2020

Green and secure computation offloading for cache-enabled IoT networks

M. Ishtiaque A. Zahed
Edith Cowan University

Iftekhar Ahmad
Edith Cowan University

Daryoush Habibi
Edith Cowan University

Quoc Viet Phung
Edith Cowan University

10.1109/ACCESS.2020.2982669
Zahed, M. I. A., Ahmad, I., Habibi, D., & Phung, Q. V. (2020). Green and Secure Computation Offloading for Cache-Enabled IoT Networks. *IEEE Access*, *8*, 63840-63855.
https://doi.org/10.1109/ACCESS.2020.2982669

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 5, 2020, accepted March 17, 2020, date of publication March 23, 2020, date of current version April 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982669

Green and Secure Computation Offloading for Cache-Enabled IoT Networks

M. ISHTIAQUE A. ZAHED, IFTEKHAR AHMAD, (Member, IEEE), DARYOUSH HABIBI, (Senior Member, IEEE), AND QUOC VIET PHUNG, (Member, IEEE)
School of Engineering, Edith Cowan University, Perth, WA 6027, Australia

Corresponding author: M. Ishtiaque A. Zahed (mazizzah@our.ecu.edu.au)

ABSTRACT The ever-increasing number of diverse and computation-intensive Internet of things (IoT) applications is bringing phenomenal growth in global Internet traffic. Mobile devices with limited resource capacity (i.e., computation and storage resources) and battery lifetime are experiencing technical challenges to satisfy the task requirements. Mobile edge computing (MEC) integrated with IoT applications offloads computation-intensive tasks to the MEC servers at the network edge. This technique shows remarkable potential in reducing energy consumption and delay. Furthermore, caching popular task input data at the edge servers reduces duplicate content transmission, which eventually saves associated energy and time. However, the offloaded tasks are exposed to multiple users and vulnerable to malicious attacks and eavesdropping. Therefore, the assignment of security services to the offloaded tasks is a major requirement to ensure confidentiality and privacy. In this article, we propose a green and secure MEC technique combining caching, cooperative task offloading, and security service assignment for IoT networks. The study not only investigates the synergy between energy and security issues, but also offloads IoT tasks to the edge servers without violating delay requirements. A resource-constrained optimization model is formulated, which minimizes the overall cost combining energy consumption and probable security-breach cost. We also develop a two-stage heuristic algorithm and find an acceptable solution in polynomial time. Simulation results prove that the proposed technique achieves notable improvement over other existing strategies.

INDEX TERMS Caching, energy, IoT, mobile edge computing, security.

I. INTRODUCTION

Internet of Things (IoT) is an integrated platform connecting hundreds of wireless sensors, smart meters, electronic equipment, and mobile devices [1]. These IoT entities collect, share, and transfer data to network nodes and connected devices for further processing.

In recent years, the remarkable advancements in IoT and wireless networks [2]–[4] have emerged as a solution for ubiquitous connectivity and intelligent data transfer. IoT applications contribute to a number of sectors, including smart healthcare, autonomous vehicles, industrial automation, monitoring ambient environment, and so on [1], [5]. The value of the IoT market is expected to outreach $8.9 trillion by 2020 [6].

These computation intensive IoT applications have created enormous workloads on the existing networking systems [7]. The bandwidth-greedy and delay-sensitive tasks for IoT applications not only result in excessive energy demand, but also cause significant environmental consequences [2]. Mobile devices with limited battery capacity and computational resources are not suitable to address such computing demand [4], [8]. Globally, the number of per day battery replacements is anticipated to be 913 million considering three-year battery lifetime [9]. The IoT services also put forward diverse quality of experience (QoE) requirements, including, but not limited to, low latency, intensive computation capability, and high data rate [10], [11]. Therefore, industries, researchers, and policy makers are constantly looking for high-performing, cost-effective, and energy-efficient strategies for IoT applications.

Conventionally, cloud computing is considered for offloading and remote execution of computing tasks [12]. This is because the migration of tasks to the cloud servers can overcome the limitations in computation capability and battery lifespan of the user devices [4], [12]. However, the long distance between the users and the cloud servers is a major problem in adapting cloud-based services. Offloading of the

The associate editor coordinating the review of this manuscript and approving it for publication was Chunsheng Zhu.

IoT tasks to the cloud servers not only causes backhaul congestion, but also fails to meet the delay requirements of the applications. Hence, cloud computing can not provide best services to the IoT systems [11].

To confront these technical requirements, mobile edge computing (MEC) has been introduced as an efficient solution for task offloading [11]–[13]. MEC achieves low transmission delay compared to cloud computing by reducing the distance between the servers and end users [11]. In MEC, mobile devices can send their computation-intensive tasks to the MEC servers placed at the network edge. This strategy not only reduces the task computation time, but also overcomes the problem related to limited battery capacity of the mobile devices [4]. Computation offloading at different edge nodes can reduce the task latency as well as communication load [14]. Furthermore, optimal allocation of the computational resources also exhibits significant potential to reduce energy consumption while maintaing latency requirements [15].

By 2025, the number of IoT connected devices will be 75.44 billion [16]. As a result, thousands of computation-intensive IoT applications will require uninterrupted connectivity and high-data rate. Conventional approaches are not efficient to fulfill these requirements. Thus, MEC is anticipated as a key solution in designing low-latency and energy-efficient solutions for IoT task offloading [5], [12].

In IoT systems, a number of users often request for the offloading of the same data and similar tasks to the MEC servers [10], [17]. The repeatedly requested tasks are expected to be generated from diverse IoT applications including smart vehicles, e-health services, interactive gaming, smart homes, industrial monitoring, and virtual reality applications [17]–[19]. Caching popular IoT data items at the network edge can play an important role in reducing the duplicate content transmission [3], [20]. Edge caching significantly improves the efficiency of the content delivery and task offloading by reducing not only the latency, but also the energy consumption [4], [21]. On the other hand, edge servers with computing and caching resources consume a considerable amount of energy [4], [22]. Besides, the tasks are diverse in terms of content popularity, input data size, and computational complexity [2], [18]. Therefore, smart management of the IoT systems with caching and computing resources is crucial for achieving energy savings and simultaneously fulfilling the QoE requirements.

Privacy and security concerns of the offloaded tasks are major threats in MEC [13], [23]. The MEC servers, which are located at the network edge, are in close proximity to the attackers and vulnerable to hostile attacks [24]. Furthermore, the security protections at the edge nodes are less stringent because of the limited computational capability compared to the cloud servers [23]. IoT applications deal with different types of confidential information [1], [25]. Potential security breach can lead to system failures and cause life threatening consequences for the end users [26], [27]. The security requirements of different applications are also diverse in terms of authentication, confidentiality, and integrity process [1], [26]. However, the preventive services for security threats inevitably originate computation overheads and lead to additional computing delay and energy consumption [23], [24]. Therefore, in IoT task offloading, addressing these conflicting issues simultaneously is a major research challenge.

In this study, we design a green and secure task offloading technique for IoT systems. The proposed model explores caching, cooperation among the base stations (BSs), and security service provisioning to achieve energy savings and reduce probable security-breach cost. The optimal allocation of the IoT tasks to the MEC servers ensures faster task execution. However, the offloaded tasks are vulnerable to malicious attacks and eavesdropping. Optimal allocation of the security services to the offloaded tasks is a must to accomplish robust security protection. Nonetheless, the security services also cause overheads in terms of energy and delay. Therefore, the problem statement of this research is developed as: *designing a novel MEC technique for green and secure task offloading in IoT networks by exploiting caching, cooperation, and security service assignment. Whereas, the strategy not only reduces energy consumption and probable security-breach cost, but also optimally allocates offloaded tasks and maintains delay requirements.* In this article, we introduce an optimization model and solve the research problem.

