Secure Computation Offloading in Blockchain based IoT Networks with Deep Reinforcement Learning

Dinh C. Nguyen, Pubudu N. Pathirana, *Senior Member, IEEE,* Ming Ding, *Senior Member, IEEE,* Aruna Seneviratne, *Senior Member, IEEE*

arXiv:1908.07466v1 [eess.SP] 15 Aug 2019

Abstract

In current Internet of Things (IoT) networks, mobile edge-cloud computation offloading (MECCO) has been regarded as a promising means to support delay-sensitive IoT applications. However, offloading mobile tasks to cloud is vulnerable to security risks due to malicious mobile devices (MDs). How to implement offloading to solve computation problems of MDs while guaranteeing high security in mobile cloud is challenging. In this paper, we investigate simultaneously the security and offloading problems in a multi-user MECCO system on mobile cloud blockchain. First, to improve offloading security, we propose a trustworthy access control using blockchain, which protects clouds against illegal offloading behaviours. Then, to tackle the intensive computation issues of authorized MDs, we formulate a computation offloading problem by jointly optimizing the offloading decisions and the allocation of computing resource and radio bandwidth. This optimization problem aims to minimize the long-term system costs of latency and energy consumption among all MDs. To solve the proposed offloading problem, we develop a novel deep reinforcement learning (DRL) algorithm by using advanced deep Q-network. We evaluate our framework towards access control and offloading performances by both real experiments and numerical simulations, showing significant advantages over existing schemes.

Index Terms

Blockchain, computation offloading, edge-cloud computing, deep reinforcement learning, security.

Dinh C. Nguyen and Pubudu N. Pathirana are with School of Engineering, Deakin University, Waurn Ponds, VIC 3216, Australia (e-mail: cdnguyen@deakin.edu.au, pubudu.pathirana@deakin.edu.au)

Ming Ding is with Data61, CSIRO, Australia (email: ming.ding@data61.csiro.au)

Aruna Seneviratne is with School of Electrical Engineering and Telecommunications, University of New South Wales (UNSW), NSW, Australia (email: a.seneviratne@unsw.edu.au)

I. INTRODUCTION

Recent years have witnessed the explosion of mobile technologies with the proliferation of Mobile Devices (MDs) such as smartphones, tablets, wearable devices, etc, which have been driving the evolution of Internet of Things (IoT) [1]. MDs can be used to run IoT applications such as smart home, smart healthcare, smart city with high flexibility and efficiency [2], [3]. However, due to the rapid growth of mobile data traffic, executing extensive IoT applications merely on overloaded MDs is incapable of providing satisfactory Quality of Services (QoS) to users. Fortunately, with recent communication technology advances, MDs now can offload the workload of IoT applications (or computation tasks) to a cloud server for execution to mitigate computing pressure on MDs [4]. To further improve efficiency of mobile computation, mobile edge computing (MEC) has emerged as a promising solution to enable MDs to offload their computation tasks to a nearby edge server [5], [6].

Particularly, the combination of cloud and edge computing can lead to a new paradigm to facilitate offloading computation for IoT networks with an unified model of mobile edge-cloud computation offloading (MECCO) [7]. MECCO offers application developers highly effective computing services in the mobile edge-cloud by obtaining advantages of both edge and cloud computing to satisfy diverse QoS requirements. Mobile applications without latency requirements (i.e. big data analysis) can be offloaded to the resourceful cloud, while the others (i.e. smart healthcare, smart home) with time-sensitive requirements can be executed at the edge server for fast-response services. Clearly, MECCO can offer promising solutions with high flexibility to achieve computation objectives for future 5G IoT [8], [9].

However, the MECCO also poses many challenges on computation offloading and one of the major challenges is security [10]. As mobile task offloading relies on MDs in a dynamic environment where mobile users are untrusted, the MEECO is vulnerable to various types of threats. Unauthorized MDs may gain malicious access to exploit computing cloud services without consent of central authority. Further, attackers can threaten computation resources on cloud to obtain mobile data, leading to privacy concerns of cloud-based IoT applications [11]. Therefore, how to guarantee security for mobile offloading is critical to any MECCO systems.

Recently, blockchain has emerged as a promising approach to tackle security issues in future IoT networks, including mobile offloading systems [12], [13]. The concept of blockchain is based on a peer-to-peer network architecture in which transaction information is distributed

among multiple nodes and not controlled by any single centralized entity. Blockchain with its decentralized and trustworthy nature has been integrated with cloud IoT systems for security guarantees such as secure access control and data management among IoT devices. Specially, smart contract [14], a self-operating computer program running on the blockchain platform has been demonstrated its feasibility in various IoT security problems. For example, smart contracts was proven its access control capacity in IoT networks, offering access verification and data audit [15]. Further, smart contracts could preserve cloud resources against malicious access [16]. Therefore, it is believed that blockchain and smart contracts can be applied in mobile cloud IoT, especially in MECCO systems, to fulfil security objectives for mobile task offloading.

A. Related work

In this subsection, we will give a brief overview about some related works in regards to blockchain-based access control and computation offloading approaches for MECCO systems.

1) Access control based security with blockchain: In mobile cloud IoT, blockchain can provide access control models for security. Specifically, in our recent work [15], a blockchain network architecture was proposed for data access control in mobile cloud IoT. An access control design using blockchain-based smart contract was implemented on a cloud platform for access right validation and authorization of mobile devices. Implementation results revealed various interesting properties of smart contract and blockchain in terms of access control capability, data protection and system integrity guarantees for IoT networks. The authors in [16], [17] also proposed access control models using blockchain-based smart contract to implement access right validation, providing an effective protection solution for IoT devices. Meanwhile, an access management system enabled by smart contracts was proposed in [18] for the industrial IoT networks on blockchain. On the same direction, the authors in [19] presented a blockchain based architecture for IoT access control and authorization. Based on achieved research results, blockchain can be extended to IoT access systems, like our proposed MECCO scenario, to achieve reliable task offloading on mobile cloud IoT networks [20], [21].

2) Computation offloading with mobile edge-cloud: Many works were proposed to investigate computation offloading issues with edge-cloud computing in IoT networks. Some offloading strategies in [22], [23], [24], [25] utilized edge or cloud services to tackle IoT computation issues with the objective of minimizing offloading costs and cloud resources by leveraging Lyapunov or conventional convex optimization methods. But such conventional offloading optimization

algorithms only work well for low-complexity online models and usually require prior knowledge of system statistics that is difficult to acquire in practical scenarios. To overcome such challenges, Reinforcement Learning (RL) has emerged as an efficient technique which allows a learning agent to adjust its policy and derive an optimal solution via trial and error to achieve the best long-term goal without requiring any prior environment information [26]. Nevertheless, in complex offloading problems with multi-user multi-IoT device scenarios, the dimension of state and action space can be extremely high that makes RL-based solutions inefficient [27], [28]. Fortunately, Deep Reinforcement Learning (DRL) methods [29] such as deep Q-network (DQN) have been introduced as a strong alternative to solve such high-dimensional problems and demonstrated its scalability and offloading efficiency in various MEC-based applications, such as multi-base station virtual MEC [30], multi-IoT networks [31]. The work in [32] introduced a multi-platform of edge and clouds for vehicular networks where offloading and resource allocation problem was formulated and then solved by RL-based methods. To provide a corporative computation service, the authors in [25], [33] investigated offloading issues in the combined fog/edge and cloud computing frameworks where offloading decision, computation and communication resources are formulated as a joint optimization problem. However, the authors proposed to decouple the original optimization problem into different sub-optimization problems and addressed them separately, which can be inefficient and complicated.

Despite the efforts made in the previous works, the integration of security scheme, i.e. access control, and computation offloading scheme has been rarely considered together in IoT networks. Specially, the integration of edge and cloud computing have been not investigated fully for IoT computation offloading scenarios on blockchain. Motivated by such limitations, in this paper, we propose a novel secure computation offloading model for IoT networks on mobile edge-cloud based on blockchain and reinforcement learning techniques.

B. Main Contributions and Paper Structure

In this paper, we mainly concentrate on security and offloading issues for the MECCO system. The main contributions of this paper can be summarized as follows.

1) We propose a novel secure computation offloading framework for mobile blockchain-based IoT networks where mobile devices (MDs) can offload their mobile tasks (i.e. IoT data) to cloud or edge server for computation under an access control mechanism.

2) We propose a trustworthy access control mechanism which can detect and prevent effectively illegal offloading behaviours of IoT devices by adopting smart contracts on mobile blockchain. Its main purpose is to perform user authentication, offloading verification and manage offloaded mobile data, ensuring security and privacy of the MECCO system.

