Cooperative Computation Offloading in Blockchain-Based Vehicular Edge Computing Networks

Ping Lang, Daxin Tian, *Senior Member, IEEE*, Xuting Duan, Jianshan Zhou, Zhengguo Sheng, *Senior Member, IEEE*, and Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*—As a novel computing paradigm, multiaccess edge computing (MEC) migrates computing and storage capabilities to edge nodes of the network to meet the requirements of executing computationally intensive or delay-sensitive tasks on intelligent vehicles. In addition, MEC fills the gap between cloud computing and terminals in vehicular networks. In the MEC system, to reduce the load on MEC servers with large-scale vehicle deployment and promote the efficient use of network resources, vehicles can also transfer tasks to neighboring resource-rich vehicles using cooperative computation offloading. However, cooperative computation offloading between vehicles faces the challenges of security and insufficient information about the server vehicle. Therefore, this paper proposes using blockchain technology to achieve efficient data sharing between vehicles and service providers (i.e., server vehicles) and ensure the security of computation offloading between vehicles. First, we design a secure data sharing architecture in blockchain-based vehicular edge computing networks. Then, a new consensus mechanism in this architecture is proposed to improve the efficiency of data sharing and prevent malicious attacks. Furthermore, we present a cooperative offloading decision-making method using an offloading game, and the Nash equilibrium of the offloading strategy is achieved using this method. The results of numerical experiments demonstrate the superior performance of the proposed method.**

***Index Terms*—Vehicular edge computing, cooperative computation offloading, data sharing, blockchain, game theory.**

Manuscript received 24 June 2022; revised 6 July 2022; accepted 8 July 2022. Date of publication 12 July 2022; date of current version 24 October 2022. This research was supported in part by the National Natural Science Foundation of China under Grants U20A20155, 62061130221, and 62173012, in part by the Beijing Municipal Natural Science Foundation under Grant L191001, in part by the Zhuoyue Program of Beihang University (Postdoctoral Fellowship), and in part by the China Postdoctoral Science Foundation under Grant 2020M680299. *(Corresponding author: Daxin Tian.)*

Ping Lang, Daxin Tian, Xuting Duan, and Jianshan Zhou are with the Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems and Safety Control, School of Transportation Science and Engineering, Beihang University, Beijing 100191, China (e-mail: langping@buaa.edu.cn; dtian@buaa.edu.cn; duanxuting@buaa.edu.cn; jianshanzhou@foxmail.com).

Zhengguo Sheng is with the Department of Engineering and Design, University of Sussex, Richmond 3A09, U.K. (e-mail: z.sheng@sussex.ac.uk).

Victor C. M. Leung is with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2022.3190308.

Digital Object Identifier 10.1109/TIV.2022.3190308

I. Introduction

ADVANCEMENTS in automated driving technology and intelligent transportation systems have led to a proliferation of applications for safety assurance, efficiency enhancement, and entertainment [1]–[6]. In contrast to traditional vehicles, in which people observe the environment and perform dynamic driving tasks with manual control, intelligent vehicles achieve intelligent environment perception by deploying various sensors such as cameras and LiDARs, and assist in driving or even achieve autonomous driving by performing computationally intensive and delay-sensitive tasks. To meet the requirements of these computationally intensive and delay-sensitive applications, more computing and storage resources must be deployed on vehicles, which increases the cost of deploying autonomous vehicles [7]. In addition, it is difficult to provide sufficient onboard resources due to the limited physical space on vehicles.

To facilitate the efficient and stable execution of onboard applications, researchers have developed a new architecture and corresponding technology called multiaccess edge computing (MEC) [8], [9], formerly known as mobile edge computing [10]. MEC helps vehicles achieve real-time data processing by employing the computing capabilities of the edge of the network. Therefore, a vehicle can offload difficult computing tasks to the MEC server to meet the delay demands of advanced onboard applications [11]. Compared with cloud computing, MEC can significantly reduce the transmission delay of data and ensure real-time processing of the data in vehicles or other terminals.

However, constrained by limited computing resources, the local capabilities of MEC servers may also be insufficient with the large-scale deployment of intelligent vehicles [12], [13]. A motivating example is as follows. During high traffic volumes (e.g., traffic jams or rush hours), many intelligent vehicles offload their computation tasks to a single roadside MEC server covering this area, which will cause the computation requirements to exceed the computation capacity of that server. As a result of competition between multiple vehicles for limited resources, the quality of service for applications will not be guaranteed [14]. That is, the computing tasks of vehicles will be completed beyond the maximum time they can tolerate, causing a large number of delay-sensitive applications to not be executed properly. A case in point is augmented reality or virtual

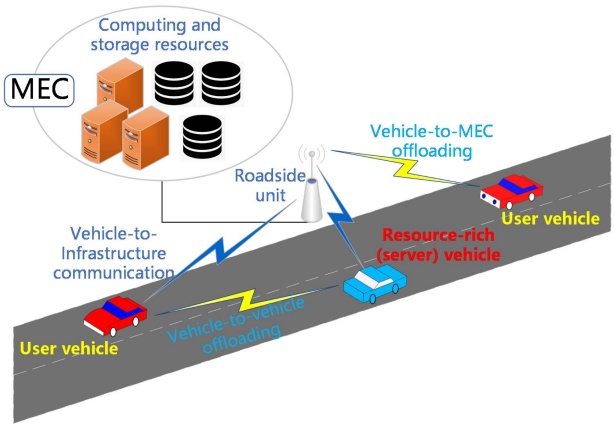

Fig. 1. Cooperative computation offloading in vehicular edge computing networks.

reality applications, such as cloud gaming in vehicles, which have critical delay requirements and involve heavy computation when the resolution and frame rate increase. These cloud gaming applications require extensive processing for rendering, which can be performed on the network or device side. The 3 rd Generation Partnership Project (3GPP) has specified performance requirements for cloud gaming (e.g., bandwidth and latency) and introduced split rendering architecture that allows rendering tasks to be performed flexibly on different entities [15], [16]. For example, some rendering tasks can be performed on the host vehicle, while other tasks can be performed on edge servers or on other vehicles when the edge servers are overloaded.

Therefore, in addition to offloading computing tasks to the MEC server, vehicles can also offload them to neighboring vehicles, which have redundant computing and storage resources, to reduce the burden on the MEC server [17], [18]. As illustrated in Fig. 1, in this scenario, the user vehicle can offload its onboard computing tasks to the MEC server or to neighboring vehicles according to the resource usage of the edge server and the number of neighboring service providers. This cooperative computation offloading effectively reduces the load on the MEC server, which helps to promote the efficient use of resources in the network and reduce waste caused by idle computing resources.

However, no methods have been developed for vehicles to accurately determine the computing capacity and trajectory of service providers for cooperative computation offloading. Transmission security and vehicle privacy are also difficult to guarantee in computation offloading between vehicles [19], [20]. Since computation offloading of vehicles often involves private data such as their trajectory, driving status, and surrounding environment, vehicles need to be sure that their offloading destination is a trusted service provider before computation offloading. In cooperative computation offloading, the server vehicle provides computation services for the surrounding vehicles as a service provider autonomously, which makes it difficult to guarantee the trustworthiness of its information. Once malicious nodes appear in the server vehicles, the privacy of user vehicles will be disclosed, which jeopardizes the information security and even the driving safety of vehicles and drivers. Therefore, it is necessary to construct a secure and tamper-proof sharing architecture of service provider information to ensure the trustworthiness of server vehicles.

Recently, blockchain technology has attracted a great deal of attention in vehicular networks in both academia and industry [21]–[23]. As a decentralized data storage technology, blockchain is characterized by distributed processing, multiparty consensus, and tamper-proof features [24], [25]. It can achieve secure synchronization and sharing of data between multiple parties and guarantee the credibility of information [26]–[28]. By integrating blockchain technology into vehicular edge computing networks, an efficient and secure data sharing mechanism can be established among MEC servers to supply information on neighboring service providers to user vehicles, which can significantly improve the cooperation and security of the system. Specifically, facing the previously mentioned security challenges, the blockchain-based secure information sharing mechanism can verify the identity of the server vehicle and ensure the trustworthiness of its service capabilities through the blockchain nodes. Moreover, attacks by malicious nodes can be identified and prevented by the consensus mechanism in the blockchain. In addition, the immutability of data in blockchain ensures the traceability of service information and thus the attacker can be traced after a malicious attack occurs.

To avoid the competition for the roadside server resources, vehicles can offload tasks to neighboring resource-rich vehicles (i.e., server vehicles or service providers) with redundant computing resources for execution, thereby reducing the use of roadside resources. Note that both user vehicles and server vehicles in this paper are intelligent vehicles, in which user vehicles are low-cost intelligent vehicles that deploy limited computing resources, and server vehicles are advanced autonomous vehicles with redundant computing resources. Therefore, the server vehicle can provide computing services to the user vehicle by cooperative computation offloading. With the addition of resource-rich vehicles as edge servers, this study faces the challenges of both trusted information sharing of server vehicles and policy evaluation for task offloading to server vehicles and roadside servers. In the existing studies of the cooperative computation offloading, the server vehicles are assumed to be trusted nodes and their service capabilities and trajectories have been accessed by the user vehicles. However, in practice, there is a lack of a trusted sharing mechanism for the information of server vehicles. In addition, the existing studies of blockchain-based data sharing in vehicular edge computing networks have not addressed the sharing mechanism for the information of service providers. Therefore, different from the existing studies, we propose a novel blockchain-based data sharing architecture for vehicular MEC networks to provide accurate information on server vehicles for cooperative computation offloading. Note that in terms of blockchain-based sharing methods and consensus mechanisms, this paper focuses on novel applications of blockchain technology in cooperative computation offloading, rather than proposing new algorithms in the blockchain field. In addition, we model the offloading decision process of each user vehicle using game theory to achieve an equilibrium of

offloading strategies in the region. The major contributions of this paper are summarized as follows:

- We propose a secure information sharing mechanism between user vehicles and service providers and a cooperative computing method for vehicles based on blockchain technology to ensure security and access to accurate information.
- In the blockchain-based data sharing architecture, we propose a consensus mechanism that combines Proof of Service [29] and Practical Byzantine Fault Tolerance (PBFT) [30] to achieve data synchronization between MEC networks and prevent malicious attacks.
- To help user vehicles make correct computation offloading decisions, we propose an offloading game model for cooperative computation offloading scenarios, in which user vehicles can select roadside MEC servers or neighboring resource-rich vehicles with different probabilities to execute applications and achieve Nash equilibrium. The performance of the proposed game method is evaluated by experimental comparison.