The key contributions of this study are summarized as follows.

- We introduce a green and secure task offloading technique for MEC in IoT networks. The designed system model incorporates caching, cooperation among the MEC servers, and security requirements of the tasks to improve system performance.
- The proposed MEC technique is formulated as a constrained non-linear program (NLP) optimization problem, which reduces both energy consumption and probable security damage. Then, we convert the NLP into an integer linear program (ILP). The proposed model also maintains delay threshold and optimally allocates tasks for caching and computing at the MEC servers.
- To reduce the complexity of the solution procedure, we decompose the optimization model into two sub-problems and develop a two-stage heuristic algorithm. The proposed heuristic achieves a sub-optimal solution in polynomial time.
- We also analyze the system performance of our introduced technique and compare them with existing solutions [4], [5], [13], [28]. The system performance is measured in terms of energy consumption, probable security-breach cost, and average delay. Numerical results show that our proposed technique achieves significant energy savings and reduction in probable security-breach costs compared to other techniques.

The remainder of this article is as follows. Section II highlights the existing studies on MEC. In Section III, we provide

a comprehensive description of the proposed system model for IoT networks and introduce different system parameters. Section IV designs an optimization problem combining caching, cooperative task offloading, and security issues. A two-stage heuristic algorithm is developed in Section V to solve the optimization problem in polynomial time. In Section VI, we illustrate the system performance and prove the effectiveness of the proposed MEC technique. Finally, Section VII provides the concluding remarks.

II. RELATED WORK

In recent years, a flurry of research has focused on the remote execution of computational tasks at the network edge. This section provides a brief outline of the existing studies in the field of MEC.

A. ENERGY-EFFICIENT AND LATENCY-CRITICAL MEC

MEC has emerged as a key technology to overcome the problems related to limited battery capacity and computational resources of the mobile devices [11]–[13]. In [29], the authors explained the role of MEC and identified energy consumption and latency as the prime performance indicators for task offloading.

A partial offloading scheme for computational tasks is designed in [30] to exploit both cloud computing and MEC for latency-critical IoT applications. Kherraf *et al.* [31] developed an optimized strategy for edge server placement and workload allocation. The research determined the minimum number and probable locations of the edge servers to reduce the overall expenditure without violating the delay requirements. Although these studies achieved significant latency reduction, the energy related issues were not addressed.

In [32], the authors introduced a reinforcement learning based energy-aware autonomous technique for the collaboration of edge devices. A reward based model is also proposed to encourage user participation, which further improved the performance gain. An architecture combining energy harvesting IoT sensors and MEC servers was developed in [33]. The study optimally allocated network edge resources and achieved energy savings, which eventually improved the lifetime of the IoT sensors.

Cheng *et al.* [11] proposed a virtual machine allocation and computation offloading scheme for remote users. The authors considered unmanned aerial vehicle assisted edge servers for MEC and utilized deep-reinforcement learning. The authors in [34] introduced an adaptive technique and optimally offloaded the tasks at the edge servers and clouds.

Balasubramanian *et al.* [35] investigated the mobility management perspectives of MEC and developed a novel architecture to ensure smooth handovers in different network slices of a 5G-IoT network. The proposed model ensured subscription-based network connections and not only improved packet delivery ratio, but also achieved latency requirements. Sun *et al.* [36] also designed a mobility management technique by combining Lyapunov optimization and multi-armed bandits theory for seamless handover and task offloading. The proposed framework reduced delay without exceeding energy budget.

Zhang *et al.* [12] exploited multi-access heterogeneous networks and designed a low-complexity three-stage solution process for the problem. The aforementioned strategies investigated the concerns related to task execution delay and energy consumption. However, these schemes did not explore the potential of cooperative task offloading and security aspects at the edge nodes.

B. COOPERATION AMONG THE EDGE SERVERS IN MEC

Cooperative computation offloading achieved further improvement in energy savings [5] and latency reduction [37]. Fan and Ansari [37] designed an application-aware scheme for IoT task offloading at the edge cloudlets. The study cooperatively allocated the network resources based on the QoE demands and decreased the average response time. Misra and Saha [5] jointly optimized task computation, resource allocation, and path selection for fog computing based IoT applications. The study considered software defined rule-capacity and not only achieved energy savings, but also decreased average delay.

In [28], the authors introduced the concept of fair cooperation to encourage the owners participation in fog computing. The fog nodes shared a specific portion of their computation resources and optimized both service time and energy cost. Guo *et al.* [8] presented a task offloading strategy to reduce the energy consumption and execution time of the offloaded tasks in ultradense IoT networks. Nonetheless, the potential of edge caching was not investigated in these works.

C. COLLABORATION OF MEC AND CACHING

Caching frequently requested information at the network edge is an effective measure to reduce the duplicate content transmission [20], [21]. The joint utilization of MEC and caching can significantly reduce the energy requirement [4] and task execution time [3].

In [2], an iterative algorithm is introduced for energy-efficient caching and MEC without degrading delay requirements. The algorithm combined optimal caching and computing to achieve an acceptable solution within lower time span over other techniques. A centralized framework for MEC and caching is proposed in [3], which jointly optimized computation and spectrum resources to reduce overall latency of the system.

In [39], the authors proposed a mobility-aware strategy incorporating caching and MEC for heterogeneous networks. They investigated the impact of content diversity, backhaul capacity, and user mobility on the network throughput and achieved significant performance gain. Tan *et al.* [40] also introduced a framework combining MEC and caching based on deep Q-learning. The mobility-aware approach not only optimally allocated network resources, but also decreased system cost.

Hao *et al.* [4] explored energy-efficient approach for task offloading and caching. They reduced energy consumption at

TABLE 1. Summary of existing and proposed mobile edge computing techniques.

Research Work	Delay	Energy	Cooperation	Security	Caching
Ning *et al.* [30]	✓				
Mishra *et al.* [34], Cheng *et al.* [11]	✓	✓			
Fan *et al.* [37], Kherraf *et al.* [31]	✓		✓		
Jiang *et al.* [26], Elgendy *et al.* [13]	✓	✓		✓	
Chen *et al.* [38]		✓	✓		✓
Zhang *et al.* [3]	✓				✓
Liu *et al.* [2], Hao *et al.* [4]	✓	✓			✓
Misra *et al.* [5], Dong *et al.* [28]	✓	✓	✓		
Proposed technique	✓	✓	✓	✓	✓

the user devices for a MEC system with constrained storage and computation resources. Chen *et al.* [38] utilized software-defined networking for cooperative computing and caching. The framework optimized spectrum, caching, and computing resources for energy-efficient solution and decreased the cost of network usage. However, the studies did not consider probable security threats and confidentiality of the offloaded tasks.

D. SECURITY ASPECTS IN MEC

The computation-intensive tasks, which are offloaded to the MEC servers, are prone to security breach [23], [27]. Security services adopted at the edge nodes also cause energy and latency overheads [13]. Therefore, the optimization of the security defense techniques is a topic of prior interest in MEC.

In [13], the authors utilized a cryptographic technique for the security of the offloaded tasks and developed an optimal strategy for computation offloading and resource allocation. Jiang *et al.* [26] designed a dynamic programming based scheduling policy for security-critical tasks. The authors considered different cryptographic algorithms and assigned security services for the tasks based on their requirements. The study achieved considerable energy savings while maintaining latency and security requirements.

He *et al.* [27] scrutinized the security aspects of IoT and MEC. The study also developed a low-cost and novel technique to control the security services of different applications. The authors in [41] proposed different models to reduce the risk of failures at IoT nodes. The integrated approach not only minimized traffic power, but also ensured reliability of the cloud-based network. A joint approach combining radio resource provisioning, security enhancement, and MEC is proposed in [42]. The authors reduced execution delay subject to physical layer security and energy budget conditions. However, most of these studies did not consider caching while investigating security issues in MEC.