3) We proposed a dynamic task offloading scheme which enables MUs to offload their tasks to cloud or edge server by taking IoT data size, available MEC computational capability and channel bandwidth resource into consideration. In particular, we proposed a novel offloading algorithm based on advanced DRL to obtain the optimal offloading policies for all MDs, subject to the QoS requirements, e.g., energy consumption and processing delay.

4) We investigate access control and offloading performances by conducting both real experiments and numerical simulations to validate the proposed MECCO framework.

The remainder of this paper is organized as follows. Section II discusses the motivation of integration of access control and computation offloading considered in our work. Section III introduces the integrated edge and cloud architecture on a blockchain platform for IoT networks. Next, we propose the system model consisting of the access control model and offloading model in Section IV. The solution is proposed in Section V with a smart contract-based access control scheme and DRL-based computation offloading scheme. Implementation results and analysis are given in Section VI, while Section VII draws our conclusion.

II. MOTIVATION OF INTEGRATION OF ACCESS CONTROL AND COMPUTATION OFFLOADING

In this section, we first discuss the motivation of the integration and then present the benefits of blockchain to solve offloading control issues in cloud IoT networks.

A. Why do we need blockchain based access control for mobile offloading

Security issues of mobile IoT offloading have been discussed extensively in recent works [34], [35], [36]. Specially, access control, an efficient security mechanism, can ensure the high security capacity of offloading systems and preserve computation resources by offering numerous features such as enhanced device authentication, user authentication as well as improved data privacy and network security [37], [38], [39], [40]. Without efficient access control, an attacker may impersonate devices to obtain the authentication rights of users [37] to perform illegal data transactions to the cloud. As a result, the attacker can retrieve user identification and access information. Moreover, adversaries can easily attack cloud resources to obtain the transferred

IoT data of mobile devices even when cloud data is encrypted [40]. In this regard, unauthorized users can exploit computation and storage resources without consent of both network participants [39]. These can degrade the cloud system and more importantly, such illegal cloud usage will lead the rapid depletion of limited cloud resources which are used to serve network users.

For these reasons, recent works [37], [38], [39], [40], [42] have proposed various access control solutions to solve security issues of mobile offloading. However, these existing solutions still remain several limitations. For example, the conventional access control architecture mainly relies on the central authentication authority (i.e. the service provider [37], central cloud server [38] or cloud service provider [39]) to manage user access and data usage among IoT devices. Nevertheless, such centralized architectures is vulnerable to single-point failure bottlenecks once the central entity is disrupted. Further, network authorities may be curious about offloaded data, thereby resulting in data privacy in offloading systems [40]. Importantly, the traditional access control often requires high trust levels among network entities for offloading security [42], which can be impossible to achieve in practice due to the lack of trust between IoT devices.

B. Benefits of blockchain to offloading control in IoT networks

The use of blockchain can bring great advantages over conventional access control solutions [37-40], [42] to mobile offloading security as the following. First, blockchain can provide a decentralized management solution for the offloading system. IoT data offloaded from mobile devices can be stored in the peer-to-peer storage in the blockchain network without relying on any central authority, which ensures fast data access and enhances significantly data privacy of mobile users [43]. Second, by incorporating blockchain into the edge-cloud computing network, the offloading system can achieve a trustworthy access control by using smart contracts which enables to authorize automatically devices and distinguish users from adversaries [15], aiming to prevent malicious offloading behaviours and potential threats to cloud computation resources. Consequently, data integrity and offloading validity of the system can be significantly improved. Final, with decentralized and secure nature, blockchain can work well in untrusted environments like our considered IoT scenario where there is no need the trust between cloud server, edge server and IoT devices for mobile task execution. Particularly, the peer-to-peer network architecture provided by blockchain can achieve robust access control with no single-point failure, high data integrity and system security for the mobile offloading [44].
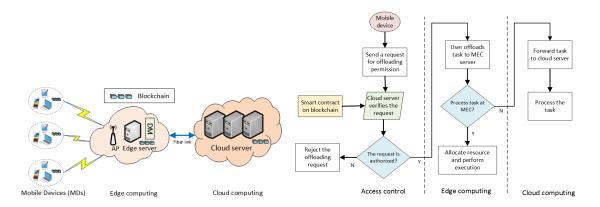
Fig. 1: The proposed MECCO architecture.

Fig. 2: Flowchart of MECCO scheme

III. NETWORK ARCHITECTURE

In this section, we first present the integrated edge and cloud architecture on a blockchain platform. Then, we describe the key concept of the proposed MECCO system which consists of a joint access control and computation offloading scheme.

A. Mobile edge-cloud for blockchain-IoT networks

In this paper, we propose an integrated edge-cloud architecture for IoT networks as shown in Fig. 1. Specially, the proposed system is deployed and managed by a blockchain network. The proposed architecture consists of three layers, namely mobile devices layer, edge computing layer and cloud computing layer. The key features of each layer are presented as follows.

- **Mobile devices layer:** This layer includes a network of IoT mobile devices (MDs) connected together on blockchain. Each MD has a blockchain account to join into the network and perform task offloading to cloud servers.

- **Edge computing layer:** This layer includes a wireless access point (AP) or base station (BS) for wireless communication with local devices, a Decision Maker (DM) for task execution decision, and a light-weight edge server for instant data processing. This layer can provide low-latency computation services at the edge of the network. However, for complex computation tasks, the edge server needs to forward them to the resourceful cloud server by a wired line to avoid the task overload on the edge layer. In addition, the edge server also acts as a blockchain entity to establish trustworthy communication with cloud nodes and MDs on the blockchain network for security guarantees. Any transactions and offloading activities in the offloading system will be recorded by blockchain and also broadcast to the edge server to achieve a common agreement on offloading management.

- **Cloud computing layer:** This layer includes multiple virtual machine (VMs) with powerful computation and storage capabilities to solve complex computation tasks from local IoT devices. In our MECCO architecture, the cloud layer also contains a network manager as a blockchain entity to control all user access, admin for smart contract management and miners for transaction mining. All cloud nodes operates on the blockchain platform in a decentralized and secure manner and link securely to edge server and MDs via the blockchain network.

B. Description of the proposed MECCO system

With the network settings, we describe the proposed MECCO system with a focus on access control and offloading concepts on the blockchain network. Due to the dynamic and scalable characteristics of the mobile multi-IoT environment, access control and computation offloading requires a comprehensive design to achieve both security and offloading goals. The MECCO workflow is shown as in Fig. 2, involving access control and computation offloading, which are described as follows.

First, a MD initializes a request as a blockchain transaction to start the computation offloading process and sends this request to the cloud server via the wireless access point (Note that for convenient management, access control for all MDs is implemented at the central cloud server). Then, the cloud server will authorize this request using an access control mechanism enabled by smart contracts. Based on predefined strict control policies, smart contract will identify, analyse and make decisions to accept or refuse the request. If the request is authorized successfully, a response will be returned to the MD so that the device can offload its tasks. Now the access control process finishes and the transaction will be recorded and stored on the blockchain network in a secure manner. In the second phase, the authorized MD will choose to offload its computation tasks to the edge or cloud server for calculation. At each offloading period, based on QoS requirements and current network conditions (task size, available edge resource, channel bandwidth resource), the decision maker (DM) at the edge layer [25] will perform optimization to decide where the mobile task should be executed, i.e. in the MEC server or cloud server, for optimal computation benefits (i.e. minimum offloading costs). If the tasks are executed at the edge layer, the MEC server needs to allocate communication resource as well as computation resource to each MD. However, if the tasks exceed the computing capability of the MEC server, such tasks should be forwarded to the resourceful cloud server for calculation. Note that data offloaded from MDs can be stored securely in a decentralized cloud storage on

blockchain [15]. The data storage management is beyond the scope of this paper, and details can be referred to our previous work [15].

IV. SYSTEM MODEL

In this paper, we consider a MECCO system on blockchain as shown in Fig. 1. The concerned network consists of a remote cloud server, a MEC server and a network of IoT mobile devices (MDs). All MDs are connected to the edge and cloud server via a wireless access point (AP). The proposed MECCO system includes two schemes, access control and computation offloading scheme as described in the following.