The remainder of this paper is organized as follows. Section II discusses related studies, while Section III presents the secure information sharing mechanism between user vehicles and service providers with a consensus mechanism in blockchain-based vehicular edge computing networks. Section IV presents the design of the offloading game method in cooperative computation offloading scenarios and describes the Nash equilibrium with a distributed solution. Section V provides the performance analysis of the proposed algorithm and the comparison with other computation offloading methods. Finally, Section VI concludes the paper.

II. Related Work

Computation offloading is a key research topic in MEC and mobile cloud computing, and there is a large amount of literature on computation offloading problems for vehicular MEC networks. Zhao *et al.* [31] researched the problem of computation offloading and resource allocation for cloud-assisted vehicular MEC, where the offloaded tasks could be executed on roadside MEC servers or migrated to cloud computing servers using roadside units. They proposed a collaborative optimization scheme for this problem and developed a distributed method to obtain the best optimization results. Ke *et al.* [32] modeled task computation offloading in an unstable environment with a varying channel state and available bandwidth to minimize the energy consumption, allocated bandwidth, and transmission delay. They then proposed a deep-reinforcement-learning-based adaptive computation offloading method to solve this problem and achieve the optimal offloading action. To reduce the overhead of MEC servers and improve the performance of computation offloading, Dai *et al.* [33] jointly modeled the offloading of vehicles and load balancing of roadside MEC servers and proposed a joint algorithm for server selection and offloading. Similarly, in our previous study [14], we proposed a computation offloading game for a large number of vehicles to reduce the computational burden on roadside MEC servers. We also designed a distributed best-response algorithm to guarantee the uniqueness of the Nash equilibrium converged by the offloading strategies of vehicles, thus achieving efficient and stable computation offloading.

As reported by the 5G Automotive Association (5GAA) [17], vehicles can migrate tasks to neighboring resource-rich vehicles for execution as well as offload them to roadside servers. Sun *et al.* [34] considered vehicle-to-vehicle (V2V) task offloading in a dynamic and uncertain vehicular wireless environment and proposed a distributed adaptive learning-based offloading method to optimize the delay in computation offloading. To relieve the workload of cloudlet nodes and improve resource utilization of vehicular fog computing, Yadav *et al.* [35] proposed a dynamic computation offloading solution to achieve energy-latency tradeoff and effective resource allocation in V2V offloading. They provided a heuristic method as the resource allocation solution and minimized the energy consumption and offloading latency. However, existing studies on V2V offloading assume that information on the service provider can be obtained by simple communication, which exposes vehicles to security and privacy risks while preventing them from effectively obtaining the service provider data.

As a decentralized and trusted storage approach, blockchain can facilitate data sharing and ensure the security and immutability of information. Thus, its applications in MEC and vehicular networks have been widely studied in the literature. Yang *et al.* [24] summarized various studies on integrated blockchain and MEC systems and provided a clear description of the complementarity of blockchain and MEC. They provided typical architectures of the integrated system and discussed the functions of computation, storage, and network in this system. To ensure the security and privacy of data storage and sharing in an integrated platform of MEC and vehicular networks, Kang *et al.* [26] established an efficient data storage and sharing mechanism using blockchain and its smart contracts. In the information sharing scheme, they developed a logic-based evaluation mechanism to select secure information providers and ensure the credibility of the shared information. In another study, Zhang *et al.* [36] designed a hierarchical software-defined vehicular network with blockchain and used a dueling deep reinforcement learning (DRL) method to improve the throughput of this integrated system. Fu *et al.* [37] investigated efficient and secure machine learning methods for connected autonomous vehicles and designed a collective learning framework with blockchain to achieve the distributed training and sharing of machine learning models for connected autonomous vehicles. In this architecture, blockchain was used to protect the integrity of the shared models and to prevent malicious node attacks. To enhance the capability of onboard positioning and support autonomous driving, Li *et al.* [38] proposed a positioning error evolution sharing architecture using blockchain. They also proposed a deep-learning-based positioning correction method as a shared model to improve the applicability of the framework.

Some studies have also combined blockchain with computation offloading to improve the security of the latter. Guo *et al.* [39] designed a computation offloading architecture in blockchain-based MEC networks to verify and audit the

execution results of tasks using smart contracts in blockchain systems. In this architecture, the authors considered the variables of resource allocation, block size, and block number and presented a DRL-based method to optimize the latency and throughput of the joint system. Qiu *et al.* [40] also provided a computation offloading solution using DRL to offload the mining and data processing applications of mobile terminals to the MEC server in blockchain-empowered MEC networks. To ensure the security and privacy of relay-based computation offloading, Feng *et al.* [41] designed a relay-based offloading framework in blockchain-enabled MEC networks and optimized the throughput and computation rate of this integrated system using the asynchronous advantage actor-critic (A3C) algorithm. For a vehicular network, Zheng *et al.* [42] reused the hierarchical distributed software-defined architecture in [36] and proposed a secure access method with smart contracts to construct a trusted computation offloading framework in integrated edge-cloud networks. They also designed a DRL-based offloading solution to achieve a tradeoff between latency and energy consumption in computation offloading.

Blockchain can thus ensure secure sharing of information in computation offloading to prevent attacks by malicious nodes. However, to the best of our knowledge, existing studies have focused on the sharing of sensing data, autonomous driving learning models, positioning errors, and computation offloading applications on audit, mining, and access control in a combined blockchain-based MEC system while ignoring the information sharing requirements of service providers in cooperative computation offloading. In this paper, we propose an information sharing architecture and consensus mechanism for service providers based on blockchain in cooperative computation offloading to provide secure and accurate offloading service information for user vehicles. The proposed blockchain-based data sharing is a new application of blockchain in cooperative computation offloading to ensure the shared information of server vehicles is consistency and tamper-proof. Based on the shared information, we propose a cooperative offloading game model that allows user vehicles to make optimal offloading decisions.

III. Blockchain-Based Data Sharing

A. Scenario Overview

In this study, we assume that user vehicles drive within the coverage area of the roadside MEC server (also called the edge node), and that there are server vehicles around the user vehicles that can provide computing services, which are also called resource-rich vehicles or service providers. A blockchain system is deployed on each roadside MEC server, and the MEC servers can achieve data sharing and synchronization using blockchain.

When a user vehicle v_i is faced with a computing task that cannot be completed in the required time, the vehicle must decide whether to send the task to the roadside server or service provider. To make an accurate decision regarding computation offloading, the user vehicle must obtain the service capacity information of the neighboring server vehicles. In this scenario, to expand the transmission range of service capacities on server vehicles and prevent attacks by malicious nodes, we adopt blockchain to share the information of idle server vehicles to user vehicles to provide a basis for their decision to computation offloading.

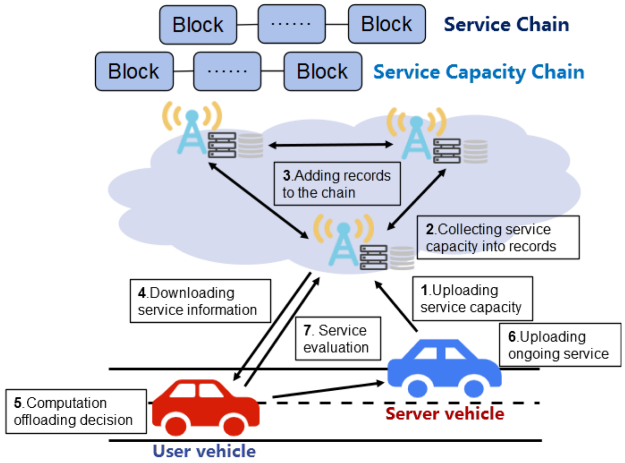

Fig. 2. Data sharing between user vehicles and service providers in blockchain-based vehicular edge computing networks.

B. Secure Data Sharing Scheme for Cooperative Computation Offloading

As mentioned above, in vehicular edge computing networks, each roadside MEC server shares the information of the server vehicle with the blockchain and broadcasts this information to vehicles within its coverage area. Note that user vehicles do not store all the received data, but filter the server vehicles with matching trajectories as potential service providers during data receiving to avoid excessive data storage. Then, the user vehicle that requires computation offloading makes a game decision according to the received information of the neighboring server vehicles and roadside MEC servers. Finally, the task can be migrated to the roadside MEC server or a server vehicle to ensure the secure and stable execution of the onboard application. The service provider data sharing scheme is designed as illustrated in Fig. 2 and described below.

Step 1: The server vehicle uploads its service capacity to the roadside MEC server. Within the range of the area covered by the MEC server MEC_m, the server vehicle v_j, which has redundant computing resources, determines its service capacity, such as the computing resources $resource_j$, service period $period_j$, planned trajectory $trajectory_j$, and initial price $price_j^{init}$. Then, v_j establishes a communication link with MEC_m and sends data regarding the service capacity to MEC_m after encryption, attaching the certificate of the vehicle and the signature of the message as

$$v_j \rightarrow MEC_m : E_{K^{pu}_{MEC_m}} \left(timestamp || Data_{Pseu^s_j} \right.$$
$$\left. || Cert_{Pseu^s_j} || Sign_{Pseu^s_j} \left(Data_{Pseu^s_j} \right) \right),$$

where $K^{pu}_{MEC_m}$ is the public key of MEC_m, $Pseu^s_j$ is the current pseudonym used by vehicle j, $Cert_{Pseu^s_j}$ and $Sign_{Pseu^s_j}$

are the corresponding certificate and signature, respectively, and

$$Data_{Pseu_j^s} = E_{K^{pr}_{Pseu_j^s}}(timestamp||resource_j ||trajectory_j||price_j^{init}||period_j) \quad (1)$$

is the data of the service capacity of v_j encrypted by its private key. Specifically, the form of the service capability data uploaded by the server vehicle v_j in (1) consists of the time of data generation, the computing resources, the travel trajectory, the desired initial service price, and the service price of v_j. These contents are further encrypted by the private key $K^{pr}_{Pseu_j^s}$ of v_j to form the service capability data uploaded by v_j. Note that since the vehicle periodically broadcasts its status containing the certificate and public key via vehicle-to-everything (V2X) communication [43], [44] in vehicular networks, we assume that both the vehicle and the roadside MEC server have obtained each other's public keys before data sharing.

