Thangaraj and Raja Sree
*Journal of Cloud Computing* (2024) 13:88
https://doi.org/10.1186/s13677-024-00599-8

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

# MSCO: Mobility-aware Secure Computation Offloading in blockchain-enabled Fog computing environments

Veni Thangaraj[1] and Thankaraja Raja Sree[2*]

**Abstract**

Fog computing has evolved as a promising computing paradigm to support the execution of latency-sensitive Internet of Things (IoT) applications. The mobile devices connected to the fog environment are resource constrained and non-stationary. In such environments, offloading mobile user's computational task to nearby fog servers is necessary to satisfy the QoS requirements of time-critical IoT applications. Moreover, the fog servers are also susceptible to numerous attacks which induce security and privacy issues.Offloading computation task to a malicious fog node affects the integrity of users' data. Despite the fact that there are many integrity-preserving strategies for fog environments, the majority of them rely on a reliable central entity that might have a single point of failure. Blockchain is a promising strategy that maintains data integrity in a decentralized manner. The state-of-art blockchain offloading mechnanisms have not considered the mobility during secure offloading process. Besides, it is necessary to ensure QoS constraints of the IoT applications while considering mobility of user devices. Hence, in this paper, Blockchain assisted Mobility-aware Secure Computation Offloading (MSCO) mechanism is proposed to choose the best authorized fog servers for offloading task with minimal computational and energy cost. To address the optimization issue, a hybrid Genetic Algorithm based Particle Swarm Optimization technique is employed. Experimental results demonstrated the significant improvement of MSCO when compared to the existing approaches in terms of on average 11 % improvement of total cost which includes the parameters of latency and energy consumption.

**Keywords** Fog computing, Internet of things, Blockchain, Computation offloading, Mobility

## Introduction

The growth of IoT has made it possible for numerous intelligent mobile gadgets to infiltrate people's daily lives and improve their quality of life. However, the Mobile Devices (MD) running the IoT applications are resource constraint (limited storage, computational capacity and power) which mandate Cloud server to cater the need of it [1]. However, the cloud has drawbacks such as geographically centralized design, a lack of mobility support, and a multi-hop distance from the data source that negatively affect the latency and response time of time-critical applications like healthcare [2]. To address this problem, a new concept known as "Fog computing" has evolved. It offers processing and storage resources close to MD, reducing the need for frequent interaction with cloud servers. As a result, IoT environments make use of fog-assisted Cloud computing environments to run latency-sensitive applications [3].

In fog environments, the devices such as routers, gateways and Road Side Unit (RSU), light weight server etc., are considered as fog nodes which processes the computational task received form MDs. Computation offloading has been envisioned as a promising approach

*Correspondence:
Thankaraja Raja Sree
trajasree87@gmail.com
[1] Department of CSE, National Institute of Technology - Calicut, Kattangal, Kozhikode 673601, Kerala, India
[2] School of CSE, Vellore Institute of Technology - Chennai, Chennai, Melakottaiyur, Tamilnadu, India

to delegate MDs' task to the fog devices to satisfy the Quality of Service (QoS) constraints of IoT applications with minimal resource consumption. In the offloading process, MDs offload their resource-hungry tasks to a remote fog or cloud computation environment to alleviate the burden of the work and decrease the computation overhead and costs compared with local execution. Both MDs and fog servers have to necessarily operate offloading frameworks to fulfill computation offloading [4–12].

The key factors such as continuous change of location of MDs (mobility), fog device heterogeneity complicate the computational offloading process. Additionally, the fog servers are open to numerous types of attacks.The integrity of end users' data is harmed when computing tasks are offloaded to a rogue fog node. Hence, it is necessary to check veracity of a fog server before doing computation offloading. Although there are many safe solutions in fog settings, the most of them are reliant on centralized servers, which have a single point of failure issue [9, 13].

Blockchain is a promising strategy that protects data integrity in a decentralized manner [14]. It's a distributed database that works without the help of a third party and stores data in blocks. It makes use of the Proof-of-Work (PoW) consensus system, which the miners employ to confirm the accuracy of the data belonging to the end users. The application of PoW consensus protocol in the aspect of secure offloading has been well studied in state-of-the art research works in [15–19]. Hence, the PoW stratgy used in this paper.

While there are other blockchain-based computation offloading strategies [15–19], the majority of them do not take into account the mobility and security of fog devices, which are crucial in fog computing environments. The state-of-art blockchain offloading mechnanisms have not considered the mobility during secure offloading process. Besides, it is necessary to ensure QoS constraints of the IoT applications while considering mobility of user devices. Thus, in this paper, Blockchain assisted Mobility-aware Secure Computation Offloading (MSCO) mechanism is proposed to choose the best authorized fog servers for offloading task with minimal computational delay and energy consumption.

MSCO exploits blockchain technology to offer secure and decentralized offloading service to end users. The task offloading to fog servers with minimal cost is NP-hard problem [20]. Hence, it is very challenging to utilize traditional greedy search methods. In MSCO, a hybrid GA-PSO technique is proposed to address the optimization problem. Key contributions of this paper are outlined as follows:

- Blockchain based fog computing framework has been designed to ensure secure computation offloading process.
- A dynamic mobility aware computational offloading technique is proposed to attain QoS of mobile user with low latency and energy consumption.
- Experiments have been carried out to validate the efficiency and effectiveness of MSCO.

The remaining section of this paper is organized as follows: "Related works" section presents the related work in block chain based computation offloading mechanisms in fog environments. The proposed MSCO framework is described in "MSCO architecture" section. "Performance evaluation" section illustrates the experimental evaluation. "Conclusion" section concludes with recommendations for further development.

## Related works

Due to the rapid development of IoT systems, efficient computation offloading in fog environment is a current and significant field of research interest.

Shah-Mansouri et al. [5] Formulated QoS maximization problem and proposed a computational offloading model in order to capture the competition between IoT users. Their experimental results achieved a significant reduction in latency of IoT applications. The computation offloading issue in the Fog computing settings was addressed by Guo et al. [6] by proposing a greedy offloading technique based on game theory. On the basis of the user's mobility, Wang et al. [7] suggested an opportunistic computation offloading approach. The statistic property of contact rates was utilized to design the optimal-offloading problem, and the convex optimization method was then applied to determine how much computation should be sent to other devices.

In order to take into account cloud and fog offloading destinations, Meng et al. [8] developed the hybrid computation offloading issue. The job distribution for computation offloading was streamlined to meet deadline requirements while consuming as little energy as possible. With the aim of minimizing energy usage while taking execution delay limitations into consideration, Chang et al. [9] suggested an energy-efficient computation offloading technique using queuing theory.

With consideration for resource instability, resource heterogeneity, and task interdependency in the vehicular cloud, Sun et al. [21] suggested a cooperative task scheduling system for computation offloading. The issue was resolved using a modified genetic algorithm. Multi-user computation offloading problem in dynamic environment, where mobile user and wireless channels become

active or inactive dynamically, was formulated by Zheng et al.

