Securing Parked Vehicle Assisted Fog Computing With Blockchain and Optimal Smart Contract Design

Xumin Huang, *Member, IEEE*, Dongdong Ye, Rong Yu, *Member, IEEE*, and Lei Shu, *Senior Member, IEEE*

***Abstract*—Vehicular fog computing (VFC) has been envisioned as an important application of fog computing in vehicular networks. Parked vehicles with embedded computation resources could be exploited as a supplement for VFC. They cooperate with fog servers to process offloading requests at the vehicular network edge, leading to a new paradigm called parked vehicle assisted fog computing (PVFC). However, each coin has two sides. There is a follow-up challenging issue in the distributed and trustless computing environment. The centralized computation offloading without tamper-proof audit causes security threats. It could not guard against false-reporting, free-riding behaviors, spoofing attacks and repudiation attacks. Thus, we leverage the blockchain technology to achieve decentralized PVFC. Request posting, workload undertaking, task evaluation and reward assignment are organized and validated automatically through smart contract executions. Network activities in computation offloading become transparent, verifiable and traceable to eliminate security risks. To this end, we introduce network entities and design interactive smart contract operations across them. The optimal smart contract design problem is formulated and solved within the Stackelberg game framework to minimize the total payments for users. Security analysis and extensive numerical results are provided to demonstrate that our scheme has high security and efficiency guarantee.**

***Index Terms*—Blockchain, parked vehicle, smart contract, Stackelberg game, vehicular fog computing (VFC).**

Manuscript received September 1, 2019; revised November 18, 2019, December 20, 2019, January 6, 2020; accepted February 6, 2020. This work was supported in part by the National Natural Science Foundation of China (61971148), the Science and Technology Program of Guangdong Province (2015B010129001), the Natural Science Foundation of Guangxi Province (2018GXNSFDA281013), the Foundation for Science and Technology Project of Guilin City (20190214-3), and the Key Science and Technology Project of Guangxi (AA18242021). Recommended by Associate Editor Peiyun Zhang. *(Corresponding author: Rong Yu.)*

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

X. M. Huang and D. D. Ye are with the School of Automation, Guangdong University of Technology, Guangzhou 510006, and also with the Guangdong Key Laboratory of IoT Information Technology, Guangdong University of Technology, Guangzhou 510006, China (e-mail: huangxumin@163.com; dongdongye8@163.com).

R. Yu is with the School of Automation, Guangdong University of Technology, Guangzhou 510006, and also with Guangdong-Hong Kong-Macao Joint Laboratory for Smart Discrete Manufacturing, Guangzhou 510006, China (e-mail:yurong@ieee.org).

L. Shu is with the College of Engineering, Nanjing Agricultural University, Nanjing 210095, China, and also with the School of Engineering, University of Lincoln, Lincoln LN67TS, UK (e-mail: lei.shu@ieee.org).

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

Digital Object Identifier 10.1109/JAS.2020.1003039

NOMENCLATURE

α_s^V, β_s^V : Percentages of the workloads of requester V offloaded to fog server s and PV i, respectively.

κ_i : Power consumption parameter of CPU execution in PV i.

ρ_i^{rx}, r_i^{dl} : Receiver power and downlink data rate of PV i in receiving data.

ρ_s^{rx}, r_s^{dl} : Receiver power and downlink data rate of fog server s in receiving data.

ρ_{rsu}^{tx} : Transmitter power of a roadside unit

A_1, A_2, τ : Power consumption parameters of CPU execution in the fog server.

$B_i, g_i, \varepsilon, d, N_0$: Downlink channel model parameters of PV i: bandwidth, channel gain, path loss exponent, distance, noise power spectral density.

b_i^V : Serviceability of PV i to serve requesting vehicle V.

c_i^V, l_i^V : Energy and time consumption of processing per percentage of the workloads from requester V in PV i.

c_s^V, l_s^V : Energy and time consumption of processing per percentage of the workloads from requester V in fog server s.

C_V : Total payment of requester V in the offloading service.

D_V^{in}, h_V, W_V, L_V : Input data, application-centric parameter, workloads and latency requirement of a task from requester V.

F_i : Total computing resource of PV i.

f_i^V : Computing resources of PV i for serving requester V.

f_s^V : Computing resources of fog server s for serving requester V.

n_s : Status-related parameter of fog server s.

p_e : Expenditure for consuming one unit of energy.

P_i : Probability of PV i that stays continuously in an entire time period T.

p_s^V : Payments given by requester V to fog server s for processing per workload.

R_V : Total reward given by requester V to employ all the PVs for processing workloads.

I. INTRODUCTION

VEHICULAR fog computing (VFC) has been emerged as one of the most promising directions of fog computing in transportation domains that provides lightweight and ubiquitous computing at the vehicular network edge [1]. Compute-intensive applications such as autonomous driving and augmented reality are supported well by utilizing proximal computation resources to improve service provision [2]. In vehicular

networks, parked vehicles (PVs) comprise a high proportion of daily vehicles and meanwhile, have underutilized resources for task execution. In contrast to regular fog servers, PVs are also competent to process user requests and become special fog nodes [3]. A new paradigm called by parked vehicle assisted fog computing (PVFC) is defined to depict convergence of PVs and original VFC [4]. However, critical security issues should be addressed for the large-scale system implementation considering the distributed and trustless computing environment.

Centralized computation offloading causes security challenges to PVFC. For enhancing the computing capacity, more PVs are expected to undertake workloads but this results in increasing risks due to the untrustworthiness of distributed performers. In the centralized scheme, requesters and performers are strictly managed by a third party as a central authority. The scheme neglects the possible trustless environment and accessible records of network behaviors. As a consequence, the scheme based on the "trusted" central authority without tamper-proof audit gives rise to the following security threats:

1) Vulnerability: The central authority easily suffers from single point of failure, remote hijacking and DDoS attacks. Moreover, if the authority is compromised, private information of requesters and performers may be revealed, leading to privacy disclosure.

2) Maliciousness: Misbehaving requesters cause flooding attacks by submitting forged requests. Malicious performers may perform spoofing attacks, for example, pretending to accept workloads, and refusing to feedback results or upload incorrect results. Then the aggregation of all the results is procrastinated.

3) Untraceability: Without recording necessary proofs, requester and performers probably launch reward repudiation of giving and reception, respectively. Free-riding behaviors of selfish requesters and cheating behaviors of greedy performers cannot be examined.

To cope with the issues, we leverage blockchain to provide security protection for PVFC. Thus, a dedicated blockchain system, called by PVFChain, is introduced to record and validate relevant information about requesters and performers in computation offloading. As a public and immutable ledger, blockchain is composed of a chain of blocks and each block consists of a set of auditing transaction records. Consensus nodes are particularly employed as regular auditors (i.e., miners) to verify a new block for token rewards. Valid blocks will be directly or indirectly stored in local blockchain replicas of nodes. Nowadays, blockchain has been widely applied to vehicular networks for diverse purposes, for example, simplifying distributed key management in heterogeneous vehicular communications [5], encouraging users with incentives for anonymous announcements [6] and achieving secure data storage and sharing among vehicles [7].

Owing to the great advantages of blockchain, the decentralized PVFC is fulfilled by depending on a majority of consensus nodes without any third parties, eliminating potential security risks. Request posting, workload undertaking, task evaluation and reward assignment are organized by using smart contracts, which are computer programs automatically running and published to miners for auditing. Each network entity registers to PVFChain and network activity is instructed through the smart contract execution to follow the contact based agreement. Through data auditing, accessible records about behaviors of requesters and performers are tamper-resistant. Misbehaviors of malicious requesters and performers are completely exposed due to the embedded transparency, irreversibility and accountability of smart contracts. Finally, PVFChain supports 1) identity authentication, 2) request validation, 3) computation verification, and 4) reward integrity to guard against a variety of network attacks.

In this paper, we investigate the PVFChain and optimal smart contract design for guaranteeing network security and efficiency of PVFC. Smart contract operations are designed to confirm behaviors of requesters and performers with security and privacy awareness. Based on recording proofs, network activities become uncompromising and traceable to overcome security threats in the trustless environment. Moreover, we study an optimal smart contract design problem when employing a fog server and several PVs to conduct workloads for a requester. The problem is formulated from an economic viewpoint and solved by the Stackelberg game approach. The requester acts a leader to optimize reward policy for performers (followers) responding with the number of undertaking workloads. The total workloads are allocated well to minimize the overall payment and meet task requirements simultaneously. The main contributions of the paper are summarized as follows.

1) We introduce PVFChain where blockchain with smart contract is exploited for PVFC to implement offloading services in a decentralized manner.

2) We design a feasible system model for large-scale application of PVFChain. Interactive smart contract operations across the network entities are described to address security issues.

3) We apply the Stackelberg game solution to achieve the optimal smart contract design with fairness. The reward policy is optimized to reduce the service fee of the requester while considering utility maximization for the performers.

The rest of this paper is organized as follows. Section II presents the related work. Section III describes crucial network entities and smart contract operations across them in PVFChain. In Section IV, the optimal smart contract design problem is formulated as a payment minimization problem. After that, the problem is solved within the Stackelberg game framework in Section V. Security analysis and performance evaluation of our scheme are provided in Sections VI and VII, respectively. Finally, Section VIII concludes this paper.

II. Related Work

A. Vehicular Fog Computing

Recently, researchers have presented the concept, network architectures and use cases of VFC. The concept of VFC is first introduced by Hou *et al.* [1]. Slowly moving and parked vehicles can become crucial computational infrastructures to

enlarge the computing capacity of VFC. In addition, the capability of VFC was analyzed and evaluated based on a real vehicular mobility trace in large-scale urban environment [8]. Huang *et al.* [9] further proposed a hierarchical architecture to clarify three-level decision-making capability in the entire network of VFC. Case studies were also provided to explain how the proposed network architecture enhances current vehicular applications, e.g., traffic control, driving safety and entertainment.

Moreover, a variety of offloading schemes are studied for VFC in different application scenarios. The deep reinforcement learning was leveraged to construct an intelligent offloading system for VFC [10], where task scheduling and resource allocation were jointly optimized to improve overall user experience. The authors in [3] and [11] considered forwarding traffic among statistic cloudlet, and dynamic parked and moving-vehicle-based fog nodes, and studied how to minimize the system transmission delay. Similarly, Zhu *et al.* [12] focused on the optimal task allocation between fixed fog servers and nearby moving vehicles as mobile fog nodes to minimize a weight sum of the maximal service latency and total quality loss. In [4], PVs were exploited as lightweight fog nodes to coordinate with regular fog nodes to realize smart parking and jointly provision delay-sensitive computing services. Particularly, auction theory was used to provide PVs with available parking places and monetary rewards in a considerate way. Reference [13] also paid attention to cooperative task execution between PVs and a fog server, and tried to determine workload allocation between them to minimize the service fee in computation offloading. But general task requirements were not considered well.