A. Access control model

We propose an access control model on a blockchain network for the MECCO system as shown in Fig. 3. For better offloading management, access control for all MDs is implemented at the cloud server as explained in the previous section, and therefore the MEC server is ignored in the access control scheme. Note that in our design, blockchain utilizes community validation to achieve synchronization on distributed ledgers which are replicated across cloud nodes (MECCO manager, admin and miners) within the peer cloud network. This decentralized cloud architecture with distributed access control gets rid of the single-point failure problems of traditional centralized cloud system, which ensures high system availability and computation service performances [15]. In the following, we introduce the key components and then explain the operation concept of the proposed access control scheme.

1) Key components of the access control scheme: As shown in Fig. 3, the access control scheme consists of four main components: MECCO manager, admin, smart contracts and miners.

- *MECCO manager:* The MECCO manager plays a significant role in our access control framework. It is responsible to monitor all offloading events on the blockchain network, including offloading requests and access authentication for MDs. The management capability of MECCO manager is enabled by smart contracts through strict user policies.

- *Admin:* It is used to manage transactions and operations on cloud by the means of adding, changing or revoking access permissions. Admin is responsible to deploy smart contracts and the only entity with the ability to update or modify policies in smart contracts.

- *Smart contracts:* The smart contracts define all operations allowed in the access control system. MDs can interact with smart contracts by the contract address and Application Binary
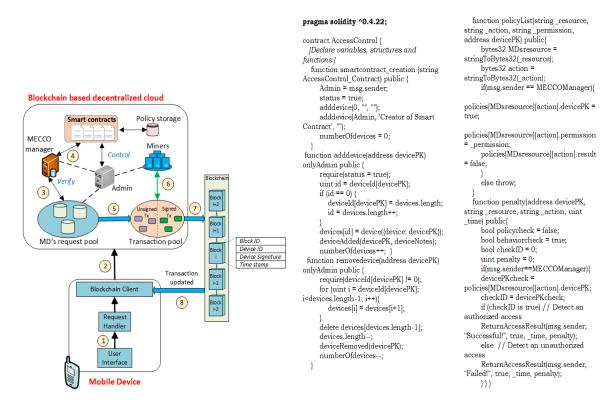
Fig. 3: Workflow of access control.

```
pragma solidity ^0.4.22;

contract AccessControl {
    [Declare variables, structures and
functions]
    function smartcontract_creation (string
AccessControl_Contract) public {
        Admin = msg.sender;
        status = true;
        adddevice(0, "", "");
        adddevice(Admin, 'Creator of Smart
Contract', "");
        numberOfdevices = 0;
    }
function adddevice(address devicePK)
onlyAdmin public {
        require(status = true);
        uint id = deviceId[devicePK];
        if (id == 0) {
            deviceId[devicePK] = devices.length;
            id = devices.length++;
        }
        devices[id] = device({device: devicePK});
        deviceAdded(devicePK, deviceNotes);
        numberOfdevices++;  }
    function removedevice(address devicePK)
onlyAdmin public {
        require(deviceId[devicePK] != 0);
        for (uint i = deviceId[devicePK];
i<devices.length-1; i++){
            devices[i] = devices[i+1];
        }
        delete devices[devices.length-1];
        devices.length--;
        deviceRemoved(devicePK);
        numberOfdevices--;
    }
```

```
    function policyList(string _resource,
string _action, string _permission,
address devicePK) public{
        bytes32 MDsresource =
stringToBytes32(_resource);
        bytes32 action =
stringToBytes32(_action);
        if(msg.sender == MECCOManager){

policies[MDsresource][action].devicePK =
true;

policies[MDsresource][action].permission
= _permission;
        policies[MDsresource][action].result
= false;
        }
        else throw;
    }
    function penalty(address devicePK,
string _resource, string _action, uint
_time) public{
        bool policycheck = false;
        bool behaviorcheck = true;
        bool checkID = 0;
        uint penalty = 0;
        if(msg.sender==MECCOManager){
        devicePKcheck =
policies[MDsresource][action].devicePK;
        checkID = devicePKcheck;
        if (checkID is true) // Detect an
authorized access
        ReturnAccessResult(msg.sender,
"Successful!", true, _time, penalty);
        else  // Detect an unauthorized
access
        ReturnAccessResult(msg.sender,
"Failed!", true, _time, penalty);
        }}}
```

Fig. 4: Smart contract implementation.

Interface (ABI). Smart contracts can identify, validate request and grant access permissions for MDs by triggering transactions or messages. The smart contract and its operations are accessible to all blockchain entities. It is considered as core software in our access control scheme.

Miners: The miners are responsible to validate the data blocks consisting of transactions of MDs. All validated blocks will be appended to the blockchain by a process called consensus mechanism under the management of a network of miners.

2) The concept of access control for MECCO system: In this subsection, we describe the operation concept of access control for MECCO systems based on smart contracts and blockchain. At each mobile device, we design a blockchain client module, which implements a full functionality to participate in the cloud blockchain network. This module is responsible to encode transactions and data requests, sign digitally on transactions and connect with blockchain for transaction tracking. Besides, the user interface module is designed for user interaction, and a request handler module for processing user request information. The work flow of access control on blockchain is illustrated in Fig. 3. A description of each step is provided as follows.

① A MD initializes a request as an offloading transaction for offloading computation tasks to the edge-cloud server.

② The blockchain client processes and sends the request to the storage pool so that the MECCO manager and smart contracts can verify.

③ The MECCO manager collects the requests of MDs in the storage pool based in a first-come first-served manner.

④ The MECCO manager verifies the request by smart contracts with a strict control policy. If the request is accepted, a response will be returned to the MD for offloading data.

⑤ Offloading transactions are grouped into data blocks, which are then inserted into the transaction pool for confirmation by miners.

⑥ The miners validate the data blocks and sign them with digital signature to append to the blockchain.

⑦ The offloading transaction is added to the blockchain network and broadcast to all MDs within the MECCO system.

⑧ The offloading transaction is updated at the MD for tracking via the blockchain client.

B. Computation offloading model

In the computation offloading model, we assume that MDs are authorized by our access control scheme as designed in the previous subsection. We consider realistic IoT applications (i.e. speech recognition or big data analysis) where computation tasks can be very large and thus inefficient to be processed locally by MDs. Therefore, the authorized MDs will have to offload their tasks to edge or cloud server for efficient execution. In this subsection, we propose a new offloading scheme for our MECCO scenario. We first formulate the task model and computation model, and then we describe the problem formulation in details.

1) Task model: We consider a task model as shown in Fig. 1. We denote a set of MDs as $\mathcal{N} = \{1, 2, ..., N\}$. It is assumed that each MD has a computation task to be completed. For each MD n, the computation task can be formulated as a variable tuple $R_n = (D_n, X_n, \tau_n)$. Here D_n (in bits) denotes the data size of computation task of the MD n. We also assume that the size of D_n is fixed when it is offloaded to edge or cloud server. X_n (in CPU cycles/bit) denotes the total number of CPU cycles required to accomplish the computation for the task R_n. Moreover, τ_n (in seconds) reflects the maximum tolerable delay of task R_n. Note that the information of D_n and X_n can be obtained by using program profilers [23]. In this paper, we focus on a joint optimization problem of offloading decision, edge computation resource and bandwidth resource, which has been not studied well in the literature studies.

Considering a set of N MDs, we define an offloading decision vector as $A = [\alpha_1, \alpha_2, ..., \alpha_N]$ where $\alpha_n = \{\alpha_n^e, \alpha_n^c\}$. Here $\alpha_n^e, \alpha_n^c \in \{0, 1\}$ and $\alpha_n^e + \alpha_n^c = 1$. If the task is offloaded to the edge or cloud server, the corresponding parameter is 1, otherwise it is 0. Note that each task is executed at only a platform at each offloading period. For tractable network analysis, similar to the existing works [23], [24], [25], we assume that all MDs and wireless network remain stationary during the offloading period. This assumption is feasible to many IoT applications such as speech recognition or face recognition in our concerned scenarios, where the offloading period is much shorter than the timescales of MD mobility and wireless network dynamics.

Further, we also consider the resource allocation problem for computation offloading. For edge computing, the MEC server needs to allocate its computing resources to each MD to perform task execution. For cloud computing, the task needs to be offloaded by the MD to the edge layer and then is forwarded to the remote cloud via a wired link. Due to powerful computing capacity of the cloud server, the problem of cloud resource allocation is ignored in our paper. Nevertheless, the allocation of limited radio bandwidth should be considered to improve offloading efficiency of our MECCO system. Assuming that the total radio bandwidth of our MECCO system is B Hz, we allocate part of the bandwidth resource to each MD to avoid interference between them [25]. We normalize the assigned bandwidth to the MD n as $w_n \in [0, 1]$, then we have $\sum_1^N w_n = 1$. We also denote the channel gain between MDs and the MEC server as h_n, then the transmission data rate of the MD n can be calculated as $r_n = w_n B log_2(1 + \frac{p_n h_n}{w_n N_0 B})$ where p_n is the transmit power (W) of MD n, N_0 is additive noisy power spectral density (dBm/Hz).