Step 2: The roadside MEC server collects the vehicle service capacity information. After receiving the encrypted message sent by v_j, MEC_m decrypts the message with its private key to obtain the service capacity provided by v_j, and verifies the signature of v_j. Then, MEC_m identifies the service capacity information and stores it as a service capacity record in the form of a transaction record as

$$record = (timestamp||recordID||provider||resource ||trajectory||price||period||quality), \quad (2)$$

where $recordID$ is the index of the service capacity record, $provider$ is the service provider, and $resource$, $trajectory$, and $period$ are the computing resources, trajectory, and service duration, respectively, that the server vehicle can provide for the service. In addition, $quality \in [0, 1]$ is the evaluation of the service quality provided by the user vehicle: the higher the value, the higher the evaluation. In (2), we represent the service capability of the server vehicle in the form of a transaction record stored in the blockchain. Specifically, the previously mentioned data generation time, the index of the record, and the serial number, the computing resources, the travel trajectory, the service price, the service period and the service quality of the service vehicle together form the service capability record. The MEC server uses the smart contract feature [45] of blockchain to dynamically price the computing services with the quality of service as

$$price = price^{init} \cdot quality. \quad (3)$$

Step 3: The MEC server adds the service capacity record of the server vehicle to the blockchain. As illustrated in Fig. 3, the MEC server (i.e., blockchain node for data sharing) uses a consensus mechanism combining Proof of Service and PBFT to share the records of the service capacity within a certain time period and ensure the security of the data to prevent malicious attacks. It is worth to noting that we use a limited number of MEC servers instead of vehicles as blockchain nodes to improve the efficiency of consensus to avoid the inefficiency of the consensus mechanism caused by excessive blockchain nodes. The specific processes are as follows.

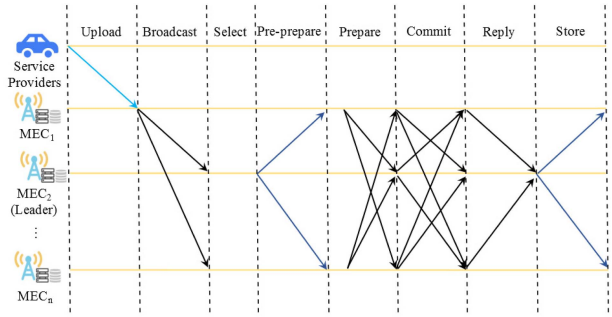

Fig. 3. Consensus mechanism for data sharing between user vehicles and service providers.

1) *Broadcast:* The MEC server (e.g., MEC_m) broadcasts the service capacity information obtained in Step 2 in the blockchain network (i.e., network composed of all edge computing nodes). Then, each node collects the service capacity records sent by other nodes and stores them.
2) *Select:* After a certain time t, a primary node (leader) is selected in the blockchain system to organize the recently generated service capacity records into a block and add this block to the current chain through a consensus mechanism between all MEC servers. Then, the latest block can be shared in the system. Here the leader is selected with the Proof of Service mechanism; that is, based on the current computing capacity of each node, the node with more redundant computing resources is selected as the primary node to generate the block. To avoid the server with strong computing power being the only leader, the p nodes with redundant computing resources are selected as the primary node in turn to generate new blocks. When all p nodes have become proposers, the server with redundant computing resources will be reselected and proceed to the next round.
3) *Pre-prepare:* Each node determines whether it is the primary node. If it is the selected leader, it broadcasts its generated block and its verification results as a pre-prepare message to other MEC servers in the blockchain network.
4) *Prepare:* After receiving the pre-prepare message, all MEC servers verify the authenticity of the leader and the validity of the content in the block. Then, they broadcast the verification results in the network as prepare messages.
5) *Commit:* After receiving the prepare messages sent by other nodes, each MEC server uses the messages along with its own verification result to make a decision and vote on whether the block is generated successfully. If the total number of valid results exceeds $2f$ (f can be interpreted as the maximum number of virulent servers that can be tolerated), the MEC server broadcasts the commit message to all other nodes in the blockchain network to indicate its voting result.
6) *Reply:* After receiving the commit messages sent by other nodes, each MEC server uses the messages along with its own voting result to make a decision. If the number of votes in favor of the generated block exceeds $2f + 1$ (including the vote of the node itself), it is considered

that the blockchain system has reached a consensus on the generation of the block, and the consensus result is sent to the leader.

7) *Store:* The primary node receives the consensus result of each MEC server and sends the latest block to all MEC servers in the system for data sharing and storage.

Step 4: The roadside MEC server sends the service capacity information of the block to the vehicle within its coverage area. After completing data sharing with the blockchain, the roadside MEC server (e.g., MEC_l) parses the latest service capacity record in the block and sends the parsed record to user vehicles in its communication coverage (e.g., vehicle v_i) with encrypted transmission, namely

$$MEC_l \rightarrow v_i : E_{K^{pu}_{Pseu^s_i}} (timestamp||Data_{MEC_l} ||Cert_{MEC_l}||Sign_{MEC_l} (Data_{MEC_l})),$$

where

$$Data_{MEC_l} = E_{K^{pr}_{MEC_l}} (timestamp||record_1 ||record_2|| \cdots ||record_z). \quad (4)$$

After receiving the record sent by MEC_l, user vehicle v_i decrypts the data with its private key and verifies the signature of MEC_l. Then, v_i uses the parsed data of the latest neighboring server vehicles and the data of service chain introduced in Step 6 to verify whether there is a server vehicle that can complete the V2V computation offloading; this help to make subsequent computation offloading decisions.

Step 5: The user vehicle makes an offloading decision based on the service capacity of the MEC server and the neighboring server vehicles. The user vehicle v_i that requires offloading determines the code $code_i$, the input data $data_i$ of the task to be computed, and the deadline $deadline_i$ for the task. If there is no server vehicle that can provide computing resources around v_i, the vehicle sends its task directly to the MEC server. If there is more than one server vehicle around it, v_i selects the server vehicle v_j that is closest to its trajectory and performs the computation offloading decision process proposed in the Section IV. To more clearly describe the secure data sharing mechanism for cooperative computation offloading, we assume that v_i decides to send its task to vehicle v_j as

$$v_i \rightarrow v_j : E_{K^{pu}_{Pseu^s_j}} \left(timestamp||Request_{Pseu^s_i} ||Cert_{Pseu^s_i}||Sign_{Pseu^s_i} \left(Request_{Pseu^s_i}\right)\right),$$

where

$$Request_{Pseu^s_i} = E_{K^{pr}_{Pseu^s_i}} (timestamp||code_i||data_i||deadline_i). \quad (5)$$

Step 6: The server vehicle uploads the information of ongoing service, and the MEC server updates the service chain. Since the computing resources of the server vehicles are significantly less than those of the roadside servers, we consider that each server vehicle can only provide computing services to one user vehicle simultaneously. With the proposed service capability chain, roadside edge servers share the service capability information of server vehicles and provide it to user vehicles, but the dynamically changing service status of offloading cannot be captured in the chain. Therefore, we add a service chain to share the service information of occupied server vehicles so that user vehicles can determine the available server vehicles for offloading decisions. If user vehicle v_i has selected a server vehicle to which to offload the computation task, the server vehicle also generates the current service information and uploads it to the nearby MEC server when it starts to execute the computation task. Similar to the operation in Steps 1–4, the MEC server packages different service information to form a block. Through the consensus mechanism, this service block is added to the service chain parallel to the service capacity chain mentioned above, and the service information of the current service provider is shared through the service chain to assist user vehicles in the computation offloading decision. Since the specific consensus processes of ongoing service information sharing are the same as those of the service capacity sharing, this step only provides the format of the service record as follows:

$$service = (timestamp||serviceID||provider ||requester||duration), \quad (6)$$

where $serviceID$ is the index of the service record, $requester$ is the specific requester of the service, and $duration$ is the estimated duration of the service.

Step 7: The user vehicle evaluates the quality of the service, and the MEC server dynamically prices the service based on the evaluation. After obtaining the execution results from vehicle v_j, user vehicle v_i evaluates its quality of service using a subjective logic framework. Specifically, the user vehicle uses three trust variables (i.e. the belief, distrust, and uncertainty of v_i to v_j) to evaluate the quality of service. Then, the user vehicle sends the evaluation results to the roadside MEC server to complete the entire offloading process of the task. To avoid malicious evaluations, user vehicles need to provide the corresponding basis while uploading evaluations. The edge server verifies data such as computing results, trajectory consistency, and connection stability against suspicious evaluations in the evaluation management. After confirming the evaluations are reasonable, the MEC server updates the corresponding service capacity record of provider v_j and uploads the data when the next block is generated so that the computing service is dynamically priced by the smart contract on the chain. In addition, the edge server maintains the reputation of user vehicles within its service area. If users frequently submit malicious evaluations, their reputation value will be reduced, resulting in a lower weight of their evaluations.

C. Security Analysis

1) Service Information Spoofing of Malicious Vehicles: In cooperative computation offloading, malicious service vehicles may provide false service information to trick user vehicles into offloading data to them and thus obtain the privacy of user vehicles. In the proposed blockchain-based sharing architecture, the MEC server prevents access of unauthorized service vehicles

by verifying the certificates. If the service vehicle does not provide effective service and violates privacy, the user vehicle can give a minimal service rating and thus prevent the service vehicle from being selected as an offloading destination by other user vehicles [46]. Meanwhile, the immutability and traceability of the service information in the blockchain-based system ensure the accurate identification of malicious service vehicles [26].

2) Malicious Evaluation of User Vehicles: In the proposed sharing scheme of service vehicle information, the evaluation of the service quality by user vehicles affects the pricing of the service, so user vehicles have an incentive to give malicious evaluations intentionally. To avoid malicious evaluations, user vehicles need to provide the corresponding evidence while uploading evaluations. The MEC server verifies data such as computing results, trajectory consistency, and connection stability for suspicious evaluations, and updates the service quality after confirming that the evaluations are reasonable. In addition, the MEC servers can maintain the reputation of user vehicles within their service area, and if users frequently submit malicious evaluations, their reputation value will be reduced, resulting in a lower weight of their evaluations.

3) Fake Block Generation and Majority Attack of MEC Servers: In the proposed data sharing scheme, the roadside MEC servers are responsible for generating blocks and sharing them within the region. If a malicious MEC server generates a fake block, the PBFT-based consensus mechanism can identify the error in the block and avoid adding it to the blockchain in case the number of malicious servers does not exceed one-third. Meanwhile, the proposed shared architecture can prevent the majority attack or 51% attack because mining-based consensus is not used [47]. In addition, the cyclic selection of block generators avoids the manipulation of the blockchain system by nodes with high computational power.