Security issues are seldom discussed in the previous works. Most of them focused on optimizing task scheduling in a centralized way for improving the overall performance. The efficiency of the schemes will be degraded with the increasing number of malicious requesters and performers. Thus, our work explores security threats in offloading services of PVFC and resorts to blockchain with smart contracts for enhancing the security protection.

B. Blockchain for Vehicular Networks

Based on the high transparency, fault tolerance and tamper-proof features, blockchain has been proposed to vehicular networks in different domains. For example, blockchain is convinced to have real benefits in promoting key transfer between two heterogeneous vehicular communication domains and realizing the verifiable and authentic key transfer processes [5]. When vehicles access different VFC data centers, a cross-datacenter authentication and key exchange mechanism was designed by using blockchain and elliptic curve cryptography, to support information confidentiality and mutual authenticity [14]. Performance evaluation demonstrated that the mechanism had lower computational cost and communication overheads.

The anonymity of blockchain also caused widespread concerns. Reference [6] introduced a blockchain-based incentive mechanism for encouraging vehicles to send announcements in traffic environment with privacy preservation. Similarly, the authors in [15] presented an anonymous reputation evaluation system for eliminating forged messages from malicious vehicles in the blockchain environment. The work in [7] proposed a dedicated consortium blockchain where roadside units are responsible for collecting and auditing data generated by vehicles, to guarantee high-quality data sharing among vehicles. Besides, Yang *et al.* [16] put forward a decentralized trust management scheme for secure message delivery among vehicles. Vehicles were employed to evaluate the credibilities of traffic-related messages, and roadside units become miners to collect ratings from vehicles and validate them.

Motivated by the existing works, we believe that the inherent distributed nature with consensus mechanisms makes blockchain suitable for orchestrating offloading services without any third party and fulfilling the decentralized PVFC. Interaction among untrusted requesters and performers can be transcoded into smart contracts and accessible on the PVFChain for validation. With the adoption of on-chain smart contracts, the credibility, programmability and traceability of network behavior records are exactly guaranteed over time. At the same time, we formulate the optimal smart contract design problem and aim to design a user-centric incentive mechanism in computation offloading.

III. PVFCHAIN DESIGN

A. Network Entities

As illustrated in Fig. 1, crucial network entities are proposed for feasible implementation of PVFChain. Overall, the related network entities include requester, performer and miner. When running a compute-intensive application, a mobile vehicle on the road posts a computation task to PVFChain, triggering a smart contract. The background system of PVFChain assigns an agent to be in charge of the task. The agent orchestrates offloading services at the vehicular network edge. In the PVFC environment, the agent seeks a fog server near a parking lot and employs idle PVs in the parking lot for

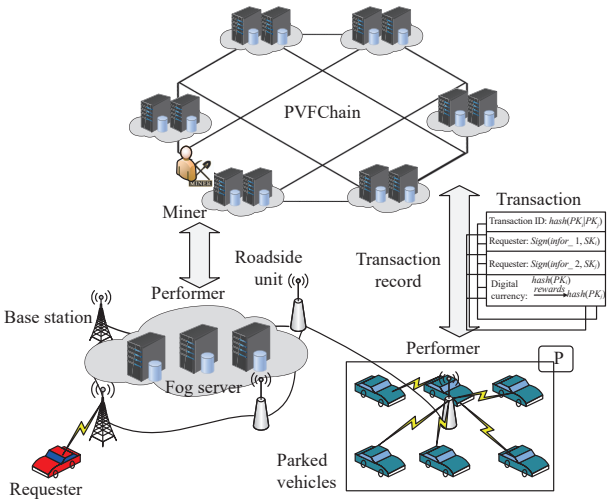


Fig. 1. Network entities in PVFChain: requester, performer, and miner.

cooperative task execution. For the necessary performer selection and decision making, the performers may be required to record some status information to the smart contract. The agent is authorized to access to the smart contract and hold prior knowledge about the task requirements and performers.

More specifically, cooperative and traceable offloading services are performed as follows. When receiving the submitted task with customized requirements, the agent notifies the fog server and several PVs to jointly handle the task for getting rewards. Based on the promising rewards, the fog server and each PV response with the corresponding numbers of undertaking workloads. After gathering the responses, the agent partitions the whole input data into various subsets with different data sizes, and transmits them to the fog server and every PV, respectively. Performers conduct the given workloads and upload output results to miners in PVFChain, which check them whether to meet the task requirements, e.g., accuracy performance. By executing the consensus protocol and third-party methods, valid output results are identified. The output results are aggregated by the agent to obtain a final result for the requester. Rewards are assigned from the digital wallet of the requester to those qualified performers. For traceability, relevant information including request posting, workload undertaking, task evaluation and reward assignment is recorded to form a list of transactions bundled into a block. Finally, confirmed blocks are added to PVFChain at specific intervals by miners. Next, we provide more details about the network entities as follows.

1) Requester in Computation Offloading: The offloading request is published to PVFChain through a client, which offers users with entrance to submit a computation task with task requirements. The requester utilizes key pairs issued by PVFChain for ensuring secure communications and avoiding revealing true identity. The request is sent to the nearest access point, e.g., roadside unit and base station, and further delivered to generate the new smart contract in the background system. Moreover, the requester is required to pay a deposit in advance.

The requester grants the trusted agent to specially process the task. Thus, those fog servers and PVs managed by the agent are able to realize the request. As a service proxy, the agent represents the requester to design an incentive mechanism for different performers, in order to minimize the overall payment and meet the task requirements simultaneously.

2) Performer in PVFC: In PVFC, there exist two kinds of performers for supporting proximal computing at the vehicular network edge. Here, fog servers refer to specialized edge servers applied to the transportation domains and are particularly deployed to serve vehicular users. In addition, modern vehicles are equipped with powerful processing units and large storage devices, and can be scheduled for task execution [17], particularly when they enter to the parked state with idle time. Compared with a fog server, a PV is of lightweight computing capability. But in a region, hundreds of PVs are distributed while only several fog servers are deployed. Service fee in employing a fog server is generally higher than that of scheduling a PV, due to different energy consumption cost of processing workloads.

PVFChain also provides client accounts for performers to register identities, view offloading requests and write information into smart contracts. Facing with a request, performers make decisions whether to accept it and the number of workloads independently, with the given monetary rewards. Before processing workloads, the performers are also required to afford a deposit in advance. Then the agent transmits input data with various data sizes to the fog server and different PVs, respectively. In particular, the agent communicates with local PVs with the help of a roadside unit locating in the center of the parking lot. All the performers upload output results via clients to verify their computation work, thereafter, are rewarded or punished according to auditing results.

3) Miner of PVFChain: In this paper, we consider that PVFChain could be a consortium blockchain so that write and read permissions of transaction records are granted to specific legal entities. According to [7], some roadside units are preselected and authorized to perform the consensus process in vehicular networks. For improving mining, miners also exploit available computing ability of fog servers connecting to the roadside units, to promote the mining procedure and reduce propagation and consensus time of new blocks [18]. For security guarantee, the typical proof of work (PoW) is utilized as the consensus protocol, where miners compete against each other for winning the privileges of block generation.

In PVFChain, the procedure that a requester posts an offloading request, and a performer participates in serving it, is treated as a transaction in Fig. 1. We take requester i and performer j, whose public and private keys are $[PK_i, SK_i]$ and $[PK_j, SK_j]$ respectively, as an example. The transaction mainly consists of the following parts: transaction ID ($hash(PK_i|PK_j)$), related information about task posting and signed by the requester $infor\_1$, auxiliary information to describe task execution and signed by the performer $infor\_2$, and digital currency circulation caused by reward assignment. Here, $infor\_1$ includes task requirements, task priority and promising rewards provided by the requester. $infor\_2$ includes the received input data, the number of processing workloads, output result, actual finish time, evaluation result for the computation work and so on. The addresses about digital wallets of the requester and performer are hash addresses corresponding to their public keys, respectively.

To confirm transactions, miners audit new transactions according to the contract-based agreement. The auditing operations are described as follows. 1) For new transactions, an auditor authenticates the identities of the requester and performers, and checks the validity of the transactions. 2) It validates whether the requester and the recruited performers with the deposits are recorded, and evaluates whether the performers upload the correct output results within task deadline. 3) The auditor finally verifies how many rewards are transferred from the requester to performers to follow the reward policy.

Confirmed transactions regarding a common requester are

listed into a new block, which is accessed by other auditors for consensus. Finally, the validated block is attached with a time stamp and linked to the previous block on the PVFChain.

B. Smart Contract Operations

We design interactive smart contract operations across the network entities to construct a reliable and traceable computation offloading environment. Then network behaviors are recorded and transcoded into smart contracts. As self-executing scripts, smart contracts are stored on the PVFChain and triggered to cope with transactions. They are automatically executed by related network entities in a predefined way. Each smart contract has a unique address on the PVFChain and allows us to express business logic for restricting network behaviors in offloading services. Nowadays, there exist several commercial platforms to support smart contracts residing on the blockchain. One of the typical platforms is Ethereum [19] and we encode smart contracts on the platform.

Smart contract operations between a requester, a service provider with employed performers and miners are shown in Fig. 2. A smart contract ensures that both the requester and performers reach an agreement and follow their prescribed behaviors. To this end, PVFChain include two parts: client as user interface and blockchain for verification. Clients allow all the network entities to interact with others while the blockchain part is to validate generated transactions.

Particularly, miners are employed in PVFChain to verify computation work for token rewards by using the third-party methods, e.g., [20] and [21]. The methods could be directly utilized to promptly check whether the output results are matched with the allocated input data and satisfy the accuracy requirement, without reexecuting the tasks. We take superresolution of license plates as an example to demonstrate how the third-party methods are used for computation verification. A requester in computation offloading may have several low-resolution images of license plates, and also own

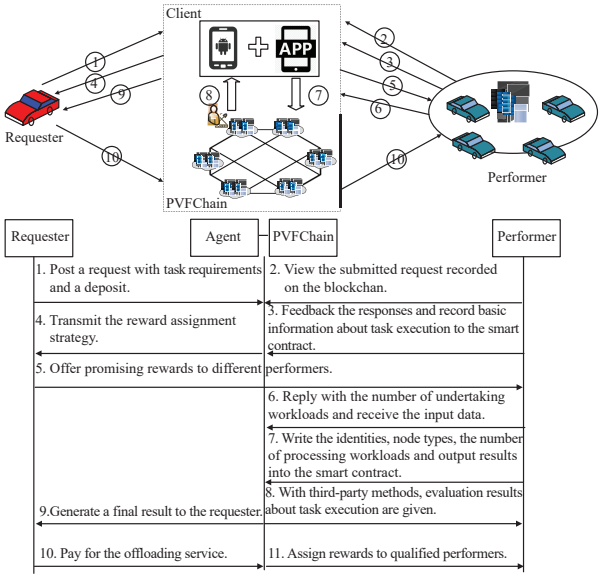


Fig. 2. Smart contract operations.

a trained DeblurGan [22], which was developed as a generative adversarial network model to achieve blind motion deblurring for given images. In PVFChain, the requester transfers images with the DeblurGan model to an agent. The agent recruits the fog server and multiple PVs to receive input data and perform the image superresolution tasks. By executing the DeblurGan model, a performer calculates new image data y (as output result) with respect to original image data x. After that, a miner is easy to measure the mean-squared error between x and y, as proposed in the above reference, and further judge whether y is qualified to meet the performance of image superresolution. Finally, computation work is evaluated accordingly.