In the following, we formulate the computation offloading model of edge computing and cloud computing with a focus on computation latency and energy consumption analysis for our MECCO system.

2) Computation model: We consider the edge computing model and the cloud computing model to formulate our computation problem.

2.1) Edge Computing:

We consider the case when the task R_n of the MD n is offloaded to the MEC server for computation ($\alpha_n^e = 1$). We denote T_n^e as the edge computing latency which includes the transmission delay for the MD n sending data to the MEC server and the execution time on the MEC server. Similar to [25], the total edge computing latency can be expressed as

$$T_n^e = \frac{D_n}{r_n} + \frac{X_n}{f_n^e} \tag{1}$$

where f_n^e (in CPU cycles/s) denotes the edge computation resource allocated to the MD n. Note that edge resource allocated to all MDs should not exceed the total computation capacity of the MEC server $\sum_{n=1}^{N} f_n^e \leq F^e$. Moreover, the energy cost for offloading to the MEC server consists of energy consumption for transmitting data and execution. Denote p_n^i as the power consumption (in watt) of the MD n in idle status, the total energy consumption of offloading data to the MEC server is given as

$$E_n^e = \frac{p_n D_n}{r_n} + \frac{p_n^i X_n}{f_n^e}. \tag{2}$$

2.2) Cloud Computing:

If the computation task is offloaded to the cloud server ($\alpha_n^c = 1$), the MD n needs to offload its task to the MEC server which then forwards it to the remote cloud server for computation via a wired link. The cloud server also allocates its computation resources to compute the task efficiently. We denote the data rate of the wired link for transmitting the task of the MD n as r_n^w and the cloud resource allocated to the MD n as f_n^c. Then then the total cloud computing latency and energy cost can be expressed respectively as [25]

$$T_n^c = \frac{D_n}{r_n} + \frac{D_n}{r_n^w} + \frac{X_n}{f_n^c}, \tag{3}$$

$$E_n^c = \frac{p_n D_n}{r_n} + p_n^i \left(\frac{D_n}{r_n^w} + \frac{X_n}{f_n^c} \right). \tag{4}$$

According to (1)-(4), the computation latency and energy consumption of the MD n in our MECCO system can be expressed respectively as

$$T_n = T_n^e + T_n^c, \tag{5}$$

$$E_n = E_n^e + E_n^c. \tag{6}$$

3) Offloading problem formulation: In this subsection, we formulate the computation offloading, edge resource allocation and radio bandwidth resource as a joint optimization problem. Our objective is to minimize the sum cost of computation latency and energy consumption for all MDs in our MECCO system.

We formulate the cost function of the MD n as the weighted sum of computation latency and energy consumption, which is given as $C_n = \beta^t T_n + \beta^e E_n$ where $\beta_n^t, \beta_n^e \in [0, 1]$ ($n \in \mathcal{N}$) denote the weight of latency and energy consumption, respectively. Mathematically, we formulate the joint optimization of computation latency and energy consumption for the multi-user

MECCO, subject to the offloading decisions $\mathbf{A} = [\alpha_1^e, \alpha_1^c, ..., \alpha_N^e, \alpha_N^c]$, edge resource allocation $\mathbf{f} = [f_1, f_2, ..., f_N]$ and radio bandwidth allocation $\mathbf{w} = [w_1, w_2, ..., w_N]$ as follows

$$(P1) : \underset{\mathbf{A}, \mathbf{f}, \mathbf{w}}{\text{minimize}} \quad \sum_{n=1}^{N} C_n$$

$$\text{subject to} \quad (C1) : \alpha_n^e, \alpha_n^c \in \{0, 1\}, \forall n \in \mathcal{N},$$

$$(C2) : \alpha_n^e + \alpha_n^c = 1, \forall n \in \mathcal{N},$$

$$(C3) : \sum_{n=1}^{N} f_n^e \leq F^e,$$

$$(C4) : f_n^e \geq 0, \forall n \in \mathcal{N},$$

$$(C5) : 0 < w_n \leq 1, \forall n \in \mathcal{N},$$

$$(C6) : \sum_{n=1}^{N} w_n \leq 1.$$

Here, the constraint (C1) and (C2) represent the binary offloading decision policy of the MD n, offloading to the MEC server or offloading to the cloud server. (C3) and (C4) indicate that the allocated edge resources should not exceed the total computing capacity of the MEC server, while (C5) and (C6) are the constraints of bandwidth allocation. Note that the optimization problem (P1) is not convex due to the non-convexity of its feasible set and objective function with the binary variable $\mathbf{A}$. Further, the size of the problem (P1) can be very large when the number of MDs in MECCO system increases rapidly. The existing studies [23], [24], [25] proposed to decouple the optimization problem into sub-optimization problems and address them separately, which is complex and inefficient to solve the large-scale MECCO problem like our concerned scenario. Therefore, in the next section, we propose a novel technique using advanced deep reinforcement learning to solve the proposed offloading problem.

V. PROPOSED SOLUTION

In this section, we propose access control and computation offloading approaches for our MECCO system.

A. Access control for MECCO system

In this subsection, we design a smart contract to formulate our access control scheme. We also provide an access protocol that presents the work flow of access control for the MECCO

system.

1) Smart contract design: We first create an *AccessControl* contract controlled by the admin to monitor transaction operations in our MECCO network on blockchain. Here we use an Ethereum blockchain platform [37] due to its adaptable and flexible features, which allow to build any blockchain applications such as our IoT scenario. We denote PK as the public key of MD. The contract mainly provides the following four functions.

- *AddMD(PK): (executed by Admin)* This function allows to add a new MD to the smart contract. MD is identified by their public key and is added into the contract with a corresponding role based on their request. MD information is also kept in cloud storage as part of system database.

- *DeleteMD(PK): (executed by Admin)* It is used to remove MDs from the network based on the corresponding public key. All device information is also deleted from cloud storage.

- *PolicyList(PK): (executed by Admin)* The policy list contains public key of all MDs for identification when the smart contract processes new transactions.

- *Penalty(PK, action): (executed by Admin)* When detecting an unauthorized offloading request to cloud, the MECCO manager will inform smart contract to issue a penalty to the requester. In our paper, we give a warning message as a penalty to the unauthorized MDs.

The smart contract design for the proposed access control scheme on Ethereum blockchain can be seen in Fig. 4.

2) Access control protocol: To operate the access control for computation offloading, we also develop an access control protocol which is performed when a MD executes a transaction for a request of offloading data to the edge-cloud. The access control protocol includes two phases: *Transaction pre-processing* (executed by the MECCO manager) and *Verification* (executed by the Admin). In the first phase, the MECCO manager receives a new transaction *Tx* from a MD. The MECCO manager will obtain the public key PK of the requester by using the *Tx.getSenderPublicKey()* function and send it to the contract for validation. In the next phase, after receiving a transaction with a MD PK from MECCO manager *(msg.sender = ME)*, the admin will verify access rights of the requester based on its *PK* in the policy list of the smart contract. If the *PK* is available in the list, the request is accepted and now a task offloading permission is granted to the requester. Otherwise, the smart contract will issue a penalty to this request through the *Penalty()* function. In this case, all offloading activities are denied and the request is discarded from the blockchain network. The access control protocol is summarized in

the Algorithm 1.

