions, allocation of computation resources and channel bandwidth. The optimization method is designed to minimize long-term system of delays, energy consumption, and flow costs for all vehicles. To better resolve the proposed offloading method, we develop a new deep reinforcement learning (DRL) algorithm via utilizing extended deep Q-networks. We evaluate the performance of our framework on access control and offloading through numerical simulations, which have significant advantages over existing solutions.**

Index Terms—**Vehicular ad hoc networks, blockchain, software-defined networking, computation offloading, edge-cloud computing, deep reinforcement learning.**

I. Introduction

RECENTLY, smart cities are developing rapidly. Secure data transmission between different objects is a vital component of the modern smart city. Therefore, communication between different entities, such as vehicle and smart devices, can be considered an important element of contemporary smart cities. Vehicle Ad Hoc Network (VANET) is a mobile ad hoc network (MANET) for vehicle environments in smart cities. As the requirements for convenient, safe and efficient transportation continue to increase, the mutual communication between connected vehicles in VANET plays an irreplaceable role in ITS [1]–[3]. However, of VANET still has challenges such as the adverse affects of malicious vehicles, the trust of connected connected vehicles, and the offloading of large-scale tasks [4]–[6].

In response to these challenges, mobile edge computing (MEC) can enable mobile devices (MD) to transfer its computation resources to nearby edge servers, and then become a promising method [7], [8]. In particular, when cloud computing and edge computing are combined, a new paradigm can be generated, and the standardized unified cloud computing offload (ECCO) model can be used to promote offload computing for VAENT networks. ECCO meets various Quality of Services (QoS) requirements by gaining the advantages of edge and cloud computing, thereby providing developers with efficient computing services in the mobile edge cloud. Mobile applications that do not require latency (for example, the large volume of vehicular data analysis) will be offloaded to a resource-rich cloud server, while others time-sensitive applications (that is, real-time monitoring of vehicle status, road emergency prediction, and road planning applications) will perform on edge servers to meet the rapid response service.

With the increasing amount of vehicles in VANET, the communication of different physical entities in a large-scale, high-mobility scenarios will product amount of real-time, high-speed, and continuous data flows. The result is that when offloading mobile tasks relies on untrusted MDs (here, roadside base units) of mobile vehicles in a dynamic environment, ECCO systems are prone to various types of threats. The result is that when offloading mobile tasks relies on untrusted MDs (here, roadside base units (RBU) of moving vehicles in a dynamic environment, ECCO is vulnerable to various types of threats. Unauthorized RBUs may achieve malicious access to utilize cloud services without central authorization. In addition, attackers can receive mobile data by threatening computing resources on cloud servers, which can cause privacy issues for VANET applications [9]. Therefore, how to ensure the safety of mobile offloading is crucial to any ECCO system.

The blockchain can be considered as a third-party system that does not require centralized trust management (i.e., agreements can be reached between different nodes to achieve distributedness) [10]–[14]. When the scale of VANET gradually increases, the traditional VANET model with centralized software-defined networking (SDN) control mechanism

Manuscript received January 30, 2020; revised June 12, 2020; accepted July 6, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61802097, Grant 61572095, and Grant 61877007, and in part by the Project of Qianjiang Talent under Grant QJD1802020. The Associate Editor for this article was S. Mumtaz. *(Corresponding author: Yuanfang Chen.)*

Xiao Zheng, Mingchu Li, and Jun Guo are with the School of Software, Dalian University of Technology, Dalian 116620, China.

Yuanfang Chen, Muhammad Alam, and Weitong Hu are with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: yuanfang.chen.tina@gmail.com).

Digital Object Identifier 10.1109/TITS.2020.3014229

obviously cannot meet the diverse needs of VANET. To solve this problem, the distributed-SDN control strategy has become a network architecture that will effectively and dynamically manage resources in VANET. In terms of security and data sharing of connected vehicle communications, distributed software-defined VANETs (SDVs) can achieve a partially trusted environment. The design of a peer-to-peer network is the core of the blockchain, where transaction information exists between multiple nodes and is not controlled by any single centralized entity. The decentralized and reliable blockchain combined with the distributed SDVs system to ensure security such as secure access control and resource allocation management between vehicle system. In particular, smart contract [15] is a computer program that runs on the blockchain background. Its feasibility has been confirmed by various vehicle network security issues. For instance, smart contracts have been proven to have access control capabilities in vehicle networks, provide access verification and data auditing [16]. In addition, smart contracts can protect cloud resources from malicious access [17]. Therefore, blockchain and smart contracts are considered to be applicable to vehicle networks, especially ECCO systems that can achieve the security goal of mobile task offload.

II. Related Work

The security of mobile edge cloud offloading has been widely discussed in recent work [18]–[20]. Blockchain can provide the security of an access control model. An access control design using a smart contract based on the blockchain is implemented on the cloud platform to verify and authorize access to mobile devices. In [21], the author proposed an access control model that uses blockchain-based that uses blockchain-based smart contracts to realize access permission verification, thereby supplying vehicle device. In [22] the author introduces a blockchain system that is used for access control and authorization. According to the research results obtained, the blockchain has been developed into IoT access been developed into IoT access systems such as the ECCO scenario we proposed to realize reliable resource networks [23], [24].

Compared with traditional access control solutions [25]–[28], using blockchain can bring the huge advantages to mobile offload security. First, the blockchain has proposed a decentralized management solution for the offloading system. The VANET data of vehicle offloading will be stored in the peer-to-peer network of the blockchain and does not need to rely on any centralized authorization, thereby ensuring fast access to data and significant enhancement of vehicle data privacy [29]. Second, by combining the blockchain with the edge cloud computing network, the offloading system will implement reliable access control through the use of smart contracts that can automatically authorize vehicles and distinguish vehicles from opponents, designed to prevent malicious clouds Computing resource offload behavior and potential threats. Therefore, the data integrity and offloading effectiveness of the system will be greatly improved. Finally, the peer-to-peer network structure supported by the blockchain will realize powerful access control without a single point of failure, the integrity of a large amount of mobile data offload, and system security [30]. Some offloading strategies in [31]–[34] utilize edge or edge or cloud services to solve network computing problems. The purpose is to use Lyapunov or traditional convex optimization methods to minimize offloading costs. However, this traditional offloading optimization method is only adapt to less-complexity online models and requires prior information of system statistical fields, but it is difficult to obtain such information in reality situations. To reslove these issues, reinforcement learning (RL) has become an effective technology that permits learners to adjust best solution via trial to achieve the best long-term goals without any previous knowlege [35]. previous knowlege [35]. However, complex computation offloading tasks in large-scale networks, as the dimensions of the state and operating space become larger, make RL-based solutions Fortunately, deep reinforcement learning (DRL) algorithms like as deep Q-Networks (DQN) [38] were presented as a powerful alternative as solving such high-dimensional spaces and proved their effectiveness in various MEC-based applications scalability and offload efficiency, eg as a scalability and offload efficiency, eg as a multi-base station virtual MEC [39]. In [40], introduced for vehicle networks, in which offloading and task allocation were proposed, and then the problem was settled via RL-based methods. To supply integrated computing services, author studied the offload computing framework in the combination of edge and cloud, in which the offloading decision, computing and communication resources are considered jointly to obtain optimization solve. However, the author decomposes the original optimization solutions into independent sub-optimal solutions and then processes them, sub-optimal solutions and then processes them, which are both time-wasting and complicated. In this paper, we focus on the security and offloading of ECCO systems. The main contributions of this paper (1) We propose a new secure computation offloading framework for a blockchain-based VANET network, in which a mobile vehicle can offload its tasks to a cloud or edge server to perform computation under an access control mechanism.

(2) We have designed a hierarchical architecture of controllable programming derived from SDN, which implements the dynamic orchestration of VANET security to achieve the communication of connected vehicles. The distributed SDN controller in the area control layer has the ability to gather vehicle consensus resource, and transmits the trust information it collects to the domain control layer.

(3) We have proposed a trusted access control mechanism that can use smart contracts on the blockchain to effectively detect and prevent illegal offloading of VANET devices. Its purpose is to verify vehicle identity, offloading tasks and manage offloading data to ensure the security and privacy of the ECCO system.

(4) We propose a dynamic offloading solution that considers offloading data size, available MEC computing power, throughput and bandwidth resources to offload its resource to the cloud or edge server. In particular, we propose an extended offloading algorithm based on DRL to attain the

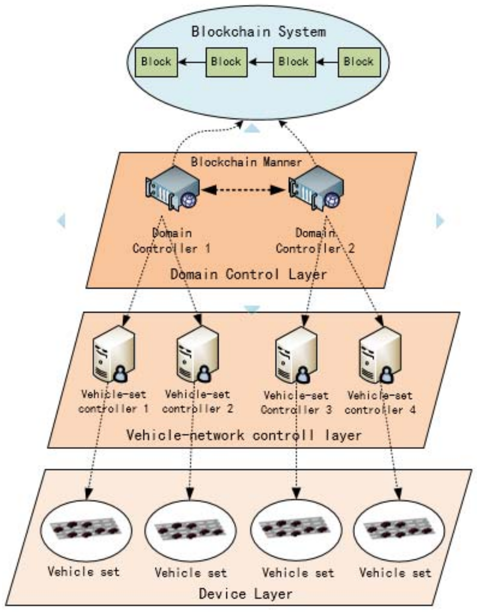


Fig. 1. Structure of hierarchical SDVs.

best offloading strategy for all vehicles, which should obey QoS requirements such as energy consumption and processing delay.

(5) We verify the proposed ECCO system via simulation experiments, and then investigate the access control and offloading performance.