In summary, smart contract operations are conducted as follows.

Step 1: The requester submits a computation task with task requirements and a deposit to PVFChain by using the client. An agent is then specifically assigned to process the task.

Step 2: Performers can view the offloading request with the task requirements through the clients.

Step 3: Some performers feed back positive responses to the request, and are required to upload basic information about task execution along with the responses.

Step 4: The agent is authorized to realize and select appropriate performers according to the prior knowledge, and determine reward assignment for different performers.

Step 5: Suggested by the agent, the client of the requester offers promising rewards to the performers.

Step 6: According to the given rewards, the performers choose a part of workloads to process, and receive the corresponding input data from the agent for task execution.

Step 7: The performers allocate computing resources to accomplish workloads and acquire output results. They write their identities, node types, the number of processing workloads and output results into the smart contract.

Step 8: In the smart contract, a specialized transaction is triggered to authorize miners to evaluate the output results. The miners select appropriate third-party methods, based on application scenarios, to put forward evaluation results, which tell whether the according computation work is qualified. Once there exist unqualified results, backup fog servers are activated automatically to supplement the corresponding output results.

Step 9: By collecting all the output results, the agent aggregates them to form a final result and transmits it to the client of the requester.

Step 10: Ultimately, the requester offers extra payments in addition to the deposit. The promising rewards with the submitted deposits are assigned to qualified performers.

IV. Optimal Smart Contract Design Problem

There exists an essential problem for designing an optimal smart contract to encourage different performer for accomplishing workloads in PVFChain. This refers to a payment minimization problem for a requester in computation offloading, where the whole workloads are optimally allocated to a fog server near a parking lot and several PVs in the parking lot for a cooperative task execution.

Particularly, when there exist a set of requesters $\mathcal{V}$ to be served simultaneously, we consider that the allocated computing resources from the fog server s and each PV i to the requester $V \in \mathcal{V}$ are equal to $f_s^V = \mu^V F_s$ and $f_i^V = \mu^V F_i$, respectively, where F_s and F_i are the total computing resources owned by the fog server and the PV, respectively, and μ^V represents the task priority. We also have $\sum_{V\in\mathcal{V}} \mu^V = 1$. Given f_s^V and f_i^V, we can pay attention to separately coping with the payment minimization problem for each requester V. Hence, for simplicity, we investigate how to incentivize fog server s and each PV i for serving a single requester V in the next part.

A. Utility Function of Fog Server

When running a compute-intensive application, a vehicle V sends an offloading request with task requirement. The task is described in four terms $\{D_V^{in}, h_V, W_V, L_V\}$. The D_V^{in} indicates the input data size. Compared with the size of input data, the size of output data is neglected [23]. The required workloads of accomplishing the task are W_V, in terms of the number of CPU cycles. We have $W_V = h_V D_V^{in}$, where h_V is an application-centric parameter related to the type of running application [24]. As for L_V, it represents the latency requirement. After realizing the request, an agent seeks the fog server and appropriate PVs for undertaking workloads.

For the fog server s, it may be particularly deployed by an offloading service provider to undertake workloads for earning monetary rewards in computation offloading. The utility function of the fog server is related with acquired rewards, energy consumption cost and incurred inconvenience because of serving the external user. Denote the payments given by requester V to the fog server for executing per workload as p_s^V. The computational speed of the CPU is f_s^V when serving requester V. For accomplishing workloads, power consumption of CPU execution is $A_1 f_s^{V\tau} + A_2$, where A_1, A_2 and τ are three positive constants and τ ranges from 2.5 to 3 [25]. In this paper, $\tau = 3$. A part of energy is also consumed for receiving input data. When the percentage of the workloads offloaded to the fog server is α_s^V, the energy consumption cost $c_s^V \alpha_s^V$ is

$$c_s^V \alpha_s^V = p_e \left[\alpha_s^V W_V \left(A_1 f_s^{V2} + \frac{A_2}{f_s^V} \right) + \rho_s^{rx} \frac{\alpha_s^V D_V^{in}}{r_s^{dl}} \right] \tag{1}$$

where p_e is the expenditure for consuming one unit of energy, ρ_s^{rx} is receiver power and r_s^{dl} is downlink data rate of receiving input data from the agent.

Besides, additional task execution gives rise to temporary inconvenience for the fog server, which may have individual workloads at that time. The incurred inconvenience resulted by serving the external request is modeled as $n_s(\alpha_s^V W_V)^2$ [26], where n_s is a status-related parameter to indicate the ratio between current workloads and the maximal workloads that can be undertaken. To summarize, the utility function of the fog server is expressed as acquired rewards minus energy consumption cost and incurred inconvenience,

$$U_s^V(\alpha_s^V, p_s^V) = p_s^V \alpha_s^V W_V - c_s^V \alpha_s^V - n_s(\alpha_s^V W_V)^2. \tag{2}$$

The fog server determines α_s^V for maximizing the utility based on the given reward parameter p_s^V.

B. Utility Function of Parked Vehicle

A part of PVs are selected to support the offloading service. Due to parking behaviors and resource availability, various PVs show different serviceabilities in being employed to execute tasks with reliability and efficiency guarantee.

According to the existing work in [13], the predicted probability of a PV that will continue to stay parked in a whole task scheduling time period, can be utilized as a crucial metric to evaluate the serviceability of the PV. As an employer, the agent is responsible for seeking PVs with higher serviceabilities periodically. The time span of this time period is denoted as T. With the everlasting records and complex analysis, the probability density function about parking durations of PVs is acquired. The function is expressed by $g(t)$, where t is the parking duration and has a considerable range, $t \in [0, t^{\max}]$. The accumulative parking durations up to the beginning of the time period is detected as ap_i. We estimate the probability of staying continuously in the following time period by

$$P_i = \int_{ap_i+T}^{t^{\max}} \frac{g(t)}{1-G(ap_i)} dt = \frac{1-G(ap_i+T)}{1-G(ap_i)} \tag{3}$$

where $G(t)$ is the cumulative distribution function of $g(t)$. In this paper, we consider that the serviceability of PV i at that time is related to the probability.

On the other hand, computing capability of PV i (namely, F_i) is regarded as the other metric to evaluate the serviceability. For computing efficiency, a qualified performer is required to have great computing capability with regard to a threshold value F_{th}. Hence, the serviceability of the PV for serving requester V (represented by b_i^V) is equal to a weight sum of P_i and F_i/F_{th},

$$b_i^V = (1-\varpi) P_i + \varpi \frac{F_i}{F_{th}} \tag{4}$$

where $0 \le \varpi \le 1$ is a preset weighting factor. At the beginning of each task scheduling time period, the serviceabilities of existing PVs are updated by the background system of PVFChain via necessary parameter detections and collections. Target PVs that satisfy $b_i^V \ge b_{th}$ are found as qualified performers and added into a candidate performer set $\mathcal{I}$.

In the incentive mechanism, all the PVs are encouraged to compete against each other for earning income from the offloading service. By jointly considering the number of workloads and serviceability for task execution, the agent allocates rewards to PV i ($i \in \mathcal{I}$) based on the following the rule: $R_V(b_i^V \beta_i^V / \sum_{i\in\mathcal{I}} b_i^V \beta_i^V)$, where β_i^V is the percentage of the workloads undertaken by the PV and R_V is the total reward offered by requester V to award all the participating PVs. In this way, the agent considers fair reward allocation, and also encourages those PVs with higher serviceabilities to undertake more workloads.

Similarly, task execution causes energy consumption cost to PV i. During the task execution, the power consumption of CPU execution is $\kappa_i f_i^{V3}$ [27], where κ_i and f_i^V are the hardware

parameter and computational speed respectively, of the CPU to serve the requester. Then the consumed energy for CPU execution is $\beta_i^V W_V \kappa_i f_i^{V2}$. The agent transmits the input data to each PV with the help of a roadside unit in the center of the parking lot. Considering the data transmission cost, we calculate the total energy consumption cost $c_i^V \beta_i^V$ by

$$c_i^V \beta_i^V = p_e \left(\beta_i^V W_V \kappa_i f_i^{V2} + \rho_i^{rx} \frac{\beta_i^V D_V^{in}}{r_i^{dl}} \right) \tag{5}$$

where r_i^{dl} is downlink data rate of PV i when receiving the input data from the local roadside unit, and ρ_i^{rx} is the receiver power.

We formulate the downlink channel model between PV i and the roadside unit with the utilization of orthogonal frequency division multiplexing technology. So r_i^{dl} is acquired by

$$r_i^{dl} = B_i \log_2 \left(1 + \frac{\rho_{rsu}^{tx} g_i d_i^{-\varepsilon}}{N_0} \right) \tag{6}$$

where B_i is the available bandwidth, $\rho_{\text{rsu}}^{\text{tx}}$, g_i and d_i are the transmitter power of the local roadside unit, channel gain and distance between the PV and the roadside unit, respectively. N_0 indicates the noise power spectral density and ε is a path loss exponent. We express the utility function of PV i as

$$U_i^V(\beta_i^V, R_V) = \frac{b_i^V \beta_i^V}{\sum\limits_{i \in \mathcal{I}} b_i^V \beta_i^V} R_V - c_i^V \beta_i^V. \tag{7}$$

For utility maximization, the PV makes a decision on β_i^V with respect to R_V and strategies of other PVs.

C. Payment Minimization Problem

As for the requesting vehicle, the utility function is directly bound up with service fee in computation offloading. Note that the requesting vehicle only consumes energy to submit the task with task requirements to the agent through the client. After that, the agent acts as a trusted service proxy to partition the task and transmit according input data to the fog server and parked vehicles for a cooperative task execution. Compared with the total payment given to the performers, communication cost of the requesting vehicle is fixed, and can be neglected in this paper.

To employ the fog server and several PVs, cost function of the requester in terms of the total payment is shown by

$$C_V = p_s^V \alpha_s^V W_V + R_V. \tag{8}$$

The agent tries to minimize the total payment for maximizing user satisfaction and meeting task requirements simultaneously. First, the whole workloads should be entirely offloaded, $\alpha_s^V + \sum\limits_{i \in \mathcal{I}} \beta_i^V = 1$. The task also has the strict latency requirement for processing on the fog server and each PV. The total delay on the fog server side includes data transmission time and task execution time, namely,

$$d_s^V = \frac{\alpha_s^V D_V^{in}}{r_s^{dl}} + \frac{\alpha_s^V W_V}{f_s} = l_s^V \alpha_s^V. \tag{9}$$

Similarly, the total delay resulted by on the PV side is

$$d_i^V = \frac{\beta_i^V D_V^{in}}{r_i^{dl}} + \frac{\beta_i^V W_V}{f_i} = l_i^V \beta_i^V. \tag{10}$$

We know that d_s^V and d_i^V are linearly increased with the increasing values of α_s^V and β_i^V, respectively. To meet the latency requirement, d_s^V and d_i^V are limited within $L_V - d_{\text{abp}}$, where d_{abp} is the overall delay on the blockchain side. The d_{abp} mainly refers to the average block propagation delay across the whole network for auditing a new block. According to [28], the delay of reaching consensus for shared transaction records is defined by $d_{\text{abp}} = \lambda/\pi m_1 + m_2 \lambda$, where m_1 is a network scale-related parameter to indicate the number of consensus nodes, π is the average data rate of all the communication links between any two miners, m_2 is a constant related to the network scale and the average verification speed of each miner, and λ is the average data size of unverified blocks.