Algorithm 1 Access control for computation offloading

1: **Input:** Tx (The offloading request on blockchain)

2: **Output:** $Result$ (Access result for offloading request)

3: **Initialization:** *(by the MECCO Manager)*

4: Receive a new transaction Tx from a mobile device MD

5: Get the public key of the requester: $PK \leftarrow Tx.getSenderPublicKey()$

6: Send the public key to Admin ($msg.sender = MECCOmanager$)

7: **Pre-processing the request** *(by Admin)*

8: **if** PK is available in the policy list **then**

9:     $policyList(PK) \leftarrow true$

10: **end if**

11: Decode the transaction $decodedTx \leftarrow abiDecoder.decodeMethod(Tx)$

12: Specify request information: $Addr \leftarrow web3.eth.getData(decodedTx([DataIndex])$

13: Specify *DeviceID*: $D_{ID} \leftarrow Addr(Index[D_{ID}])$;

14: **Verification** *(by the smart contract)*

15: **while** true **do**

16:     **if** $policyList(PK) \rightarrow true$ **then**

17:         **if** $policyList(D_{ID}) \rightarrow true$ **then**

18:             $Result \leftarrow Penalty(PK,"Successful!")$

19:             break;

20:         **else**

21:             $Result \leftarrow Penalty(PK,"Failed")$

22:             break;

23:         **end if**

24:     **else**

25:         $Result \leftarrow Penalty(PK,"Failed")$

26:         break;

27:     **end if**

28: **end while**

B. Advanced DRL-based computation offloading for MECCO system

1) Problem formulation: In this paper, we describe the computation offloading problem as a reinforcement learning process. Our offloading algorithm is to minimize the weighted sum cost of all MDs in terms of computation latency and energy consumption in the proposed MECCO. Specially, in this work, we consider a realistic multi-user MECCO scenario where the size of computation task, edge computation resource and system bandwidth resource are highly dynamic.

Consequently, the MECCO system faces the large of the system state and action space. More importantly, it needs to allocate resources (edge and bandwidth resources) to each MD at each system state with the aim of optimizing the total offloading cost. These pose challenges to traditional approaches to solve effectively the multi-scale offloading problem. Therefore, in this subsection, we propose an offloading method using advanced DRL which works well for high-dimensional systems like our considered scenario. We first introduce the offloading framework using DRL where state space, action space and reward are defined.

- **State:** The system state is chosen as $s = \{tc, ec, bw\}$ where tc is the total offloading cost of MECCO system ($tc = C$), ec is the available computation resource of the MEC server ($ec = F^e - \sum_{n=1}^{N} f_n^e$). Further, bw denotes the available bandwidth resource of the MECCO system, and can be specified as $bw = B - \sum_{n=1}^{N} w_n$.

- **Action:** The action space is formulated as the offloading decision vector $\mathbf{A} = [\alpha_1^e, \alpha_1^c, ..., \alpha_N^e, \alpha_N^c]$, edge resource allocation $\mathbf{f} = [f_1, f_2, ..., f_N]$ and radio bandwidth allocation $\mathbf{w} = [w_1, w_2, ..., w_N]$. Thus, the action vector can be expressed as $a = [\alpha_1^e, \alpha_1^c, f_1, w_1, ..., \alpha_N^e, \alpha_N^c, f_N, w_N]$.

- **Reward:** The objective of the RL agent is to find an optimal offloading decision action a at each state s with the aim of minimizing the sum cost $C(s, a)$ of time and energy consumption in the MECCO system. Specially, the reward function should be negatively related to the objective function of the optimization problem (P1) in the previous section. Accordingly, we can formulate the system reward as $r(s, a) = -C(s, a)$.

2) Basics of Deep Reinforcement Learning: The principle of Reinforcement Learning (RL) can be described as a Markov Decision Process (MDP) [27]. In the RL model, an agent can make optimal actions by interacting with the environment without an explicit model of the system dynamics. In our MECCO scenario, at the beginning, the agent has no experience and information about the MECCO environment. Thus it needs to *explore* for every time epoch by taking some actions at each offloading state, i.e. the size of current IoT data size, available edge resource. As long as the agent has some experiences from actual interactions with the environment, it will *exploit* the known information of states while keep exploration. As a combination of Monte Carlo method and dynamic programing, a temporal-difference (TD) approach can be employed to allow the agent to learn offloading policies without requiring the state transition probability which is difficult to acquire in realistic scenarios like in our dynamic mobile blockchain. Therefore, we can develop a dynamic offloading scheme using a free-model RL. Specially, in this paper, our

focus is to find the optimal policy that minimize the offloading cost C. To this end, the state-action function can be updated using the experience tuple of agent (s^t, a^t, r^t, s^{t+1}) at each time step t in our offloading application as

$$Q(s^t, a^t) \quad \leftarrow \quad Q(s^t, a^t) \quad + \quad \alpha[r(s^t, a^t) \quad + \quad \gamma \quad * \quad minQ(s^{t+1}, a^{t+1}) \quad - \quad Q(s^t, a^t)] \quad (7)$$

which is called as Q-learning algorithm [27]. Here α is the learning rate, γ is the discount factor between (0,1) and $\sigma^t = r(s^t, a^t) + \gamma * maxQ(s^{t+1}, a^{t+1}) - Q(s^t, a^t)$ is the TD error which will be zero for the optimal Q-value. Further, under the optimal policy π^* which can be obtained from the maximum Q-value ($\pi^*(s) = argmaxQ^*(s, a)$), the Bellman optimality equation [27] for the state-action equation can be expressed as

$$Q^*(s^t, a^t) = \mathbb{E}_{s^{t+1} \sim E}[r(s^t, a^t) + \gamma * minQ^*(s^{t+1}, a^{t+1})]. \tag{8}$$

It is noting that the Q-learning algorithm is proved to converge with probability one over an infinite number of times [27] and achieves the optimal Q^*.

Although the reinforcement learning can solve the offloading problem by obtaining the optimum reward, there are still some remaining problems. The state and action values in the Q-learning method are stored in a two-dimensional Q table, but this method can become infeasible to solve complex problems with a much larger state-action space. This is because if we keep all Q-values in a table, the matrix $Q(s, a)$ can be very large, which makes the learning agents difficult to obtain sufficient samples to explore each state, leading to the failure of the learning algorithm. Moreover, the algorithm will converge slowly due to too many states that the agent has to process.

To overcome such challenges, we can use deep learning with Deep Neural Network (DNN) to approximate the Q-values instead of using the conventional Q-table, leading to a new algorithm called deep reinforcement learning (DRL) [29]. In the DRL-based algorithm, a DNN is used to approximate the target Q-values $Q(s^t, a, \theta)$ with weights θ. Further, to solve the instability of Q-network due to function approximation, the experience replay solution is employed in the training phase with the buffer $\mathcal{B}$ which stores experiences $e^t = (s^t, a^t, r^t, s^{t+1})$ at each time step t. Next, a random mini-batch of transitions (s^j, a^j, r^j, s^{j+1}) from the replay memory is selected to train the Q-network. Here the Q-network is trained by iteratively updating the weights θ to minimize the loss function, which is written as

$$L(\theta) = \mathbb{E}[y^j - Q(s^j, a^j | \theta^j))^2] \tag{9}$$

where $y^j = (r^j + \gamma * min_{a^{j+1}} Q(s^{j+1}, a^{j+1}|\theta'))$ and the $\mathbb{E}[.]$ denotes the expectation function.

3) Advanced DRL-based computation offloading: Recent years have witnessed great efforts in deep reinforcement learning to improve the performance of DLR-based algorithm. In this paper, we focus on two recent improvements in DRL research to apply to our formulated offloading problem, including double DQN [45] and dueling DQN [46].

- *Double DQN:* In conventional DQN, we use the same samples from the replay memory to both specify which action is the best and estimate this action value, which leads to the large over-estimation of action values. To solve this problem, two Q-functions are proposed to select and evaluate action values by the new loss function as follow

$$L_{dou}(\theta) = \mathbb{E}[y_{dou}^j - Q(s^j, a^j|\theta^j))^2] \tag{10}$$

where $y_{dou}^j = (r^j + \gamma.Q(s^{j+1}, min_{a^{j+1}} Q(s^{j+1}, a^{j+1}|\theta^1), \theta^2)$. It is noting that the action choice is still based on the weight θ^1, while the evaluation of the selected action relies on the weight value θ^2. This technique reduces over-estimation problem and this improves the training process and thus efficiency of the DQN algorithms.

- *Dueling DQN:* During the computation of Q function in some MDP problems, it is unnecessary to estimate action and state values at the same time, thus we can estimate separately the action and state value functions. Motivated by this concept, in dueling DQN, the state action value $Q(s, a)$ can be decomposed into two value functions as follows: $Q(s, a) = V(s) + A(a)$. Here, $V(s)$ is the state-value function which evaluates the significance of being at a given state s. $A(a)$ is action-value function to estimate the significance of choosing an action a compared to other actions. The $V(s)$ and $A(a)$ are first computed separately, then are combined to generate the final output $Q(s, a)$. This approach leads to a better performance on policy evaluation of MDPs, especially complex problems with large action space like our considered offloading scenario.