III. Network Structure

In this section, we first depict the structure of hierarchical-SDVs and the way in which multiple controllers are interconnected using a blockchain, as shown in Fig.1. Learned from the picture: In the device layer of the network, this layer includes vehicle devices connected to the blockchain to communicate with each other and perform tasks offloading (i.e. refined data) to the edge computing. These refined data are the packets data which are transferred by a vehicle to its nearby. Usually these refined information are encapsulated in a message package and sent to the corresponding control region that can decapsulate the data packet and extract the associated message. Then, we describe a network system that combines access control, computation offloading, and consensus mechanisms. In the vehicle-network control layer, this layer is designed to collect vehicle and connection information and send them to the cloud control layer. The domain control layer runs by means of a distributed blockchain. The controller in the layer collects the refined information from the edge control, and then the message is shared to other controllers. Especially, hierarchical SDVs share model parameters trained by domain controllers to other domain controllers through the blockchain in a transactional manner. In the hierarchical-SDVs structure, each domain controller and regional controller has different modules and interacts with the blockchain and device layers in the hierarchical-SDVs.

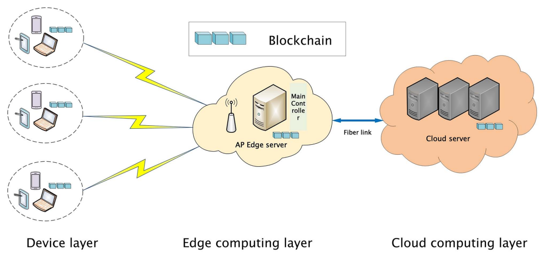


Fig. 2. Architecture of ECCO system [18].

Then we suggest an integrated edge-cloud structure for hierarchical-SDVs networks, as despicted in Fig.2. We describe the primary components of the ECCO system, including joint access control, computing offloading schemes, and consensus mechanisms.

A. Hierarchical-SDVs for Edge-Cloud Computing System

Below we give a detailed description of our proposed hierarchical-SDVs network: device layer, edge computing control layer, and cloud control layer. The main features of each layer are described below.

-**Mobile Device layer:** This layer is composed of hierarchical SDVs mobile devices (MDs) connected to the blockchain. Each MD has a blockchain account that can access the network and offload tasks to the cloud server.

- **Edge computing layer:** It consist of wireless access point (AP) or base station (BS) for wireless communication with mobile devices. Main controller (MC) are used to perform tasks, while lightweight edge servers are used to handle instant data. This layer be able to supply low-latency computing services at network edge. But, for complicated computing resource, the edge server forwards them to the resource-rich cloud server over a wired line to prevent overloading tasks on the edge side. Furthermore, edge servers are also considered to be blockchain entities, which ensure security by establishing trusted communications with cloud nodes and MDs on the blockchain network. All transactions and offloading actions in the offloading system are recorded by the blockchain and then broadcast to the edge server to reach a common agreement.

-**Cloud control layer:** This layer includes powerful computing and storage functions that can solve the complex computing tasks of local vehicle network equipment. In the ECCO system, the cloud also includes a blockchain entity network manager that controls access to all vehicles, which is used for smart contract admins and miners for mining transactions.

B. Consesus Mechanism Description

The consensus mechanism in hierarchical-SDVs structure is to guarantee that all consensus nodes in the blockchain are executed in the same order and each transaction is written to the distributed ledger, while the accounting nodes only need to synchronize the data from the consensus nodes, so there is no need to join in the consensus process. After the verification signature, message authentication code (MAC) and smart contract steps, Blockchain-VAENT sends the verified block back to the main control layer, and then appends it to the blockchain. The main controllers investigate the payload informations.

Next, we give a consensus mechanism called redundant Byzantine fault tolerance (RBFT) in the blockchain. This algorithm provides $f = \frac{U-1}{3}$ (in which K is node in blockchain and is defined as $\mathcal{K} = \{1, 2, \ldots, k, \ldots, K\}$, the consensus nodes can be difined as $\mathcal{U} = \{1, 2, \ldots, u, \ldots, U\}, \mathcal{U} \in \mathcal{K})$ fault tolerance under the premise of ensuring liveness & safety [41]. For this consensus mechanism, the blockchain system has only one primary node in control of sequencing device requests. At the same time, the replica node executes the demands in succession supplied by the primary node.

To our knowledge, practical Byzantine fault tolerance (PBFT) is usually the consensus mechanism used in blockchains. Compared with RBFT, RBFT adds a transaction authentication process to this RBFT. The primary node packs transactions into blocks and verifies them, then adds the verification result to the PRE-PREPARE message, and broadcasts it to the entire network. The consensus mechanism using RBFT is divided into five processes within the authorized blockchain structure, which are shown as:

1. The request starts, i.e., the number of consensus nodes requested by all main controllers: Suppose there are L main controllers in main control layer and is defined as $\mathcal{L} = \{1, 2, \ldots, l, \ldots, L\}$, the purpose of these controllers is to select the appropriate consensus node based on the authentication message of all nodes in the blockchain.

2. The request ends, i.e., all main controllers send request information to all consensus nodes: A controller sends $((transactions, ID_l)_{\mathcal{X}_l}, ID_l)_{\mathcal{X}_{\vec{l}}}$ to all the consensus nodes. This shows that information $(transactions, ID_l)$ is signed with node l's public key, and authenticated by a MAC vector with a send node l.

3. Propagate process, i.e., all the consesus nodes propagate request information to all replicas: After verifying the request, each node will propagate the received demands to all other consensus nodes. As long as there is at least one correct node receiving the demand, this process ensures that each correct node can eventually receive the demand. Such as, a node u obtains $(propagate, (transactions, ID_l)_{\mathcal{X}_l}, u)_{\overrightarrow{u\prime}}$ propagate massage from a node $u\prime$. Node u verifies the signature of the transactions when the MAC is valid. Otherwise, node u transmits propagate information to all other nodes. In this process, each replica node makes $U - 1, (U \in K)$ MAC and $U - 1, (U \in K)$ signatures, and each replica node obtains $f + 1$ propagate message from other replica nodes.

4. Three-phase agreement, i.e, pre-prepare, prepare, commit three phases: As shown in Fig.3, if the $U - 1, (U \in K)$ replica node obtain request, then the primary node sends $(PRE - PREPARE, ID_{Pr}, ID_l, H(l))_{\mathcal{X}_{\overrightarrow{Pr}}}$ PRE-PREPARE message authenticated by MAC for replica nodes, in which $H(l)$ is Hash value of block. The primary node products $U-1$, $(U \in K)$ MACs for all other replica nodes.

When a replica node acquires a PRE-PREPARE demand from the primary node, it verifies MAC's validity. Then it replies $(PREPARE, ID_{Pr}, ID_l, H(l), ID_{replica})_{\mathcal{X}_{\overrightarrow{replica}}}$ to all other replica nodes, in which each replica node products $U - 1, (U \in K)$ MACs, and each replica node needs to track $2f$ matching PREPARE messages received from different replicanodes consistent with the correct replica node.

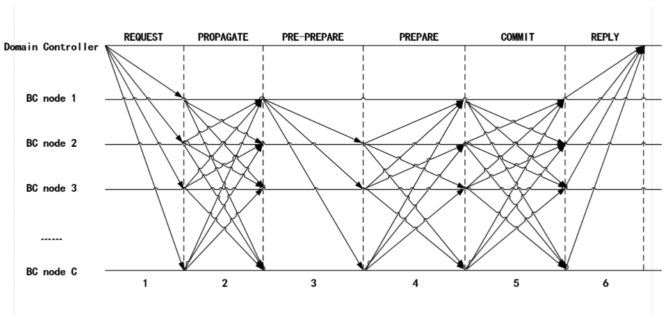


Fig. 3. Consensus process in blockchain.

After completing the PREPARE message processing, each replica node will send a commit demand, which is authenticated by the MAC. Such as a replica node products $U - 1, (U \in N)$ MACs to all other replica nodes, and it verifies one MAC to examine the validity of the arriving $(COMMIT, ID_{Pr}, ID_l, H(l), ID_{replica})_{\mathcal{X}_{\overrightarrow{replica}}}$. When receiving $2f + 1$ matching COMMIT messages from different replica nodes, $c\prime$ will verify the smart contract. If it's valid, it will attach to the blockchain.

5. Reply process, i.e., the consensus nodes deal with the request and transmit reply informations to all other main controllers: After performing the request demand, each node transmits $(REPLY, ID_u)_{\mathcal{X}_{u,l}}$ to domain controller. When obtaining the reply message from the blockchain system, each controller needs to verify $f + 1$ effective reply messages to prove that the smart contract has been successfully performed in each replica.

C. ECCO System Description

Through the network settings, we focus on the ECCO system with the concept of access control and offloading on the blockchain network. It is precisely because the mobile multi-vehicle network environment is dynamic and scalable that access control and computing offloading require a more comprehensive design to achieve system security and offloading purposes. The ECCO process is dispict as Fig.4, which involves access control and computation offloading, as shown below.

At first, the vehicle begins to convert the request into a blockchain transaction to compute the offloading process, and transimits this demand to the cloud server through the wireless access point (AP) (note that for unified standard, all vehicle access control is performed on the central cloud server). The cloud server uses an access control mechanism enabled by the smart contract to authorize this demand. If the demand is successfully authorized, the reaction is returned to the vehicle so that the vehicle will offload its task. Access control with completing transactions can be recorded and stored on the blockchain network safely. Secondly, authorized vehicles will choose to transfer their computing tasks to an edge or

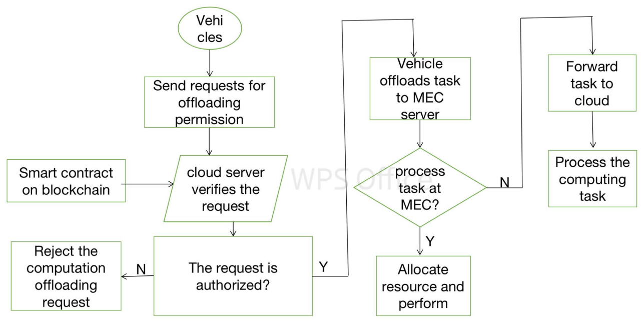

Vehi cles
Send requests for offloading permission
Smart contract on blockchain
cloud server verifies the request
Reject the computation offloading request
N
The request is authorized?
Y
Vehicle offloads task to MEC server
process task at MEC?
N
Y
Allocate resource and perform
Forward task to cloud
Process the computing task

Fig. 4. Flowchart of ECCO system.

cloud server for computing. During each offloading period, the controller at the edge layer optimizes according to QoS requirements and current network conditions to determine the execution location of the mobile task, that is, it is executed in the MEC or cloud server to obtain the best computing benefits. The MC in the edge layer optimizes to determine where mobile tasks will be performed, i.e., in the MEC or cloud server to obtain the best computing advantage. If the task is performed at the edge side, the MEC server will allocate communication and computing resources for vehicles. But if the task exceeds the computing capacity of the MEC server, it will be transmitted to the cloud server with rich resources for computing. Note that the data offloaded from the vehicle can be safely stored in the decentralized cloud storage on the blockchain.