For minimizing the service fee of the requester, the agent optimizes various rewards for different performers by solving the following problem:

$$\begin{aligned} &\min_{p_s^V, R_V} \ C_V \\ &\text{s.t. } 0 \le \alpha_s^V, \beta_i^V \le 1 \\ &\quad \alpha_s^V + \sum_{i \in I} \beta_i^V = 1 \\ &\quad d_s^V \le L_V - d_{\text{abp}} \\ &\quad d_i^V \le L_V - d_{\text{abp}}, \forall i. \end{aligned} \tag{11}$$

In PVFChain, the optimal smart contract design problem is transformed to solve the payment minimization problem above for enriching user satisfaction.

V. Stackelberg Game Solution

We utilize the Stackelberg game approach to solve the payment minimization problem in PVFChain. The proposed Stackelberg game model can be solved by the subgame perfect Nash equilibrium. According to previous works, a subgame perfect Nash equilibrium indicates a Nash equilibrium of every subgame in the original game. We exploit backward induction as the typical method to discuss subgame perfect equilibrium.

A. Stackelberg Game Model

In this paper, we design the incentive mechanism for two kinds of performers by modeling the strategic interaction between them and the requesting vehicle as a two-stage single-leader multi-follower Stackelberg game. The agent of requesting vehicle V determines the reward policy (p_s^V, R_V) given to the fog server and PV group, respectively. In turn, the performers reply to the agent with the numbers of undertaking workloads for maximizing the utilities with respect to the promising rewards, as explained in (2) and (7), respectively. Particularly, each PV belonging to the group competes in a non-cooperative workload determination subgame. In summary, we define the two-stage reward-participation game as follows.

Stage 1: Reward policy. The agent of requesting vehicle V optimizes the reward policy for various performers with different responses in order to minimize the payments and

satisfy necessary constraints shown by problem in (11)

$$(p_s^{V*}, R_V^*) = \arg\min_{p_s^V, R_V} C_V(p_s^V, R_V, \alpha_s^V, \boldsymbol{\beta}^V)$$

where $\boldsymbol{\beta}^V$ is a strategy set of all the PVs belonging to $\mathcal{I}$.

Stage 2: Participation level. For utility maximization, the fog server determines the participation level α_s^V to conduct workloads, given the promising reward parameter p_s^V. Similarly, each PV i competes in a non-cooperative workload determination subgame and chooses the participation level β_i^V, given the total reward R_V and the strategies of other PVs except PV i, β_{-i}^V.

$$\alpha_s^{V*} = \arg\max_{\alpha_s^V}\{U_s^V(\alpha_s^V, p_s^V)\}$$
$$\beta_i^{V*} = \arg\max_{\beta_i^V}\{U_i^V(\beta_i^V, \beta_{-i}^V, R_V)\}.$$

The objective of the Stackelberg game between the requester and all the performers is to find the unique Stackelberg equilibrium at which the leader obtains the minimal payments given the best responses of the followers. At that time, both of them have no motivations to unilaterally change their decisions. Given the single-leader multi-follower Stackelberg game, we further define the Stackelberg equilibrium as follows.

Definition 1 (Stackelberg equilibrium): There exists a set of strategies $(p_s^{V*}, R_s^{V*}, \alpha_s^{V*}, \boldsymbol{\beta}^{V*})$ which is the Stackelberg equilibrium if and only if the requester V, fog server s, and any PV i satisfy the following inequalities, respectively.

$$C_V(p_s^{V*}, R_V^*, \alpha_s^{V*}, \boldsymbol{\beta}^{V*}) \le C_V(p_s^V, R_V, \alpha_s^{V*}, \boldsymbol{\beta}^{V*})$$
$$U_s^V(\alpha_s^{V*}, p_s^{V*}) \ge U_s^V(\alpha_s^V, p_s^{V*})$$
$$U_i^V(\beta_i^{V*}, \beta_{-i}^{V*}, R_V^*) \ge U_i^V(\beta_i^V, \beta_{-i}^{V*}, R_V^*), \forall i.$$

We try to find the subgame perfect Nash equilibrium to achieve the Stackelberg equilibrium. In the proposed game, there exists a subgame among PVs and they strictly compete in a non-cooperative fashion. The Nash equilibrium of the subgame is denoted as $(\beta_i^{V*}, \beta_{-i}^{V*})$, where the utility of each PV i cannot be maximized further by adopting a different strategy other than the given strategies β_{-i}^{V*}. For the fog server, the best response is acquired as α_s^{V*} by maximizing the utility with respect to p_s^{V*}. As for the leader, it determines the optimal reward policy for the fog server and PV group to obtain the minimal payments by predicting their best responses. To reach the Stackelberg equilibrium, we adopt the backward induction in this paper.

B. Best Response Analysis

We first analyze the best response of the fog server α_s^{V*}. Given the reward parameter p_s^V, the fog server always has the optimal strategy because U_s^V can be converted into an optimal utility function in terms of α_s^{V*}. We take the first and second derivatives of U_s^V with respect to α_s^V, and easily have

$$\frac{\partial U_s^V}{\partial \alpha_s^V} = p_s^V W_V - c_s^V - 2n_s \alpha_s^V W_V^2$$
$$\frac{\partial^2 U_s^V}{\partial \alpha_s^{V2}} = -2n_s W_V^2 < 0. \tag{12}$$

The utility function is concave and this indicates that the maximal value of the function exists. By using $\frac{\partial U_s^V}{\partial \alpha_s^V} = 0$, we have

$$\alpha_s^{V*} = \frac{p_s^V W_V - c_s^V}{2n_s W_V^2}. \tag{13}$$

The best response is acquired by the fog server when determining the number of undertaking workloads for serving requesting vehicle V, in order to maximize the utility under the condition of given p_s^V.

As for PV $i \in \mathcal{I}$, the best response for serving requester V is denoted as β_i^{V*}. We also take the first and second derivatives of U_i^V with respect to β_i^V

$$\frac{\partial U_i^V}{\partial \beta_i^V} = \frac{b_i^V \sum b_{-i}^V \beta_{-i}^V}{(\sum_{i\in\mathcal{I}} b_i^V \beta_i^V)^2} R_V - c_i^V$$
$$\frac{\partial^2 U_i^V}{\partial \beta_i^{V2}} = -\frac{2b_i^{V2} \sum_{i\in\mathcal{I}} b_i^V \beta_i^V \sum b_{-i}^V \beta_{-i}^V}{(\sum_{i\in\mathcal{I}} b_i^V \beta_i^V)^4} R_V < 0. \tag{14}$$

Due to the negative value of the second-order derivative, U_i^V is a strictly concave function. So PV i has a unique and optimal strategy with respect to $R_V > 0$ and β_{-i}^V. We use the first-order optimality condition $\frac{\partial U_i^V}{\partial \beta_i^V} = 0$ and have

$$\frac{\sum b_{-i}^V \beta_{-i}^V}{(\sum_{i\in\mathcal{I}} b_i^V \beta_i^V)^2} = \frac{c_i^V}{b_i^V R_V}. \tag{15}$$

Furthermore, we calculate

$$\beta_i^{V*} = \max\left(\sqrt{\frac{R_V \sum b_{-i}^V \beta_{-i}^V}{b_i^V c_i^V}} - \frac{\sum b_{-i}^V \beta_{-i}^V}{b_i^V}, 0\right) \tag{16}$$

where $\max(\cdot)$ is a function to get the maximal value among input parameters. β_i^{V*} is mainly influenced by the total reward, individual serviceability and strategies of other PVs. The PV would like to accept workloads by clarifying $\beta_i^{V*} > 0$.

We temporarily assume that each PV would like to participate in the offloading service. Summing up (15) over all the PVs, we get

$$\sum_{i\in\mathcal{I}} b_i^V \beta_i^{V*} = \frac{|\mathcal{I}| - 1}{\sum_{i\in\mathcal{I}} \frac{c_i^V}{b_i^V}} R_V \tag{17}$$

where $|\mathcal{I}|$ indicates the number of members belonging to $\mathcal{I}$. We substitute $\sum_{i\in\mathcal{I}} b_i^V \beta_i^{V*}$ into (15) and get

$$\beta_i^{V*} = \frac{(|\mathcal{I}|-1)R_V}{b_i^V \sum_{i\in\mathcal{I}} \frac{c_i^V}{b_i^V}} - \frac{(|\mathcal{I}|-1)^2 c_i^V R_V}{(b_i^V \sum_{i\in\mathcal{I}} \frac{c_i^V}{b_i^V})^2}. \tag{18}$$

Next, we try to seek a subset $\mathcal{J} \subseteq \mathcal{I}$ and each PV in the subset accepts workloads, namely, $\beta_i^{V*} > 0$. To this end, the subset $\mathcal{J}$ meets the following constraint:

$$\frac{c_i^V}{b_i^V} \le \frac{\sum_{i\in\mathcal{J}} \frac{c_i^V}{b_i^V}}{|\mathcal{J}|-1}, \quad \forall i \in \mathcal{J}. \tag{19}$$

Finally, the best response of each PV in the set $\mathcal{I}$ is summarized as follows:

$$\beta_i^{V*} = \begin{cases} \frac{(|\mathcal{J}|-1)R_V}{b_i^V \sum_{i\in\mathcal{J}} \frac{c_i^V}{b_i^V}} - \frac{(|\mathcal{J}|-1)^2 c_i^V R_V}{(b_i^V \sum_{i\in\mathcal{J}} \frac{c_i^V}{b_i^V})^2}, & i \in \mathcal{J} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

As proved by [29], the above best response of each PV achieves the unique Nash equilibrium in the subgame. We also know that $\sum_{i\in\mathcal{J}} \beta_i^{V*} = \phi R_V$, where

$$\phi = \sum_{i\in\mathcal{J}} \psi_i^V = \sum_{i\in\mathcal{J}} \left[\frac{(|\mathcal{J}|-1)}{b_i^V \sum_{i\in\mathcal{J}} \frac{c_i^V}{b_i^V}} - \frac{(|\mathcal{J}|-1)^2 c_i^V}{(b_i^V \sum_{i\in\mathcal{J}} \frac{c_i^V}{b_i^V})^2} \right] \tag{21}$$

When realizing all the parameters b_i^V and c_i^V, the agent estimates $\mathcal{J}$ by using an iterative algorithm presented in [29]. In this paper, the agent proactively evaluates b_i^V, and requires each PV to upload c_i^V for calculating $\mathcal{J}$ and ϕ.