To reduce energy usage, Al-shatri et al. [22] created a distributed computational offloading technique that chooses whether jobs should be partially transferred to clouds or fog. A multi-layer computation offloading framework called FlopCoin was suggested by Chatzopoules et al. [15] and includes a credit-based incentive program for mobile users. In order to schedule resources in mobile blockchain networks, Luong et al. [16] suggested an optimal action method that takes advantage of deep learning. For fog computing environments based on blockchain, Duo et al. [17] presented a mobility aware computation offloading approach. To achieve data integrity and implement balanced offloading techniques, Xu et al. [18] presented a blockchain-based migration mechanism.

Price-based resource management for blockchain networks was suggested by Xiong et al. [19]. Deep reinforcement learning-based task offloading in blockchain-aided fog environments was proposed by Nguyen et al. [23]. A blockchain-based migration approach is suggested by Xu et al. [10] to maintain data integrity and achieve the goal of balanced offloading strategies.

The mobility aware task scheduling for virtual fog environments has been presented in [24]. The mobility aware proximal policy optimization (MAPPO) which managed mobility, reduced the transmission rate and increased throghput. An autonomous computation offloading strategy using a deep learning based hybrid approach for mobile edge computing has been proposed in [25]. Mobility aware blockchain enabled offloading using Linear Search Based Task Scheduling (LSBTS) for vehicular fog cloud computing has been proposed in [26]. A deadline and priority aware task offloading using multilevel feedback queuing for fog environments has been proposed in [27]. The mobility aware task scheduling for healthcare applications has been proposed in [28].

It is observed from the existing literatures that most techniques have failed to consider the end-user's mobility and security features, which is critical in fog computing environments. In MSCO, the features such as mobility and security are jointly considered to achieve the QoS of IoT applications with minimal delay and energy consumption. The overview of existing literature is represented in Table 1.

## MSCO architecture

This section details MSCO, a framework for mobility aware secure computation offloading in fog computing environments. The major goal of MSCO is to choose the best authorized fog server by means of blockchain technology to satisfy the QoS constraints of IoT applications with minimal delay and energy consumption. MSCO architecture is depicted in Fig. 1. It consists of cloud layer, fog layer and IoT layer. All the three layers in MSCO architecture are connected via wireless medium. The details of each layer are described as follows:

- **IoT Layer:** User MDs are in charge of processing time-critical IoT applications in this layer. The MDs are resource-constrained devices that transfer the task to the fog or cloud layer when processing exceeds their computing capacity. Each MD in this

**Table 1** Summary of the related literature

| Research work | Methodology | Mobility | Delay | Energy | Security |
|---|---|---|---|---|---|
| Shah-Mansouri et al. [5] | Computation offloading game | ✓ | ✓ | | |
| Guo et al. [6] | Two-tier game theoretic greedy offloading scheme | | ✓ | | |
| Wang et al. [7] | Convex Optimization | | ✓ | ✓ | |
| Meng et al. [8] | Computation energy efficiency based task offloading | | ✓ | ✓ | |
| Chang et al. [9] | Queuing Theory | | ✓ | ✓ | |
| Sun et al. [21] | A modified genetic algorithm based scheduling scheme | ✓ | ✓ | | |
| Zheng et al. [29] | A stochastic game-theoretic approach | ✓ | ✓ | | |
| Al-shatri et al. [22] | Distributed decision algorithm | ✓ | ✓ | ✓ | |
| Chatzopoules et al. [15] | Hidden Market design approach | | ✓ | ✓ | ✓ |
| Luong et al. [16] | Deep learning approach | | | ✓ | ✓ |
| Dou et al. [17] | Blockchain based mobility aware offloading method | ✓ | ✓ | | ✓ |
| Xu et al. [18] | Blockchain based computation offloading method | | ✓ | | ✓ |
| Xiong et al. [19] | Two-stage stackelberg game theory approach | | ✓ | | ✓ |
| Nguyen et al. [23] | Deep reinforcement learning approach | | ✓ | ✓ | ✓ |
| Xu et al. [10] | Blockchain, Nondominated sorting genetic algorithm III | | ✓ | ✓ | ✓ |
| **MSCO** | Blockchain, Hybrid GA+PSO | ✓ | ✓ | ✓ | ✓ |