IV. System Model

In this section, we introduce the hierarchical-SDVs in ECCO system as shown in Fig.1. Related network structures include remote cloud servers, MEC servers, and mobile vehicle networks. All vehicles are connected to edge servers and cloud servers via BS or AP. Our proposed blockchain VAENT includes the following three schemes.

A. Access Control Mechanism

We introduce an access control mechanism on blochchain network for ECCO system as shown in Fig. 5. For better offload computing, all mobile vehicles were previously controlled on the cloud server, and all access controls on the MEC server were ignored. In our proposed algorithm, the blockchain verifies the synchronization on the distributed ledger through the community unit, and these ledgers are copied between cloud nodes (main controller, dominator, miner) in the peer-to-peer cloud network. The distributed cloud architecture based on distributed access control ensures the high availability and computing serviceability of the system by eliminating the single point obstacle problem of traditional centralized cloud systems. Below we introduce several key components of the access control mechanism: main controller, dominate, miners.

-Main controller: The main controllers play an important part in our access control structure. It is in charge of monitoring all offloading tasks on the blockchain network,

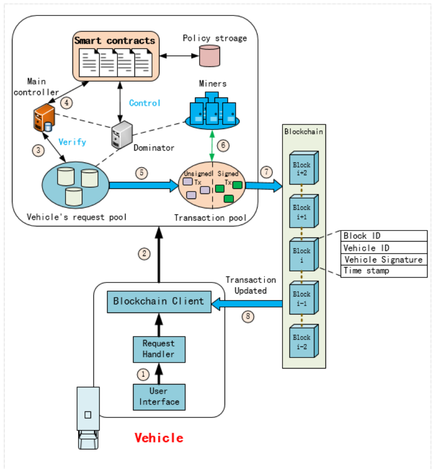


Fig. 5. Consensus process in blockchain.

involving offloading demands and access authentication of mobile vehicles.

-Dominator: It controls transactions and actions on the cloud server by adding, subtracting, or revoking access authority. The dominator is in control of deploying smart contracts and the only entity that can update or modify the policies in the smart contract.

-Smart contracts: Smart contracts give definitions of all actions permited to an access control frame. Mobile vehicle interacts with smart contracts through contract addresses and application binary interfaces (ABI). Smart contracts identify, verify request information and grant vehicle access authority by triggering transactions. All blockchain entities have access to smart contracts and their operations.

-Miners: The miner manages to verify data blocks composed of mobile vehicle transactions. Under the management of miners, all verified blocks will be attached to the blockchain through a process called consensus mechanism.

Next, we introduce the specific steps of ECCO model: First, at MD layer, we designed a block chain-VAENT module, which implements a complete function blockchain network involved in the clouds layer. This module is in charge of encoding transactions and data demands, signing transactions, and connecting to the blockchain for transaction tracking. In addition, vehicle interaction is designed by the user interface module, and user request information is processed by the request handler module. The specific flow of its access control is as follows:

(1) The mobile vehicle initializes the demand task as an offloading transaction, and performs computation offloading to the edge-cloud server.

(2) The blockchain control end processes the demand information and sends it to the storage pool for smart contract verification.

(3) The main controller collects demands for mobile vehicles in the storage pool on a first come first served basis.

(4) The main controller verifies the demand through a smart contract with a control strategy. When the demand is received, the reaction is returned to the mobile vehicle to offload the data.

(5) The transactions to be offloaded are divided into data blocks, and then placed in a transaction pool for verification by miners.

(6) The miner verifies the data block and sign them with digital signature to attack to the blockchain.

(7) Add offloading transactions to the blockchain network and broadcast to all mobile vehicles in the our proposed blockchain-VANET system.

(8) Offloading transactions are tracked by the main controller and updated at the mobile vehicle.

B. Trust Computing Mechanism

For the trust computing, we define the set of mobile vehicles as $\mathcal{N} = \{1, 2, \dots, N\}$. The vehicle is in each area of the device layer, and there exist $\mathcal{B} = \{1, 2, \dots, b, \dots, B\}$ areas in the device layer. In addition, each vehicle updates its trustworthiness by interacting with neighboring vehicles. Vehicles n_b and $n_{b'}$, interact with each other in a predetermined time interval $[t, t+\Delta t]$ in area b. Computing the direct trust of each vehicle based on the interaction of two neighboring vehicles is as follows:

$$Trust_{n_b n'_b}(t) = \frac{f^{Correct}_{n_b n'_b}(t)}{f_{n_b n'_b}(t)}, \quad t \leq W_{timestamp} \tag{1}$$

where $Trust_{n_b n'_b}(t)$ represents the direct trust of vehicle n_b to nearby vehicle n'_b. $f_{n_b n'_b}(t)$ indicates the total number of packets forwarded from n_b to node n'_b, and $f^{Correct}_{n_b n'_b}(t)$ represents the number of correctly forwarded packets during this time period t, where $W_{timestamp}$ denotes the size of the current timestamp. The sequence number of the data package is apply to define the number of lost or correctly forwarded data packets. Based on the sequence number of each package, the packagw was symbolized as correctly accepted or lost in the most recent timestamp $W_{timestamp}$. This is handled by comparing the recently obtained sequence number with the last sequence number achieved in the neighbor routing table. Therefore, each node can verify the correct number of packages accepted.

After each interaction is completed, vehicle n'_b examines whether vehicle n_b correctly forwarded the data packet at time slot t. If so, the trust value $Trust_{n_b n'_b}(t)$ increases, otherwise it is decreased. In the blockchain-VAENT trust model at the device layer, the trust value range for each vehicle is 0 to 1 (i.e., $0 \leq Trust_{n_b n'_b}(t) \leq 1$). Trust value of 0 indicates whole distrust, while trust value of 1 indicates absolute trust. If two vehicles do not interact in each area, define the initial trust value as 0.6 (smaller trustworthiness vehicles). We present a threshold δ for distinguishing malicious vehicles to describe the trust threshold. That is, if a vehicle has a trust value less than the threshold value δ, it is regarded as a malicious node.

C. Computing Offloading Model

For the computation offloading model, we assume that the mobile vehicle is authorized by the access control mechanism proposed in the above subsection. We consider the specific application scenarios of the VANET such as large-scale data analysis, where the processing tasks are very large and cannot be fully performed on the local device. Therefore, an authorized mobile vehicle must offload its computing tasks to an edge or cloud server for efficient performance. We first design the task model, computing model, and trust computing model, and finally give a detailed description of the problem.

1) Task Model: For each moving vehicle, the computing task can be defined as a tuple variable $R_n = (X_n, Y_n)$, where X_n (bits) represents the data size of the task of the mobile vehicle n. We also suppose that when X_n is offloaded to an edge or cloud server, its size is fixed. Y_n (CPU cycles / bits) denotes the total number of CPU cycles needed to fulfill the task R_n. For a set of$N$ mobile vehicles, define an offloading strategy vector $A = [\alpha_1, \alpha_2, \dots, \alpha_n]$ in which $\alpha_n \in \{0, 1\}$. When the computation task is offloaded to the edge/ cloud server, then $\alpha_n = 1$. Otherwise when it is executed on device layer, $\alpha_n = 0$. It is emphasized here that each computing task is performed on only one platform during each offloading cycle. For easy-to-handle network analysis similar to existing work, we assume that all vehicles and wireless networks remain stable during offloading. For many VAENT applications we care about, such as RBS unit status detection or the large volume of vehicular data analysis, this assumption is feasible. Compared to mobile vehicle's mobility and wireless network dynamic time scales, these applications have much shorter offloading cycles. Next, we consider the problem of resource allocation for computing offloading (that is, the problem of bandwidth allocation). For edge computing, the MEC server assigns its computing tasks to each vehicle to execute computing offloading. Nevertheless, we should consider allocating limited radio bandwidth to improve the offload efficiency of our proposed blockchain-VAENT system. We assume the bandwidth of the entire system is B Hz. To avoid interference, we allocate some bandwidth to each mobile vehicle. We normalize the bandwidth allocated to mobile vehicles to $w_n \in [0, 1]$ and $\sum w_n = 1$. Let g_n represent the channel gain between the mobile vehicle and the MEC server. The data transmission rate of the mobile vehicle is expressed as $r_n = w_n B log_2(1 + \frac{p_n g_n}{N_0\ w_n})$ where p_n (W)is the the transmission power of the mobile vehicle, N_0 (dm/Hz)is additive noisy power spectral density. Finally, we consider the consensus mechanism in the blockchain nodes: it includes the trust feature of the vehicle $T_{n_b n'_b}(t)$, and the trust feature of each node $\phi^u_p(t)$. Because there is no centralized security service, all nodes and controllers have unique trust features. Moreover, $\gamma_K(t)$ indicates which node is the consensus node, if $\gamma_k(t) = 1$ indicates that node k can be chosed by consensus nodes.