C. Stackelberg Equilibrium

The agent realizes the task requirements. For performer selection, the performers are also required to write basic information about task execution into the smart contract before receiving workloads. The basic information refers to $infor\_2$ in a transaction. For the fog server, $infor\_2$ includes l_s^V, c_s^V and n_s. For a PV, $infor\_2$ includes F_i, l_i^V and c_i^V. By accessing to the smart contract, the agent knows the prior knowledge, and can estimate the optimal strategies of each PV and the fog server. After that, the original problem in (11) is transformed as

$$\begin{aligned} \min_{p_s^V, R_V, i\in\mathcal{J}} \ & \frac{(p_s^V)^2 W_V - c_s^V p_s^V}{2n_s W_V} + R_V \\ \text{s.t.} \ & p_s^{V,\min} \le p_s^V \le p_s^{V,\max} \\ & 0 \le R_V \le R_V^{\max} \\ & \frac{p_s^V}{2n_s W_V} + \phi R_V = 1 + \frac{c_s^V}{2n_s W_V^2} \end{aligned} \tag{22}$$

where $p_s^{V,\min}$, $p_s^{V,\max}$ and $R_V^{\max}$ are calculated by combining the constraints in problem (11) with (9), (10), (13) and $\sum_{i\in\mathcal{J}} \beta_i^{V*} = \phi R_V$. Thus, we know

$$p_s^{V,\min} = \frac{c_s^V}{W_V} \tag{23}$$

$$p_s^{V,\max} = \frac{2n_s W_V (L_V - d_{\text{abp}})}{l_s^V} + \frac{c_s^V}{W_V} \tag{24}$$

$$R_V^{\max} = \frac{(L_V - d_{\text{abp}})}{\max\left(\left\{\psi_i^V l_i^V\right\}_{i\in\mathcal{J}}\right)}. \tag{25}$$

Based on the equality constraint, we easily transform the goal function of the above problem $\mathcal{G}$ in terms of p_s^V and have

$$\begin{aligned} \frac{\partial \mathcal{G}}{\partial p_s^V} &= \frac{p_s^V}{n_s} - \frac{c_s^V}{2n_s W_V} - \frac{1}{2\phi n_s W_V} \\ \frac{\partial^2 \mathcal{G}}{\partial p_s^{V2}} &= \frac{1}{n_s} > 0. \end{aligned} \tag{26}$$

This means that $\mathcal{G}$ is a convex function of p_s^V and has linear constraints. We can utilize existing methods to cope with such a typical convex optimization problem. With the help of the first-order optimality condition $\frac{\partial \mathcal{G}}{\partial p_s^V} = 0$, we get

$$p_s^{V*} = \frac{c_s^V}{2W_V} + \frac{1}{2\phi W_V}. \tag{27}$$

Considering all the constraints in the problem, $p_s^{V,\min}$ is updated by

$$p_s^{V,\min} = \min\left(\frac{c_s^V}{W_V},\ 2n_s W_V + \frac{c_s^V}{W_V} - 2n_s W_V \phi R_V^{\max}\right)$$

where $\min(\cdot)$ is a function to get the minimal value among input parameters. Hence, we acquire the final solution of p_s^{V*}:

$$p_s^{V*} = \begin{cases} p_s^{V,\min}, & \frac{c_s^V}{2W_V} + \frac{1}{2\phi W_V} \le p_s^{V,\min} \\ \frac{c_s^V}{2W_V} + \frac{1}{2\phi W_V}, & p_s^{V,\min} < \frac{c_s^V}{2W_V} + \frac{1}{2\phi W_V} \le p_s^{V,\max} \\ p_s^{V,\max}, & \frac{c_s^V}{2W_V} + \frac{1}{2\phi W_V} > p_s^{V,\max}. \end{cases} \tag{28}$$

After that, R_V^* is calculated by

$$R_V^* = \frac{1}{\phi} + \frac{c_s^V}{2n_s \phi W_V^2} - \frac{p_s^{V*}}{2n_s \phi W_V}. \tag{29}$$

When the agent has global knowledge of the performers recorded in the smart contract, the simplified optimization problem is formulated. Ultimately, there exists a unique solution (p_s^{V*}, R_V^*) to be solved as the best response of the requesting vehicle based on the prediction of α_s^{V*} and $\beta_i^{V*}, i \in \mathcal{J}$. At this time, the unique Stackelberg equilibrium is reached when all the players, including the fog server, each PV and requesting vehicle, have their optimized payoff and cost, respectively, considering the strategies chosen by other players in the game. They have no incentives to change the decisions and take other actions. The solution (p_s^{V*}, R_V^*) is transmitted to the client of the requester as suggestion. Then the requester releases the predetermined reward policy to the fog server and PVs accordingly. As predicated, they response with α_s^{V*} and β_i^{V*}, and write the decisions into the smart contract.

To efficiently release the rewards R_V^* to the PVs, the agent could utilize the following distributed algorithm to achieve the Nash equilibrium of the non-cooperative subgame in an iterative way, owing to mutual interaction with each PV. First, the agent publishes the task requirements, serviceabilities of current PVs and total reward to the PVs. Algorithm 1 is executed iteratively and converges within a limited number of iteractions. In one iteration, each PV makes the decision according to (16) after knowing current strategies of the other PVs from the agent. All the decisions are collected to the agent. In the next iteration, each PV is notified with the new

strategies of the others, and updates the decision and uploads it when necessary. The same interactive procedure between the agent and all the PVs (as illustrated by line 3 to 7) continues until the decision of each PV is changeless within a specific range ζ. Finally, the Nash equilibrium is approximately achieved and the approximate accuracy is determined by a precision threshold ζ, which also influences the number of iterations.

Algorithm 1. Distributed algorithm to solve β_i^{V*}

Input: Given R_V^* and $\{c_i^V, b_i^V\}_{i\in\mathcal{I}}$.
Output: $\{\beta_i^{V*,k}\}$.
1 Initialization: Precision threshold ζ, $k=0$ and each $\beta_i^{V*,0}$ is randomly assigned with a small value.
2 repeat
3 Broadcast $\sum b_{-i}^V \beta_{-i}^V$ to each PV.
4 **for** PV $i \in \mathcal{I}$ **do**
5 Update $\beta_i^{V*,k+1}$ according to (16) and upload it to the agent.
6 **end**
7 Goto the next round, $k = k+1$.
8 until $\left|\beta_i^{V*,k} - \beta_i^{V*,k-1}\right| \le \varsigma$;
9 final;
10 return $\{\beta_i^{V*,k}\}$

VI. Security Analysis and Discussion

A. Security Analysis for PVFChain

In this paper, we introduce PVFChain to guarantee network security and efficiency for PVFC by using blockchain with an optimal smart contract design. Smart contract operations are proposed to record relevant information and evaluate behaviors of requesters and performers in computation offloading. Security and privacy requirements can be satisfied. First, all the network entities should be registered with valid identities, and use them to maintain authenticated and secure communications. This preserves a certain level of individual privacy by permitting users to register with issued key pairs instead of revealing true identity. Besides, tasks are automatically released, executed, evaluated and awarded via the smart contract execution, and to generate committed transactions recorded on the blockchain, achieving immutability, auditability and transparency of implementing decentralized offloading services. Owing to contract enforcement, network behaviors become consistent and non-repudiable in the nonfully trusted environment. The great advantages of PVFChain are summarized as follows.

1) Identity Authentication: Each identity is strictly authenticated to avoid illegal entities. In the meantime, the true identities of all the requesters and performers still remain unknown. This ensures authenticity and anonymity to remove impersonation attacks and avoid privacy leakage during the service provision.

2) Request Validation: Every request is submitted with a deposit, eliminating false-reporting and free-riding behaviors of malicious requesters. By checking the attached signature, the malicious requesters are easier to be identified and punished. The utilization of deposit prevents the malicious requesters from flooding the entire network with forged requests.

3) Computation Verification: Honest performers are recognized when their computation work is successfully accepted. Output results uploaded by them should be proved to satisfy accuracy performance. By using the deposit mechanism, the proposed scheme also guards against spoofing attacks from malicious performers that may offer misleading results to procrastinate the procedure of output result aggregation.

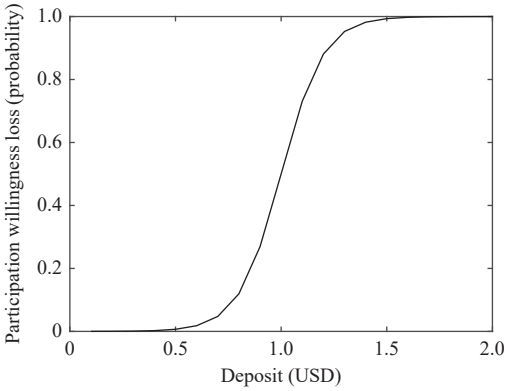
4) Reward Integrity: Tamper-resistant transaction records guarantee the process integrity of reward assignment. So reward assignment from the requesters to performers becomes untampered, accountable and traceable. Both of them should realize and follow the promising reward policy according to the contract-based agreement. In other words, they have no chances to launch reward repudiation of giving and reception. Free-riding behaviors and greedy charge are overcome.

B. Discussion About the Deposit Mechanism

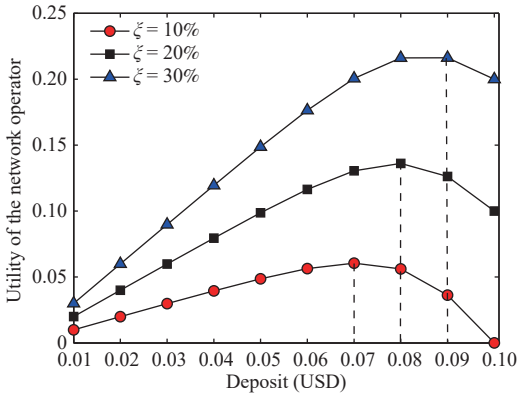
As Ethereum is an open and programmable Blockchain platform, we develop the programmable smart contracts by using the specially designed blockchain virtual machine, Ethereum virtual machine (EVM), which provides an isolated runtime environment for handling the computation and state of smart contracts in Ethereum. After calling an estimateGas API, we know the sum of all the gas for the executed operations required by a designed smart contract is about 374 200. As measured by the work in [30], when miners employ fog servers for mining and we set that there are only 10 miners whose fog servers are widely connected, the average delay of verifying a block can be small to multiple milliseconds. To estimate the average block propagation delay d_{abp}, the average data size of blocks $\lambda = 20$ kb, the transmission rate between any two miners $\pi = 10$ Mbps and two coefficients $m_1 = m_2 = 0.1$. Then $d_{\text{abp}} \approx 20$ milliseconds.