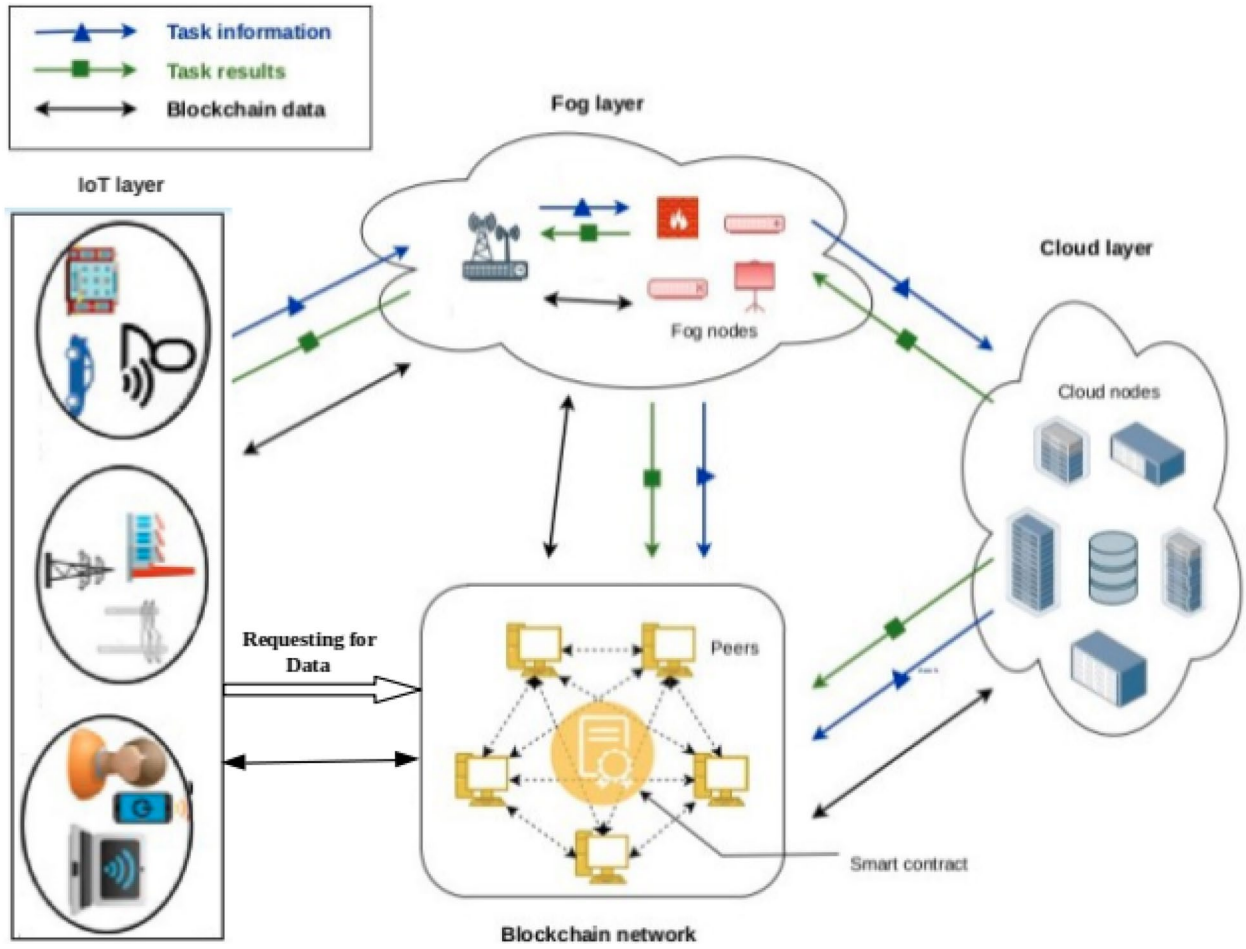



**Fig. 1** Blockchain-based Secure Offloading Framework

tier has a blockchain account that allows them to join the network and offload tasks to the fog layer.

- **Fog Layer:** This fog layer is made up of geographically dispersed fog devices including routers, gateways, RSU, micro-data centers, etc. that are intelligent enough to handle tasks from MDs. The fog gadgets typically have small-scale computing capabilities. If processing requires more computational power than it has available, the task is forwarded to the Cloud layer.
- **Cloud Layer:** This layer consists of numerous top-tier servers that can process and store a significant amount of data.

### Block generation for offloading process in fog environments

In essence, blockchain is a distributed database. Each data block in a blockchain comprises information about a transaction that is used to validate the data and create the following block. Blockchain is used to track the unloading transactions and guarantee the security of the data. Each computational task that is offloaded to a fog server is registered as a transaction in a block, which will be added to MSCO as a blockchain after a PoW-based consensus verification.

In addition, there is a PoW with verifiability and traceability, which is the result of computing a hash function. When the transaction is created, it is treated as an unconfirmed transaction for each node. It is equivalent to solving the PoW proof mechanism of math problem. The node in the blockchain network who solves the PoW faster will get the power to generate a new block and broadcast all the time-stamped transactions recorded in the block to the whole network.

After obtaining the verification and consent of other nodes, the new block is attached to the current blockchain [30]. Each block contains the hash value of previously generated block. Thus if a block is modified by an attacker, all the previously generated blocks need to be

modified which is almost impossible. Thus data integrity is guaranteed. Fog server resource monitoring can be done by using ledgers in blockchain. The ledger is dynamically updated. The concerned fog server can be assigned to the task when the request comes in, depending on the values of fog servers' resource usage and their service waiting time. A new record is created simultaneously, and the ledger is updated accordingly. By using blockchain techniques, MDs and Fog servers can store the entire history of transactions. Fog servers use their private keys to signature the actual updated workloads, their geographical positions, and the updated information. In addition, Mds use their private keys to signature the offloading transactions. Since each Mds can store the entire history of transaction, it could easily determine which Fog server should be selected to utilize to offload its specific computation by using the proposed GA-PSO optimization algorithms.

Because of the fact that mining secures the whole system, one of the node in the blockchain network can be elected as the temp centralized node. The first one who solve the mathematical problem will win the election and has the right to add the block which associates with all the transactions after the last block to the blockchain. Considering the fact that the geo-distributed Fog servers have limited computing power and their main tasks are helping mobile devices to release their computation workload by leveraging offloading technology.

Hence, in proposed system, the mining nodes whose computing resource are plentiful to validate new transactions and record them on the global ledger, and these mining nodes could be the dedicated high performance machines and protected by the service operators. If one fog server leaves the network due to power failure etc., its private key would be regenerated when it rejoins the network. Therefore, MDs joins the decentralized network and synchronize the history of transaction records, and also, Mds have the service waiting time of every Fog server at any time by means of blockchain technology.

#### Computation offloading model

In this part, the mathematical formulation of task offloading process is presented, Table 2 represents the various notations used in this paper.

The offloading decision making process has been depicted in Fig. 2. The fog servers are clustered based on the current location of MDs. The edge gateway queries the blockchain to obtain the fog server information for offloading decision process.Fog server resource monitoring can be done by using ledgers in blockchain. The ledger is dynamically updated. The concerned fog server can be assigned to the MD's task when the request comes in, depending on the values of fog servers' resource usage and their service waiting time.

**Table 2** Mathematical notations

| Notation | Description |
|---|---|
| MD | Set of Mobile Devicess |
| $D_n$ | Data size of $n^{th}$ computational task |
| $C_n$ | Total amount of CPU cycles needed to finish the task |
| $\tau_n$ | Maximum tolerable delay of an application |
| F | Set of all fog server |
| $f_i$ | Fog server i |
| $f_{vi}$ | Computational speed of $f_i$ |
| $ST_i$ | Time spent inside the scope of a fog server's service i |
| $x_i$ | Assigned offloading computation for $f_i$ |
| $P_i$ | Assignment decision for $a_i$ |
| $M_i$ | Migration decision for $a_i$ |
| $\Theta_c$ | Power consumption of fog server during computation process |
| $\Theta_t$ | Power consumption of MD during offloading task to fog server |
| MT | Mobility Trajectory |
| R | Set of fog server candidates under the service coverage area of MDs |
| $\Delta_w$ | Estimated waiting time in one location |
| $D_t$ | Distance Threshold |
| $\Delta_d$ | Estimated distance between two successive locations |
| $l_i$ | Geographical location of MD |
| $E_i$ | Energy cost of the application $a_i$ |
| $T_i$ | Total computational latency of the $a_i$ |
| $tw_i$ | Waiting time for fog server $f_i$'s availability |
| $N$ | Number of Fog Clusters |

The set of MDs application are represented as a set of A= $\{a_1, a_2, ...a_n\}$. Each $a_i$ must accomplish a computation task. The computation task has its attributes such as data size of the computation task ($D_n$), the total amount of CPU cycles needed to finish the computation task ($C_n$), maximum tolerable delay ($\tau_n$). The set of fog servers are represented as F $=\{f_1, f_2...f_n\}$. Each fog device has the computation speed which is represented as $f_{vi}$.

During the execution of a certain application, the program profiler keeps track of the program's many performance metrics, such as execution time, acquired memory, thread CPU time, number of instructions, and method calls. Hence, The information of ($D_n$) and ($C_n$) of a particular application can be obtained using program profilers [31].