2) Computation Model: We use edge computing and cloud computing models to consider our proposed computing problems.

a) Edge computing: Here we consider the case where the computation task R_n of the mobile vehicle is offloaded to the MEC server for computing. Let $T^{(e)}_n$ represent the edge computing delay which involves the transmission delay of the mobile vehicle delivering data to the MEC server and the delay performed on the MEC server and the entire delay of the vehicle transmitting the data package. So the entire computing delay is expressed as

$$T_n^{(e)} = \frac{X_n}{r_n} + \frac{Y_n}{f_n^{(e)}} + t_{trans}^{n_b n_b'} \tag{2}$$

where $f_n^{(e)}$ (CPU cycles/s) represents the CPU cycle frequency of the edge computing task assigned to the mobile vehicle. The edge computing tasks assigned to all mobile vehicles will not exceed the total computing capacity of the MEC server, i.e., $f_n^{(e)} \leq F^{(e)}$.

$$t_{trans}^{n_b n_b'} = \sum_{i=1}^{U_s} t_i^s + \sum_{i=1}^{U_f} t_i^f \tag{3}$$

where indicates the total number of packets successfully transmitted to the neighboring vehicle n_b of n_b', U_f expresses the total number of failed packages to be sended to its neighbor n_b'.

We suppose that with IEEE802.11p enhanced distributed channel access (EDCA), the number of transmission attempts for vehicle n_b to successfully send packet i to its neighbor n_b' is U_s. Therefore, the time when the data packet is successfully transmitted can be defined as follows:

$$t_i^s = ACW_{U_{s'}} + T_f + \sum_{u=1}^{U_s'} (ACW_u + T_s) \tag{4}$$

where ACW_u represents the average contention window (ACW) size for package transmission attempts, and $ACW_{U_{s'}}$ indicates the average contention window size of the last successful package i, T_s represents the successful transmission time of the access category, which can be defined as follows:

$$T_s = T_{header} + T_{packet} + T_{ACK} + T_{SIFS} + 2\tau \tag{5}$$

where τ indicates the propagation delay, T_{header} and T_{packet} indicate the time of the package header and data message, respectively. T_{SIFS} and $T_{ACK}$are the time for short inter-frame interval and the ACK acknowledgement.

Similar to this, the transmission time of unsuccessful packets dropped after the maximum number of retransmissions U_f' can be expressed as:

$$t_i^f = \sum_{u=1}^{U_f'} (ACW_u + T_f) \tag{6}$$

where T_f indicates that the transmission failure time of the access category is as follows:

$$T_f = T_{header} + T_{packet} + T_{SIFS} + T_{timeout} + \tau \tag{7}$$

where $T_{timeout} = T_{SIFS} + T_{slot}$ is the ACK timeout delay.

Finally, the ACW size is defined as:

$$ACW_u = \frac{\min(CW_{max}, 2^u \times (CW_{min} - 1)) \times T_{slot}}{2} \tag{8}$$

where CW_{max}, CW_{min} are the maximum and minimum contention window sizes determined by on the IEEE 802.11p access classification, respectively.

Failure to consider available bandwidth when selecting a trusted neighbor vehicle as the next hop may bring about higher package delays and losses. Thus, we determine the available throughput (AT) to assess the maximum data rate that each vehicle vb will send successfully as:

$$AT_{n_b n_b'} = \frac{Trust_{n_b n_b'} \times f_{n_b n_b'}}{t_{trans}^{n_b n_b'}} \tag{9}$$

where $Trust_{n_b n_b'} \times f_{n_b n_b'}$ indicates the total number of packages successfully sent by the vehicle n_b.

In addition, the energy consumption offloaded to the MEC server includes the energy consumption for data transmission and implementation, and the cost of the computing power of the blockchain system. Here, blockchain computing capability is the blockchain nodes mentioned in the Section 2 six steps trying to reach consensus with each other. In each step, energy costs are divided into primary node energy costs and replica node energy costs. We define p_n^j as the power consumption of the vehicle in inactive state, the total energy cost of offloading data to the MEC server is:

$$E_n^{(e)} = \frac{p_n X_n}{r_n} + \frac{p_n^j Y_n}{f_n^{(e)}} + E_{primary} + E_{replica} \tag{10}$$

Next we analyze these six processes to obtain the energy consumption of the primary node and the replica node. First perform a theoretical analysis on the controller that sends requests to all nodes: Suppose there are D transactions transferred from the domain control layer to the blockchain system, and the υ fraction of the transaction sent from the controller is correct [42]. Considering that each consensus node generates a MAC, verifying a MAC and a signature requires α, α, and β cycles, respectively. Therefore, the energy cost of the primary node in this step is:

$$E_{primary}^{req} = (\alpha + \beta) + \frac{D}{\upsilon}(\alpha + \beta) \tag{11}$$

and the energy cost of each replica $E_{replica}^{req}$ is the same as $E_{primary}^{req}$.

Secondly, we analyze the propagation process: Suppose that the primary node firstly acquires the arriving transaction, verifies the MAC and signature, and then sends the PROPAGATE request to all other consensus nodes. Therefore, the energy cost of primary nodes is:

$$E_{primary}^{pro} = (\frac{D}{\upsilon} + U - 1)(\alpha + \beta) \tag{12}$$

and the energy cost of replica is:

$$E_{replica}^{pro} = (\frac{D}{\upsilon} + 1)(\alpha + \beta) \tag{13}$$

Thirdly, pre-prepare process analysis: The master node produces $U - 1$ MACs for all replicas whose replicas verify the MAC and signature of the T/υ transaction and the $U - 1$ MAC of the PROPAGATE message. At the same time, each replica stores the resulting transactions. So the energy cost of primary node is:

$$E_{primary}^{pre} = (U - 1)\alpha \tag{14}$$

and the energy coat og replica is:

$$E_{replica}^{pre} = \alpha + \frac{D}{\upsilon}(\alpha + \beta) \tag{15}$$

Fourth, we analyze the preparation process: The primary node verifies $2f$ matching PARPARE messages, and each replica produces $U-1$ MACs for all other consensus nodes and verifies $2f$ MACs. Therefore, the energy cost of the primary node is:

$$E^{par}_{primary} = 2f\alpha \tag{16}$$

and the energy cost of repilca is:

$$E^{par}_{replica} = (U-1+2f)\alpha \tag{17}$$

Fifth, we analyze the commit process: The primary node and each replica generate $U-1$ MACs and verify $2f+1$ MACs. Thus, the energy cost of the primary node is:

$$E^{com}_{primary} = (U+2f)\alpha \tag{18}$$

and the energy cost of repilca is the same as $E^{com}_{primary}$.

Sixth, we analyze the reply process: The primary and replica servers generate D/υ MACs for the transactions of the M domain controller, Thus, the energy cost of the primary node is:

$$E^{rep}_{primary} = \frac{MD}{\upsilon}\alpha \tag{19}$$

and the energy cost of replica is the same as $E^{rep}_{primary}$.

Therefore, the total energy cost of primary node for each transaction can be shown as:

$$\begin{aligned} E_{primary} &= E^{req}_{primary} + E^{pro}_{primary} + E^{pre}_{primary} \\ &\quad + E^{par}_{primary} + E^{com}_{primary} + E^{rep}_{primary} \\ &= (\frac{M+2}{\upsilon} + \frac{U}{D})\alpha + (\frac{2}{\upsilon} + \frac{U}{D})\beta + (\frac{4f+2U-1}{D})\alpha \end{aligned} \tag{20}$$

Similary, the total energy cost of replica for each transaction can be shown as:

$$\begin{aligned} E_{replica} &= E^{req}_{replica} + E^{pro}_{replica} + E^{pre}_{replica} \\ &\quad + E^{par}_{replica} + E^{com}_{replica} + E^{rep}_{replica} \\ &== (\frac{M+3}{\upsilon} + \frac{2}{D})\alpha + (\frac{3}{\upsilon} + \frac{2}{D})\beta + (\frac{4f+2U}{D})\alpha \end{aligned} \tag{21}$$

Assume that a multi-core module runs in parallel on different cores. Each core has a computing speed of η Hz. Simultaneously, we discuss that the probability that the primary node becomes a malicious node is $k, k \in (0, 1]$. The purpose of reducing system performance is because malicious nodes will reduce system throughput. As a result, the throughput of the blockchain system will reach at most:

$$BT_{blockchain} = \min\left\{\frac{k\eta}{E_{primary}}, \frac{k\eta}{E_{replica}}\right\} \tag{22}$$

Here, the $BT_{blockchain}$ estimates the number of transactions processed by the blockchain system per second.