E. NOVELTY OF THIS WORK

Table 1 summarizes the state-of-the art research in mobile edge computing and compares the proposed technique with existing works. Most of these studies reduced either energy consumption or associated delay in MEC. Some of these works also considered caching and cooperation between the nodes. Although few studies considered security requirements as a constraint in MEC, the probable security risk was not optimized in any of them. Furthermore, none of these studies jointly investigated caching, cooperation, and security issues in MEC for IoT applications. In this article, we optimize the assignment of computing, storage, and security resources to reduce the energy consumption and probable security risk without violating the delay requirements.

III. SYSTEM MODEL

In this section, we introduce a green and secure model for MEC and caching in an IoT network. The proposed model comprises system overview, communication model, computation model, caching model, and security model. The brief description of the system model is as follows.

A. SYSTEM OVERVIEW

Fig. 1 presents a network comprising a number of BSs at the edge nodes, multiple subscribers, and mobile devices. Each of the BSs possesses an MEC server with finite computation and storage capacity. The set of the edge nodes is represented as $\mathcal{N} = \{1, 2, 3, \ldots, n, \ldots, N\}$. These edge nodes receive offloading requests through the respective BSs and execute the tasks at the connected MEC servers. These nodes also collaborate with other nodes to share the network resources and minimize caching redundancy. The computing and caching capacity of an edge node (n-th node) is defined by F_n and C_n, respectively.

The internet subscribers are connected to the nearest edge node and the set of the user equipment (UE) in the n-th node is denoted as $\mathcal{U}_n = \{U_1, U_2, \ldots, U_{k,n}, \ldots, U_{K,n}\}$. $U_{k,n}$ represents the k-th user device, which is connected to the edge node n. Every user device $U_{k,n}$ is defined as a 2-tuple $\{\vartheta^l_{k,n}, P_{k,n}\}$. $\vartheta^l_{k,n}$ and $P_{k,n}$ represent the computation capability and the transmission power of $U_{k,n}$, respectively.

The user devices utilize network resources for task computation, and some of the popular tasks are repeatedly requested by multiple users. The set of the computation tasks considered in this model is $\mathcal{T} = \{t_1, t_2, \ldots, t_i, \ldots, t_I\}$. Each

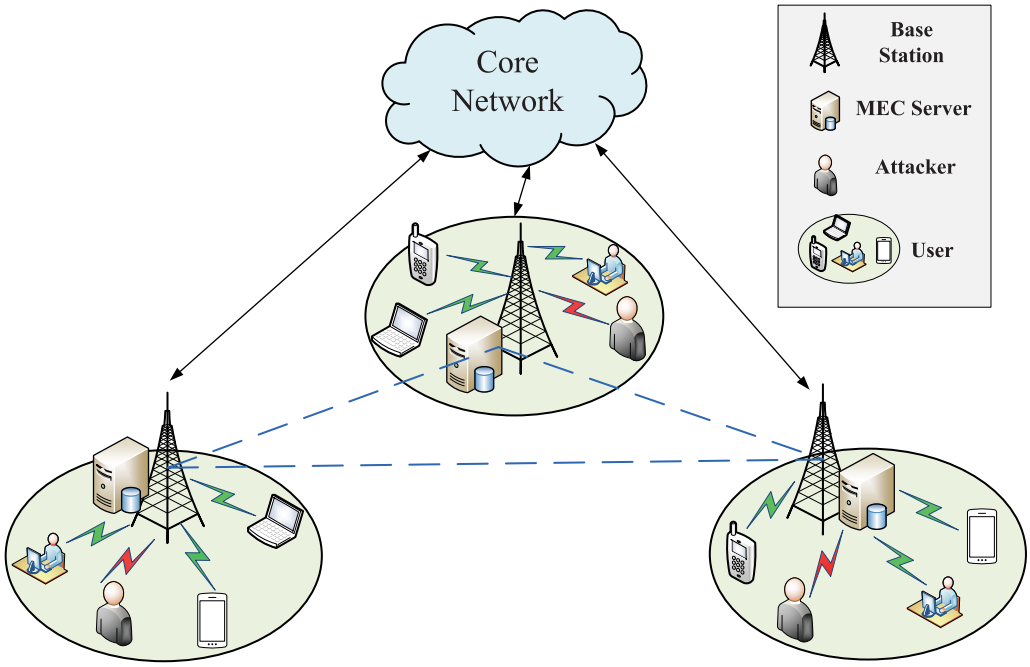

FIGURE 1. Green and secure task offloading and caching model for MEC.

of these computation tasks has specific requirements to be executed [4], [26]. A task t_i is defined as $\{\delta_i, \omega_i, \mu_i, \rho_i\}$. δ_i denotes the input data size, ω_i is the computational requirement for task completion, μ_i represents the task deadline, and ρ_i is the expected security level protection.

B. COMMUNICATION MODEL

In this model, we consider orthogonal spectrum allocation for the users to avoid the interference effect, and each UE is assigned one channel [12]. The uplink transmission rate [4] of the UE $U_{k,n}$ to the edge node n is expressed as

$$R_{k,n} = B \log_2 \left(1 + \frac{P_{k,n} g_{k,n}}{\sigma^2}\right) \tag{1}$$

where B, $g_{k,n}$, and σ^2denote the channel bandwidth, transmission channel gain, and noise power, respectively. For simplicity, we consider constant channel gain [4] and noise power [2], [3]. The bandwidth is identical for all the channels connecting the users to the BSs [12].

The output data size of the executed task is very small compared to the input data size and the corresponding downlink data rate is greater than the uplink data rate [4]. Therefore, similar to the existing studies [5], [13], we ignore the download time and energy of the output data.

C. COMPUTATION

In this model, a task can be either executed locally at the mobile devices or offloaded to any of the edge nodes [2], [5]. The overheads related to task execution are calculated as execution latency and energy consumption [2], [5].

- Local Computing

The execution latency and energy consumption to process the task t_i at the UE $U_{k,n}$ is

$$d^{Loc}_{i,k,n} = \frac{\omega_i}{\vartheta^l_{k,n}} \tag{2}$$

and

$$E^{Loc}_{i,k,n} = \epsilon(\vartheta^l_{k,n})^2 \omega_i \tag{3}$$

where $\vartheta^l_{k,n}$ is the computing capacity of the mobile device and $\epsilon(\vartheta^l_{k,n})^2$ is the consumed energy in every computation cycle. The energy coefficient ϵ depends on the device chip architecture and $\epsilon = 10^{-25}$ [4].

- Edge Computing

In MEC, the computing task is offloaded to the connected edge node using a wireless channel. The time required to offload the task t_i of user device $U_{k,n}$ to the edge node n is expressed as

$$d^{Off}_{i,k,n} = \frac{\delta_i}{R_{n,k}} \tag{4}$$

Associated energy consumption for task offloading is

$$E^{Off}_{i,k,n} = P_{k,n} \frac{\delta_i}{R_{n,k}} \tag{5}$$

In contrast to most of the existing studies [2], [4], [13], while calculating energy consumption, we consider not only the mobile devices, but also the MEC servers. Furthermore, in this model, the edge nodes work cooperatively.

If the task t_i of the UE $U_{k,n}$ is executed at the edge node m, the task propagation time from n to m [43] is expressed as

$$d^{Pr}_{i,n,m} = \delta_i \theta_{n,m} \tag{6}$$

where $\theta_{n,m}$ is the time to transmit per unit content and $\theta_{n,m} = \theta_0\gamma_{n,m} + \theta_1$. Similar to the existing studies, we consider that $\theta_{n,m}$ is linearly proportional to the distance as the content requires more time to travel a longer distance [37], [44]. $\gamma_{n,m}$ represents the distance between the nodes n and m, while θ_0 and θ_1 are the linear coefficients [44].