To eliminate malicious performers, deposit mechanism is used in PVFChain. If the malicious performers are not able to submit correct output results in time, the preserved deposits are deducted by the background system for specific purposes, e.g, compensating for requesters with poor user experience. Clearly, a larger deposit causes more financial losses to the malicious performers. The strict deposit mechanism is effective to remove security risks resulted by the malicious performers. However, participation willingness of normal performers, in terms of probability to accept the recruitment in computation offloading, may also be influenced by the strict deposit mechanism. For a performer, the participation willingness of undertaking workloads is roughly decreased with an increasing deposit. Here, we use the sigmoid function to model various participation willingness losses for a normal performer with respect to different deposits, as shown in Fig. 3(a). Let θ represent the deposit, the participation willingness loss in probability $\mathcal{L}$ is evaluated by

$$\mathcal{L} = \frac{1}{1+\exp\left(-\dfrac{\theta-\sigma}{\eta}\right)} \tag{30}$$



(a) Comparison of participation willingness loss $\mathcal{L}$ with respect to different θ



(b) Comparison of system utility $\mathcal{U}$ with respect to different θ and ξ

Fig. 3. Discussion about the tradeoff in the deposit setting.

where σ and η are two constants. Here, we set $\sigma = 0.1$ and $\eta = 0.01$. From the figure, when $\theta = 0.2$ USD, few performers are attracted to join the offloading service because the participation willingness loss approximates to 100%.

For a network operator, there exists a tradeoff problem when determining the deposit θ to achieve an acceptable tradeoff between penalty acquisition from malicious performers and overall participation willingness losses caused to normal performers. Considering there exist N performers and the ratio of malicious performers is ξ, due to the participation willingness loss, the utility of the network operator in the deposit mechanism is expressed by

$$\mathcal{U} = N\xi\theta - \gamma N(1-\xi)\mathcal{L} \tag{31}$$

where γ is a tradeoff parameter. In Fig. 3(b), we set $N = 10$ and $\gamma = 0.2$, and realize that an optimal strategy of θ (marked by the dash line) is always achieved for the network operator to maximize the utility under different conditions of ξ. In general, when there exist more malicious performers, namely, ξ is predicated with a higher value, it is a better choice of the network operator to require a larger deposit. In the following simulations, we set $\theta = 0.07$ USD as $\xi = 10\%$.

VII. Numerical Results

A. Performance Comparison of Different Schemes

We evaluate the performance of the proposed scheme for PVFChain by extensive simulations. We consider a scenario where a requesting vehicle V sends a computation task to an agent which employs a fog server s and several PVs $i \in \mathcal{I}$ in a parking lot for jointly processing the workloads. The power consumption models of the fog server and each PV in CPU execution refer to [25] and [14], respectively. Within the coverage of a local roadside unit, nearby PVs use the downlink channel model in [31] to receive the input data. Particularly, parking behaviors are formulated according to a real dataset from ACT Government Open Data Portal dataACT. The real dataset uses the SmartParking app to collect 180295 parking records in the Manuka shopping precinct within a month [32]. More simulation parameters are found in Table I.

In the simulations, we select about 60 000 PVs in the dataset for an observation, and make data statistic and analysis on their parking behaviors. To summarize, the parking duration of a PV ranges from 8 to 240 min. Based on the statistic data, we draw the frequency distribution histogram about the parking durations of all the PVs in the dataset, as shown in Fig. 4(a). Here, the class interval for grouping statistics is 10 min. Furthermore, we use the professional curve fitting tool provided by the MATLAB software, cftool, to fit and get the probability density function (PDF) of the parking durations of these PVs. The fitting PDF is illustrated by the black curve in the figure.

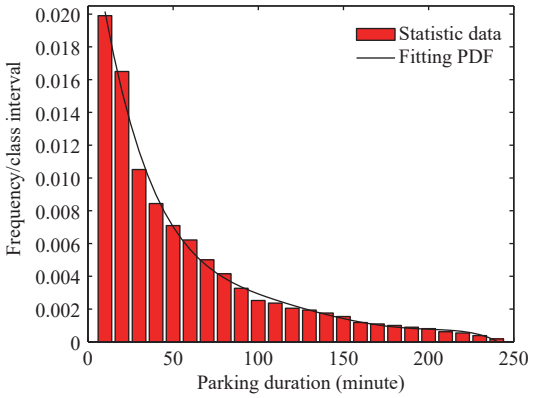
According to the fitting PDF, we measure the serviceability of PV i for task execution in a given task scheduling time period. T indicates the time span of the time period. Based on (4), the serviceability of a PV is equal to the weight sum of the predicted probability of staying continuously within the time span T and relative value of the computing capability.

As shown in Fig. 4(b), the serviceability of PV i is decreased with the extending time span T, accumulative parking durations ap_i and reduced computing capability F_i. First, the serviceability of a PV is decreased with the expanding time window. The increasing value of T means that the PV is required to stay continuously within longer time durations, decreasing the estimated value of P_i. Besides, those PVs that have been parked with longer time durations are exactly inadequate for being selected as performers. This is because that for them, the probability of a sudden departure is increased and thus, this leads to a lower serviceability. For example, within the 45 min time window, the evaluated serviceability of PV i that has been parked with 50 min will be reduced about 7% when ap_i is increased to 100 min. At last, the serviceability can be clearly improved with the increasing computing capability F_i. In the following simulations, the task scheduling time period is set 30 min. The PVs with the serviceability higher than 0.75 are deserved to be employed as reliable and efficient performers.

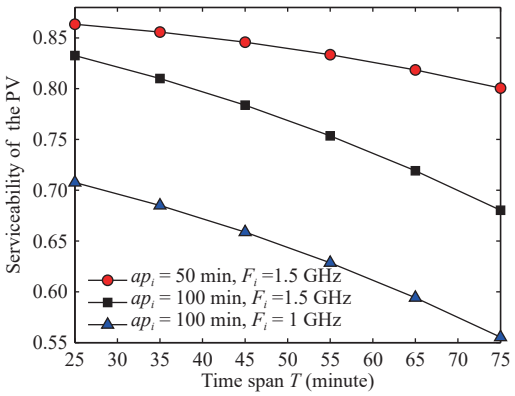
To show great advantages of our scheme, we compare the performances between PVFChain and three baseline schemes. Current VFC schemes only schedule the fog server task execution (e.g., [33]), or PVs (e.g., [1]) for task execution. Moreover, we introduce a baseline PVFC scheme where the uniform reward policy is published to both the fog server and

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Computing capability of the fog server f_s^V	5 GHz
Power consumption parameters of the fog server $[A_1, A_2, \tau]$	$[3.2, 68, 3]$
Receiver power and downlink data rate of the fog server ρ_s^{rx} and r_s^{dl}	0.4 W, 10 Mbps
Status-related parameter of the fog server n_s	0.5
Parking behavior: $g(t), t \in [0, t^{\max}]$	From the dataset [32]
Weighting and threshold values for PV selection ϖ, F_{th} and b_{th}	0.5, 2 GHz and 0.75
Computing capability of PV i f_i^V	[0.5, 2] GHz
Receiver power of PV i ρ_i^{rx}	[30, 33] dBm
Distance between PVi and the local roadside unit d_i	[50, 350] meter
Transmitting power of the roadside unit ρ_{rsu}^{tx}	33 dBm
Hardware parameter of PV i in power consumption of CPU execution κ_i	10^{-26}
Downlink channel model parameters of PV i $[B_i, g_i, \varepsilon, N_0]$	[1,2] MHz, 1/47.86 dBm, 2.75, –95 dBm/Hz
Expenditure for consuming per unit energy p_e	0.1 USD/Joule
Precision threshold ζ	10^{-3}



(a) Frequency distribution histogram about parking duration of one PV in the dataset



(b) Comparison of serviceability of a PV with respect to different T, ap and F

Fig. 4. Observations about PVs in the dataset.

all the participating PVs for fairness. More specifically, the whole workloads are allocated to all the PVs belonging to the set $\mathcal{I}$ and the fog server in parallel and distributed computing environment within the latency requirement. According to (13), p_s^{V*} is calculated to stimulate the fog server for undertaking the given rewards. After that, the total payment of the requester is $p_s^{V*}W_V$ as the same reward policy is released to each participating PV. Here, to describe the offloading request, the computation task of requester V is represented by D_V^{in} = 5000 KB, h_V = 0.2 megacycle per KB and L_V = 200 ms.

As shown in Table II, the task is executed at a considerate cost, to meet the latency requirement in our scheme. In the fog server based VFC, the total payment is rapidly increased since the whole workloads are executed by the fog server with the higher expense. Finally, the total payment of the scheme is 64% more than that of our scheme. In this paper, we combine idle resources from PVs with current resources in the fog server to achieve parallel computing. The execution time is decreased to satisfy the latency requirement. Moreover, after optimizing workload allocation among different performers, the total payment of the requester is further minimized. The smallest execution time is clearly acquired in the introduced scheme with the uniform reward policy. This is because without any competition, all the PVs are arbitrarily permitted to participate in the offloading service given the uniform reward policy while a subset of the PVs would like to undertake workloads in the competitive environment of our scheme. Nevertheless, compared with the introduced baseline scheme, our scheme still causes less monetary cost, decreasing about 35%, owing to diverse reward setting for different performers.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT SCHEMES

Scheme	Total payment (USD)	Execution time (ms)
Our scheme	5.84	180
Fog server based VFC [33]	10.36	204
PV based VFC [1]	1.98	1948
PVFC with uniform reward policy	9.05	141

We also realize that the latency requirement cannot be satisfied as the whole task is entirely assigned to all the

participating PVs, by letting $\sum\beta_i^V = 1$. The execution time resulted in PV based VFC is much longer, and approximates to 2 s, which cannot be exactly tolerated in compute-intensive applications. This means that compared with existing schemes, our scheme fully meets the payment minimization goal and latency constraint.