To improve the system throughput, we will jointly consider the network throughput and the blockchain system throughput to formulate the immediate reward function. From (9) and (22), the immediate reward function is:

$$\begin{aligned} R(t) = \sigma[\sum_{b=1}^{B} \frac{\Pi_{k_b=1}^{K_b} T_{k_b,k+1_b} \cdot f_K}{\sum_{k_b=1}^{K_b}(t^{k_b,k+1_b}_{trans})}] \\ + \delta \min \times \left\{\frac{k\eta}{E_{primary}}, \frac{k\eta}{E_{replica}}\right\} \end{aligned} \tag{23}$$

where $f_K = \sum_{k=1}^{L} f_k$, $k = \sum_{u=1}^{U} a_p^u(t)\phi_p^u(t)$, here, $a_p^u(t)$ is the consensus node vector in the blockchain, and its value is $\{0, 1\}$, and $U = \sum_{n=1}^{N} a_n(t)\gamma_n(t)$, the value of $a_n(t)$ is also $\{0, 1\}$, which indicates the number of selected consensus nodes, if $a_n(t) = 1$ indicates that it can be selected as a consensus node, otherwise, it cannot be selected.

b) Cloud Computing: If the data task is offloaded to the cloud server, the mobile vehicle needs to offload his task to a MEC server that can forward its task to the remote cloud server though a wired connection. The cloud server also assigns its tasks to efficiently compute tasks. We denote the data rate of the wired link used to transmit the mobile vehicle n task as $r_n^{(w)}$ and the cloud resource allocated to mobile vehicle n as $f_n^{(c)}$. The total cloud computing latency and energy cost can then be shown as:

$$T_n^{(c)} = \frac{X_n}{r_n} + \frac{X_n}{r_n^{(w)}} + \frac{Y_n}{f_n^{(c)}} + t^{n_b n'_b}_{trans} \tag{24}$$

$$E_n^{(c)} = \frac{p_n X_n}{r_n} + p_n^j(\frac{X_n}{r_n^{(w)}} + \frac{Y_n}{f_n^{(c)}}) + E_{primary} + E_{replica} \tag{25}$$

According to (2)-(23), the computing delay and energy consumption of mobile vehicle n in our blockchain-VAENT system indicate:

$$T_n = T_n^{(e)} + T_n^{(c)}, E_n = E_n^{(e)} + E_n^{(c)} \tag{26}$$

3) Offloading Model: In this section, we combine computation offloading, edge resource allocation, and available throughput into joint optimization problems. Our goal is to minimize the total cost of computation delay, energy consumption, and available throughput for all vehicles in the ECCO system. We design the cost function of mobile vehicle n as a weighted sum of delay time, energy consumption, and available throughput. The formula is $C_n = \rho^t T_n + \varrho^e E_n - \gamma^t R(t)$ represent the weight for the delay time and energy consumption as well as reward cost for multi-vehicle SDVs, subject to the offloading decision $\boldsymbol{A} = [\alpha_1^{(e)}, \alpha_1^{(c)}, \ldots, \alpha_N^{(e)}, \alpha_N^{(c)}]$, edge task allocation $\boldsymbol{f} = [f_1, f_2, \ldots, f_N]$, bandwith resource allocation $\boldsymbol{w} = [w_1, w_2, \ldots, w_N]$ and consensus mechine decision $\boldsymbol{\Psi} = [\psi_{\mathcal{V}}, \psi_p, \psi_{\mathcal{N}}]$ as follow:

$$\begin{aligned} \text{(P1):}\quad & \min_{\boldsymbol{A},\boldsymbol{f},\boldsymbol{\Psi}} \sum_{n=1}^{N} C_n \\ & \text{subject to}\quad (C_1): \alpha_n^{(e)}, \alpha_n^{(c)} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \\ & \qquad (C_2): \sum_{n=1}^{N} f_n^{(e)} \le F^{(e)}, \\ & \qquad (C_3): f_n^{(e)} \ge 0, \quad \forall n \in \mathcal{N}, \\ & \qquad (C_4): 0 \le Trust_{n_b n'_b}(t) \le 1, \end{aligned}$$

$$(C_5) : \psi_V(t) \in \{0, 1\}, \quad \forall V \in \mathcal{V},$$
$$(C_6) : \psi_p^u(t) \in \{0, 1\}, \quad \forall u \in U,$$
$$(C_7) : \psi_n(t) \in \{0, 1\}, \quad \forall n \in \mathcal{N},$$
$$(C_8) : \sum_{n=1}^{N} \omega_n \leq 1, 0 < \omega_n \leq 1, \quad \forall n \in N.$$

Here, Constraint (C_1) indicates the binary offloading decision strategy of mobile vehicle n, offload to MEC server or offload to cloud server. (C_2) and(C_3) indicate that the assigned edge resources do not exceed the total computing capacity of the MEC server. (C_4) indicate the trust range for each vehicle is limited to 0 to 1, a trust value of 0 indicates complete distrust, while a trust value of 1 indicates absolute trust. When there is no interaction between two vehicles in each area, we set the initial trust value to 0.6 (smaller confidence vehicles). Here, we propose the threshold θ for distinguishing malicious vehicles to describe the trust threshold. (C_5) indicate the vehicle's action vector in each area of the device layer. If $\psi_1(t) = 0$ at time slot t, this implies that vehicle 1 is not choosed as a trusted neighbor vehicle. Otherwise, it means being selected as a trusted neighbor. In addition, the goal of each vehicle is to select one of the most trusted nodes (ie, the highest trust value) to communicate with as its neighbor. Therefore, suppose a route composed of K nodes is expressed as $\mathcal{K} = \{1_b, \ldots, l_b, \ldots, L_b\}, \mathcal{K} \in \mathcal{V}$. The total trust of this routing path in area b can be expressed as $\Pi_{k_b=1}^{K_b} T_{k_b,k+1_b}(t)$. (C_6) indicate the vector representing consensus nodes in the blockchain. If $\psi_p^1(t) = 1$ shows that node 1 can be choosed as the primary node. If $\psi_p^1(t) = 0$ implies that node 1 acts as a replica. As shown in Section 3, when a blockchain system has only one primary node meanwhile: $\sum_{u=1}^{U} \psi_p^u(t) = 1$. (C_7) indicate the vector representing the number of selected consensus nodes. If $\psi_1(t) = 1$ means that node 1 can be choosed as the consensus node. If $\psi_1(t) = 0$ implies that node 1 is not selected as the consensus node in time slot t. (C_8) indicate a constraint on bandwidth allocation. When the number of mobile vehicles in the ECCO system is rapidly increasing, the size of the (P_1) problem may be very large. Existing research proposes to decompose the optimization problem into sub-optimization problems and solve them respectively, which is complicated and ineffectual for solving large-scale ECCO problems similar the situation we have proposed. Consequently, in the following section we suggest a new technique exploiting extended deep reinforcement learning to resolve the suggested offloading problem.

V. Proposed Solution

In this section, we present a computation offloading method for ECCO system.

Expanded DRL-based computation offloading for ECCO system:

A. Problem Formation

In the paper, we depict the computing offloading problem as a reinforcement learning (RL) process. we proposed offloading solution is to minimize the weighted sum cost of all mobile vehicles in the proposed blockchain-VANET computation delay and energy consumption and consensus mechanism. In particular, in this work, we considered a realistic multi-vehicle ECCO, in which computing tasks, edge computing resources, the size of system bandwidth resources, and the state of consensus mechanisms are highly dynamic. Therefore, the blockchain-VAENT system has a large system state and action space. Moreover, it allocates resources (ie, edge or bandwidth) to each mobile vehicle in each system state to optimize the total offload cost. This poses a new challenge to the traditional method of effectively solving the problem of multi-scale offloading. Thence, in this section, we put forward an offloading method utilizing extended DRL, which is applicable to multi-vehicle scenarios as our proposed high-dimensional system. We first present the offloading framework exploiting DRL, which defines the state space, action space, and rewards.

- State space: System state is selected as $s = (t, e, \omega, Trust_{n_b n'_b}(t), \phi_p^U(t), \gamma_K(t))$ where t is the total offloading cost of ECCO system, e is available computing resources for MEC server, i.e., $e = F^{(e)} - f_n^{(e)}$. ω indicates the available bandwidth resources, which can be specified as $\omega = B_{bandwidth} - \sum_{n=1}^{N} \omega_n$. $Trust_{n_b n'_b}(t)$ indicates the trust feature of vehicles, $\phi_p^U(t)$ is trust feature of each node in the blockchain, and $\gamma_K(t))$ indicates which node is the consensus node.
- Action space: The action space is denoted as an offloading decision vector $\boldsymbol{A} = [\alpha_1^{(e)}, \alpha_1^{(c)}, \ldots, \alpha_N^{(e)}, \alpha_N^{(c)}]$, edge computing resource allocation $\boldsymbol{f} = [f_1, f_2, \ldots, f_N]$, bandwidth resource allocation $\boldsymbol{w} = [\omega_1, \omega_2, \ldots, \omega_N]$ and consensus mechine decision $\boldsymbol{\Psi} = [\psi_{\mathcal{V}}, \psi_p, \psi_{\mathcal{N}}]$. Therefore, the action vector is expressed as: $a = [\alpha_1^{(e)}, \alpha_1^{(c)}, f_1, \omega_1, \psi_{\mathcal{V}}, \psi_p, \psi_{\mathcal{N}}, \ldots, \alpha_N^{(e)}, \alpha_N^{(c)}, \ldots, f_N, \omega_N, \psi_{\mathcal{V}}, \psi_p, \psi_{\mathcal{N}}]$.
- Reward space: The goal of the RL agent is to observe the optimal offloading decision action a for each state s to minimize the total cost $C(s, a)$ of the ECCO system. In particular, the reward function is negatively related to the objective function of the optimization problem P_1 in the Section 3. Therefore, we devise the system reward as $R(s, a) = -C(s, a)$.