According to well-established energy proportional model, the transmission energy linearly depends on data size and delivery distance [38], [45]. Energy required to transfer the task from node n to node m is expressed as

$$E^{Pr}_{i,n,m} = \delta_i \gamma_{n,m} P_{Tr} \tag{7}$$

where P_{Tr} is the energy coefficient between two edge nodes, which is fixed for a specific link [38], [45].

The task execution time and energy consumption for task t_i, which is executed at the edge node m, are as follows

$$d^{Ex}_{i,m} = \frac{\omega_i}{\vartheta_m} \tag{8}$$

and

$$E^{Ex}_{i,m} = \epsilon_s(\vartheta_m)^2\omega_i \tag{9}$$

where ϑ_m represents the allocated computation frequency by the node m to each of the executed tasks. ϵ_s is the energy coefficient of the MEC server. According to existing study [12], MEC servers are energy-efficient compared to mobile devices. In this study, we consider $\epsilon_s = 10^{-26}$ [2].

D. CACHING

In this model, we consider task caching, which denotes that the input data of the executable task is stored at the MEC server [2]. When the mobile device requests the edge node for task offloading, the MEC server checks whether the task data is cached at the MEC server or any of the nearby servers. If the task is cached, the mobile device can save the time for offloading delay and propagation delay. Associated energy consumption is also reduced by caching. After executing the tasks at the designated MEC server, the output result is forwarded to the user device. However, the caching capability of a MEC server is constraint and all the requested tasks can not be cached at the servers. The caching capacity of a MEC server at node n is represented by C_n.

Although different users request for diverse task offloading, the popular tasks are requested by a number of users. Consequently, the caching of a popular task reduces repeated data transmission and network overheads. The probability that task t_i is requested by an user connected to the edge node n is represented as p^n_i, where $0 < p^n_i < 1$ and $\sum_{i=1}^{I} p^n_i = 1$.

The consumed energy at the MEC server for task caching is proportional to the stored data size [38]. Hence, the energy consumption at node n for caching task t_i is

$$E^{Ca}_{i,n} = \delta_i P_{Ca} T \tag{10}$$

where P_{Ca} and T denote caching power density and caching duration, respectively.

E. SECURITY

The offloaded tasks at the MEC servers are vulnerable to malicious attacks, eavesdropping, and spoofing [26]. The tasks also have different security requirements [27]. Hence, different cryptographic algorithms are used for data encryption and decryption in MEC [13]. Along with the strength and robustness of security protection algorithm, the energy and delay overheads increase significantly [26]. On the other hand, the preventive measures can not stop 100% security breach. Therefore, the quantification of the security risk is a major challenge in designing optimization strategies.

In this model, we define different cryptographic algorithms as different security levels [26]. The offloaded tasks are encrypted and decrypted at the user devices and MEC servers, respectively [13]. This process is computationally expensive and causes associated overheads (e.g., delay, energy) [13], [26]. If this encryption-decryption process is more stringent, the overheads are higher. On the other hand, different applications have different requirements, such as industrial monitoring applications are more security sensitive compared to interactive gaming.

The set of security protection levels is defined as $\mathcal{S} = \{s_1, s_2, \ldots, s_l, \ldots, s_L\}$. In this study, security protection level (s_l) represents the robustness of the designated cryptographic algorithm. If there are L number of cryptographic algorithms, then the algorithms with minimum and maximum robustness are denoted as $s_1 = 1$ and $s_L = L$, respectively [24], [26]. Other algorithms are also defined accordingly. Moreover, the associated overheads (e.g., delay, energy) for these cryptographic algorithms are already determined in previous studies [24], [26]. Associated time and energy overheads of s_l are represented as α_l (in ms/MB) and β_l (in J/MB), respectively.

The probable security breach cost of task t_i having security protection s_l is expressed as

$$\psi^l_i = \begin{cases} 1 - e^{-(\rho_i - s_l)}, & \text{if } s_l < \rho_i \\ 0, & \text{if } s_l \geq \rho_i \end{cases} \tag{11}$$

where ρ_i defines the expected security level of task t_i.

According to this model, the security breach cost exists for a task, which does not achieve expected security protection. Otherwise, this cost is zero. This security breach cost represents the vulnerability of the offloaded task to malicious attacks or eavesdropping when the expected protection is not achieved [26].

IV. PROBLEM FORMULATION AND TRANSFORMATION

In this section, we design an ILP model and jointly optimize energy consumption and security breach cost while maintaining delay conditions.

The proposed model is based on following considerations.

- The users are connected to the nearest BSs.
- All the BSs with MEC servers are under a single administrative domain and the geographical position of the users are known.

- The tasks can be offloaded to the host edge-node or any of the neighboring MEC servers.
- The link capacity is constrained, and $\Gamma_{n,m}$ denotes the bidirectional link-capacity between edge node n and m.

Although the execution of a task at an edge node reduces the energy cost of mobile devices, it can cause security breach or violate delay condition. Furthermore, the caching of a frequently requested task over a less popular task can alleviate network traffic and reduce both energy cost and delay. In this article, our proposed model optimally caches popular computation-intensive tasks, offloads the tasks to the MEC servers, and assigns security protections to the offloaded tasks.

The caching variable for task t_i at node n is defined as

$$x_i^n = \begin{cases} 1, & \text{if node } n \text{ caches input data } (\delta_i) \text{ of task } t_i \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

When task t_i of user device $U_{k,n}$ is executed at the MEC server of node m, the decision variable is expressed as

$$y_{n,m}^{i,k} = \begin{cases} 1, & \text{if task } t_i \text{ of } U_{k,n} \text{ is executed at node } m \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

For task t_i, the selection of security protection level s_l is defined by the binary variable

$$z_i^l = \begin{cases} 1, & \text{when } s_l \text{ is selected for task } t_i \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

According to the previous section, when an offloaded task is cached at the MEC server, the offloading delay and propagation delay can be reduced. Therefore, the delay expression for computing and caching of task t_i of UE $U_{k,n}$ is written as

$$D_{i,k,n}^C = \sum_{m=1}^{N} y_{n,m}^{i,k}\{(1 - x_i^m)(d_{i,k,n}^{Off} + d_{i,n,m}^{Pr}) + d_{i,m}^{Ex}\} \quad (15)$$

The delay overhead for security protection of task t_i of UE $U_{k,n}$ is

$$D_{i,k,n}^S = \sum_{m=1}^{N} y_{n,m}^{i,k} \sum_{l=1}^{L} z_i^l \alpha_l \delta_i \quad (16)$$

Similarly, the corresponding expressions for energy consumption are

$$E_{i,k,n}^C = \sum_{m=1}^{N} y_{n,m}^{i,k}\{(1 - x_i^m)(E_{i,k,n}^{Off} + E_{i,n,m}^{Pr}) + E_{i,m}^{Ex}\} \quad (17)$$

and

$$E_{i,k,n}^S = \sum_{m=1}^{N} y_{n,m}^{i,k} \sum_{l=1}^{L} z_i^l \beta_l \delta_i \quad (18)$$

If we also consider the local execution of IoT tasks, delay and energy expressions for task t_i of UE $U_{k,n}$ can be expressed as

$$D_{i,k,n} = \Big(1 - \sum_{m=1}^{N} y_{n,m}^{i,k}\Big) d_{i,k,n}^{Loc} + D_{i,k,n}^C + D_{i,k,n}^S \quad (19)$$

and

$$E_{i,k,n} = \Big(1 - \sum_{m=1}^{N} y_{n,m}^{i,k}\Big) E_{i,k,n}^{Loc} + E_{i,k,n}^C + E_{i,k,n}^S \quad (20)$$