Motivated by the advantages of above two approaches, we develop a novel DQN algorithm using such two improvements to solve the offloading problem of our MECCO system. The details of the proposed algorithm is shown in Algorithm 2. The ADRLO algorithm can achieve the optimal task offloading strategy in an iterative manner. As shown in Algorithm 2, the procedure generates a task offloading strategy for MDs based on system states and observes the system reward at each time epoch so that the offloading policy can be optimized (lines 8-15). Then the procedure updates the history experience tuple and train the Q-network (lines 17-21) with

loss function minimization. This trial and error solution will avoid the requirement of prior information of offloading environment. Over the training time period, the trained deep neural network can characterize well the environment and therefore, the proposed offloading algorithm can dynamically adapt to the real MECCO environment.

Algorithm 2 Advanced DRL-based computation offloading (ADRLO) algorithm for MECCO system

---

1: **Initialization:**

2: Set replay memory $\mathcal{D}$ with capacity N

3: Initialize the deep Q network $Q(s, a)$ with random weight θ and θ', initialize the exploration probability $\epsilon \in (0, 1)$

4: **for** episode = 1,..., *M* **do**

5:     Initialize the state sequence s^0

6:     **for** $t = 1, 2, ...$ **do**

7:         /$***$ *Plan the computation offloading* $***$/

8:         Estimate the current offloading cost tc^t

9:         Estimate the available edge resource ec^t

10:        Estimate the available bandwidth resource bw^t

11:        Set $s^t = \{tc^t, ec^t, bw^t\}$

12:        Select a random action a^t with probability ϵ, otherwise $a^t = argminQ(s^t, a, \theta)$

13:        Offload the computation task $\alpha_e^t(D^t)$ to MEC server or to cloud $\alpha_c^t(D^t)$

14:        Observe the reward r^t and next state s^{t+1}

15:        Evaluate the system cost $C(s, a)^t$

16:        /$***$ *Update* $***$/

17:        Store the experience (s^t, a^t, r^t, s^{t+1}) into the memory $\mathcal{D}$

18:        Sample random mini-batch of state transitions (s^j, a^j, r^j, s^{j+1}) from $\mathcal{D}$

19:        Calculate the target Q-value by $(y_{dou}^j = r^j + \gamma.Q(s^{j+1}, min_{a^{j+1}}Q(s^{j+1}, a^{j+1}|\theta), \theta')$

20:        Perform a gradient descent step with the weight θ^j on $(y_{dou}^j - Q(s^j, a^j|\theta^j))^2$ as the loss function

21:        Train the deep Q-network with updated θ and θ'

22:     **end for**

23: **end for**

To this end, we propose a novel MECCO algorithm by combining access control and computation offloading on a mobile blockchain IoT network. The concept of the integrated scheme is shown in Algorithm 3, which can be explained as follows. We first create a private Ethereum blockchain environment on cloud, i.e. Amazon cloud platform, to perform access control and offloading functionalities. With blockchain setup, we can deploy a smart contract and connect with a network of MDs, i.e. smartphones to formulate a MECCO system (lines 1-3). It is assumed that all MDs have the demand to offload their computation tasks to edge or cloud servers for

---

Algorithm 3 Joint access control and computation offloading on blockchain for IoT networks

---

1: Initialize private blockchain, and setup N Ethereum nodes for all mobile devices

2: Deploy smart contract with Ethereum blockchain on cloud

3: Blockchain starts mining

4: **Access control phase:**

5: **for** each MD n in N **do**

6: Initialize the transaction Tx for offloading request

7: Submit the transaction Tx to MECCO manager

8: Perform request verification for access control (*see Algorithm 1*)

9: **if** verification is successful **then**

10: Go to the offloading phase

11: **else**

12: Reject the offloading request

13: **end if**

14: **end for**

15: **Computation offloading phase:**

16: **for** each authorized MD n in $N^{'}$ **do**

17: Perform computation task offloading (*see Algorithm 2*)

18: Verify the offloading transaction Tx by mining process

19: Upload the transaction Tx to blockchain and wait for confirmation

20: **end for**

---

execution. In the access control phase, each MD will send an offloading request as a transaction in blockchain context to the cloud server for authentication. The smart contract deployed on cloud will verify the transaction to accept or refuse the request. If the request is accepted, the process now goes to the offloading phase, otherwise a penalty will be given to the requestor (lines 4-14). In the offloading phase, it is assumed that all authorized MDs in the access control phase are grouped into the new set of devices ($N^{'}$). We perform the offloading algorithm using a DRL network to optimize the offloading cost for our MECCO. The process finishes with the mining of the transaction and appending it to the blockchain (lines 15-20). The implementation and evaluation of the proposed scheme is presented in the next section.

VI. PERFORMANCE EVALUATION

In this section, we investigate the proposed MECCO system by conducting both real experiments to evaluate the access control performance and numerical simulations to evaluate the efficiency of computation offloading.

Fig. 5: Access control experiment.　　　　　　Fig. 6: Cloud blockchain deployment.

A. Implementation settings

1) Experiment settings for access control: We considered an access control framework for MECCO on mobile cloud as shown in Fig. 5. We deployed a private Ethereum blockchain network on the Amazon cloud computing platform [47] where two virtual machines AWS EC2 were employed as the miners, two virtual machines Ubuntu 16.04 LTS were used as the admin and MECCO manager, respectively. Our smart contract was written by Solidity programming language [48] as shown in Fig. 4 and was deployed on AWS Lambda functions. Each function interacts with the cloud blockchain via the web3.js API. MDs can interact with smart contracts through their Android phone where a Geth client [49] (a command line interface implemented in the Go language) was installed to transform each smartphone into an Ethereum node. By using the Geth client, a mobile device can create an Ethereum account to communicate with our blockchain network for accessing data. The web3.js library [50], a lightweight Java library for working with smart contracts and blockchain, was also used for developing the mobile application to connect with the Ethereum blockchain network. In our experiment in this paper, we used two Sony mobile phones running on an Android OS version 8.0 platform to investigate results of access control.

2) Simulation settings for computation offloading: In our simulation for computation offloading, a MECCO system is considered with a cloud server, a MEC server with a number of MDs authorized by the access control mechanism. Here we consider *N=10 MDs*, each of them has a computation task to be executed at edge or cloud server. We assume that the data size of IoT computation tasks D is randomly distributed between 0.1MB and 12MB. The total bandwidth resource B is set to 15 MHz [25]; the additive noisy power spectral density N_0 is -100dBm/Hz [25]. Besides, the total computation capacity of MEC server F^e and cloud server F^c are set to 2GHz and 10GHz, respectively. Moreover, in the advanced DQN-learning algorithm, simulations

Fig. 7: Illustrations of access control results on smartphones with blockchain: a) Device registration form with Ethereum account, b) Offloading access results of an authorized MD, c) Transaction record of authorized offloading access, d) Access result of an unauthorized MD, e) Transaction record of unauthorized offloading access.

were implemented in Python with TensorFlow 2.0 [51] and we used the AdamOptimizer to optimize the loss function for the training process. All numerical simulations was performed on a computer with an Intel Core i7 4.7GHz CPU and 128 GB memory.

B. Access control performance

In this subsection, we implemented access control on blockchain and verified the proposed scheme. We first deployed a private Ethereum blockchain on Amazon cloud as illustrated in Fig. 6. Offloading access and transactions are recorded and shown on the web interface for monitoring. Based on our blockchain settings, we deployed smart contracts, established network entities of the access control as explained in Fig. 3 and connected with mobile applications to build our access control framework.

We evaluated the performance of the proposed access control scheme via two use cases with authorized and unauthorized access for offloading request (Fig. 7). The main objective of the access control is to verify effectively the offloading request from the MDs and prevent any potential threats to our MECCO system on mobile edge-cloud. The evaluation is presented as follows. First, it is assumed that an owner of a smartphone (i.e. MD) wants to offload its data task to the cloud or edge server for calculation. He should create an Ethereum blockchain account and register an access right of his device by providing the device name, device ID and make a transaction for an offloading request (Fig. 7(a)). Note that based on the blockchain concept, the transaction is signed by a private key coupled with a public key for device identification. After receiving the transaction from the MD, the MECCO manager will authorize the request using

the smart contract as designed in Algorithm 1. If the MD is verified by the smart contract, the request is now confirmed for task offloading. Accordingly, a response is given to the requestor for confirmation so that the MD can offload its data to edge or cloud server (Fig. 7(b)). Now the access control process finishes and the offloading transaction is appended to blockchain by the cloud miner and broadcast to all entities in the network. Therefore, a MD can keep track of the offloading access (Fig. 7(c)), which improves network trustworthiness.