The mobility is an important feature which should be considered during offloading procedure. In order to predict the movement patterns of the MDs from their historical movement log, We have used the mobility model according to [18]. Based on the predicted next location sequences and the current location, a set of fog devices is

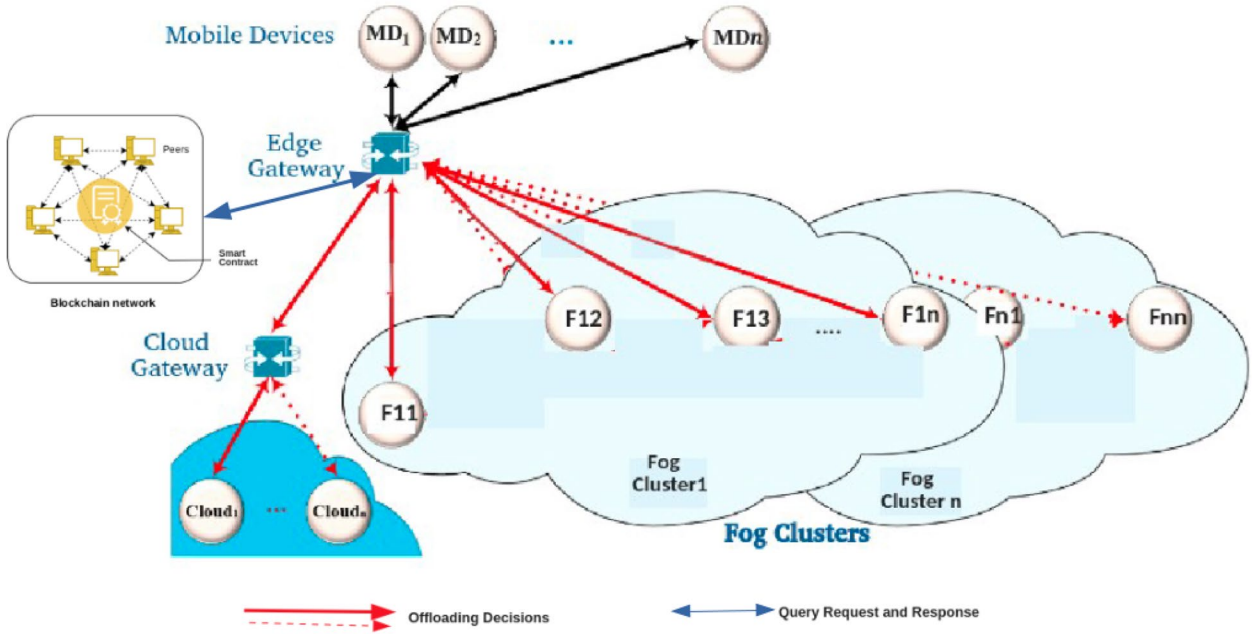



**Fig. 2** Offloading Decision-Making Process

selected. First, the users' regular movement patterns can be modelled from their historical mobility traces using a variant of multi-variable Bayesian network, where the location sequences of the visit, type of the places visit in different temporal scales are considered. In the next step, the location sequence can be predicted effectively using the variable-order Markov chain approach.

We have used the mobility model according to [18]. The staying time (ST) is defined as follows:

**Definition 1** Staying Time is defined as the maximal contact time of MDs within service coverage of fog servers.

The individual mobility model can be described based on each MDs mobility trajectory.

**Definition 2 (Mobility Trajectory):** Each MDs mobility trajectory can be described as a series of locations and jumps based on the mobility of MDs. It is denoted as "MT". For example, the mobility trajectory can be described as $l_1, \Delta w_1, \Delta d_1, l_2, \Delta w_2, \Delta d_2, l_3, \Delta w_3, \Delta d_3,$ ..., where $l_i$ means geographical location, $\Delta w_i$ is the staying time, $\Delta d_1$ is the distance between the current location to the next location. $\Delta w_i, \Delta d_1$ are chosen based on the probability distribution $P(\Delta w), P(\Delta d)$ respectively. MDs average velocity is mv.

The computation offloading model can be developed for the fog layers, with an emphasis on computation delay and energy usage analysis. Due to powerful computing capacity of cloud, the task offloading process in cloud is ignored in this paper. Due to mobility feature mobile users, the computational task can be offloaded to any of the fog servers under its coverage area . Let $x_i$ be the computation task to be assigned to the particular fog server. Let $P_i$ be the offloading decision variable it could be one when the computational task is offloaded to particular fog server within its coverage area $1 \leq i \leq R$.

$$P_i = \begin{cases} 1, \text{ if fog server } f_i \text{ is utilized} \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

$$M_{ijk}^t = \begin{cases} 1, \text{ if application } a_i \text{ migrates from } f_j \text{ to } f_k \text{ at a time t} \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

Therefore,

$$\sum_{i=0}^{|R|} P_i.x_i = C_n \tag{3}$$

In order to guarantee the deadline of task $(\tau_n)$, the computation should be completed before the location $l_s$, where

$$s = \arg\min_k \left| \sum_{k=1} \Delta w_k + \left( \frac{\Delta d_k}{mv} \right) - \tau_n \right| \tag{4}$$

The computational task offloading $(x_i)$ for fog server $f_i$ can be expressed as

$$x_i = (ST_i - tw_i).f_{vi} \tag{5}$$

where $ST_i$ be the staying time of $MD_i$ under the service coverage of fog server $f_i$, $tw_i$ is the waiting time for fog server $f_i$'s availability, $f_{vi}$ is Computational speed of $f_i$.

The total computational latency can be expressed as

$$T_i^c = \sum_{i=1}^{n}\sum_{j=0}^{|R|}\sum_{t=0}^{T}\left(\frac{D_i}{r_i}+\frac{x_i}{f_{vj}}\right)P_{ij}^t + \sum_{i=1}^{n}\sum_{j=0}^{|R|}\sum_{k=0}^{|R|}\sum_{t=0}^{T}\left(\frac{D_i}{r_i}+\frac{(C_i-x_i)}{f_{vk}}\right)M_{ijk}^t \tag{6}$$

where $r_j$ be the transmission rate of $a_j$ which is calculated as per [23], $D_j$ is the Data size of $j^{th}$ computational task, $x_j$ is Assigned offloading computation for $f_j$, $f_{vj}$ is Computational speed of $f_j$.