4) Data Tampering of MEC Servers: The proposed sharing scheme of service vehicle information uses blockchain to share data among different MEC servers. Each block on the blockchain contains a hash digest of the previous block, so blockchain nodes can only read existing blocks or add new blocks, and cannot modify the contents of blocks on the chain, which ensures data immutability and prevents data tampering [46]–[48].

IV. Cooperative Computation Offloading Using Game Theory

Based on the trusted sharing scheme designed in the previous section, user vehicles can obtain information about the surrounding service vehicles. In this way, each user vehicle can choose to offload its tasks to the roadside MEC server or the surrounding service vehicles for execution. To ensure efficient computation offloading of user vehicles, the main problem in this paper is how to achieve the distributed optimal offloading decision in the blockchain-based vehicular edge computing network. Therefore, we propose a game-based cooperative computation offloading decision method and prove the existence of equilibrium in this section.

A. Cooperative Computation Offloading Model

As mentioned in the last section, when user vehicle v_i matches with a nearby server vehicle according to the trajectory of the server vehicle from the received shared data, it must decide whether to execute its task on the MEC server or the server vehicle based on the utility and overhead of the service, such as the price, task execution latency, and quality of service. To better characterize the utility and cost of cooperative computation offloading, we present specific models of application, communication, and computation. Based on these models, we propose a game model for cooperative computation offloading and design a payoff function for user vehicles.

In cooperative computation offloading, the user vehicle sends the code, input data, and deadline of the task to the MEC server or server vehicle. Therefore, the application model of the user vehicle v_i that requires computation offloading can be expressed as a triple $(L_i, \alpha_i, t_{i,\max})$, where L_i is the input data size of the application to be offloaded, α_i is the computational complexity, and $t_{i,\max}$ is the maximum tolerable execution time. The computational complexity refers to the required central processing unit (CPU) cycles to handle 1-bit data in the application, which is related to the code of that task. $t_{i,\max}$ is the maximum execution time of the offloaded computation that is useful for user vehicle v_i. If the task is completed within $t_{i,\max}$, v_i attains the expected utility; conversely, it is penalized accordingly. A quantitative expression of the expected utility is provided in the next subsection. In practical systems, the values of these parameters vary with different tasks. For example, in the task of point cloud based object detection, L_i is the input data size of LiDAR point clouds per frame, α_i is the complexity of the object detection algorithm (i.e., the number of CPU cycles required to process each bit of data), and $t_{i,\max}$ is the processing interval of the adjacent frames.

For simplicity and without loss of generality, we consider that the communication model for computation offloading is a block flat Rayleigh fading channel with path loss exponent θ and fading coefficient h_i of user vehicle v_i. For all user vehicles, the length of the block is greater than $t_{i,\max}$. Therefore, the uplink and downlink data rates between v_i and the MEC server can be expressed as

$$R_{i,E}^{U} = W_i \log_2 \left(1 + \frac{P_i d_{i,E}^{-\theta} |h_i|^2}{N_0}\right), \tag{7}$$

$$R_{i,E}^{D} = W_i \log_2 \left(1 + \frac{P_E d_{i,E}^{-\theta} |h_i|^2}{N_0}\right), \tag{8}$$

where W_i and $d_{i,E}$ are the bandwidth and distance between the user vehicle and MEC server, respectively, N_0 is the white Gaussian noise power, and P_i and P_E are the transmission power of the user vehicle and MEC server, respectively. It is worth noting that the noise power N_0 is dependent of the bandwidth, but since the bandwidth of V2V and vehicle-to-infrastructure (V2I) communication is fixed (10 MHz in China), we determine the noise power here with a fixed bandwidth [34], [49]. It is worth noting that we consider V2V and V2I communication implemented with LTE-V2X [43], [44], which achieves data

transmission through the multiple access approach of orthogonal frequency-division multiplexing (OFDM). Referring to [11], [34], [50], we use the same bandwidth and channel gain to evaluate the computation offloading delay, which also fits the practical application of LTE-V2X (fixed bandwidth for V2I and V2V communication). In addition, the offloading decision method proposed in this paper evaluates the utility of computation offloading based on the relative delay, which is also applicable to the environment with different bandwidths and channel gains in the uplink and downlink. Similar to [11] and [34], in the downlink transmission we consider roadside units transmitting data with fixed power, which is also consistent with the practical application of LTE-V2X communication. Meanwhile, since the power allocation does not affect the game-based decision methodology, the offloading decision method proposed in this paper is also applicable to the transmission environment with dynamic power.

Similarly, the data transmission rates of the offloading request and returned result between the user vehicle v_i and server vehicle v_j can be expressed as

$$R_{i,V}^{req} = W_i \log_2 \left(1 + \frac{P_i d_{i,V}^{-\theta} |h_i|^2}{N_0}\right), \tag{9}$$

$$R_{i,V}^{res} = W_i \log_2 \left(1 + \frac{P_j d_{i,V}^{-\theta} |h_i|^2}{N_0}\right), \tag{10}$$

where $d_{i,V}$ is the distance between v_i and v_j, and P_j is the transmission power of the server vehicle. As mentioned earlier, the bandwidth and channel gain of the links for offloading requests and returned results in V2V communication are also the same, and the decision method proposed in this paper is applicable to communication environments with different bandwidths and channel gains.

In the computation model, we use the CPU frequency to represent the computational capability of the server vehicle and MEC server. Combining this with the computational complexity and the data size of the task being offloaded, we can obtain the expected time for the task to execute at the node and evaluate the utility of computation offloading with this latency. Therefore, the processing time of the task on the MEC server is

$$\tau_{i,E} = \frac{\alpha_i L_i}{f_E}, \tag{11}$$

where f_E is the CPU frequency of the MEC server. Similarly, with the CPU frequency f_j of server vehicle v_j, the execution time on v_j can be expressed as

$$\tau_{i,V} = \frac{\alpha_i L_i}{f_j}. \tag{12}$$

In this way, the total latency of computation offloading on the MEC server and server vehicle sides can be expressed as

$$\begin{aligned} t_{i,E} &= t_{i,E}^U + \tau_{i,E} + t_{i,E}^D \\ &= \frac{\beta_i^U L_i}{W_i \log_2 \left(1 + \frac{P_i d_{i,E}^{-\theta} |h_i|^2}{N_0}\right)} + \frac{\alpha_i L_i}{f_E} \\ &\quad + \frac{\beta_i^D L_i}{W_i \log_2 \left(1 + \frac{P_E d_{i,E}^{-\theta} |h_i|^2}{N_0}\right)}, \end{aligned} \tag{13}$$

$$\begin{aligned} t_{i,V} &= t_{i,V}^{req} + \tau_{i,V} + t_{i,V}^{res} \\ &= \frac{\beta_i^{req} L_i}{W_i \log_2 \left(1 + \frac{P_i d_{i,V}^{-\theta} |h_i|^2}{N_0}\right)} + \frac{\alpha_i L_i}{f_j} \\ &\quad + \frac{\beta_i^{res} L_i}{W_i \log_2 \left(1 + \frac{P_j d_{i,V}^{-\theta} |h_i|^2}{N_0}\right)}, \end{aligned} \tag{14}$$

where β_i^U and β_i^{req} are the uplink-cost in vehicle-to-MEC (V2M) offloading and request-cost in V2V offloading, respectively, β_i^D is the joint factor for the downlink-cost and the output-input data ratio on in V2M offloading, and β_i^{res} is the joint factor for result return-cost and the output-input data ratio in V2V offloading.

Based on these offloading latencies, the user vehicle can quantify the utility of both V2M and V2V offloading, and decide whether to migrate the task to the MEC server or server vehicle. Next, we use the latencies to construct a game of cooperative computation offloading to guide the decision of user vehicles.

B. Cooperative Computation Offloading Game

We obtain the latencies incurred by offloading tasks from the user vehicle to different destinations with the application, communication, and computation models of cooperative computation offloading. Combining these latencies with the maximum execution time of the tasks, the utility of offloading computation to different destinations can be evaluated. In this section, we establish the interaction of user vehicles using game theory in cooperative computation offloading, and derive the utility and overhead of offloading to construct a distributed solution with the goal of optimizing the latencies of all user vehicles.

The game model of cooperative computation offloading can be expressed as $\mathcal{G} = \{\mathcal{N}, (p_i)_{i\in\mathcal{N}}, (u_i)_{i\in\mathcal{N}}\}$ [5], where $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the set of all user vehicles in the coverage area of MEC servers, and $p_i \in [0, 1]$ is the mixed offloading strategy of user vehicle v_i that combines the two pure strategies of V2M and V2V offloading. If p_i is close to 1, v_i migrates the task to the MEC server with a high probability. u_i is the payoff that v_i can obtain from this game with the payoff function $u_i(\mathbf{p})$. The payoff function consists of both the utility and the cost of computation offloading, thus evaluating the benefit and overhead of strategy p_i in cooperative computation offloading. In this game model, we derive a specific payoff function based on the latency of computation offloading and the competition of multiple user vehicles for the MEC server.

First, we use the latency of task execution to characterize the utility of the user vehicle in cooperative computation offloading. To unify the wide range of fluctuations in latency, we design a value function to map the task execution time to a fixed value interval. Intuitively, the lower the latency of the

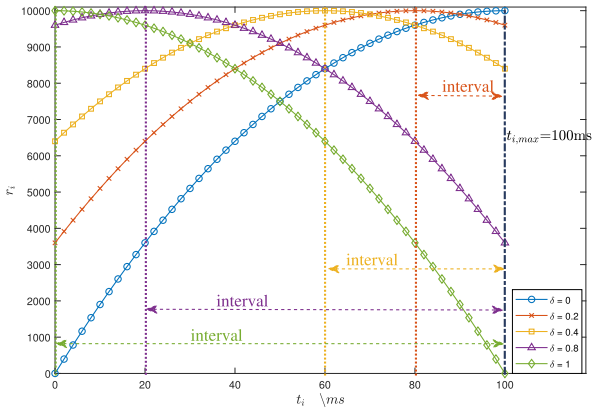

Fig. 4. Value function for different value factors.

task execution, the higher the utility to the user vehicle, but in practice, the user vehicle can benefit from the successful execution of the task as long as the latency of the task does not exceed its maximum tolerable execution time. Moreover, the blind pursuit of minimizing the execution latency can also make user vehicles compete fiercely for edge server resources, leading to overtime of some tasks with the long waiting time. Therefore, considering the reciprocity among user vehicles, we construct this value function in the form of a quadratic function and use a parameter to dynamically adjust the value versus latency curve. The specific value function can be expressed as

$$r_i(t_i) = 2t_{i,\max}\left(t_i + \delta t_{i,\max}\right) - \left(t_i + \delta t_{i,\max}\right)^2, \qquad (15)$$

where $t_i \in \{t_{i,E}, t_{i,V}\}$ is the total latency of V2M offloading in (13) or V2V offloading in (14), and $\delta \in [0,1]$ is defined as the value factor to regulate the latency of the maximum value (i.e., the distance between the latency of the apex of the curve and $t_{i,\max}$). As illustrated in Fig. 4, when δ is large, the task reaches its maximum value at a lower latency, and, conversely, the maximum value corresponds to a longer task execution time. If the task takes longer than $t_{i,\max}$ to execute, the vehicle will not obtain any value from computation offloading.