In the case of unauthorized access, the smart contract will verify and detect by the access protocol with a predefined policy list. Such illegal request is prevented and discarded from blockchain, and a warning message is returned to the requester (Fig. 7(d)). A corresponding transaction for unauthorized access is also issued by the smart contract (Fig. 7(e)). Obviously, the use of blockchain can address effectively challenges mentioned in the literature in controlling access information and monitoring offloading behaviours, which can enhance system reliability and data privacy.

Based on the analysed access control results, it is clearly that our design achieves a trustworthy access control among mobile devices, with the capability of device identity, authentication and reliability. Besides, to show further the advantage of our blockchain-based access control implementation over conventional access control solutions, we provided an extensive efficiency comparison in our recent work [15] under various performance metrics, including theoretical analysis and real evaluations on delay efficiency. The implementation results demonstrate the effectiveness of blockchain in improving security and robustness of user-cloud systems in offloading scenarios.

C. Computation offloading performance

We evaluated the advanced DRL-based offloading algorithm (ADRLO) in various performance metrics. For comparisons, we consider three other schemes: (1) *DRL-based offloading scheme (DRLO)* which offloading is performed by regular DRL, (2) *Edge offloading scheme (EO)* which all MDs offload their computation task to the MEC server, and (3) *Cloud offloading scheme (CO)* which all MDs offload their computation task to the cloud server. Note that the simulation results are averaged over 50 runs of numerical simulations.

We first evaluate the performance of the total MECCO cost versus the number of MDs and tasks in Fig. 8. More specific, in Fig. 8(a), we consider a MECCO system with a varying number of MDs, and each MD has a computation task (task sizes varies between 0.1MB and 1MB).
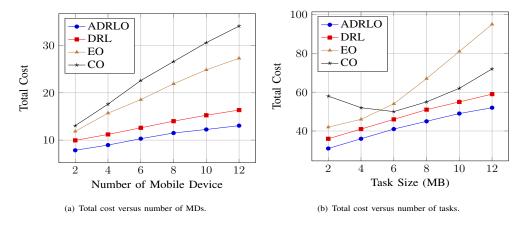
(a) Total cost versus number of MDs.

(b) Total cost versus number of tasks.

Fig. 8: Total offloading cost versus number of MDs and tasks.

(a) Total cost versus edge resource.

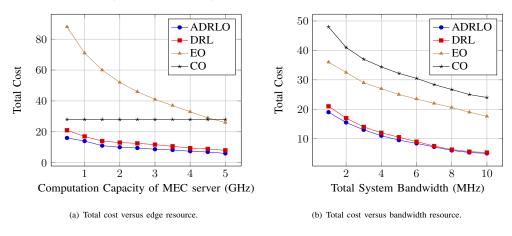(b) Total cost versus bandwidth resource.

Fig. 9: Total offloading cost versus resource allocation.

As indicated from the simulation results, the curves of all offloading schemes increase with the growing number of MDs. Specially, the CO scheme has the highest offloading cost among all offloading schemes. An explanation is that the computation tasks of MDs in this test case are relatively small, so offloading to the remote cloud will incur a larger transmission latency, and thus leads to a higher offloading cost. Meanwhile, thanks to low-latency computation services of MEC server, the EO scheme shows better offloading efficiency with lower offloading costs for any user cases. More importantly, the DRLO and ADRLO schemes exhibit much lower offloading costs and the ADRLO scheme achieve the best performance among all offloading schemes, which can be explained by the following reasons. First, in the proposed advanced DRL algorithm, the double DQN network obtains the optimal policy with larger system gains than that of the regular DQN. Second, the dueling DQN architecture leads to a better performance on policy evaluation by evaluating separately the state-value function and the action-value function. These advanced techniques make the ADRLO scheme more efficient in terms of evaluation of optimal offloading

policy and offloading performance, accordingly.

Next, we analyze the computation offloading performance for the MECCO system with a single MD $N = 1$ and its task size changes between 2MB and 12MB as shown in Fig. 8(b). It is observed that when the task size increases, the cost of four schemes increases due to the growing amount of IoT data to be executed completely. Particularly, when the task size is small ($<$5MB), the CO scheme has the higher offloading cost than that the EO scheme. The reason behind this observation is the small tasks can be processed efficiently by the MEC server with sufficient computation resources. Therefore, offloading small tasks to the remote cloud will result in unnecessary transmission latency and consequently, incur a higher total offloading cost for the CO scheme. However, when the task size become larger ($>$5MB), the computational capacity of the MEC server becomes less sufficient to accommodate all tasks, while the resourceful cloud server can compute large size-tasks effectively. As a result, the CO scheme can achieve a much lower offloading cost, compared to the EO scheme. The ADRLO scheme still achieves the minimum total offloading cost, followed by the DRLO scheme with a small gap when the task size increases.

Moreover, we compare offloading schemes under resource allocation scenarios in Fig. 9. First, we evaluate the impact of edge resource allocation on offloading performance. From Fig. 9(a), the total offloading cost of schemes based on edge computing decreases significantly with the increase of the computation capacity of MEC server. Furthermore, the CO scheme has a stable offloading cost with the increase of MEC capacity as all computation tasks in this scheme are processed in the cloud. In particular, when the edge computation capacity is small, the EO scheme has the highest offloading cost. This is because a less computing resource of MEC server will lead to a higher execution latency, and thus the EO scheme suffers from a much higher computation cost, compared to other schemes. However, when the edge computation capacity becomes large, the total system cost of the EO scheme will reduce significantly and can achieve a lower cost than that the CO scheme (i.e. when the edge capacity is 5GHz). This result can provide more insights into how to allocate properly allocation for MEC server to enhance the overall offloading performance for the MECCO system. Again, the proposed ADRLO scheme can obtain the lowest cost with a downward trend and thus exhibits the best performance when compared to the DRLO scheme and other baselines.

Meanwhile, Fig. 9(b) shows the total offloading cost versus the system bandwidth allocation. Based on the simulation result, it is clear that the all the curves of offloading schemes decrease
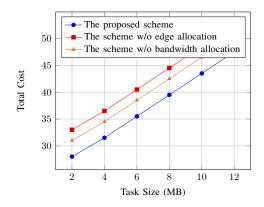
Fig. 10: Total costs of offloading schemes.

gradually with the increment of the total system bandwidth. The reason behind this observation that bandwidth allocation is always significant in improving the data transmission rate between MDs and edge-cloud server. As a result, this solution will reduce the transmission delay and the total offloading cost, accordingly. Further, by applying the proposed dynamic offloading policy, our approach using reinforcement learning can adjust dynamically how much bandwidth resource should be allocated to a certain MD based on MDs task size so as to achieve the optimal offloading for the MECCO system.

We also analyze the efficiency of the proposed ADRLO scheme by comparing to other two baseline solutions: *the proposed scheme without edge allocation* and *the proposed scheme without bandwidth allocation*. Note that for these baselines, edge computation and bandwidth resources are allocated equally to all MDs in the MECCO system regardless of task sizes. As indicated in Fig. 10, the proposed scheme, which takes edge resource and bandwidth resource into account, can achieve the best performance in terms of minimum total offloading costs, compared to other benchmarks. Obviously, by jointly optimizing both offloading decision and resource allocation, our proposed algorithm can achieve effective computation cost savings and improve significantly the offloading performance of the MECCO system.

VII. CONCLUSION AND FUTURE WORKS

In this paper, we have jointly studied access control and computation offloading by combining blockchain and DRL for the MECCO systems in IoT networks. We have considered a general IoT scenario where multiple MDs can offload its task to edge or cloud server to be executed cooperatively. First, to improve security for computation task offloading, we propose a new access control mechanism enabled by smart contracts and blockchain to manage access of MDs

with the objective of preventing malicious offloading access and preserving cloud resources. Then, we propose a novel DRL-based offloading scheme to obtain the optimal offloading policy for all MDs in the IoT network. We formulate task offloading decision, edge resource allocation and bandwidth allocation as a joint optimization problem, which is solved efficiently by an advanced DQN algorithm in a fashion the total offloading cost of computation latency and energy consumption is minimized. We conducted both real experiments and numerical simulations to evaluate the effectiveness of the proposed scheme. The implementation results showed that our scheme can provide high security to the MECCO system while achieving significant performance improvement with minimum offloading costs, compared to the other baseline methods.