Therefore, the overall energy consumption for task caching, offloading, and security services is

$$E = \sum_{i=1}^{I} \sum_{n=1}^{N} \Big(E_{i,n}^{Ca} + \sum_{k=1}^{K} p_i^n E_{i,k,n}\Big) \quad (21)$$

Probable security breach cost for the edge computing of task t_i of UE $U_{k,n}$ is

$$\psi_{i,k,n} = \sum_{m=1}^{N} y_{n,m}^{i,k} \sum_{l=1}^{L} z_i^l \psi_i^l \quad (22)$$

Hence, the overall security breach cost is

$$\psi = \sum_{i=1}^{I} \sum_{n=1}^{N} \sum_{k=1}^{K} p_i^n \psi_{i,k,n} \quad (23)$$

The total cost of the IoT system is quantified as a combination of the energy consumption and security breach cost. To formulate the joint optimization problem, the overall cost is formulated as

$$\phi = \eta E + (1 - \eta)\psi \quad (24)$$

where η is the weighting parameter, which investigates the trade-off in between the energy consumption and security breach cost. When η is higher, the system provides more emphasis on energy over security and vice versa. η varies from 0 to 1 based on operators' requirements.

Before considering the energy consumption and probable security cost in the joint optimization problem, we normalize these parameters [21] because their units are different.

In this article, the optimization problem is modeled as

$$\underset{x,y,z}{\text{minimize}}\ \phi \quad (25)$$

$$\text{subject to } D_{i,k,n} \le \mu_i, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n,\ \forall n \in \mathcal{N} \quad (26)$$

$$\sum_{i=1}^{I} x_i^n \delta_i \le C_n, \quad \forall n \in \mathcal{N} \quad (27)$$

$$\sum_{m=1}^{N} y_{n,m}^{i,k} \le 1, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n,\ \forall n \in \mathcal{N} \quad (28)$$

$$\sum_{l=1}^{L} z_i^l = 1, \quad \forall t_i \in \mathcal{T} \quad (29)$$

$$\sum_{i=1}^{I} \sum_{n=1}^{N} \sum_{k=1}^{K} y_{n,m}^{i,k} p_i^n \vartheta_m \le F_m, \quad \forall m \in \mathcal{N} \quad (30)$$

$$\frac{\sum_{i=1}^{I} \sum_{k=1}^{K} y_{n,m}^{i,k}(1 - x_i^m)\delta_i}{\sum_{i=1}^{I} \sum_{k=1}^{K} y_{n,m}^{i,k}(1 - x_i^m) d_{i,n,m}^{Pr}} \le \Gamma_{n,m}, \quad \forall n, m \in \mathcal{N} \quad (31)$$

Here, the delay condition for all the tasks is maintained by constraint (26). Constraint (27) bounds the input data

size of the cached tasks. According to constraint (28), to avoid redundancy, a specific task of a particular subscriber is not offloaded to more than one MEC server. Similarly, constraint (29) guarantees that each of the tasks are assigned a specific security protection service. Constraint (30) denotes that the total amount of computation offloaded to an edge node is limited by the computing capacity of the node. According to constraint (31), the data rate of task propagation from one node to the other can not exceed the link capacity.

The proposed optimization model is an NLP. This is because the energy and delay expressions in (15) - (18) and constraint (31) have quadratic terms. Optimization problems of this category are computationally expensive and solved by heuristics based on relaxations and approximations. Nonetheless, we transform this NLP into an ILP and solve the problem using an ILP solver. This solution is utilized as a benchmark to quantify the acceptance of the heuristics.

We consider two binary decision variables $w_{n,m}^{i,k}$ and $q_{n,m}^{i,k,l}$ for the ILP transformation. The associated constraints are defined as follows

$$w_{n,m}^{i,k} \leq 1 - x_i^m, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n,\ \forall n, m \in \mathcal{N} \tag{32a}$$

$$w_{n,m}^{i,k} \leq y_{n,m}^{i,k}, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n,\ \forall n, m \in \mathcal{N} \tag{32b}$$

$$y_{n,m}^{i,k} - x_i^m \leq w_{n,m}^{i,k}, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n,\ \forall n, m \in \mathcal{N} \tag{32c}$$

and

$$q_{n,m}^{i,k,l} \leq z_i^l, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n, \forall n, m \in \mathcal{N}, \forall s_l \in \mathcal{S} \tag{33a}$$

$$q_{n,m}^{i,k,l} \leq y_{n,m}^{i,k}, \quad \forall t_i \in \mathcal{T}, \forall U_{k,n} \in \mathcal{U}_n, \forall n, m \in \mathcal{N},\ \forall s_l \in \mathcal{S} \tag{33b}$$

$$z_i^l + y_{n,m}^{i,k} - 1 \leq q_{n,m}^{i,k,l}, \quad \forall t_i \in \mathcal{T},\ \forall U_{k,n} \in \mathcal{U}_n, \forall n, m \in \mathcal{N}, \forall s_l \in \mathcal{S} \tag{33c}$$

Therefore, delay and energy expressions in (15) - (18) are transformed into following equations

$$D_{i,k,n}^C = \sum_{m=1}^{N} \left\{ w_{n,m}^{i,k} (d_{i,k,n}^{Off} + d_{i,n,m}^{Pr}) + y_{n,m}^{i,k} d_{i,m}^{Ex} \right\} \tag{34}$$

$$D_{i,k,n}^S = \sum_{m=1}^{N} \sum_{l=1}^{L} q_{n,m}^{i,k,l} \alpha_l \delta_i \tag{35}$$

and

$$E_{i,k,n}^C = \sum_{m=1}^{N} \left\{ w_{n,m}^{i,k} (E_{i,k,n}^{Off} + E_{i,n,m}^{Pr}) + y_{n,m}^{i,k} E_{i,m}^{Ex} \right\} \tag{36}$$

$$E_{i,k,n}^S = \sum_{m=1}^{N} \sum_{l=1}^{L} q_{n,m}^{i,k,l} \beta_l \delta_i \tag{37}$$

The constraint in (31) is rewritten as

$$\frac{\sum_{i=1}^{I} \sum_{k=1}^{K} w_{n,m}^{i,k} \delta_i}{\sum_{i=1}^{I} \sum_{k=1}^{K} w_{n,m}^{i,k} d_{i,n,m}^{Pr}} \leq \Gamma_{n,m}, \quad \forall n, m \in \mathcal{N} \tag{38}$$

Thus, the optimization problem in (25) is updated into an ILP. In this article, IBM ILOG CPLEX Optimization Studio is used to solve the ILP problem. Nonetheless, the computation time to achieve an acceptable solution depends on the input parameters. Therefore, it is difficult to solve the problem involving a large number of servers and users within practical time. In the following section, we propose a two-stage heuristic solution to overcome this limitation and achieve an acceptable solution within a reasonable time limit.

V. A TWO-STAGE HEURISTIC SOLUTION

In this section, we decompose the proposed model into two sub-problems, which are successively solved to achieve an acceptable solution in polynomial time. The first stage of the heuristic solves the caching and computing problem. Then, in the second stage, we assign security protections to the offloaded tasks based on the solution of the first stage.

A. CACHING AND COMPUTING PROBLEM

The caching and computing problem investigates the optimal allocation of the resources at the MEC servers. We select the tasks for caching and MEC considering the energy consumption and delay requirements. In this part, we do not investigate the security requirements and consider the security components as zero. Thus, in equations (19) and (20) respectively $D_{i,k,n}^S = 0$ and $E_{i,k,n}^S = 0$. We also relax the binary decision variables $x_i^n, y_{n,m}^{i,k}, w_{n,m}^{i,k}$ and convert the ILP into a linear programming (LP) model.