B. Impacts to the Stackelberg Game

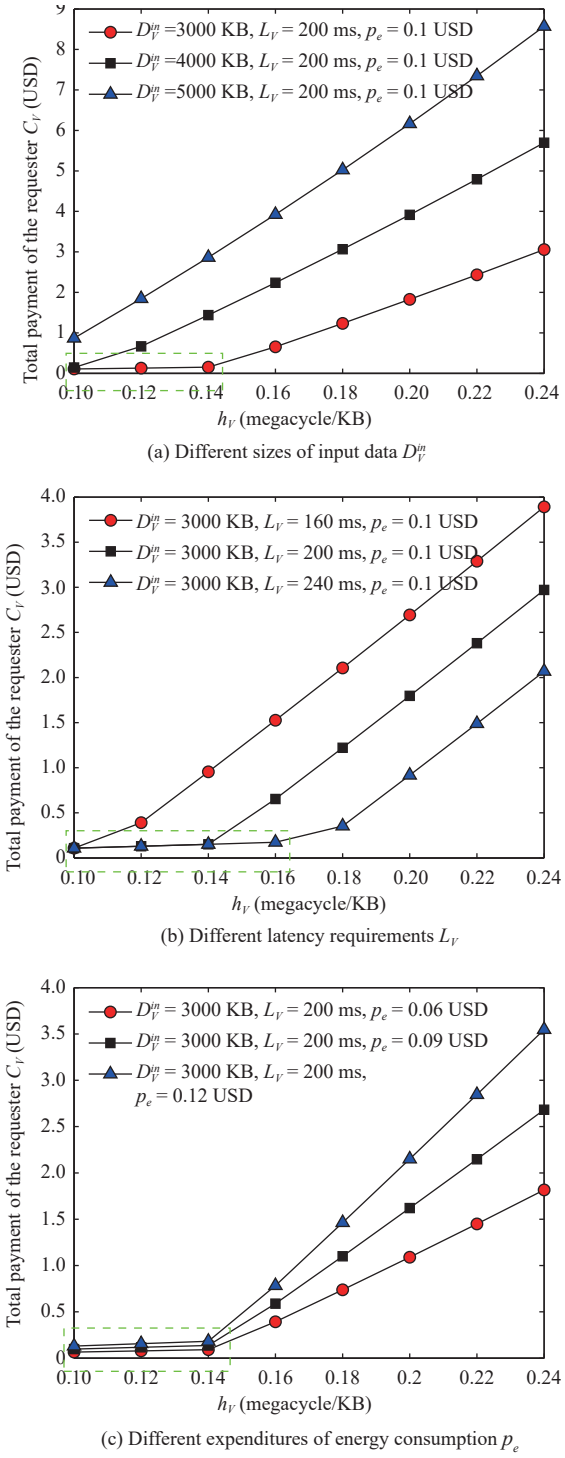
Next, the impacts of different system parameters for the best responses of a requester V, fog server s and each PV i in the Stackelberg game are discussed.

The total payment C_V is influenced by the following parameters: application-centric parameter h_V, size of input data D_V^{in}, latency requirement L_V and expenditure of energy consumption p_e. Clearly, with the increasing values of h_V and D_V^{in}, the requester pays more for stimulating the performers to process more workloads. So C_V is directly increased with the higher values of h_V and D_V^{in} in Fig. 5(a). In particular, when the whole workloads are small or the latency requirement is relaxed enough, for example, when $h_V = 0.1$, $D_V^{in} = 3000$ KB and $L_V = 200$ ms, all the workloads are directly undertaken by the PVs. At this time, the total payment is reasonably lower, due to the lightweight power consumption of CPU execution in PVs. Meanwhile, increasing workloads only lead to the minor increment of C_V in the scenario, as shown in the green dotted rectangles in Figs. 5(a), 5(b) and 5(c).

But once a part of workloads need to be allocated to the fog server, the obvious increment of C_V is caused. For example, given the same values of $h_V = 0.2$, $L_V = 200$ ms and $p_e = 0.1$ USD, the total payment is increased about 57% when more workloads appear since the size of input data is changed from 4000 to 5000 KB. To satisfy the same latency requirement, the workloads allocated to the fog server are increased. This directly leads to higher payments. In turn, the relaxed latency requirement results in less workloads assigned to the fog server. This gives rise to the reduction of the total payment, as shown in Fig. 6(b). When $h_V = 0.2$, $D_V^{in} = 3000$ KB and $p_e = 0.1$ USD, the total payment is declined about 66% once L_V is extended from 160 to 240 ms. Furthermore, for task execution, the higher expenditure of energy consumption p_e directly makes the increment of C_V. When the fixed cost about task execution is increased, more monetary cost is exactly necessary to incentivize the performers for undertaking the same workloads, as illustrated in Fig. 5(c).

Various best responses of the fog server with respect to different parameters are shown in Fig. 6. We take a computation task with specific parameters $h_V = 0.2$ and $D_V^{in} = 3000$ KB as an example and let $p_e = 0.1$ USD. As shown in Fig. 6(a), the optimal rewards for employing the fog server to process per workload p_s^{V*} is related to computing capability of f_s^V. According to (1), with the improvement of f_s^V, power consumption of CPU execution in the fog server is increased. As a consequence, more energy is consumed for conducting workloads. A higher value of p_s^{V*} is then required to offer sufficient monetary rewards for the fog server. Besides, Fig. 6(b) illustrates that when the latency requirement L_V is decreased, the percentage of the workloads assigned to the fog server α_s^{V*} is reduced accordingly. Due to the prolonged latency requirement, more workloads can be allocated to the PVs with lower computing capability and energy consumption, aiming to minimize the total payment. Then α_s^{V*} is decreased to about 5% when the tolerated latency becomes 420 ms.

On the other hand, α_s^{V*} is influenced by the increasing computing capability of the PVs. When the average

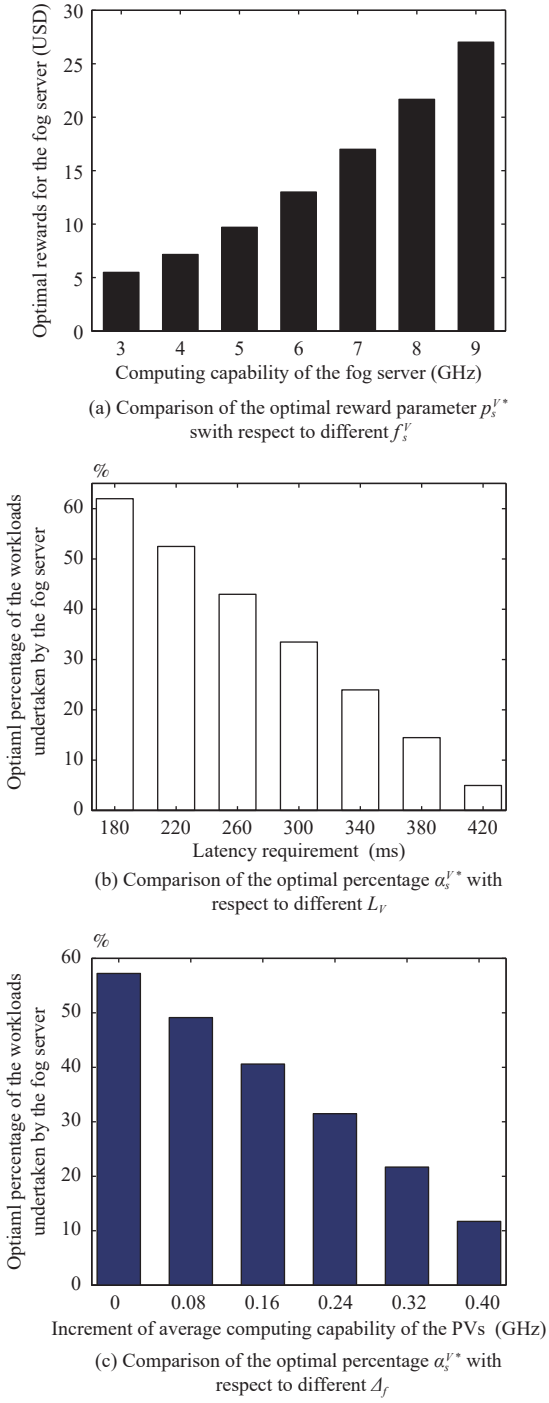


(a) Different sizes of input data D_V^{in}

(b) Different latency requirements L_V

(c) Different expenditures of energy consumption p_e

Fig. 5. Comparison of total payment of the requester C_V with respect to different h_V, D_V^{in}, L_V and p_e.



(a) Comparison of the optimal reward parameter p_s^{V*} swith respect to different f_s^V

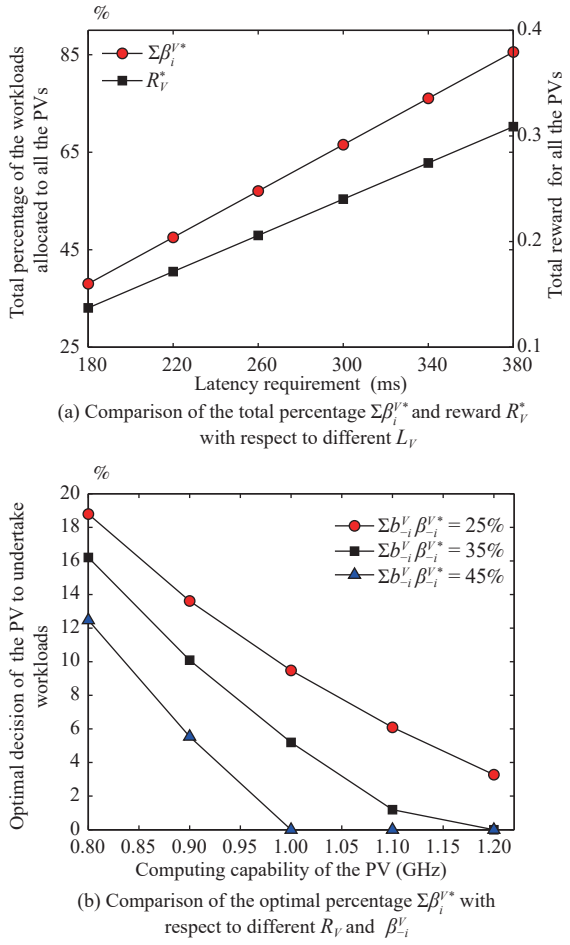
(b) Comparison of the optimal percentage α_s^{V*} with respect to different L_V

(c) Comparison of the optimal percentage α_s^{V*} with respect to different Δ_f

Fig. 6. Comparison of the best response of the fog server with respect to different f_s^V, L_V and Δf.

computing capability of all the PVs is improved, more workloads are handled by them within the same latency requirement. Thus, $\sum\beta_i^{V*}$ is increased, and this decreases α_s^{V*}. We denote the increment of the average computing capability of all the participating PVs as Δf and let $L_V = 200$ ms. In Fig. 6(c), we find that when $\Delta f = 0.32$ GHz, α_s^{V*} is changed from 51.7% to 21.7%. The decline in percentage is about 62%.

At last, we pay attention to the best strategies of the PVs under different conditions. First, as mentioned above, the total percentage of the workloads undertaken by all the PVs $\sum\beta_i^{V*}$ is increased with the prolonged latency requirement L_V. For payment minimization, the agent assigns most of the workloads to the PVs, meanwhile, the total reward R_V is improved for encouraging them to undertake more workloads, as indicated by Fig. 7(a).



(a) Comparison of the total percentage $\Sigma\beta_i^{V*}$ and reward R_V^* with respect to different L_V

(b) Comparison of the optimal percentage $\Sigma\beta_i^{V*}$ with respect to different R_V and β_{-i}^V

Fig. 7. Comparison of the best responses of the PVs with respect to different L_V, R_V and $\sum\beta_{-i}^{V*}$.

We choose PV i for further observations. Based on (16), the best response of PV i in the non-cooperative subgame β_i^{V*} is influenced by the following parameters: individual computing capability f_i^V and the strategies of other PVs $\sum b_{-i}^V \beta_{-i}^{V*}$. In Fig. 7(b), the higher value of f_i^V means that the power consumption of CPU execution is larger and the energy consumption cost is subsequently increased when processing given workloads. As a result, facing with the same reward policy, the PV chooses to undertake less workloads or rejects to undertaking workloads for utility maximization. In addition, when realizing that other PVs accept less workloads, namely, $\sum b_{-i}^V \beta_{-i}^{V*}$ is known with a lower value, the PV would like to accept more workloads for earn more economic benefits in the competitive environment. So β_i^{V*} is increased with rationality. For example, when total reward $R_V = 0.3$ and $f_i^V = 1$ GHz, β_i^{V*} is increased more than 82% as $\sum\beta_{-i}^{V*}$ is decreased from 35% to 25%.