Energy cost ($E_j$) for offloading to the particular fog server includes the energy consumption for transmitting data and for execution. Let $\Theta_t$ be the power consumption of $MD_j$ for the transmitting task from MD to fog server. $\Theta_c$ be the power consumption of fog server during the computational process. The energy cost is described as follows:

$$E_i = \sum_{i=1}^{n}\sum_{j=0}^{|R|}\left(\frac{\Theta_t.D_i}{r_i}+\frac{\Theta_c.x_i}{f_{vj}}\right)P_{ij}^t + \sum_{i=1}^{n}\sum_{j=0}^{|R|}\sum_{k=0}^{|R|}\left(\frac{\Theta_t D_i}{r_i}+\frac{\Theta_c.(C_i-x_i)}{f_{vk}}\right)M_{ijk}^t \tag{7}$$

The main objective is to minimize the computational latency and energy cost for all MDs in MSCO system. The cost function $Cost_i$ of $MD_i$ can be formulated as follows

$$Cost_i = \alpha^c T_i + \alpha^e E_i \tag{8}$$

where $\alpha^c, \alpha^e \in [0,1] (i \in M)$ denote the weight of the computational latency and energy consumption, respectively.

Hence,the optimization problem can be modeled as

$$\begin{aligned} Y = \min_{x_i,P_i,\forall i\in R} \quad & \sum_{i}^{R} Cost_i \\ \text{s.t.} \quad & (C1): \alpha_i^c, \alpha_i^e \in [0,1], \forall i \in R, \\ & (C2): \alpha_i^c + \alpha_i^e = 1, \forall i \in R, \\ & (C3): P_i \in [0,1], \\ & (C4): \left(\frac{D_j}{r_j}+\frac{x_j}{f_{vj}}\right) \le ST_i, \\ & (C5): \sum_i x_i.P_i = \tau_i. \end{aligned} \tag{9}$$

Here, the constraint (C1) and (C2) represent the binary offloading decision policy to offload to the fog server or to the cloud server. Constraint (C3) indicates Assignment decision for the fog server. Constraint (C4) refers to the offloading restriction of the fog server based on the MDs' duration of stay in the associated service coverage. Constraint (C5) shows that the magnitude of the whole computation carried out by MDs and all chosen fog servers is equal to the total computing demand of an IoT application.

Since the objective function Y in equation (8) is linear, and related variables are integer. Moreover, the decision to offloading among fog or cloud is binary. Thus, the proposed optimization function with mentioned restrictions can be mapped as the Mixed-Integer Programming (MIP) ( (i. e., binary programming), which is inherently an NP-Hard problem. In order to solve this problem, the proposed optimization probem should be relaxed in order to fit for the proposed GA and PSO algorithms. The relaxation of the MIP problem, Y', as follows:

$$\begin{aligned} Y' = \min_{x_i,P_i,\forall j\in N\forall i\in R} \quad & \sum_{i}^{R}\sum_{j}^{N} Cost_i \\ \text{s.t.} \quad & (C1): 0 \le \alpha_i^c \le 1, 0 \le \alpha_i^e \le 1, \forall i \in R, \\ & (C2): \alpha_i^c + \alpha_i^e = 1, \forall i \in R, \\ & (C3): 0 \le P_i \le 1 \\ & (C4): \left(\frac{D_j}{r_j}+\frac{x_j}{f_{vj}}\right) \le ST_i, \\ & (C5): \sum_i x_i.P_i = \tau_i. \end{aligned} \tag{10}$$

The task offloading to fog servers with minimal cost is NP-hard problem [19]. The meta-hueristic algorithms such as GA, PSO, Ant Colony Optimization (ACO) have been employed in existing literature to solve such optimization problems [12]. Hence, in MSCO, the hybrid of GA and PSO can be used to solve above optimization problem. GA is having the problem of more convergence time because of its large solution space and low fitness values of parent chromosomes. PSO has fast convergence, but it suffers from the problem of local optima. The hybrid

GA-PSO method is anticipated to operate more quickly than the GA (or) PSO algorithm.

Additionally, due to the employment of the GA mutation operator, which improves the accuracy of the solutions, the hybrid GA-PSO algorithm may not become stuck in the local optimal solution. Consequently, the aforementioned optimization problem can be solved using PSO in MSCO, a hybrid of GA. Algorithm 1 shows the blockchain assisted GA-PSO based computation offloading mechanism.

Algorithm 1 accpets the input such as application task attributes, each fog server's processing capacity and Mobility Trajectory. Initial step of Algorithm 1 is to find the list of authorized fog server for computation offfloading under ther service coverage area of MDs. Then the service waiting time of each fog server based on its current load by means blockchain methodlogy can be found. Finally, the optimal assignment of task to the fog server can be found using GA-PSO method.

**Algorithm 1** Blockchain assisted GA-PSO based computation offloading

**Input:** Application task attributes $(D_n, C_n, \tau_n)$, $ST_i$ set, $f_{vi}$, MT,$D_t$
**Output:** Optimal assignment of task to the fog server

1: Generate the cluster of fog servers whose distance is less than $D_t$ as $R_t$
2: Aggregate all fog servers according to MT as $R_p$
3: R $\leftarrow R_t \cap R_p$
4: Query blockchain to get service waiting time for R by (4)
5: Generate optimal assignment based on GA-PSO method using Algorithm2
6: Return assignment list

**Algorithm 2** Generating Candidate assignment list using hybrid GA-PSO method

**Input:** Initialized population X
**Output:** Optimal assignment of task to the fog server in the last iteration X*

1: //Apply GA over the population
2: i=1
3: **for** i $\leq$ I **do**
4: **for** the chromosome in X **do**
5: Evaluate objective functions by (6) and (7)
6: **end for**
7: **end for**
8: Perform crossover operation
9: Perform mutation operation
10: //Apply PSO over the GA generated population
11: For each particle in population, initialize particle weight and velocity
12: Calculate the fitness value ($Cost_i$) for each particle (6)
13: For each particle, consider the best position value $Pbest$ .
14: $TakeG_{best}$as the smallest fitness value.
15: Update position and velocity of each particle.
16: Repeat steps from 2 to 11 (until termination condition is satisfied).
17: Return assignment list X*

Algorithm 1 complexity depends on the size of R for each application. O(R|*logR*|) is the complexity of Algorithm 1.

## Performance evaluation

The details of the experimental parameters, comparison methods, performance metrics taken into account, and assessment results are presented in this section in order to show the effectiveness of MSCO.

### Experimental parameters

An Ubuntu Debian-powered computer with an Intel Core i5 processor and 8 GB of RAM was used for all of the experiments. The EUA-data set has been used in order to simulate the trajectories of mobile users and the locations of fog servers. The locations in the EUA-data set are randomly choose to generate the individual mobility model. According to the EUA-dataset, there are 817 mobile user locations which is depicted by (latitude, longitude).

An individual mobility model is generated by randomly selecting locations from the EUA- data set. The value of mobility model parameters is fixed based on the experimental results [17]. In MSCO, the staying time is an integer that is assumed to be chosen at random and follow a uniform distribution in [13, 29].

A private Ethereum blockchain network is set up in order to emulate the entire blockchain's run-time environment.The Ethereum network began by using a consensus mechanism that involved Proof-of-work (PoW). This allowed the nodes of the Ethereum network to agree on the state of all information recorded on the Ethereum blockchain. In order to simulate the whole blockchain's runtime environment, Docker has been used to build an image that supports the Etherenum environment. Ethereum is an open source distributed public blockchain network. It allows decentralized apps to be built on it with the help of Smart Contract functionality.