The caching and computing problem is relaxed as

$$\underset{x,y,w}{\text{minimize}}\ E \tag{39}$$

$$\text{subject to (26)-(28), (30), (32), (38), and } x_i^n, y_{n,m}^{i,k}, w_{n,m}^{i,k} \in [0, 1] \tag{40}$$

In Algorithm (1), we present the relaxation and rounding based solution for the caching and computing problem. First, we relax the binary variables and solve the LP problem. If the model is feasible, we get the solutions for the variables in float data type. Otherwise, the system will return error message. After that, we sort the caching variable $\{x_i^n\}$ in descending order and convert into $\{X_i^n\}$ (Line 6). This is because, a higher value of $\{x_i^n\}$ is expected to represent a more suitable task $\{t_i\}$ for caching. Next, we consider the caching capacity constraint and assign binary values to $\{X_i^n\}$ (Line 7-15). We consider 0.5 as a benchmark to convert the variables from float to binary and assign the caching variables accordingly (Line 16).

In the following part of the algorithm, we consider $\{x_i^n\}$ as a known parameter and solve the problem LP2, which considers both $\{y_{n,m}^{i,k}\}$ and $\{w_{n,m}^{i,k}\}$ as float type variables (Line 18-20). If LP2 is feasible, we extract the solution and set flag to zero. We sort the values in $\{y_{n,m}^{i,k}\}$ and convert into $\{Y_{n,m}^{i,k}\}$. Then, the rounding procedure in Algorithm 2 is utilized to convert $\{Y_{n,m}^{i,k}\}$ and $\{W_{n,m}^{i,k}\}$ into binary values.

In Algorithm 2, the rounding procedure also uses 0.5 as a benchmark for the conversion and checks the computation offloading constraint in (30). Eventually, the value of $\{Y_{n,m}^{i,k}\}$

Algorithm 1 Caching and Computing Based on Relaxation and Rounding

Relax the binary variables $x_i^n, y_{n,m}^{i,k}, w_{n,m}^{i,k}, \forall t_i \in \mathcal{T}$, $\forall U_{k,n} \in \mathcal{U}_n$, and $\forall n, m \in \mathcal{N}$
Solve the LP problem

$$LP1: \underset{x,y,w}{\text{minimize}} \quad E$$

subject to (26)-(28), (30), (32), and (38)
if LP1 is feasible **then** Obtain the solution
else Return error
end if
Sort the caching variable (x_i^n) in descending order and convert into X_i^n
for $n = 1$ to N **do**
  for $i = 1$ to I **do**
    if $(X_i^n \geq 0.5)$ and $(\sum_{i=1}^{I} X_i^n \delta_i \leq C_N)$ **then**
      $X_i^n \leftarrow 1$
    else
      $X_i^n \leftarrow 0$
    end if
  end for
end for
Update x_i^n from X_i^n
while flag=1 **do**
  Set the value of x_i^n and consider $y_{n,m}^{i,k}, w_{n,m}^{i,k} \in [0, 1]$
  Solve the LP problem

$$LP2: \underset{y,w}{\text{minimize}} \quad E$$

subject to (26),(28),(30), (32), and (38)
  if LP2 is feasible **then**
    Obtain the solution and flag $\leftarrow 0$
    Sort the task offloading variable $(y_{n,m}^{i,k})$ in descending order for different tasks and convert into $Y_{n,m}^{i,k}$
    Transform $Y_{n,m}^{i,k}$ and $W_{n,m}^{i,k}$ into binary values according to Rounding procedure
    Update $y_{n,m}^{i,k}$ and $w_{n,m}^{i,k}$ from $Y_{n,m}^{i,k}$ and $W_{n,m}^{i,k}$
    Calculate energy and delay for all the tasks.
  else
    Convert next element in $X_i^n \leftarrow 1$
    if $(\sum_{i=1}^{I} X_i^n \delta_i \leq C_N)$ **then**
      Update x_i^n
    else
      Return error
      break
    end if
  end if
end while

is updated to 1 and $\{W_{n,m}^{i,k}\}$ is set accordingly. If this allocation violates the data rate constraint in (38), the variables are set back to zero.

Algorithm 1 takes these values from the rounding procedure and updates $\{y_{n,m}^{i,k}\}$ and $\{w_{n,m}^{i,k}\}$. Then, we calculate the associated energy and delay for all the tasks.

Algorithm 2 Rounding Procedure

procedure Rounding
  for $n = 1$ to N **do**
    for $m = 1$ to M **do**
      for $k = 1$ to K **do**
        for $i = 1$ to I **do**
          if $(Y_{n,m}^{i,k} \geq 0.5)$ and constraint (30) upholds **then**
            $Y_{n,m}^{i,k} \leftarrow 1$
            $W_{n,m}^{i,k} \leftarrow Y_{n,m}^{i,k}(1 - x_i^m)$
            if constraint (38) is violated **then**
              $Y_{n,m}^{i,k}, W_{n,m}^{i,k} \leftarrow 0$
            end if
          else
            $Y_{n,m}^{i,k}, W_{n,m}^{i,k} \leftarrow 0$
          end if
        end for
      end for
    end for
  end for
end procedure

If LP2 is not feasible, we consider following tasks in the sorted list of $\{X_i^n\}$ and cache the tasks. Then, the process is repeated until we successfully find an acceptable solution. Otherwise, the system returns error.

B. SECURITY SERVICE ASSIGNMENT PROBLEM

The security service assignment problem investigates which security level protection should be assigned to an offloaded task at the MEC server, subject to that the offloading decisions have been made. In Algorithm 2, we assign the security services to minimize the probable security breach cost and energy consumption. The delay condition is also strictly maintained.

First, we calculate the probable security breach costs, associated overheads, and overall cost $\{\phi_{i,l}^s\}$ for all the tasks and possible security services (Line 3-6). Then, we choose the security protection service l' for all the tasks considering the minimum value of $\{\phi_{i,l}^s\}$ and initialize the associated variable to 1 (Line 8).

In the following step, we evaluate the associated delay $(D_{i,k,n})$ for every task and check the delay condition (Line 13-14). If any of these offloaded tasks violates the delay requirement, we update the minimum value of $\{\phi_{i,l}^s\}$ while maintaining delay requirements. The associated security protection level is labeled as l' and the security service assignment is completed (Line 17-21).

Finally, we calculate energy consumption (E), security breach cost (ψ), and overall cost (ϕ) for the system.

C. POLYNOMIAL TIME COMPLEXITY

This section presents the polynomial time complexity of the two-stage heuristic solution.

Algorithm 3 Security Service Assignment

for $i = 1$ to I **do**
  for $l = 1$ to L **do**
    Calculate ψ_i^l from equation (11)
    Determine delay overhead, $D_{i,l}^S = \alpha_l \delta_i$
    Determine energy overhead, $E_{i,l}^S = \beta_l \delta_i$
    Evaluate $\phi_{i,l}^S = \eta E_{i,l}^S + (1-\eta)\psi_i^l$
  end for
  Choose the security protection l' for task t_i with minimum value of $\phi_{i,l}^S$ and assign $z_i^{l'} \leftarrow 1$
end for
for $n = 1$ to N **do**
  for $k = 1$ to K **do**
    for $i = 1$ to I **do**
      Calculate the associated delay $D_{i,k,n}$ from equation (34)
      if $(D_{i,k,n} > \mu_i)$ **then**
        Choose the security protection l' for task t_i with minimum value of $\{\phi_{i,l}^S\}$,which satisfies the delay condition in equation (26).
        for $l = 1$ to L **do**
          if $l = l'$ **then**
            $z_i^l \leftarrow 1$
          else
            $z_i^l \leftarrow 0$
          end if
        end for
      end if
    end for
  end for
end for
Calculate E, ψ, and ϕ from equations (21), (23), and (24)

In Algorithm 1, first, we solve the LP1 optimization problem. The solution of such an LP model is achieved in polynomial time [46]. Then, the sorting procedure has a computational complexity of $O(IlogI)$ and the complexity for N number of edge nodes is $O(NIlogI)$. The following steps converting the caching variable into the binary format involves two for loops. This conversion has a complexity of $O(NI)$. The LP2 optimization problem also has polynomial-time complexity. The sorting of the task offloading variable possesses a computational complexity of $O(N^2KIlogI)$. Complexity of the following steps involving rounding and binary conversion is $O(N^2KI)$. Therefore, the solution for the caching and computing problem can be achieved in polynomial time.