B. Introduction to Deep Reinforcement Learning (DRL)

The RL is intended to be expressed as a Markov Decision Process (MDP). For the RL model, the agent interacts with the environment to find the optimal operation, and does not need an explicit system dynamics model. In our ECCO system, the agents did not have experience and information about the VANET environment at the beginning. Therefore, it needs to take some action in each offload state to explore each state information, namely the current network data size or available edge resources. As long as the agent and the environment can obtain some experience from the actual interaction, it will continue to explore while using known state information. When the Monte Carlo method and dynamic programming are combined, the time difference (TD) method can be used to enable the agent to learn the offloading strategy without the need for state transition probability. In reality, for example, in our dynamic mobile blockchain, the state transition probability is difficult to obtain. Therefore, we can use the free model RL to develop a dynamic offloading scheme. In particular, in this paper, we aim to find the optimal strategy to minimize the offloading cost. To do this, you can use the experience tuple of the agent $(s^t, a^t, R^t, s^(t+1))$ at each time step t to update the state-action function in our offloading model as below:

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha[R(s^t, a^t) + \gamma \cdot \min Q(s^{t'}, a^{t'}) - Q(s^t, a^t)] \quad (27)$$

which is the so-called Q-learning algorithm. $\gamma \in [0, 1]$ is the discount factor, α is called the learning rate, and $\sigma^t = R(s^t, a^t) + \gamma \cdot \min Q(s^{t+1}, a^{t+1}) - Q(s^t, a^t)$ is called TD error, compared to the best Q value will be zero. In addition, under the optimal strategy π^* obtained by the maximum Q-value, i.e., $\pi^*(s) = argmax Q^*(s, a)$ the Bellman optimal equation of the state action equation will be denoted as

$$Q^*(s^t, a^t) = E_{s' \sim E}[R(s^t, a^t) + \gamma \cdot \min Q^*(s', a')] \quad (28)$$

It turns out that Q-learning can converge once indefinitely and can reach the optimal Q^*. Although RL can resolve the offloading problem though getting the best rewards, there are still other problems. The state and action values of the Q-learning algorithm are stored in a two-dimensional Q-table, but this technology may not be feasible for solving complex problems with large state-action space. Mainly because if we make all Q values in a table, the matrix $Q(s, a)$ may be large, which will make it difficult for the learning agent to obtain enough samples to explore each state, making the learning algorithm unavailable. To overcome the above challenges, we employ deep learning and deep neural networks (DNN) to approximate the Q value rather than the traditional Q table, resulting in a new algorithm named deep reinforcement learning (DRL). For the DRL algorithm, the DNN uses the weight θ to approximate the target Q-value $Q(s^t, a^t, \theta)$. In addition, in order to resolve the instability of the Q network caused by the function approximation, an empirical replay method is adopted during the training period, in which the buffer to store the experience $e^t = (s^t, a^t, R^t, s^{t'})$ at each time t. Then, choose the transition $(s^j, a^j, R^j, s^{j'})$ random mini-batch from the replay memory to train the Q network. Minimize the loss function by iteratively updating Q network training with weights θ, which can be expressed as

$$L(\theta) = E[((R^j + \gamma \cdot \min_{a^{j'}} Q(s^{j'}, a^{j'}|\theta')) - Q(s^j, a^j|\theta^j))^2] \quad (29)$$

C. Expanded DRL-Based Computation Offloading

In recent years, we have seen tremendous endeavors in DRL to better the performance of DLR-based algorithms. In the paper, we are concerned about two current advancements in DRL research to suitable for our proposed offloading problems based on dual DQN [43] and duel DQN [44].

- Double DQN: For general DQN, we adopt the same samples in the replay memory to define which is the best action to predict the action value, and then cause overestimation of the action value. To settle the above problems, two Q functions which are selected and evaluated by a new loss function are proposed, as shown below

$$L_{double}(\theta) = E[((R^j + \gamma \cdot Q(s^{j'}, min_{a^{j'}} Q(s^{j'}, a^{j'}|\theta^1), \theta^2)) - Q(s^j, a^j|\theta^j))^2] \quad (30)$$

Note that the action selection is still based on the weight θ^1, but the assessment of the choosed action depends on the weight value θ^2. This method reduces the problem of overestimation and increases the training process, thereby improving the computation efficiency of DQN.

-Dueling DQN: In some MDP problems, the Q function computation process does not need to estimate the action and state values simultaneously, so we can estimate the action and state value functions separately. Inspired by the idea, when confronting DQN, the state-action value $Q(s, a)$ can be expressed as the following two value functions: $Q(s, a) = S(s) + A(a)$. Here, $S(s)$ is a state value function for evaluating the importance in a given state s. $A(a)$is an action value function for estimating the importance of selecting one action contrasted to other actions. First compute $S(s)$ and $A(a)$ respectively, and then combine them to produce the final output $Q(s, a)$. This method can get better performance in the strategy evaluation of MDP, particularly in complex problems with a large space for action (such as the offloading situation we consider).

In view of the advantages of the above two methods, we developed an extended DQN algorithm based on these two improved methods to resolve the offloading problem of the ECCO system we proposed. The specific details are shown in Algorithm 1. The EDRL algorithm uses an iterative approach to achieve the best computation offloading strategy. This structure products a task offloading strategy for mobile vehicles based on the system state and inquires the system rewards in each time t so that the offloading strategy can be optimized (see lines 8-15). Immediately after the process updates the historical experience tuples and trains the Q-network with a loss function minimization (see lines 17-21). This trial and error solution will avoid a priori requesting information for the offload mechanism. Since the trained deep neural network (DNN) can better describe the setting during training, the proposed offloading algorithm can dynamically suit to the actual VANET situation.

VI. Performance Evaluation

In this section, we will assess the computation offloading efficiency in our proposed EECO system through numerical simulation. Below we give the simulated settings for computing the offloading.

A. Simulation Settings Used to Compting Offloading

In our computing offloading simulation, consider the ECCO system and cloud server, MEC server with multiple mobile vehicles is authorized by the access control mechanism. Here we define the existence of $N = 15$ mobile vehicles, each of which has a computing task to be performed on the edge or

Algorithm 1 Expanded DRL-Based Computation Offloading (EDRLCO) Algorithm for ECCO

```
1: **Initialization:**
2: Define the capacity of the replay memory to $N$
3: Initialize the deep Q network $Q(s, a)$ with random weights $\theta$ and initialize the exploration probability $\epsilon \in (0, 1)$with $\theta'$
4: **for** $t = 1, \dots, N$ **do**
5:     Initialize the state vector $s^0$
6:     **for** $t = 1, 2, \dots$ **do**
7:         Schedule computation offloading
8:         Estimate current unloading costs $t$
9:         Estimate available edge computing resources $e$
10:        Estimate available bandwidth resources $w$
11:        Estimating vehicle trust feature $Trust_{n_b n'_b}(t)$
12:        Estimating the trust characteristics of each node in the blockchain $\phi_p^U(t)$ and verify the consensus nodes $\gamma_K(t))$
13:        Set $s^t = (t, e, w, Trust_{n_b n'_b}(t), \phi_p^U(t), \gamma_K(t))$
14:        Randomly choose a random action $a$ with probability $\epsilon$, otherwise $a = argminQ(s^t, a^t, \theta)$
15:        Offload computing resource $\alpha_t^{(e)}(D_t)$ to MEC server or cloud $\alpha_t^{(c)}(D_t)$
16:        Observe the reward $R^t$ and the next state $s'$
17:        Assessing system costs $C(s^t, a^t)$
18:        /*** Update***/
19:        Store the experience information $(s^t, a^t, R^t, s')$ in memory $\mathcal{D}$
20:        Randomly sample the mini-batch state transition probability $(s^j, a^j, R^j, s')$ from memory $\mathcal{D}$
21:        Compute the target Q-value by $R^j + \gamma \cdot Q(s^{j'}, min_{a^{j'}} Q(s^{j'}, a^{j'}|\theta), \theta')$
22:        Use weight $\theta^j$ in $((R^j + \gamma \cdot Q(s^{j'}, min_{a^{j'}} Q(s^{j'}, a^{j'}|\theta), \theta')) - Q(s^j, a^j|\theta^j))^2$ as loss function to perform gradient descent
23:        Training deep Q networks with updates of $\theta$ and $\theta'$
24:    **end for**
25: **end for**
```

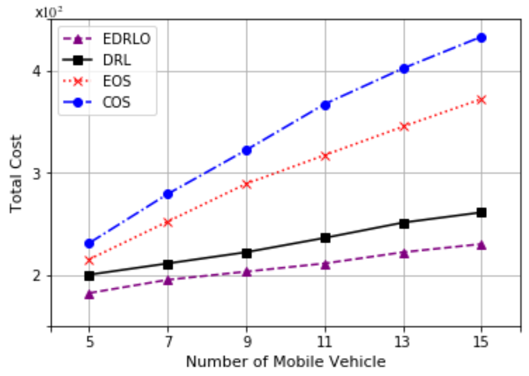
cloud server. We suppose that the data size of the computation task $\mathcal{D}$ of the VAENT network is randomly distributed between 0.5MB and 15MB. Total bandwidth resource $\mathcal{B}$ is set to 20 MHz. The additional noise power spectral density N_0 is expressed as -100dBm / Hz. In addition, the total computing power of the MEC server is set to $F^{(e)} = 5$ GHz and the cloud server is set to $F^{(c)} = 12$GHz. We assume that the state of each vehicle will be trusted (trust level higher than 0.6), suspicious (trust level between $[\delta, 0.6]$) and malicious (trust level lower than δ). In our simulation experiment, we suppose that the threshold of the trust level δ is 0.5. A trusted vehicle is a trusted node that aims to build a secure route from a source node to a destination. The hierarchical-SDVs supplies a good way to prevent using suspicious and malicious vehicles entities. It is possible for each vehicle to maintain its state or change it at the next instant t. For this we define the transition probability of each vehicle in each area. In our simulation, the trust features of each node in the blockchain will be divided into three types of situations: trusted, suspicious, and untrusted. The most trusted node is choosed as the primary node. Then the transition probability matrix of each node is set as $\Omega = ((0.5, 0.25, 0.25), (0.7, 0.1.0.2), (0.5, 0.35, 0.15))$.

Each node in the blockchain has the potential to turn into a consensus node, which will select consensus nodes from each domain controller to vote in a preference manner. We fix the transition probability matrix of each blockchain node as $\Phi = ((0.5, 0.25, 0.25), (0.7, 0.1, 0.2), (0.35, 0.25, 0.4))$. In addition, we set $\alpha = 4$ MHz, $\beta = 0.05$ MHz, and the blockchain size is 1 Mb. Therefore, the IEEE.802.11 MAC protocol we use is 802.11p, and the topology coverage at the mobile device layer is 5×5 km^2, and the data rate is 5.5Mbps. In addition, for the extended DQN learning algorithm, simulation is implemented in Python using TensorFlow 2.0, and we use the AdamOptimizer to optimize the loss function of the training procedure. All simulation experiments were executed on a computer with an Intel Core i7 4.7GHz CPU and 256 GB of memory.