With the value function, we can use the latencies to express the expected utility of different computation offloading strategies as

$$U_i(\mathbf{p}) = p_i \frac{r_i(t_{i,E})}{r_{i,\max}} + (1-p_i)\frac{q_j r_i(t_{i,V})}{r_{i,\max}}, \qquad (16)$$

where $r_{i,\max} = r_i((1-\delta)t_{i,\max})$ is the maximum value of the task, and q_j is the quality of service of v_j evaluated by other user vehicles. The expected utility is composed of the utility resulting from V2M and V2V offloading together. The utility of each computation offloading is expressed as the ratio of the value corresponding to its execution latency to the maximum value, and the quality of service is also considered in the utility of V2V offloading. If the quality of service of the corresponding server vehicle is low, the utility obtained from computation offloading will be reduced.

Due to the limited MEC servers resources, multiple user vehicles in V2M computation offloading will compete for limited resources. Thus, the expected cost of cooperative computation offloading can be computed by considering the competition for V2M offloading and the price of V2V offloading, which can be expressed as

$$C_i(\mathbf{p}) = p_i^2\left[1 - \prod_{k\neq i}(1-\lambda_k p_k)\right] + (1-p_i)\rho_j, \qquad (17)$$

where λ_k denotes the average arrival rate of task in user vehicle v_k, and $\rho_j = \dfrac{price_j}{price_E}$ denotes the ratio of the price of V2V offloading to that of V2M offloading. In (17), as the number of user vehicles and the price increase, the cost of computation offloading also increases; thus, vehicles must consider both utility and overhead to make better decisions.

Finally, by integrating (16) and (17), the payoff function for cooperative computation offloading is given by

$$\begin{aligned} u_i(\mathbf{p}) &= U_i(\mathbf{p}) - C_i(\mathbf{p}) \\ &= p_i \frac{r_i(t_{i,E})}{r_{i,\max}} + (1-p_i)\frac{q_j r_i(t_{i,V})}{r_{i,\max}} \\ &\quad - p_i^2\left[1 - \prod_{k\neq i}(1-\lambda_k p_k)\right] + (1-p_i)\rho_j. \end{aligned} \qquad (18)$$

In this way, user vehicles can fully account for the effects of latency, competition, and price in the cooperative computation offloading game and adjust their computation offloading probabilities (i.e., mixed strategies) to obtain a higher payoff. We consider the current strategy to be a Nash equilibrium $\mathbf{p}^*$ if no user vehicle can achieve more payoff by changing its current strategy, that is, for all p_i

$$u_i(p_i^*, \mathbf{p}_{-i}^*) \geq u_i(p_i, \mathbf{p}_{-i}^*), \qquad (19)$$

where $\mathbf{p}_{-i} = (p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_N)$ is the vector of mixed strategies of all user vehicles except vehicle v_i [51].

Next, we construct a distributed solution for user vehicles based on the cooperative computation offloading game and guarantee the uniqueness of the converged Nash equilibrium from the strategies of all user vehicles.

C. Distributed Computation Offloading Solution

With blockchain-based data sharing of resource-rich vehicles, we propose a cooperative offloading game to describe the influence of the capacity of MEC servers and server vehicles on the computation offloading decision of user vehicle v_i. Following [52], there is definitely a Nash equilibrium of mixed strategies in the cooperative computation offloading game proposed in this paper; thus, we must construct an efficient algorithm to guarantee the convergence of the Nash equilibrium from the strategies of user vehicles. For the static game proposed in this paper, we use the best-response mechanism to perform the offloading decision of user vehicles and construct a distributed cooperative computation offloading algorithm for them. To ensure the convergence and uniqueness of the user vehicle strategies, we also prove that this distributed computation offloading algorithm allows the strategy of each vehicle to converge to the unique Nash equilibrium. Since the user vehicles in the proposed

Algorithm 1: Distributed Cooperative Computation Offloading Algorithm For User Vehicles.

Initialization
Initialize the offloading requirements $(\boldsymbol{L}, \boldsymbol{\alpha}, \boldsymbol{t}_{i,\max})$ of tasks of all user vehicles.
Initialize the vector $\mathbf{p}$, $\boldsymbol{\rho}$, and $\boldsymbol{\lambda}$ of all user vehicles.
for all $i \in \{1, \ldots, N\}$ **do**
  Parse the information of server vehicles.
  if no server vehicles are around v_i **then**
    v_i takes the roadside MEC as the offloading destination and sends its task.
  else
    if the number of server vehicles > 1 **then**
      Select the server vehicle v_j that is closest to its trajectory.
    end if
    Estimate the data transmission rate with the roadside MEC server and server vehicle v_j based on (7)–(10).
    Estimate the latencies when the task is offloaded to the MEC server and v_j with (13) and (14), respectively.
    Generate a random number $rand_i \in [0, 1]$ and obtain the offloading probability p_i with the best-response updated in (20).
    if $rand_i > p_i$ **then**
      v_i offloads its task to server vehicle v_j for execution.
      Upload the service evaluation to the MEC server after the offloading task is completed.
    else
      v_i offloads its task to the roadside MEC for execution.
    end if
  end if
end for

game cannot obtain the strategies of other vehicles in real time, we express the best response of user vehicle v_i based on the previous strategies of other vehicles as

$$p_i = \arg\max_{p_i} u_i(\mathbf{p}) = \left[\frac{r_i\left(t_{i,E}\right)/r_{i,\max} - q_j r_i\left(t_{i,V}\right)/r_{i,\max} + \rho_j}{2\left(1 - \prod_{k\neq i}\left(1-\lambda_k p_k\right)\right)} \right]_0^1, \quad (20)$$

where the operator $[\cdot]_0^1$ limits the result to [0,1]. Based on this best-response mechanism in the cooperative computation offloading game, we can construct the distributed cooperative computation offloading solution as given in Algorithm 1.

In Algorithm 1, each user vehicle v_i makes the computation offloading decision in parallel, so its computational complexity is mainly reflected in parsing and screening the information of server vehicles. If each user vehicle can obtain information of M server vehicles, the computational complexity of data parsing for v_i is $O(M)$. In addition, for each user vehicle, the computational complexity of both latency estimation and strategy update is $O(1)$. Thus, for each task, the computational complexity of Algorithm 1 is $O(M)$. Considering that each user vehicle needs to perform K independent tasks, the total complexity of its computation offloading is $O(KM)$.

In this way, within the framework of game theory, user vehicle v_i determines the utility and cost that can be obtained from its mixed strategy and then calculates the payoff function of the game. Based on this payoff function, v_i adopts the strategy p_i using the best-response method to obtain the maximum payoff. It can be proved using the contraction mapping theorem [53] that p_i uniquely converges to the Nash equilibrium p_i^* under certain conditions.

Theorem 1: Starting from any initial strategy, the best-response mechanism in the distributed cooperative computation offloading algorithm converges to the unique equilibrium p_i^* if $\forall i \in \{1, \ldots, N\}$,

$$\left| \frac{r_i(t_{i,E})}{r_{i,\max}} - \frac{q_j r_i(t_{i,V})}{r_{i,\max}} + \rho_j \right| < \frac{2\left[1 - \prod_{k'\neq i}\left(1-\lambda_{k'}p_{k'}\right)\right]^2}{\sum_{k\neq i}\lambda_k \prod_{l\neq i,k}(1-\lambda_l p_l)}. \quad (21)$$

Proof: Following [14], [54], the convergence and uniqueness of the Nash equilibrium in best-response updating can be guaranteed by proving that (20) is a contraction mapping. In addition, according to the contraction mapping theorem [53], we only need to prove that a certain norm (here, we use $\|\cdot\|_\infty$) of the Jacobian matrix of (20) is less than 1.

Therefore, we first derive the norm of best-response updating. The Jacobian matrix $\mathcal{J}$ of best-response updating in (20) is defined as

$$J_{i,k} = \frac{\partial p_i^s}{\partial p_k^{s-1}}, \quad (22)$$

where p_k^{s-1} denotes the strategy adopted in the previous stage for each user vehicle except v_i. Therefore, the Jacobian matrix can be derived as

$$J_{i,k} = \begin{cases} 0, & i = k \\ \dfrac{-\lambda_k\left(r_{i,E} - r_{i,V} + \rho_j\right)\prod_{l\neq i,k}(1-\lambda_l p_l)}{2\left[1 - \prod_{k'\neq i}\left(1-\lambda'_k p'_k\right)\right]^2}, & i \neq k \end{cases}, \quad (23)$$

where $r_{i,E} = \frac{r_i(t_{i,E})}{r_{i,\max}}$ and $r_{i,V} = \frac{q_j r_i(t_{i,V})}{r_{i,\max}}$ are defined to simplify the expression of $J_{i,k}$. Therefore,

$$\|\mathcal{J}\|_\infty = \max_{i\in\mathcal{N}} \frac{\sum_{k\neq i}\left|r_{i,E} - r_{i,V} + \rho_j\right|\lambda_k \prod_{l\neq i,k}(1-\lambda_l p_l)}{2\left[1 - \prod_{k'\neq i}\left(1-\lambda_{k'}p_{k'}\right)\right]^2}. \quad (24)$$

According to the contraction mapping theorem, if $\|\mathcal{J}\|_\infty < 1$, then the strategy updating in (20) can converge to the unique Nash equilibrium. Therefore, if $\forall i \in \mathcal{N}$, $\frac{\sum_{k\neq i}\left|r_{i,E} - r_{i,V} + \rho_j\right|\lambda_k \prod_{l\neq i,k}(1-\lambda_l p_l)}{2[1 - \prod_{k'\neq i}(1-\lambda_{k'}p_{k'})]^2} < 1$, that

is, $|\frac{r_i(t_{i,E})}{r_{i,\max}} - \frac{q_j r_i(t_{i,V})}{r_{i,\max}} + \rho_j| < \frac{2[1-\prod_{k'\neq i}(1-\lambda_{k'}p_{k'})]^2}{\sum_{k\neq i}\lambda_k\prod_{l\neq i,k}(1-\lambda_l p_l)}$, then the best-response strategy in the distributed cooperative computation offloading algorithm can converge to the unique Nash equilibrium. ■

The distributed computation offloading method thus achieves strategy equilibrium to help user vehicles make better offloading decisions. In this way, user vehicle v_i obtains a computation offloading strategy p_i^* that can achieve Nash equilibrium in most cases. Based on p_i^*, v_i can decide whether to transfer the onboard application task to the roadside MEC server or neighboring server vehicles to improve the execution efficiency of the application.