VIII. Conclusions

Towards secure PVFC, we propose PVFChain with an optimal smart contract design to conduct computation offloading in a totally decentralized way, finally improving network security and efficiency. Request posting, workload undertaking, task evaluation and reward assignment are organized and validated automatically through the smart contract execution. To implement PVFChain, we propose the crucial network entities and consider smart contract operations across them. Moreover, we study the optimal smart contract design problem, which is formulated as a payment minimization problem and solved within the Stackelberg game framework, to employ a fog server and several PVs as performers for jointly processing workloads. The requester (leader) optimizes the reward policy for performers (followers) to improve user satisfaction while also considering utility maximization for them. Security analysis is provided to demonstrate that PVFChain maintains anonymous interaction, guards against flooding and spoofing attacks and ensures accountable reward assignment. Numerical results also indicate that the Stackelberg game approach is efficient and effective to maximize user satisfaction from an economic viewpoint.

In the future, we pay attention to the combination of PVFChain and social transportation system by adding and coping with the human and social dimension into offloading service scenario, as proposed by [34]. Privacy sensitivity of requesters and social ties among various PVs as local performers could be studied. Based on necessary social information, the decision-making process is further performed in a comprehensive way.

References

[1] X. S. Hou, Y. Li, M. Chen, D. Wu, D. P. Jin, and S. Chen, "Vehicular fog computing: a viewpoint of vehicles as the infrastructures," *IEEE Trans. Vehicular Technology*, vol. 65, pp. 3860–3873, Jun. 2016.

[2] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 4, pp. 926–937, 2019.

[3] Z. L. Ning, J. Huang, and X. J. Wang, "Vehicular fog computing: enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, Feb. 2019.

[4] Y. Zhang, C.-Y. Wang, and H.-Y. Wei, "Parking reservation auction for parked vehicle assistance in vehicular fog computing," *IEEE Trans. Vehicular Technology*, vol. 68, no. 4, pp. 3126–3139, Apr. 2019.

[5] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. L. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things J.*, vol. 4, pp. 1832–1843, Dec. 2017.

[6] L. Li, J. Q. Liu, L. C. Cheng, S. Qiu, W. Wang, X. L. Zhang, and Z. H. Zhang, "Creditcoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Trans. Intelligent Transportation Systems*, vol. 19, pp. 2204–2220, Jul. 2018.

[7] J. W. Kang, R. Yu, X. M. Huang, M. Q. Wu, S. Maharjan, S. L. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things J.*, vol. 6, pp. 4660–4670, Jun. 2019.

[8] X. Y. Kui, Y. Sun, S. G. Zhang, and Y. Li, "Characterizing the capability of vehicular fog computing in large-scale urban environment," *Mobile Networks and Applications*, vol. 23, pp. 1050–1067, Aug. 2018.

[9] C. Huang, R. X. Lu, and K.-K. R. Choo, "Vehicular fog computing: architecture, use case, and security and forensic challenges," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[10] Z. L. Ning, P. R. Dong, X. J. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: an intelligent offloading system," *ACM Trans. Intelligent Systems and Technology*, vol. 10, no. 6, pp. 60, May 2019.

[11] X. J. Wang, Z. L. Ning, and L. Wang, "Offloading in internet of vehicles: a fog-enabled real-time traffic management system," *IEEE Trans. Industrial Informatics*, vol. 14, pp. 4568–4578, Oct. 2018.

[12] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. S. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.

[13] X. M. Huang, R. Yu, J. Q. Liu, and L. Shu, "Parked vehicle edge computing: exploiting opportunistic resources for distributed mobile applications," *IEEE Access*, vol. 6, pp. 66649–66663, 2018.

[14] Y. T. Wang, M. Sheng, X. J. Wang, L. Wang, W. J. Han, Y. Zhang, and Y. Shi, "Energy-optimal partial computation offloading using dynamic voltage scaling," in *Proc. IEEE Int. Conf. Communication Workshop* (*ICCW*), pp. 2695–2700, Jun. 2015.

[15] Z. J. Lu, W. C. Liu, Q. Wang, G. Qu, and Z. L. Liu, "A privacy-preserving trust model based on blockchain for vanets," *IEEE Access*,, vol. 6, pp. 45655–45664, 2018.

[16] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchainbased decentralized trust management in vehicular networks," *IEEE Internet of Things J.*, vol. 6, pp. 1495–1505, Apr. 2019.

[17] W. C. Xu, H. B. Zhou, N. Cheng, F. Lyu, W. S. Shi, J. Y. Chen, and X. M. Shen, "Internet of vehicles in big data era," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2017.

[18] Z. H. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, pp. 33–39, Aug. 2018.

[19] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[20] S. Y. Zhao, V. Lo, and C. G. Dickey, "Result verification and trust-based scheduling in peer-to-peer grids," in *Proc. 15th IEEE Int. Conf. Peer-to-Peer Computing* (*P2P'05*), pp. 31–38, IEEE, Jan. 2005.

[21] M. Walfish and A. J. Blumberg, "Verifying computations without reexecuting them," *Communications of the ACM*, vol. 58, no. 2, pp. 74–84, 2015.

[22] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: blind motion deblurring using conditional adversarial networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 8183–8192, 2018.

[23] M. Chen and Y. X. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[24] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Vehicular Technology*, vol. 64, pp. 4738–4755, Oct. 2015.

[25] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of internet data centers under multiregional electricity markets," *Proc. the IEEE*, vol. 100, pp. 269–282, Jan. 2012.

[26] X. M. Wang, X. M. Chen, W. W. Wu, N. An, and L. S. Wang, "Cooperative application execution in mobile cloud computing: a stackelberg game approach," *IEEE Communications Letters*, vol. 20, pp. 946–949, May 2016.

[27] W. W. Zhang, Y. G. Wen, K. Guan, D. Kilper, H. Y. Luo, and D. O. Wu, "Energyoptimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Communications*, vol. 12, pp. 4569–

4581, Sep. 2013.

[28] X. J. Liu, W. B. Wang, D. Niyato, N. Zhao, and P. Wang, "Evolutionary game for mining pool selection in blockchain networks," *IEEE Wireless Communications Letters*, vol. 7, pp. 760–763, Oct. 2018.

[29] D. J. Yang, G. L. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 1732–1744, Jun. 2016.

[30] J. L. Pan, J. Y. Wang, A. Hester, I. Alqerm, Y. N. Liu, and Y. Zhao, "Edgechain: an edge-iot framework and prototype based on blockchain and smart contracts," *IEEE Internet of Things J.*, vol. 6, pp. 4719–4732, Jun. 2019.

[31] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the performance of IEEE 802.11p and LTE-V2V for the cooperative awareness of connected vehicles," *IEEE Trans. Vehicular Technology*, vol. 66, pp. 10419–10432, Nov. 2017.

[32] ACT Government Open Data Portal dataACT. [Online]. Available: https://www.data.act.gov.au/Transport/SmartParking-History/grth-myzs, 2017.

[33] X. L. He, Z. Y. Ren, C. H. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles," *China Communications*, vol. 13, pp. 140–149, Nov. 2016.

[34] G. Xiong, F. H. Zhu, X. W. Liu, X. S. Dong, W. L. Huang, S. H. Chen, and K. Zhao, "Cyber-physical-social system in intelligent transportation," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 3, pp. 320–333, 2015.

Xumin Huang (M'19) received the Ph.D. degree from Guangdong University of Technology, in 2019. He is currently a postdoctoral with Guangdong University of Technology, China. His research interests include network performance analysis, simulation, and enhancement in wireless communications and networking.

Dongdong Ye received the M.S. degree in 2018 from Guangdong University of Technology, where he is currently working toward the Ph.D. degree. His research interests include game theory, resource management in wireless communications and networking, and data sharing and trading.

Rong Yu (M'08) received the B.S. degree in communication engineering from Beijing University of Posts and Telecommunications, in 2002 and Ph.D. degree in electronic engineering from Tsinghua University, in 2007, respectively. From 2007 to 2010, he worked in the School of Electronic and Information Engineering at South China University of Technology. In 2010, he joined the School of Automation at Guangdong University of Technology, where he is currently a Professor. His research interests include wireless networking and mobile computing such as mobile cloud, edge computing, deep learning, connected vehicles, smart grid and internet of things. He is the author or co-author of over 100 international journal and conference papers and the co-inventor of over 50 patents in China. He was a member of the Home Networking Standard Committee in China, where he led the standardization work of three standards.

Lei Shu (M'07–SM'15) received the B.Sc. degree in computer science from the South Central University for Nationalities, in 2002, the M.Sc. degree in computer engineering from Kyung Hee University, South Korea, in 2005, and the Ph.D. degree from the Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland, in 2010. Until 2012, he was a specially assigned researcher with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Japan. He is currently a Distinguished Professor with Nanjing Agricultural University, China, and a Lincoln Professor with the University of Lincoln, U.K. He is also the Director of the NAU-Lincoln Joint Research Center of Intelligent Engineering. He has published over 400 papers in related conferences, journals, and books in the area of sensor networks. His research interest include the wireless sensor networks and the Internet of Things. He has served as a TPC Member of more than 150 conferences, including ICDCS, DCOSS, MASS, ICC, Globecom, ICCCN, WCNC, and ISCC. He received the Globecom 2010, ICC 2013, ComManTel 2014, WICON 2016, and SigTelCom 2017 Best Paper Awards, the 2014 Top Level Talents in "Sailing Plan" of Guangdong, China, and the 2015 Outstanding Young Professor of Guangdong, the 2017 and 2018 IEEE Systems Journal Best Paper Awards, the 2017 Journal of Network and Computer Applications Best Research Paper Award, and the Outstanding Associate Editor Award of the 2017 IEEE Access. He has served as the CoChair for international conferences/workshops, more than 50 times, including the IWCMC, ICC, ISCC, ICNC, and Chinacom, especially the Symposium Co-Chair for IWCMC 2012 and ICC 2012, the General Co-Chair for the Chinacom 2014, Qshine 2015, Collaboratecom 2017, DependSys 2018, SCI 2019, and the TPC Chair for the InisCom 2015, NCCA 2015, WICON 2016, NCCA 2016, Chinacom 2017, InisCom 2017, WMNC 2017, and NCCA 2018. His H-index is 54 and i10-index is 197 in Google Scholar Citation. He has been serving as an Associate Editor for the *IEEE Transactions on Industrial Informatics*, *IEEE Communications Magazine*, *IEEE Network Magazine*, *IEEE Systems Journal*, *IEEE Access*, *IEEE/CAA Journal of Automatica Sinica*, *Sensors*, and so on.