Table 3 shows details of all the experimental parameters. The performance of MSCO has been analyzed by the setting the parameters such as number of tasks, number of fog nodes, deadline of the application, service waiting time of the fog servers. The overall experimental settings is depicted in Table 4.

Experiments have been conducted for various levels of deadline of an application. For example, ***d*** be the execution time of an application when it is fully executed on MDs. Then 20%,80% and 110% of ***d*** have been fixed as the deadline of an application. MSCO compared against the following algorithms.

- **First-Fit** [3]: the task is offloaded to the first fog server that having enough capacity to execute the task.
- **Instant Offloading** [7]: the task is offloaded to the nearest fog server under its coverage area.
- **Random** [8]: the task is randomly allocated on the fog server under its coverage area while ensuring resource constraints.
- **BMO** [16]: The task is allocated to authorized fog server based on the utility value which is calculated based on the latency.

The metrics such as total cost, deadline satisfaction ratio has been used to validate the efficiency MSCO.

### Experimental results and discussion

***Impact of varying number of tasks:***

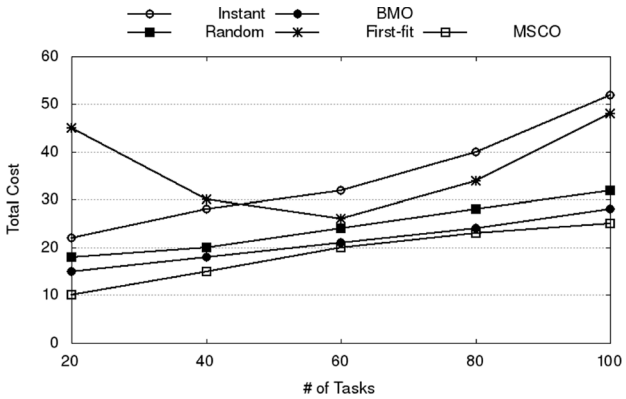
Figure 3 depicts the performance and cost of MSCO compared to the various offloading algorithms. It can be observed from the figure that with the increasing number of tasks, the total cost and service violation rate also increasing. However, MSCO outperforms the existing algorithms in terms of lower service violation rate and lower total cost. The reason is that, with the increasing number of tasks, MSCO offloads the task to better fog server under the service coverage of mobile users.

**Table 3** Experimental parameters

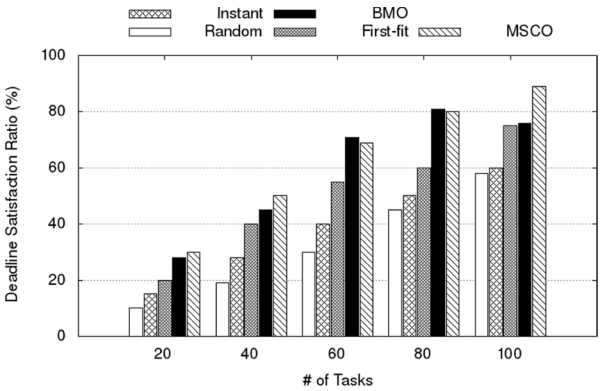
| Parameter | Value |
|---|---|
| Service coverage of each fog server (Meter) | [400, 800] |
| Computational capacity of fog servers (MIPS) | [3, 10] |
| IoT application computation instructions (MIPS) | [1, 3] |

**Table 4** Experimental settings

| Exp. | No. of Tasks | No. of fog servers | Deadline | Service waiting time |
|---|---|---|---|---|
| 1 | 10,20,...,100 | 10 | 80% | 100% |
| 2 | 100 | 1,2,...,10 | 80% | 100% |
| 3 | 100 | 10 | 20%, 30%, ...,110% | 100% |
| 4 | 100 | 10 | 80% | 5%, 15%,...,95% |




(a) Total cost




(b) Deadline Satisfaction Ratio

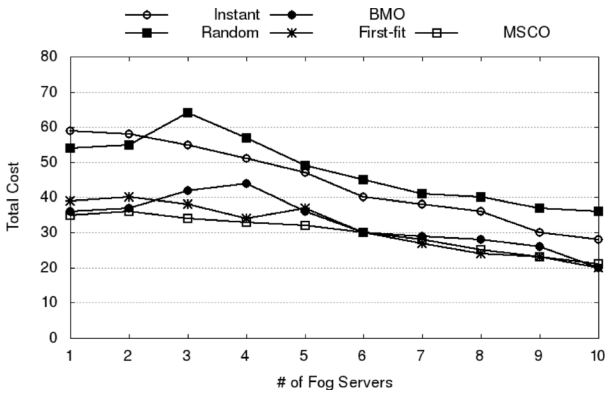
**Fig. 3** Impact of varying number of tasks

Due to the mobility of the mobile users, significant communication cost, service violation rate is increased in all other methods, and MSCO exploits this mobility and thus achieves significant performance comparing to other methods.

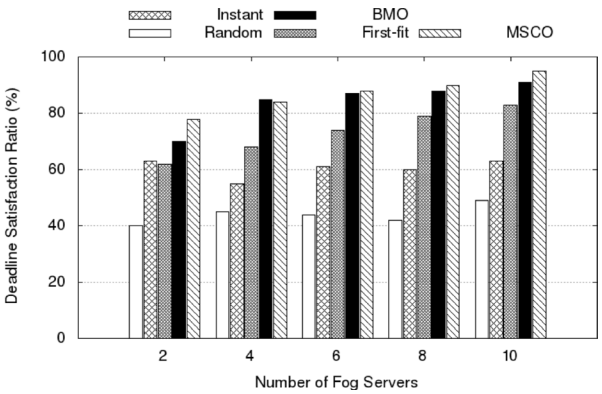
***Impact of varying number of fog nodes:***
Figure 4 shows that the total offloading cost versus the number of fog servers. According to the findings, MSCO significantly outperformed all other approaches in terms of performance. This is because MSCO has additional options for selecting the most affordable fog servers for computation offloading. This can be reflected in the deadline satisfaction ratio also Fig. 4.

***Impact of varying deadline of tasks:***
Figure 4 illustrates the performance of various offloading algorithms in terms of service violation cost under the various deadline constraints of the tasks. It can be observed that MSCO achieves lesser service violation and lessor cost comparing to all other offloading methods. This is due to the fact that MSCO selects best cost effective fog server for tasks with urgent deadline constraints with the consideration of moving users comparing to all other methods. This can be reflected in total cost Fig. 5.