V. Numerical Results and Discussion

In this section, we present a series of experiments run in MATLAB to evaluate the performance of the method proposed in this paper. Specifically, we demonstrate the convergence of Algorithm 1 and estimate the effect of different parameters on the offloading strategy and expected time (latency) consumed to complete the task of user vehicles. In addition, we compare the expected latency and payoff of cooperative computation offloading in different offloading schemes and demonstrate the performance of the proposed offloading method.

For simplicity and without loss of generality, we assume that all user vehicles do not have sufficient local resources to execute their tasks and need to offload these tasks to MEC servers or resource-rich vehicles for execution. These tasks share the same application model (L, α, t_{max}) and other parameters for user vehicle v_i, such as $L_i = L$, $\alpha_i = \alpha$, $t_{i,\max} = t_{max}$, $P_i = P_E = P_j = P$, $\lambda_i = \lambda$, and $\rho_j = \rho$. Furthermore, we assume that each user vehicle in the scenario is near a resource-rich vehicle that can provide computing services to meet the requirements of cooperative computation offloading. The main parameters in the experiments are listed in Table I.

The convergence of the offloading probability (i.e., strategy) for six user vehicles is presented in Fig. 5. We can see that the strategies of the user vehicles quickly converge to a stable value, which demonstrates the convergence and uniqueness of the Nash equilibrium under the condition of Theorem 1, that is, $\forall i \in \mathcal{N}$, $|\frac{r_i(t_{i,E})}{r_{i,\max}} - \frac{q_j r_i(t_{i,V})}{r_{i,\max}} + \rho_j| < \frac{2[1-\prod_{k'\neq i}(1-\lambda_{k'}p_{k'})]^2}{\sum_{k\neq i}\lambda_k\prod_{l\neq i,k}(1-\lambda_l p_l)}$. In addition, due to the different channel environments and communication distances around each user vehicle, each strategy converge to a different equilibrium.

Fig. 6 displays the variation of the average probability of cooperative computation offloading with the number of vehicles for all user vehicles. Initially, as the number of user vehicles grows, the offloading probabilities decrease sharply. Once the number of vehicles is greater than 20, the average offloading probabilities saturate and remain low; that is, user vehicles prefer to offload tasks to neighboring server vehicles rather than to the MEC server. This is because overloaded user vehicles intensify the competition for the MEC server resources between user vehicles, and vehicles find it difficult to achieve higher utility from V2M offloading; therefore, they turn to V2V offloading without competition for resources. In addition, different task arrival rates λ have no effect on the saturation value of the average offloading probability for the same price ratios. This is because an increased number of user vehicles saturates the MEC server utilization, and a lower λ only slows down the process of resource saturation without affecting the saturation strategy.

Figs. 7 and 8 display the variation in the average offloading probability and the expected latency of 10 user vehicles with the value factor δ. As expressed in (15) and Fig. 4, δ can adjust the mapping of the latency to the maximum value. If

TABLE I
Experimental Parameters [7], [14], [34], [55]

Notation	Definition	Value and unit
L	Input data size of the application	1 Mbits
α	Computational complexity of the application	240 cycles/bit
θ	Path loss exponent	2
f_j	Frequency of the CPU in server vehicle	1 GHz
f_E	Frequency of the CPU in MEC server	5 GHz
W_i	Bandwidth of the communication channel	10 MHz
β_i^U	Uplink-overhead in V2M offloading	1
β_i^D	Joint factor for downlink-cost and the output-input data ratio in V2M offloading	0.05
β_i^{req}	Request-overhead in V2V offloading	1
$\beta_i res$	Joint factor for result return-cost and the output-input data ratio in V2V offloading	0.05
δ	Value factor	0.7
λ	Average arrival rate of task	0.7
P	Transmit power	0.2 W
ρ	Ratio of the price of V2V offloading to that of V2M offloading	0.7

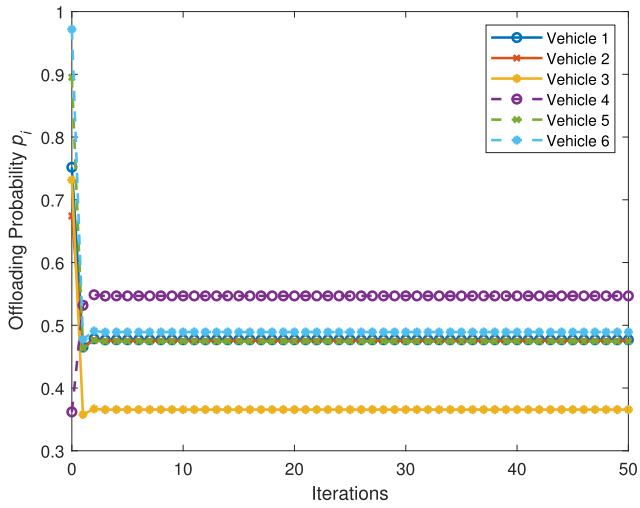

Fig. 5. Convergence of computation offloading probability for user vehicles.

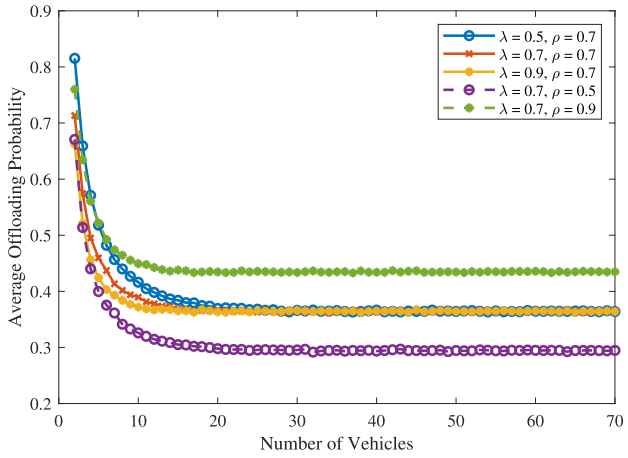

Fig. 6. Comparison of the average probability of cooperative computation offloading for different λ and ρ.

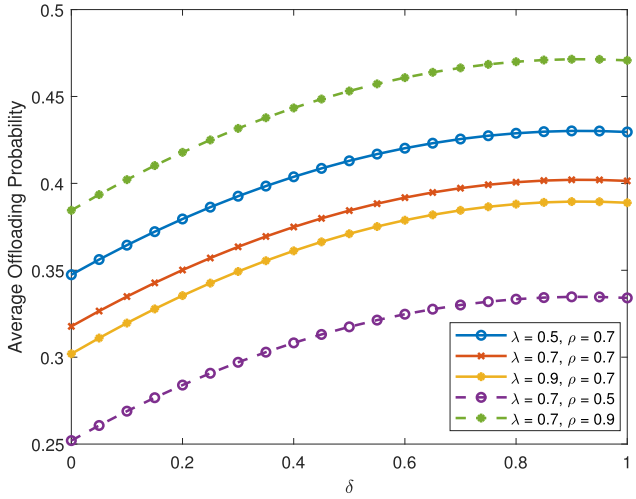

Fig. 7. Average offloading probability versus value factor δ for different λ and ρ.

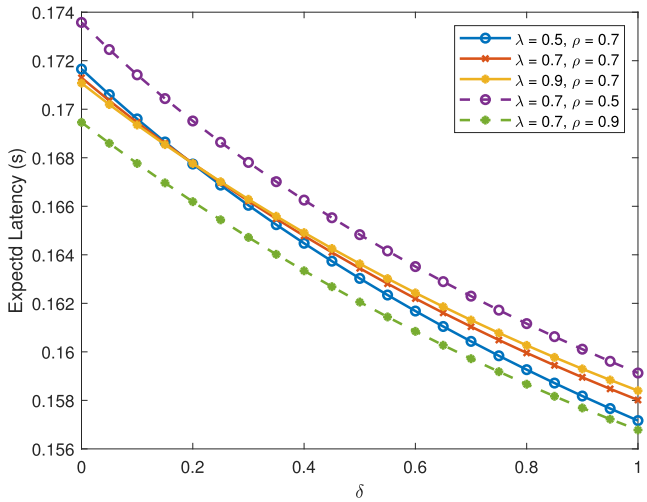

Fig. 8. Expected latency versus value factor δ for different λ and ρ.

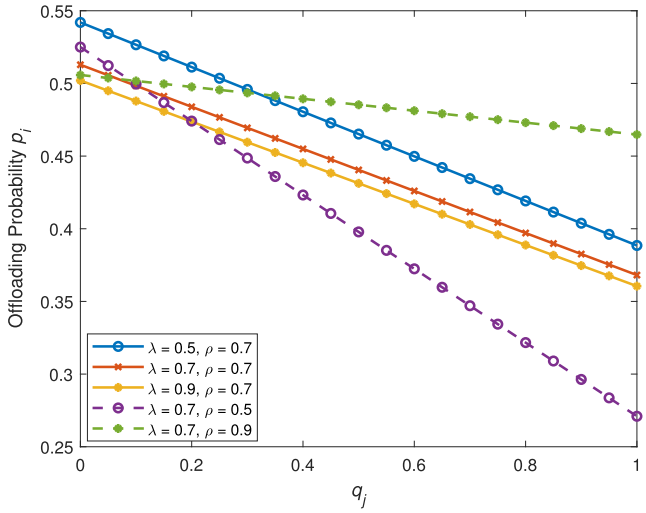

Fig. 9. Offloading probability versus quality of service q_j of the server vehicle for different λ and ρ.