In the future, we will consider light-weight blockchain designs so that the access control framework can be designed and deployed directly at the edge layer, which will be promising to provide time-sensitive network management services for offloading systems. Moreover, in the future 5G IoT network, due to the rapidly growing traffic, it is very challenging for static base stations (i.e. access point, router) to support offloading demands of numerous IoT devices. Motivated by our previous work [52], the use of unmanned aerial vehicle (UAV) with its high mobility and flexibility can be a promising solution [53] that can assist base stations to offload IoT data workload to edge-cloud servers.

References

[1] A. Al-Fuqaha et al., "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015.

[2] S. M. Riazul Islam, Daehan Kwak, MD. Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," *IEEE Access*, vol. 3, pp. 678-708, 2015.

[3] A. Zanella et al., "Internet of Things for Smart Cities," *IEEE IoT Journal*, vol. 1, no. 1, pp. 22-32, Feb. 2014.

[4] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava, "A Survey of Computation Offloading for Mobile Systems" *Mob. Netw. Appl.*, pp. 129-140, 2013.

[5] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[6] Y. Mao, C. You et al., "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.

[7] B. P. Rimal et al., "Mobile-edge computing vs. centralized cloud computing in fiber-wireless access networks," in *IEEE Conference on Computer Communications Workshops* , pp. 991-996, 2016.

[8] H. Zhou et al.,"A Survey on Mobile task offloading Technologies," *IEEE Access*, vol. 6, pp. 5101-5111, 2018.

[9] Akherfi, K. et al.,"Mobile cloud computing for computation offloading: Issues and challenges," *Appl. Comput. Inform.*, pp 116, 2018.

[10] J. Zhang, B. Chen, Y. Zhao, X. Cheng and F. Hu, "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues," *IEEE Access*, vol. 6, pp. 18209-18237, 2018.

[11] Z. Xiao and Y. Xiao, "Security and Privacy in Cloud Computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 843-859, 2013.

[12] T. M. Fernndez-Carams et al., "A Review on the Use of Blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32979-33001, 2018.

[13] S. Huh, S. Cho and S. Kim, "Managing IoT devices using blockchain platform," in 19th *International Conference on Advanced Communication Technology (ICACT)*, 2017, pp. 464-467.

[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Yellow Paper. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[15] Dinh C. Nguyen, Pubudu N. Pathirana, Ming Ding, and Aruna Seneviratne, "Blockchain for Secure EHRs Sharing of Mobile Cloud based E-health Systems," *IEEE Access*, vol. 7, pp. 66792-66806, 2019.

[16] H. G. Do and W. K. Ng, "Blockchain-Based System for Secure Data Storage with Private Keyword Search," in *IEEE World Congress on Services (SERVICES)*, Honolulu, HI, 2017, pp. 90-93.

[17] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen, "BlendCAC: A Smart Contract Enabled Decentralized Capability-Based Access Control Mechanism for the IoT," *Computers*, 2018.

[18] C. H. Liu, Q. Lin and S. Wen, "Blockchain-enabled Data Collection and Sharing for Industrial IoT with Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, 2019.

[19] O. J. A. Pinno, A. R. A. Gregio and L. C. E. De Bona, "ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT," in *IEEE Global Communications Conference*, pp. 1-6, 2017.

[20] I. Satoh, "Toward Access Control Model for Context-Aware Services Offloaded to Cloud Computing," in *IEEE 35th Symposium on Reliable Distributed Systems Workshops (SRDSW)*, pp. 7-12, 2016.

[21] J. Xu, L. Chen, K. Liu and C. Shen, "Designing Security-Aware Incentives for Computation Offloading via Device-to-Device Communication," *IEEE Transactions on Wireless Communications*, vol. 17, no. 9, pp. 6053-6066, Sept. 2018.

[22] X. He et al., "Energy-Efficient MobileEdge Computation Offloading for Applications with Shared Data," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1-6.

[23] X. Lyu et al., "Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435-3447, April 2017.

[24] J. Zheng et al., "Dynamic Computation Offloading for Mobile Cloud Computing: A Stochastic Game-Theoretic Approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771-786, 1 April 2019.

[25] J. Du, L. Zhao, J. Feng and X. Chu, "Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems With Min-Max Fairness Guarantee," *IEEE Trans. on Communications*, vol. 66, no. 4, pp. 1594-1608, April 2018.

[26] I. Comsa et al., "Towards 5G: A Reinforcement Learning-Based Scheduling Solution for Data Traffic Management," *IEEE Trans. on Network and Service Management*, vol. 15, no. 4, pp. 1661-1675, 2018.

[27] R. S. Sutton, A. G. Barto et al., "Reinforcement learning: An introduction," *MIT press*, 1998.

[28] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38, Nov. 2017.

[29] Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[30] X. Chen et al., "Optimized Computation Offloading Performance in Virtual Edge Computing Systems via Deep Reinforcement Learning," *IEEE Internet of Things Journal*, 2019.

[31] M. Min et al., "LearningBased Computation Offloading for IoT Devices With Energy Harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930-1941, Feb. 2019.

[32] Y. Cui, Y. Liang and R. Wang, "Resource Allocation Algorithm With Multi-Platform Intelligent Offloading in D2D-Enabled Vehicular Networks," *IEEE Access*, vol. 7, pp. 21246-21253, 2019.

[33] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795-2808, October 2016.

[34] D. Shibin and G. J. W. Kathrine, "A comprehensive overview on secure offloading in mobile cloud computing," in *4th International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, 2017, pp. 121-124.

[35] S. Han et al., "Energy Efficient Secure Computation Offloading in NOMA-based mMTC Networks for IoT," *IEEE Internet of Things Journal*, 2019.

[36] Elgendy et al., "An Efficient and Secured Framework for Mobile Cloud Computing," *IEEE Trans. on Clou. Comput.*, 2019.

[37] Y. Lin et al., "Local Authentication and Access Control Scheme in M2M Communications With Computation Offloading," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3209-3219, Aug. 2018.

[38] I. Satoh, "Toward Access Control Model for Context-Aware Services Offloaded to Cloud Computing," in *IEEE 35th Symposium on Reliable Distributed Systems Workshops (SRDSW)*, Budapest, 2016, pp. 7-12.

[39] S. Tu and Y. Huang, "Towards efficient and secure access control system for mobile cloud computing," *China Communications*, vol. 12, no. 12, pp. 43-52, December 2015.

[40] T. Li et al., "On Efficient Offloading Control in Cloud Radio Access Network with Mobile Edge Computing," in *IEEE 37th International Conference on Distributed Computing Systems*, 2017, pp. 2258-2263.

[41] Y. Zhu, H. Hu, G. Ahn, D. Huang and S. Wang, "Towards temporal access control in cloud computing," in *Proceedings IEEE INFOCOM*, Orlando, FL, 2012, pp. 2576-2580.

[42] Mora-Gimeno FJ, Mora-Mora H, Marcos-Jorquera D, Volckaert B, "A Secure Multi-Tier Mobile Edge Computing Model for Data Processing Offloading Based on Degree of Trust," *Sensors Journal*. 2018.

[43] Muthanna Ammar et al., "Secure and Reliable IoT Networks Using Fog Computing with Software-Defined Networking and Blockchain," *Journal of Sensor and Actuator Networks*, 2019.

[44] M. Liu et al., "Computation Offloading and Content Caching in Wireless Blockchain Networks With Mobile Edge Computing," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 11, pp. 11008-11021, Nov. 2018.

[45] H. Van Hasselt et al., Deep reinforcement learning with double q-learning," in *Conf. Artificial Intell.*, pp. 2094-2100, 2016.

[46] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," arXiv:1511.06581, 2015.

[47] Amazon Web Services (AWS) - Cloud Computing Services. [Online]. Available: https://aws.amazon.com/.

[48] Solidity 0.5.3 documentation. [Online]. Available: https://solidity.readthedocs.io/en/develop/.

[49] Ethereum on Android. [Online]. Available: https://github.com/ethereum/go-ethereum/wiki/Ethereum-on-Android.

[50] Lightweight Java library for Ethereum clients. [Online]. Available: https://github.com/web3j/web3j.

[51] A. Martn, A. Ashish et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/.

[52] Dinh C. Nguyen, Pubudu N. Pathirana, Ming Ding, and Aruna Seneviratne, "Secrecy Performance of the UAV enabled Cognitive Relay Network," in *IEEE 3rd Int. Conf. on Communication and Information Systems*, Dec. 2018.

[53] Azade Fotouhi, Haoran Qiang, Ming Ding et al., "Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges," *IEEE Communications Surveys & Tutorials*, 2019.