***Impact of varying service waiting time of fog servers:***
The experiments have been conducted by varying the ratio of service waiting time with staying time and observed the service violation and total offloading cost. This is depicted in Fig. 6. It can be observed is that MSCO achieved better performance in comparison with the all other methods under various mobility constraints. This is due to the fact that comparing to all other offloading methods, MSCO effectively handles the mobility




(a) Total cost




(b) Deadline Satisfaction Ratio

**Fig. 4** Impact of varying number of fog servers

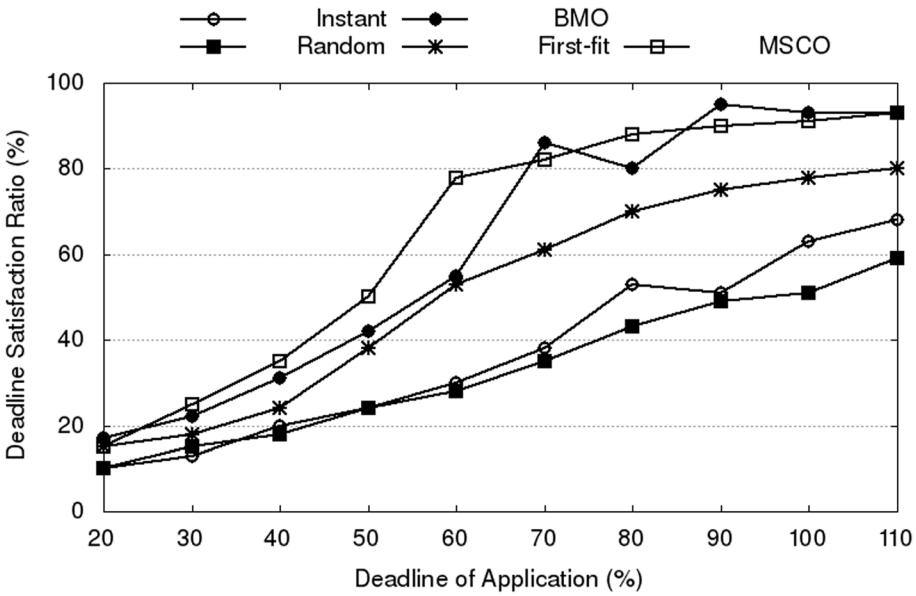
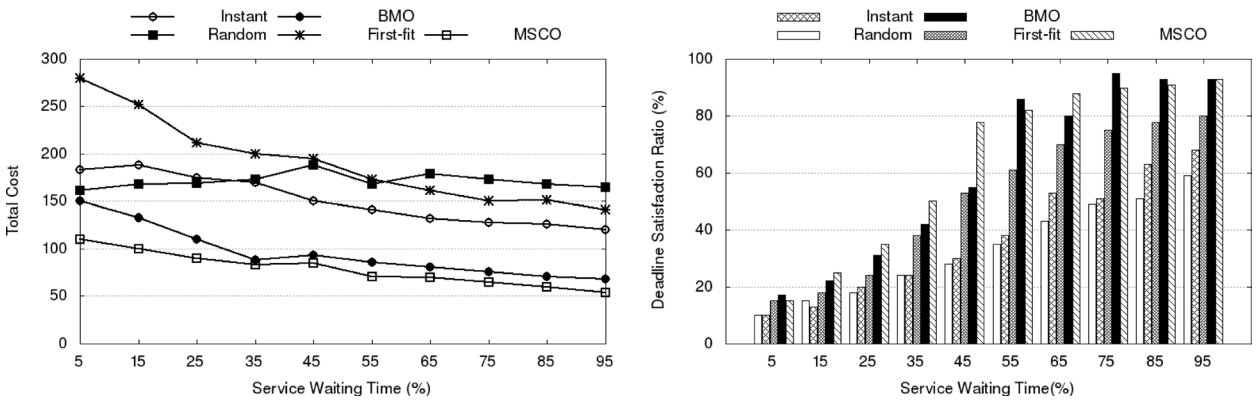



**Fig. 5** Impact of varying deadline of tasks




(a) Total cost (b) Deadline Satisfaction Ratio

**Fig. 6** Impact of varying service waiting time of fog servers

constraints and always chooses the best fog server for task offloading. The same can be observed in total offloading cost also.

#### *Impact of total processing time for various offloading user requests*

Every offloading transaction is signed using a private key in conjunction with a public key for device identification in accordance with the blockchain idea. The request permission can be completed using the smart contract algorithm after receiving the transaction from MDs [30, 31]. The request for task offloading is approved if the MD is confirmed by the smart contract.

The efficiency of blockchain in enhancing the security and sturdiness of fog environments in offloading scenarios has been demonstrated through experiments. The total processing time for the offloading request can be measured with number of offloading request. The results are depicted in Fig. 7. It is observed from Fig. 7 that MSCO achieves less processing time comparing to all other offloading methods. This is due to the fact that though, MSCO consumes more time for MD-fog server

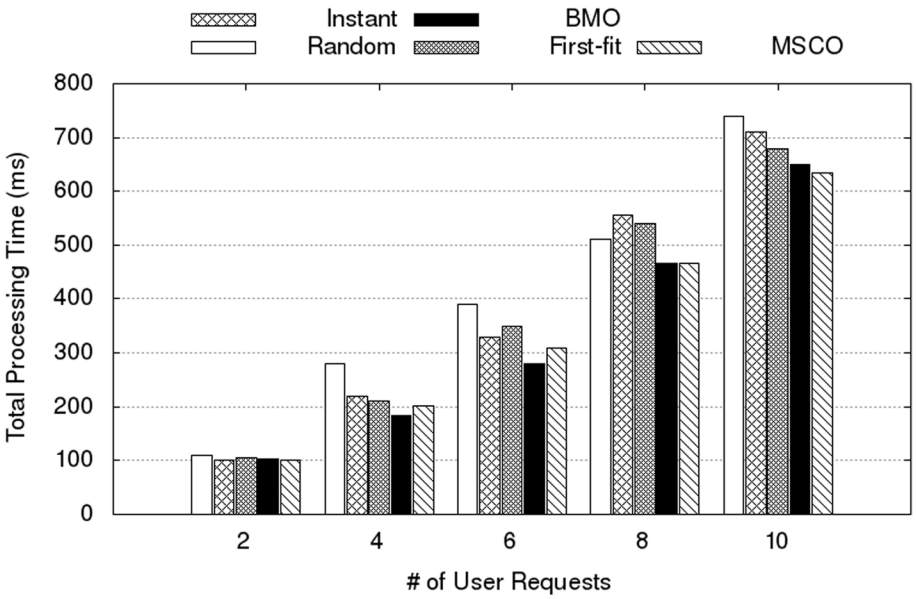



**Fig. 7** Impact of total processing time for various offloading user requests

authorization, it saves considerable amount of time by selecting optimal authorized fog server for offloading process compared to all other offloading methods.