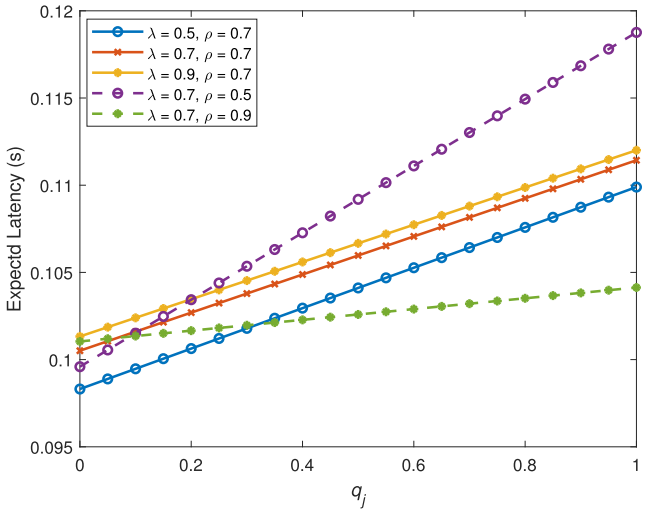

Fig. 10. Expected latency versus quality of service q_j of the server vehicle for different λ and ρ.

δ is large, the user vehicle is more inclined to minimize its own latency to improve utility. In Fig. 7, as δ alters the value curve, the average offloading probability gradually increases, thereby reducing the task execution time through V2M offloading. As demonstrated in Fig. 8, the increase in offloading probability leads to a decreasing trend in the expected latency, and each user vehicle can thus achieve higher utility in the game.

Figs. 9 and 10 depict the change in the offloading probability and expected latency of a single user vehicle with the quality of service q_j of its server vehicle while the quality of service of other vehicles remains constant. In Fig. 9, the offloading probability of the user vehicle gradually decreases as q_j increases, indicating its preference to perform tasks through V2V offloading. The change in offloading probability is more significant at a lower price ratio ρ, which indicates that low ρ values are more sensitive to changes in q_j because the user

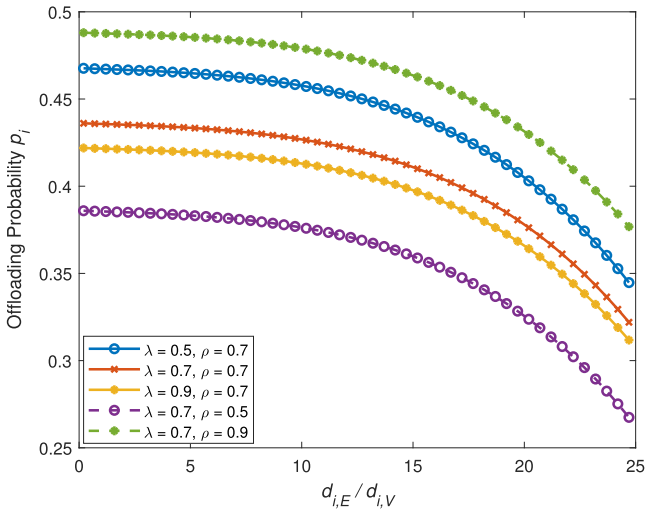

Fig. 11. Influence of ratio of $d_{i,E}$ to $d_{i,V}$ on offloading probability for different λ and ρ.

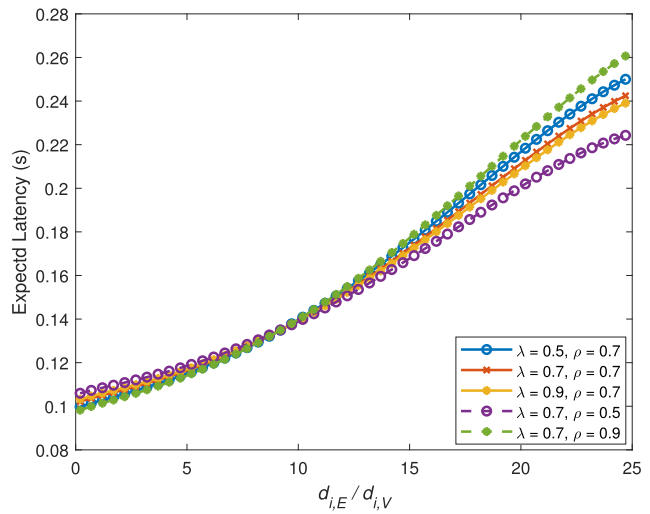

Fig. 12. Influence of ratio of $d_{i,E}$ to $d_{i,V}$ on expected latency for different λ and ρ.

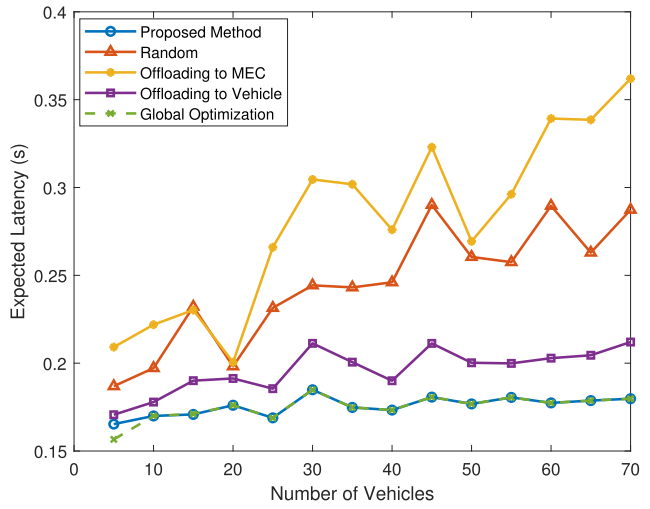

Fig. 13. Comparison of different solutions in terms of expected latency.

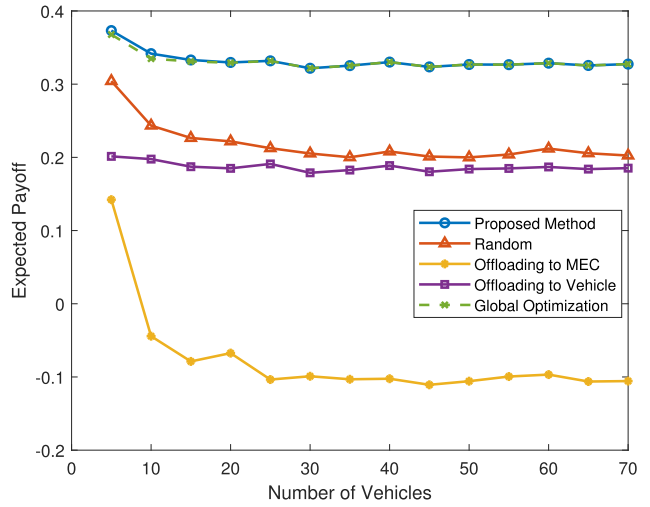

Fig. 14. Comparison of different solutions in terms of expected payoff.

vehicle can achieve higher utility in this case. Similarly, a reduced offloading probability causes a slight increase in the expected latency in Fig. 10, and the increase is more significant at lower ρ values. This is because the increase in latency is accompanied by a decrease in computation offloading overhead, and the user vehicle can balance its payoff by sacrificing some of its latency to reduce its V2M offloading cost as long as the task is completed on time.

Figs. 11 and 12 display the variation in the offloading probability and expected latency of a single user vehicle with a ratio of $d_{i,E}$ to $d_{i,V}$ while the communication distance of other vehicles remains constant. In this experiment, we set the distance $d_{i,V}$ between user vehicle v_i and server vehicle v_j to 10 m and adjust the distance $d_{i,E}$ between v_i and the MEC server to change the ratio of $d_{i,E}$ to $d_{i,V}$. As illustrated in Fig. 11, an increase in the distance of V2M offloading prolongs the transmission time of the application data and thus reduces the offloading probability of the user vehicle. As the increase in distance leads to a decrease in the quality of V2M communication, the expected delay in Fig. 12 gradually increases. In addition, as the distance of V2M communication increases, the user vehicle is more willing to execute tasks via V2V offloading, which reverses the effect of λ and ρ on the expected latency.

Fig. 13 depicts the expected latency of user vehicles for different numbers of vehicles using different offloading solutions. In this experiment, the completion deadline t_{max} is set to 1 s. We can observe that the expected delay of the proposed method is stable below the completion deadline and is significantly lower than that of other offloading schemes. It can be seen that the expected latency of the proposed game method is the same as the result of the global optimization solved with global information for large numbers of vehicles. However, due to changes in path loss and channel fading caused by different communication distances, the expected latency of the three non-game solutions is difficult to maintain at a stable level. Therefore, the proposed

game method can promote efficient computation offloading and ensure the real-time execution of tasks.

The expected payoffs of different offloading solutions are compared in Fig. 14. We can see that the game method proposed in this paper achieves almost the same maximum payoff as the global optimization method, whereas the other solutions fail to maintain the obtained payoff at a high level. This is because the proposed method efficiently obtains the server vehicle information using blockchain-based data sharing and evaluates the competition of multiple vehicles for MEC server resources using a distributed approach. In Fig. 14, the proposed distributed algorithm achieves an equilibrium close to the global optimum, demonstrating its performance advantage.

VI. Conclusion

In this paper, we proposed a data sharing architecture between user vehicles and service providers for cooperative computation offloading in blockchain-based vehicular MEC networks. A consensus mechanism used in blockchain, which combines Proof of Service and PBFT, is designed in this architecture to improve the security and efficiency of data sharing. Furthermore, we proposed a game of cooperative computation offloading and constructed a method to obtain offloading strategy equilibrium using the best-response mechanism. The performance evaluation reveals the advantage of the proposed method in terms of latency. Using the computation offloading strategy, a user vehicle can determine whether to transfer its task to the roadside MEC server or a nearby server vehicle. The proposed method can thus help increase the security and efficiency of cooperative computation offloading in blockchain-based vehicular edge computing networks. In future work, we will also investigate cooperative computation offloading problem in presence of multiple MEC servers and server vehicles in a region to provide a more accurate decision method in vehicular MEC networks.

References

[1] S. Santini, A. Salvi, A. S. Valente, A. Pescapé, M. Segata, and R. L. Cigno, "Platooning maneuvers in vehicular networks: A distributed and consensus-based approach," *IEEE Trans. Intell. Veh.*, vol. 4, no. 1, pp. 59–72, Mar. 2019.

[2] J. A. Guerrero-Ibanez, S. Zeadally, and J. Contreras-Castillo, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and Internet of Things technologies," *IEEE Wireless Commun.*, vol. 22, no. 6, pp. 122–128, Dec. 2015.

[3] S. A. Fayazi and A. Vahidi, "Mixed-integer linear programming for optimal scheduling of autonomous vehicle intersection crossing," *IEEE Trans. Intell. Veh.*, vol. 3, no. 3, pp. 287–299, Sep. 2018.