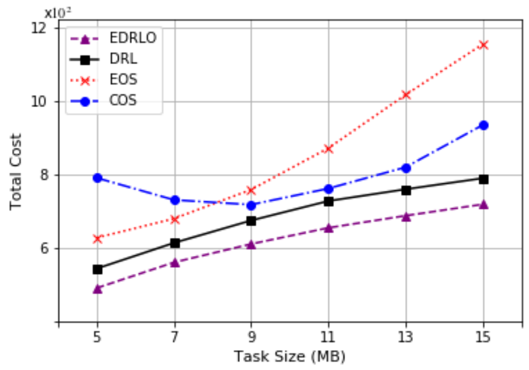
B. Computation Offloading Performance

We rated the extended DRL-based computation offloading method (EDRLCO) with different performance indicators. For comparison, we use the other three options as a basis: (1) DRL-based offloading scheme whose offloading is executed through regular DRL, (2) Edge Offload Solution (EOS), all mobile vehicles offload their computing tasks to the MEC server, and (3) Cloud Offload Solution (COS) is where all mobile vehicles offload their computing tasks to a cloud server.

We first rate the effect of the ECCO's total cost relative to amount of mobile vehicles and tasks, as shown in Fig.6. More specifically, in Fig. 6 (a), we suggest the ECCOT system with a different number of mobile vehicles, and each mobile vehicle has a computing task (the range of the task size is 0.1Mb- 1Mb). It can be seen from the simulation results that the curves of all offloading schemes increase with the number of mobile vehicles. In particular, COS is the offloading solution with the highest cost of offloading. One explanation is that in such test cases, the computing tasks of a mobile vehicle are relatively small, therefore offload data to a remote cloud will result in greater transmission delays and therefore higher offloading costs. At the same time, owing to the low-latency computing service of the MEC server, the EOS solution indicates better offloading efficiency and lower offloading costs in any vehicle situation. Especially importantly, DRLO and EDRLCO show much lower offloading costs, and EDRLCO achieves the best performance in all offloading scenarios, the reason is explained below. Firstly, in the proposed extended DRL algorithm, the dual DQN network obtains the optimal strategy of system gain larger than that of the conventional DQN. Firstly, in the proposed extended DRL algorithm, the optimal strategy for dual DQN networks to achieve greater system gain than traditional DQN. Secondly, dueling DQN structure evaluates the state value function and the action value function separately, thereby achieving better performance in

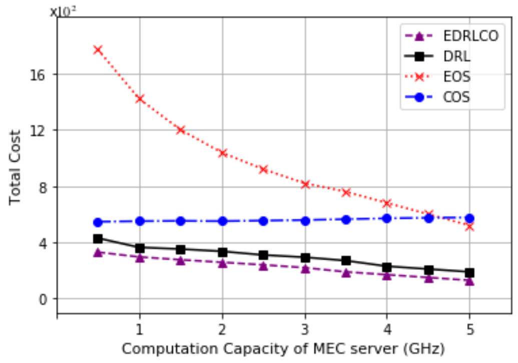


(**6.a**) Total cost versus number of mobile vehicles

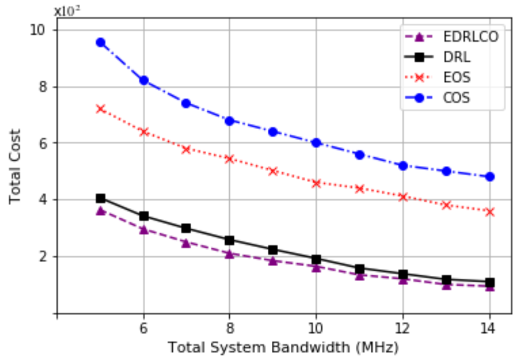


(**6.b**) Total cost versus number of computing tasks

Fig. 6. Total offloading cost versus number of mobile vehicles and computing tasks.



(**6.a**) Total cost versus edge computing resource



(**6.b**) Total cost versus bandwidth resource

Fig. 7. Total offloading cost versus resource allocation.

stategy evaluation. Accordingly, these advanced technologies make the EDRLCO algorithm more effective in evaluating the optimal offload strategy and performance.

Next, we explain the computation offloading performance of the ECCO system with a single mobile vehicle $N = 1$ and a task size varying between 5MB and 15MB, as shown in Fig. 6 (b). It can be seen that when the task scale increases, the cost of the four schemes also increases because of the increasing the number of network data to be fully performed. In particular, when the task size is small (< 8MB), the offloading cost of the COS solution is higher than that of the EOS solution. The reason behind this is that the MEC server uses sufficient computing resources to efficiently handle small tasks. Thence, offloading small tasks to a remote cloud can cause transmission delays, resulting in a higher total offload costs for the COS solution. But, as the task size becomes larger (> 8MB), the computing power of the MEC server is not enough to contain all resource, but the resource-rich cloud server will efficiently compute large-scale resource. Consequence, compared to the EOS, the COS can achieve lower offloading costs. The EDRLCO algorithm also obtains the lowest total offloading cost, and only adopts the DRLO scheme with smaller gaps when the task size increases.

In addition, we compare the offloading methods in resource allocation as shown in Fig. 7. We firstly rate the effect of edge computing resource allocation on offloading capability. It can be seen from Fig. 7 (a) that the total offloading cost based on the edge computing scheme decreases significantly with the increase of the computing power of the MEC server. In addition, since all computing tasks in this solution are performed on the cloud server, as the MEC capacity increases, COS has a stable offloading cost. Especially when edge computing power is small, EOS has the highest offload cost. This is precisely because the MEC server with less computing resources will result in higher execution latency compared to other solutions, so the computation cost of EOS is higher. However, as the edge computing capacity gradually increases, the total system cost of EOS will be greatly reduced, and it can reach lower costs than COS. The results can provide more foresight on how to achieve the correct allocation of MEC servers to enhance the overall offloading performance of the ECCO system. Similarly, compared with other benchmarks, the proposed EDRLCO algorithm can obtain the lowest cost and move in the downward direction, thus showing the best performance.

At the same time, Fig. 7 (b) indicates the relationship between total offloading cost and system bandwidth assignment. It is obvious from the simulation results that all the curve results of the offloading scheme gradually decrease with the increase of the total system bandwidth. This is why bandwidth allocation is always critical to increasing the data transfer rate between the vehicle and the edge-cloud server. This will cause the solution to reduce transmission latency and total offload costs correspondingly. In addition, by using the proposed dynamic offloading strategy, the method proposed applying RL will dynamically adjust how much bandwidth resources will be assigned to some mobile vehicles according to the task size of the mobile vehicle, thereby achieving the optimal offloading of the ECCO system.

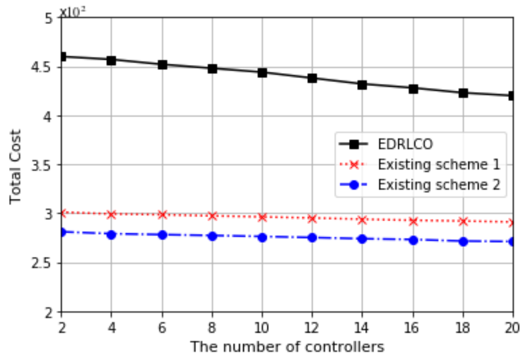


Fig. 8. Total cost versus the number of controllers.

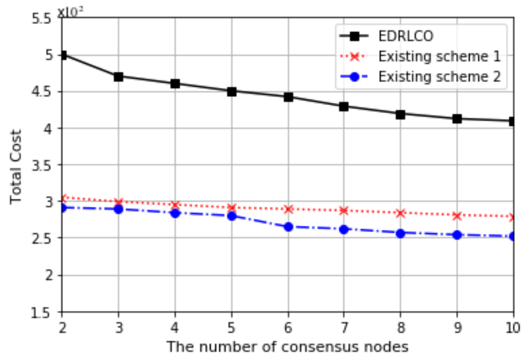


Fig. 9. Total cost versus the number of consensus nodes.

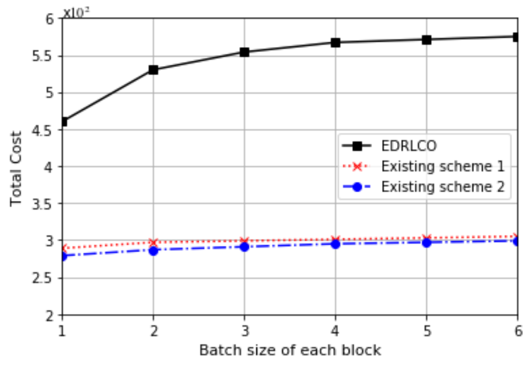


Fig. 10. Total cost versus the batch size of each block.

We also analyze the efficiency of the proposed EDRLCO scheme by comparing to other two existing schemes: (1) Existing scheme 1 has local computing capability, which does not need to select a trusted vehicle, and has a traditional vision change. This scheme grants connected vehicles to communicate with each other randomly without considering the trust features, and the use of traditional view change protocols to select the primary node without having to consider the trust function of each blockchain node. Besides, this solution does not adopt the hierarchical network structure. At the same time, the scheme also uses the dueling deep Q-learning (DDQL) method. (2) Existing scheme 2 has local computing capability, which does not need to select a trusted vehicle, and has a traditional vision change. At the same time, the scheme does not use the DDQL method. This scheme permits domain controllers to randomly access the blockchain system. Each domain controller randomly selects a blockchain node as a consensus node. Fig. 8 illustrates the total cost comparison with the number of controllers in the main control layer. Specifically, the graph can also simulate performance in a large real-world network environment, as there are up to 30 main controllers interacting with the edge control layer and blockchain system. From the figure we will infer that as the number of domain controllers increases, the system performance cost decreases simultaneously. This is due to the fact that more domain controllers require more computing resources to verify signatures and MACs. However, our proposed algorithm utilizes a hierarchical blockchain control platform and RBFT consensus mechanism with better performance, as shown in the top image. At the same time, it can be seen that our proposed algorithm has better performance in the context of large-scale blockchain systems.