## Conclusion

Efficient computation offloading with the consideration of mobility and security is a critical but challenging problem in fog computing environments. In this paper, MSCO, blockchain assisted mobility-aware secure computation offloading mechanism has been proposed to choose the best authorized fog servers for computation offloading by means of blockchain methodlogy. The hybrid genetic algorithm based particle swarm optimization technique has been used to achieve optimal offloading with minimal delay and energy consumption. Experimental results demonstrated the significant improvement of MSCO when compared to the existing approaches on average of 11 % improvement of total cost which includes the parameters of latency and energy consumption. In addition, the experimental results demonstrated the significance of security during offloading process.

In future, in MSCO, the budget and fault-tolerant aspects of IoT applications will be considered. These two constraints are critical when task offloading can be done in real IoT environments. In addition, MSCO will be extended to consider the synchronization overhead during handling of multiple requests in parallel in the blockchain networks.

**Acknowledgements**
We thank all the reviewers for their insightful comments which helps us to improve the quality of the article.

**Research involving Human Participants and/or Animals**
Author declare that there is no research involving Human Participants and Animals.

**Authors' contributions**
1 author - Idea formation, Implementation 2 author - Block chain implementation and Manuscript editing.

**Funding**
There has been no significant financial support for this work that could have influenced its outcome.

**Availability of data and materials**
There are no datasets required to carry out this experiments.

## Declarations

**Ethics approval and consent to participate**
The authors were voluntarily involved in this study to know more about the recent research related to anonymity.

**Consent for publication**
The authors ensure that the publisher has the author's permission to publish this research article.

**Informed consent**
Author have the consent to publish the paper in this journal.

**Competing interests**
The authors declare no competing interests.

Received: 27 February 2023 Accepted: 20 January 2024
Published online: 12 April 2024

**References**

1. Al-Fuqaha A et al (2015) Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutor 17(4):2347–2376
2. Gu L et al (2015) Cost efficient resource management in fog computing supported medical cyber-physical system. IEEE Trans Emerg Top Comput 5(1):108–119
3. Gill SS, Garraghan P, Buyya R (2019) ROUTER: fog enabled cloud based intelligent resource management approach for smart home IoT devices. J Syst Softw 154:125–138
4. Shakarami A, Ghobaei-Arani M, Masdari M, Hosseinzadeh M (2020) A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective. J Grid Comput 18(4):639–671
5. Shah-Mansouri H, Wong VW (2018) Hierarchical fog-cloud computing for IoT systems: a computation offloading game. IEEE Internet Things J 5(4):3246–3257
6. Guo H et al (2018) Mobile-edge computation offloading for ultradense IoT networks. IEEE Internet Things J 5(6):4977–4988
7. Wang C, Li Y, Jin D (2014) Mobility-assisted opportunistic computation offloading. IEEE Commun Lett 18(10):1779–1782
8. Meng X, Wang W, Zhang Z (2017) Delay-constrained hybrid computation offloading with cloud and fog computing. IEEE Access 5:21355–21367
9. Chang Z, Zhou Z, Ristaniemi T, Niu Z (2017) Energy efficient optimization for computation offloading in fog computing system. In: GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, p 1-6
10. Nguyen DC, Pathirana PN, Ding M, Seneviratne A (2021) Secure computation offloading in blockchain based IoT networks with deep reinforcement learning. IEEE Trans Netw Sci Eng 8(4):3192–208
11. Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W (2019) BeCome: blockchain-enabled computation offloading for IoT in mobile edge computing. IEEE Trans Ind Inform 16(6):4187–4195
12. Sarrafzade N, Entezari-Maleki R, Sousa L (2022) A genetic-based approach for service placement in fog computing. J Supercomput 78(8):10854–10875
13. Guo N, Zhao C, Gao T (2020) An anonymous authentication scheme for edge computing-based carhome connectivity services in vehicular networks. Futur Gener Comput Syst 106:659–671
14. Gupta A, Tripathi M, Shaikh TJ, Sharma A (2019) A lightweight anonymous user authentication and key establishment scheme for wearable devices. Comput Netw 149:29–42
15. Chatzopoulos D, Ahmadi M, Kosta S, Hui P (2017) Flopcoin: a cryptocurrency for computation offloading. IEEE Trans Mob Comput 17(5):1062–1075
16. Luong NC, et al. (2018) Optimal auction for edge computing resource management in mobile blockchain networks: a deep learning approach. In: 2018 IEEE International Conference on Communications (ICC). IEEE
17. Dou W, Tang W, Liu B, Xu X, Ni Q (2020) Blockchain-based Mobility-aware Offloading mechanism for Fog computing services. Comput Commun 164:261–273
18. Mukherjee A et al (2022) Mcg: mobility-aware computation offloading in edge using weighted majority game. IEEE Trans Netw Sci Eng 9(6):4310–4321
19. Xiong Z, Feng S, Niyato D, Wang P, Han Z (2017) Edge computing resource management and pricing for mobile blockchain. arXiv preprint arXiv:1710.01567
20. Wu H et al (2020) EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing. IEEE Internet Things J 8(4):2163–2176
21. Sun F et al (2018) Cooperative task scheduling for computation offloading in vehicular cloud. IEEE Trans Veh Technol 67(11):11049–11061
22. Al-Shatri H, Müller S, Klein A (2016) Distributed algorithm for energy efficient multi-hop computation offloading. In: 2016 IEEE International Conference on Communications (ICC). IEEE, p 1-6
23. Wood G (2014) Ethereum: a secure decentralised generalised transaction ledger. Ethereum Proj Yellow Pap 151(2014):1–32
24. Matrouk KM, Matrouk AD (2023) Mobility Aware-Task Scheduling and Virtual Fog for Offloading in IoT-Fog-Cloud Environment. Wirel Pers Commun 130(2):801–836
25. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2021) An autonomous computation offloading strategy in Mobile Edge Computing: a deep learning-based hybrid approach. J Netw Comput Appl 178:102974
26. Lakhan A et al (2021) Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing. IEEE Trans Intell Transp Syst 22(7):4212–4223
27. Adhikari M, Mukherjee M, Srirama SN (2019) DPTO: a deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. IEEE Internet Things J 7(7):5773–5782
28. Abdelmoneem RM, Benslimane A, Shaaban E (2020) Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures. Comput Netw 179:107348
29. Zheng J, Cai Y, Wu Y, Shen X (2018) Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach. IEEE Trans Mob Comput 18(4):771–786
30. Zheng Z, Xie S, Dai H, Chen X, Wang H (2017) An overview of blockchain technology: architecture, consensus, and future trends. In: 2017 IEEE international congress on big data (BigData congress). IEEE, p 557–564
31. Lai P, He Q, Abdelrazek M, Chen F, Hosking J, Grundy J, Yang Y (2018) Optimal edge user allocation in edge computing with variable sized vector bin packing. InInternational Conference on Service-Oriented Computing. Springer, Cham, pp 230–245

## Publisher's Note