In Algorithm 2, the steps calculating the cost functions and selecting security protection for every task have the computational complexity of $O(L)$. Hence, the complexity involving these steps (Line 1-9) is $O(2IL)$. The following part of the algorithm has three for loops and the step choosing security protection (Line 15) has computational complexity of $O(L)$. The next steps select security protections among L security levels and the corresponding complexity is also $O(L)$.

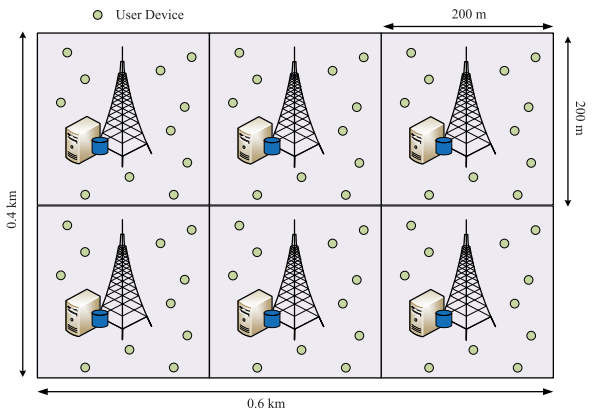

FIGURE 2. Simulation scenario for green and secure MEC.

TABLE 2. Summary of simulation parameters.

Definition	Value
Number of edge nodes, N	6
Number of task type, I	20
Number of users at each node, K	100
Channel Bandwidth, B	40 MHz [8]
Computing capacity of an MEC server, F_n	16 GHz [30]
Computing capacity of an user device, $\vartheta_{k,n}^l$	0.1-1.0 GHz [47]
Input data size of a task, δ_i	0.02 Mb [48]
Computational requirement of a task, ω_i	20 Mcycles [48]
Deadline for task execution, μ_i	1-100 ms [49]
Caching power density, P_{Ca}	10^{-9} W/bit [45]
Transmit power density, P_{Tr}	2×10^{-8} J/bit [45]
Transmit power of an user device, $P_{k,n}$	0.1 W [8]
Noise power, σ^2	-100 dBm [8]

Therefore, the overall complexity of the security service assignment involving the above mentioned for loops is $O(2NKIL) \approx O(NKIL)$.

Eventually, it is evident that, the proposed two-stage heuristic algorithm can be solved in polynomial-time.

VI. SIMULATION RESULTS AND DISCUSSION

In this section, we investigate the system performance of the proposed MEC technique and validate the effectiveness of the technique over existing models. The system performance is measured in terms of energy consumption, probable security breach cost, average delay, and task offloading percentage. In the first part of this section, we describe the simulation settings. Then we provide the numerical results and discussions.

We consider a small-scale network with six edge nodes, which is presented in Fig. 2. Each of these nodes possesses a BS with an MEC server, and the coverage area of a node is 200 m × 200 m [3]. The users are evenly distributed and connected to the nearest BS. The user devices offload tasks to the BSs using wireless channel. The channel gain is defined as $g_{k,n,} = 127 + 30logb_{k,n}$, where $b_{k,n}$ is the distance (in km) between the edge node n and user device $U_{k,n}$ [4].

In this article, the user tasks are generated based on uniform distribution [12] and the task popularity follows Zipf probability distribution with a skewness parameter of 0.56 [3]. The linear coefficients for content transmission delay are chosen

TABLE 3. Cryptographic algorithms for security protection.

Cryptographic algorithms	Security level, s_l	Energy, β_l (J/MB)	Delay, α_l (ms/MB)
RC4	1	2.02	6.30
RC5	2	4.03	12.50
Blowfish	3	5.47	17.00
IDEA	4	6.28	19.6
SKIPJACK	5	6.97	21.7

as $\theta_0 = 5\,ms/km$ and $\theta_1 = 22.3\,ms$ [37]. We consider $\eta = 0.2$ as a default value for the trade-off coefficient and provide more emphasis on security over energy. However, we also consider two other values of η (0.1 and 0.5) and highlight their impact. The summary of the key simulation parameters are presented in Table 2. The simulation parameters carry these default values, unless mentioned otherwise.

To assign security protections based on the requirements of the offloaded tasks, we consider five different crytographic algorithms [26], [50]. These algorithms can be categorized according to their security strengths. However, the energy and delay overheads usually increase with an upsurge in security strength. In Table 3, we summarize diverse levels of security strength and associated overheads for different security protection algorithm.

In this article, we compare the performance of our proposed MEC system with the following strategies presented in the existing literature.

- **Task caching and offloading (TCO) technique:** This method [4] utilizes task caching along with MEC. The popular tasks are cached at the BSs and energy critical tasks are offloaded to the nearest MEC servers. However, the study does not consider cooperation between the BSs and security aspects of task offloading.
- **Cooperative mobile edge computing (CMEC) technique:** In this approach [5], [28], the MEC servers connected to different nodes work collectively. The most suitable MEC server offloads the task of an user considering associated delay and energy consumption. Nonetheless, the potential of task caching and the security breach issues of the offloaded tasks are not investigated.
- **Secure task offloading (STO) technique:** The security aspects of the offloaded tasks are addressed in this model [13]. The model utilizes cryptographic algorithm for data encryption and decryption of an offloaded task. However, the STO technique neither considers the collective operation of the edge servers nor caching. Moreover, the technique does not address