Fig. 9 denotes the performance cost comparison with the number of consensus nodes in the blockchain. The figure shows the system performance cost decreases as the number of consensus nodes increases. The reason is that as the number of consensus nodes increases, more signatures and MACs will be generated. These signatures and MACs require more CPU to process, which reduces system performance. However, our proposed algorithm is still better than existing solutions.

Figure 10 shows a comparison of the system performance cost and the batch size of each block. It can be seen from the figure that the system performance cost in the simulation experiment will increase as the blocks expand to different sizes. This is because a larger block size can accommodate more transactions, which can synchronize more local network events between main controllers, ultimately leading to increased system performance costs. Simultaneously, our proposed scheme is still superior to the existing scheme.

VII. Conclusion

In this paper, we combine blockchain and DRL for the ECCO system in the VANET network, and jointly investigate access control and computation offloading. We consider a general VANET scenario where multiple vehicles can offload their tasks to an edge or cloud server for collaborative performance. Then, we designed a hierarchical distributed software-defined VANET (SDVs) framework based on the blockchain. First, to improve the security of task offloading, we propose an access control enabled by smart contracts and blockchain to manage vehicle access to prevent malicious offloading access. We then propose a new DRL-based offloading scheme to achieve the optimal offloading strategy for all vehicles in VANET. We use the extended DQN algorithm to formulate task offloading decisions, consensus mechanism decisions, and edge resource as well as bandwidth allocation as joint optimization problems to minimize the total offloading cost of computation latency, throughput and energy consumption. We conducted an experimental simulation to evaluate the effectiveness of the proposed scheme. The results show that, compared with other benchmark methods, our scheme provides high security for the ECCO system and achieves performance improvements with minimum offloading costs. In the future, we will consider designing light-weight blockchains so that the access control architecture is devised and arrayed directly at the edge side. It will hopefully support time-sensitive network management services for offloaded systems.

References

[1] F. R. Yu, "Connected vehicles for intelligent transportation systems [Guest editorial]," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3843–3844, Jun. 2016.

[2] K. Abboud and W. Zhuang, "Impact of node mobility on single-hop cluster overlap in vehicular ad hoc networks," in *Proc. 17th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM)*, 2014, pp. 65–72.

[3] Y. Guo, Q. Yang, F. R. Yu, and V. C. M. Leung, "Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5445–5459, Jun. 2018.

[4] P. Tyagi and D. Dembla, "Investigating the security threats in vehicular ad hoc networks (VANETs): Towards security engineering for safer on-road transportation," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 2084–2090.

[5] M. N. Mejri, J. Ben-Othman, and M. Hamdi, "Survey on VANET security challenges and possible cryptographic solutions," *Veh. Commun.*, vol. 1, no. 2, pp. 53–66, Apr. 2014.

[6] Y. He, F. R. Yu, Z. Wei, and V. Leung, "Trust management for secure cognitive radio vehicular ad hoc networks," *Ad Hoc Netw.*, vol. 86, pp. 154–165, Apr. 2019.

[7] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[9] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 843–859, 2nd Quart., 2013.

[10] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[11] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[12] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[13] J. Xie *et al.*, "A survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2794–2830, 3rd Quart., 2019.

[14] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial Internet of Things: A dueling deep Q -Learning approach," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4627–4639, Jun. 2019.

[15] W. G. Ethereum, "A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[16] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for secure EHRs sharing of mobile cloud based E-health systems," *IEEE Access*, vol. 7, pp. 66792–66806, 2019.

[17] H. G. Do and W. K. Ng, "Blockchain-based system for secure data storage with private keyword search," in *Proc. IEEE World Congr. Services (SERVICES)*, Jun. 2017, pp. 90–93.

[18] D. Shibin and G. J. W. Kathrine, "A comprehensive overview on secure offloading in mobile cloud computing," in *Proc. 4th Int. Conf. Electron. Commun. Syst. (ICECS)*, Feb. 2017, pp. 121–124.

[19] S. Han *et al.*, "Energy efficient secure computation offloading in NOMA-based mMTC networks for IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5674–5690, Jun. 2019.

[20] I. Elgendy, W. Zhang, C. Liu, and C.-H. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Trans. Cloud Comput.*, early access, Jun. 18, 2018, doi: 10.1109/TCC.2018.2847347.

[21] R. Xu, Y. Chen, E. Blasch, and G. Chen, "BlendCAC: A smart contract enabled decentralized capability-based access control mechanism for the IoT," *Computers*, vol. 7, no. 3, p. 39, Jul. 2018.

[22] O. J. A. Pinno, A. R. A. Gregio, and L. C. E. De Bona, "ControlChain: Blockchain as a central enabler for access control authorizations in the IoT," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[23] I. Satoh, "Toward access control model for context-aware services offloaded to cloud computing," in *Proc. IEEE 35th Symp. Reliable Distrib. Syst. Workshops (SRDSW)*, Sep. 2016, pp. 7–12.

[24] J. Xu, L. Chen, K. Liu, and C. Shen, "Designing security-aware incentives for computation offloading via device-to-device communication," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6053–6066, Sep. 2018.

[25] Y.-H. Lin, J.-J. Huang, C.-I. Fan, and W.-T. Chen, "Local authentication and access control scheme in M2M communications with computation offloading," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3209–3219, Aug. 2018.

[26] I. Satoh, "Toward access control model for context-aware services offloaded to cloud computing," in *Proc. IEEE 35th Symp. Reliable Distrib. Syst. Workshops (SRDSW)*, Sep. 2016, pp. 7–12.

[27] S. Tu and Y. Huang, "Towards efficient and secure access control system for mobile cloud computing," *China Commun.*, vol. 12, no. 12, pp. 43–52, Dec. 2015.

[28] T. Li, C. S. Magurawalage, K. Wang, K. Xu, K. Yang, and H. Wang, "On efficient offloading control in cloud radio access network with mobile edge computing," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2258–2263.

[29] A. Muthanna *et al.*, "Secure and reliable IoT networks using fog computing with software-defined networking and blockchain," *J. Sensor Actuator Netw.*, vol. 8, no. 1, p. 15, Feb. 2019.

[30] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.

[31] X. He, H. Xing, Y. Chen, and A. Nallanathan, "Energy-efficient mobile-edge computation offloading for applications with shared data," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[32] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[33] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[34] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[35] I.-S. Comsa *et al.*, "Towards 5G: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1661–1675, Dec. 2018.

[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT, 2018.

[37] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[38] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[39] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[40] Y. Cui, Y. Liang, and R. Wang, "Resource allocation algorithm with multi-platform intelligent offloading in D2D-enabled vehicular networks," *IEEE Access*, vol. 7, pp. 21246–21253, 2019.

[41] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.

[42] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making Byzantine fault tolerant systems tolerate Byzantine faults," in *Proc. NSDI*, vol. 9, 2009, pp. 153–168.

[43] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[44] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*. [Online]. Available: http://arxiv.org/abs/1511.06581

Xiao Zheng received the B.S. degree from Shandong University in 2010, and the M.S. degree from Xihua University, Chengdu, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Software, Dalian University of Technology. Her current research interests include mobile edge computing and Game Theory.

Mingchu Li received the B.S. degree in mathematics from Jiangxi Normal University in 1983, the M.S. degree in applied science from the University of Science and Technology Beijing in 1989, and the Ph.D. degree in mathematics from the University of Toronto in 1997. He was an Associate Professor with the University of Science and Technology Beijing from 1989 to 1994. He was engaged in the research and development of information security at Long View Solution Inc. and Compute Ware Inc. from 1997 to 2002. Since 2002, he has been with the School of Software, Tianjin University, as a Full Professor, and since 2004, he has been with the School of Software Technology, Dalian University of Technology, as a Full Professor, a Ph.D. Supervisor, and the Vice Dean. His main research interests include theoretical computer science and cryptography.

Yuanfang Chen (Member, IEEE) received the M.S. and Ph.D. degrees from the Dalian University of Technology, China, and the second Ph.D. degree from University Pierre and Marie CURIE (Paris VI), France. She is currently a Professor with Hangzhou Dianzi University. She was an Assistant Researcher of the Illinois Institute of Technology, USA, along with Prof. X. Li. She has been invited as the Session Chair of some conferences, the Associate Editor of Industrial Networks and Intelligent Systems, and the Guest Editor of the MONET.

Jun Guo received the B.S. degree and the M.S. degree from the School of Information, Liaoning University, Shenyang, China, in 2012 and 2017, respectively. He is currently a Ph.D. candidate with the School of Software Technology, Dalian University of Technology, Dalian, China. His research interests include reinforcement learning, data mining and big data computing.

Muhammad Alam (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Aveiro, Portugal, in 2014, with a specialization in inter layer and cooperative design strategies for green mobile networks. In 2009, he joined the Instituto de Telecomunicacoes-Aveiro, Portugal, as a Researcher. He has participated in several European Union FP7 projects, such as Hurricane, C2POWER, ICSI, and PEACE and Portuguese government funded projects such SmartVision. He is currently a Senior Researcher with the Instituto de Telecomunicacoes and participating in European Union and Portuguese government funded projects. His research interests include the IoT, real-time wireless communication, 5G, vehicular networks, context-aware systems, and radio resource management in next generation wireless networks. He is the Editor of the Book Intelligent Transportation Systems, Dependable Vehicular Communications for Improved Road Safety.

Weitong Hu received the B.S. degree and the Ph.D. degree from the School of Software Technology, Dalian University of Technology, Dalian, China, in 2008 and 2015, respectively. He is currently a Lecturer with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China. His research interests include secret sharing, data hiding, and network security.