[4] L. Cheng et al., "SCTSC: A semicentralized traffic signal control mode with attribute-based blockchain in IOVs," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 6, pp. 1373–1385, Dec. 2019.

[5] H. Zhang, B. Xin, L.-H. Dou, J. Chen, and K. Hirota, "A review of cooperative path planning of an unmanned aerial vehicle group," *Front. Inf. Technol. Electron. Eng.*, vol. 21, no. 12, pp. 1671–1694, 2020.

[6] C. Lin, D. Tian, X. Duan, and J. Zhou, "3D environmental perception modeling in the simulated autonomous-driving systems," *Complex Syst. Model. Simul.*, vol. 1, no. 1, pp. 45–54, 2021.

[7] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62624–62632, 2019.

[8] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, Jul.–Sep. 2017.

[9] W. Yang et al., "Edgekeeper: A trusted edge computing framework for ubiquitous power Internet of Things," *Front. Inf. Technol. Electron. Eng.*, vol. 22, no. 3, pp. 374–399, 2021.

[10] P.-Q. Huang, Y. Wang, and K.-Z. Wang, "Energy-efficient trajectory planning for a multi-UAV-assisted mobile edge computing system," *Front. Inf. Technol. Electron. Eng.*, vol. 21, no. 12, pp. 1713–1725, 2020.

[11] S. Wang, J. Li, G. Wu, H. Chen, and S. Sun, "Joint optimization of task offloading and resource allocation based on differential privacy in vehicular edge computing," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 1, pp. 109–119, Feb. 2022.

[12] H. Yu, C. Chang, S. Li, and L. Li, "CD-DB: A data storage model for cooperative driving," *IEEE Trans. Intell. Veh.*, to be published, doi: 10.1109/TIV.2022.3150509.

[13] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 2041–2057, Feb. 2022.

[14] Y. Wang et al., "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[15] 3GPP, "Study on network controlled interactive service (NCIS) in the 5G system (5GS)," *3rd Gener. Partnership Project (3GPP)*, Tech. Rep. (TR) 22.842, Dec. 2019.

[16] 3GPP, "Service requirements for the 5G system," 3rd Generation Partnership Project (3GPP), *Tech. Specification (TS) 22.261*, Jul. 2020.

[17] 5GAA, "Toward fully connected vehicles: Edge computing for advanced automotive communications," 5G Automotive Association (5GAA), *White Paper*, Dec. 2017. [Online]. Available: https://5gaa.org/news/toward-fully-connected-vehicles-edge-computing-for-advanced-automotive-communications/

[18] R. Chattopadhyay and C.-K. Tham, "Joint sensing and processing resource allocation in vehicular Ad-Hoc networks," *IEEE Trans. Intell. Veh.*, to be published, doi: 10.1109/TIV.2021.3124208.

[19] 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.

[20] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (v2x) communication platforms," *IEEE Trans. Intell. Veh.*, vol. 5, no. 4, pp. 693–713, Dec. 2020.

[21] T. Aste, P. Tasca, and T. Di Matteo, "Blockchain technologies: The foreseeable impact on society and industry," *Computer*, vol. 50, no. 9, pp. 18–28, 2017.

[22] R. A. Memon, J. P. Li, J. Ahmed, M. I. Nazeer, M. Ismail, and K. Ali, "Cloud-based vs. blockchain-based IoT: A comparative survey and way forward," *Front. Inf. Technol. Electron. Eng.*, vol. 21, no. 4, pp. 563–586, 2020.

[23] X. Huang, D. Ye, R. Yu, and L. Shu, "Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 2, pp. 426–441, Mar. 2020.

[24] 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, Apr.–Jun. 2019.

[25] D. Xu, W. Shi, W. Zhai, and Z. Tian, "Multi-candidate voting model based on blockchain," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 12, pp. 1891–1900, Dec. 2021.

[26] J. Kang et al., "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.

[27] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 6, pp. 1407–1419, Dec. 2019.

[28] S. K. Dwivedi, R. Amin, and S. Vollala, "Blockchain-based secured IPFS-enable event storage technique with authentication protocol in VANET," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 12, pp. 1913–1922, Dec. 2021.

[29] 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, 2017.

[30] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.
[31] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[32] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.
[33] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
[34] Y. Sun et al., "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.
[35] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14198–14211, Dec. 2020.
[36] D. Zhang, F. R. Yu, and R. Yang, "Blockchain-based distributed software-defined vehicular networks: A dueling deep *Q*-learning approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1086–1100, Dec. 2019.
[37] Y. Fu, F. R. Yu, C. Li, T. H. Luan, and Y. Zhang, "Vehicular blockchain-based collective learning for connected and autonomous vehicles," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 197–203, Apr. 2020.
[38] C. Li, Y. Fu, F. R. Yu, T. H. Luan, and Y. Zhang, "Vehicle position correction: A vehicular blockchain networks-based GPS error sharing framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 898–912, Feb. 2021.
[39] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.
[40] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.
[41] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Jul. 2020.
[42] X. Zheng, M. Li, Y. Chen, J. Guo, M. Alam, and W. Hu, "Blockchain-based secure computation offloading in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4073–4087, Jul. 2021.
[43] S. Chen, J. Hu, Y. Shi, and L. Zhao, "LTE-V: A TD-LTE-based V2X solution for future vehicular network," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 997–1005, Dec. 2016.
[44] S. Chen, J. Hu, Y. Shi, L. Zhao, and W. Li, "A vision of C-V2X: Technologies, field testing, and challenges with chinese development," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3872–3881, May 2020.
[45] S. Dustdar, P. Fernández, J. M. García, and A. Ruiz-Cortés, "Elastic smart contracts in blockchains," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 12, pp. 1901–1912, Dec. 2021.
[46] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1495–1505, Apr. 2019.
[47] W. Li, M. Nejad, and R. Zhang, "A blockchain-based architecture for traffic signal control systems," in *Proc. IEEE Int. Congress Internet Things*, 2019, pp. 33–40.
[48] 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. 14, no. 8, pp. 3690–3700, Aug. 2018.
[49] A. Bansal, N. Agrawal, and K. Singh, "Rate-splitting multiple access for UAV-based RIS-enabled interference-limited vehicular communication system," *IEEE Trans. Intell. Veh.*, to be published, doi: 10.1109/TIV.2022.3168159.
[50] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[51] S. Tadelis, "Game theory : An introduction," *Econ. Books*, vol. 1, 2012.
[52] D. Fudenberg and J. Tirole, *Game Theory*. Cambridge, MA,USA: MIT Press, 1991.
[53] R. Abraham, J. E. Marsden, and T. S. Ratiu, *Manifolds, Tensor Analysis, and Applications*. Berlin, Germany: Springer, 1988.
[54] J.-W. Lee, A. Tang, J. Huang, M. Chiang, and A. R. Calderbank, "Reverse-engineering MAC: A non-cooperative game model," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 6, pp. 1135–1147, Aug. 2007.
[55] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

Ping Lang received the B.Eng. and M.Sc. degrees in computer science from Jilin University, Changchun, China, in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree with the School of Transportation Science and Engineering, Beihang University, Beijing, China.

His research interests include vehicular networks, edge computing, and intelligent transportation systems.

Daxin Tian (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Jilin University, Changchun, China, in 2002, 2005, and 2007, respectively.

He is currently a Professor with the School of Transportation Science and Engineering, Beihang University, Beijing, China. His research interests include mobile computing, intelligent transportation systems, vehicular ad hoc networks, and swarm intelligent. He is an IEEE Intelligent Transportation Systems Society Member and an IEEE Vehicular Technology Society Member.

Xuting Duan received the Ph.D. degree in traffic information engineering and control from Beihang University, Beijing, China.

He is currently an Assistant Professor with the School of Transportation Science and Engineering, Beihang University. His research interests include vehicular ad hoc networks, cooperative vehicle infrastructure system, and Internet of Vehicles.

Jianshan Zhou received the B.Sc., M.Sc., and Ph.D. degrees in traffic information engineering and control from Beihang University, Beijing, China, in 2013, 2016, and 2020, respectively.

From 2017 to 2018, he was a Visiting Research Fellow with the School of Informatics and Engineering, University of Sussex, Brighton, U.K. He has authored or coauthored more than 20 international scientific publications. His research interests include the modeling and optimization of vehicular communication networks and air–ground cooperative networks, the analysis and control of connected autonomous vehicles, and intelligent transportation systems. He is currently a Postdoctoral Research Fellow supported by the Zhuoyue Program of Beihang University and the National Postdoctoral Program for Innovative Talents, and is or was the Technical Program Session Chair with the IEEE EDGE 2020, the TPC member with the IEEE VTC2021-Fall track, the session organizer with ICAUS 2022, and IEEE ICUS 2022, and the Youth Editorial Board Member of the Unmanned Systems Technology.

Zhengguo Sheng (Senior Member, IEEE) received the B.Sc. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006, and the M.S. and Ph.D. degrees from Imperial College London, London, U.K., in 2007 and 2011, respectively.

He is currently a Reader with the University of Sussex, Brighton, U.K. He was with UBC, Vancouver, BC, Canada, as a Research Associate and with Orange Labs as a Senior Researcher. He has authored or coauthored more than 120 publications. His research interests include IoT, vehicular communications, and cloud/edge computing.

Victor C. M. Leung (Life Fellow, IEEE) is a Distinguished Professor of computer science and software engineering with Shenzhen University, Shenzhen, China. He is also an Emeritus Professor of electrical and computer engineering and Director of the Laboratory for Wireless Networks and Mobile Systems with the University of British Columbia (UBC), Vancouver, BC, Canada. His research focuses on wireless networks and mobile systems, and he has published widely in these areas.

Dr. Leung is serving on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, IEEE ACCESS, IEEE NETWORK, and several other journals. He was the recipient of the 1977 APEBC Gold Medal, 1977–1981 NSERC Postgraduate Scholarships, IEEE Vancouver Section Centennial Award, 2011 UBC Killam Research Prize, 2017 Canadian Award for Telecommunications Research, 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award, and 2018 ACM MSWiM Reginald Fessenden Award. He coauthored papers that won the 2017 IEEE ComSoc Fred W. Ellersick Prize, 2017 IEEE Systems Journal Best Paper Award, 2018 IEEE CSIM Best Journal Paper Award, and 2019 IEEE TCGCC Best Journal Paper Award. He is a Life Fellow of IEEE, and a Fellow of the Royal Society of Canada (Academy of Science), Canadian Academy of Engineering, and Engineering Institute of Canada. He is named in the current Clarivate Analytics list of Highly Cited Researchers.