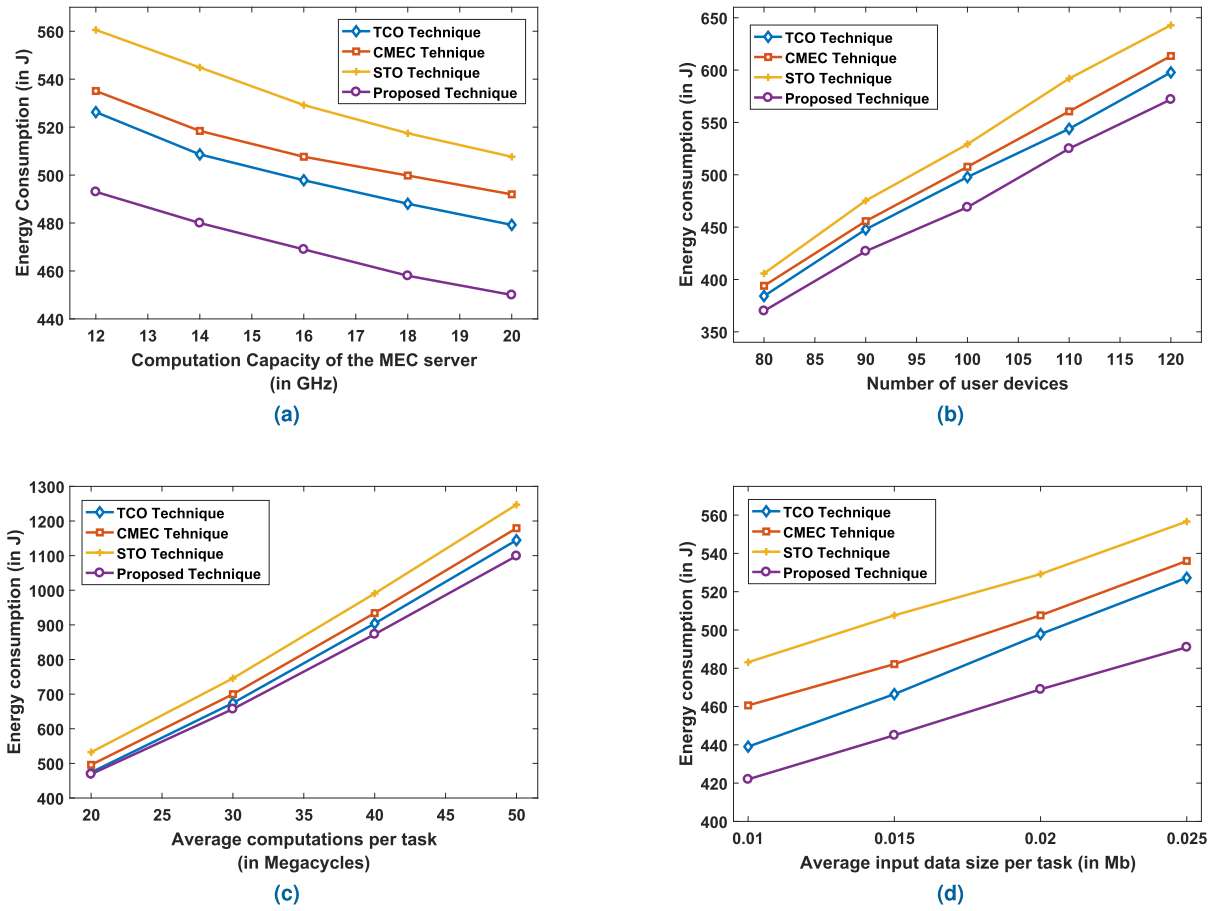

FIGURE 3. The comparison of energy consumption for different MEC techniques with the changing (a) computation capacity of the MEC server; (b) number of user devices; (c) average computations per task; (d) average input data size.

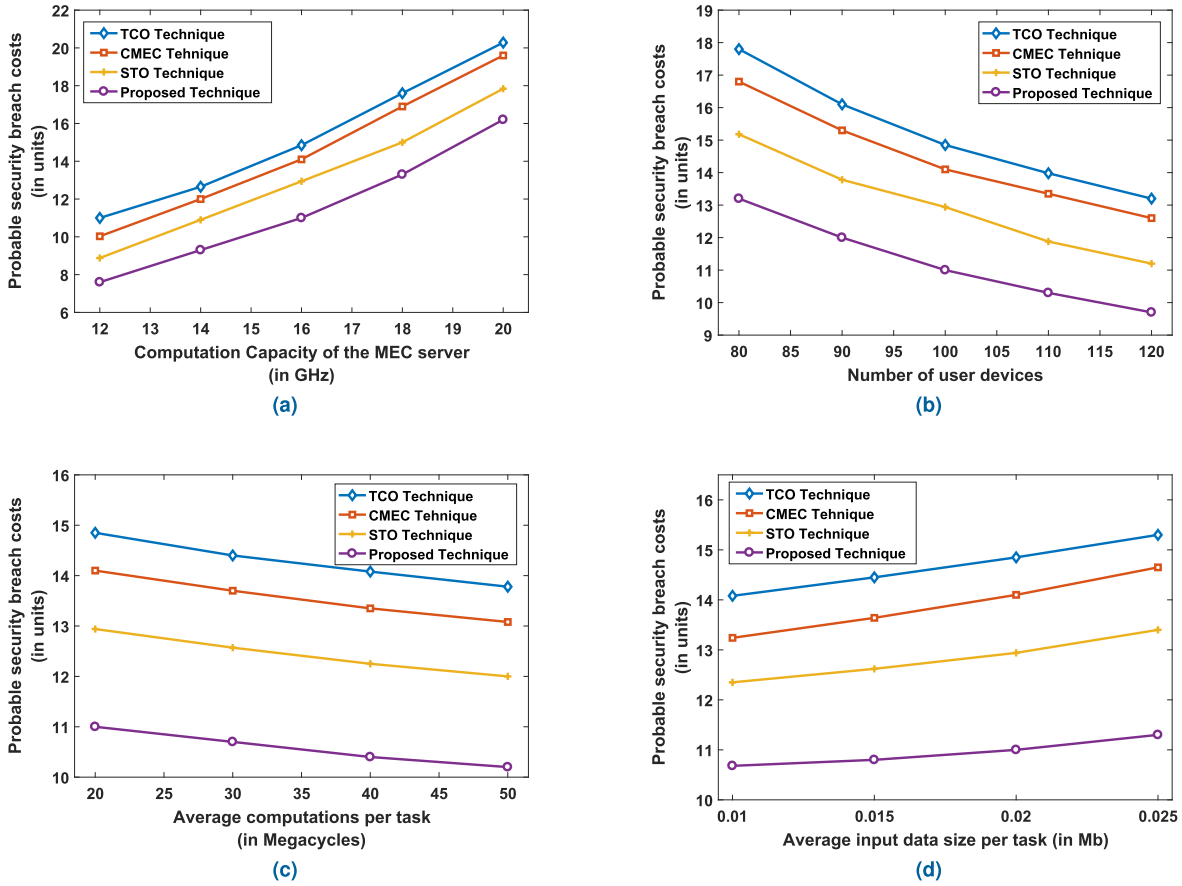

FIGURE 4. The comparison of probable security breach costs for different MEC techniques with the changing (a) computation capacity of the MEC server; (b) number of user devices; (c) average computations per task; (d) average input data size.

different security requirements of the offloaded tasks and the security breach cost.

In contrast to the aforementioned techniques, we not only reduce energy consumption, but also decrease probable security breach cost while maintaining delay requirements of the IoT applications.

In this article, we have generated the results using IBM ILOG CPLEX optimization studio and MATLAB R 2016b. The simulations are conducted on a PC with Intel(R) Core i5 3.8 GHz processor and 16 GB RAM. The simulations are run for 50 times and the average values are taken as numerical results.

Fig. 3 presents the energy consumption for different MEC techniques with respect to diverse system parameters. The *STO technique* has maximum energy consumption because it utilizes neither caching nor cooperation of the MEC servers. Furthermore, this technique also experiences energy overhead because of the security algorithm. The energy consumption in the *CMEC technique* is further improved because the MEC servers are operated collectively. Although the *TCO technique* does not consider cooperation among the MEC servers, it utilizes caching and saves the transmission energy required for the cached-task input data. However, our proposed technique explores both caching and cooperation in MEC. Therefore, the energy consumption decreases significantly, in spite of the energy overheads caused by the security services. The energy savings achieved in the proposed technique is evident in Fig. 3.

Fig. 3(a) shows that the energy consumption decreases with an upraise in the computation capacity of the MEC server, which varies from 12 GHz to 20 GHz. This is because a higher computation capacity allows more tasks to be offloaded at the MEC servers, which provides faster and energy-efficient task execution.

According to Fig. 3(b), the energy consumption increases with the number of the user devices, while the computing capacity remains constant. The MEC servers with finite capacity can not offload all the tasks when more user devices are in the network. Hence, a large portion of the tasks are executed locally and energy consumption increases.

From Fig. 3(c), we observe that the energy consumption increases when the average computations per task is higher. This is because the MEC servers with a fixed computation capacity offloads lower number of user tasks with higher computation requirements. As a result, the number of the locally executed tasks increases. If the computation requirement is smaller, then more tasks are executed at the edge and significant energy savings is achieved.

In Fig. 3(d), we show the energy consumption for changing input data size for different tasks. The energy consumption is higher when the data size increases because it requires more energy to transmit and offload the tasks to the MEC servers.

Fig. 4 shows the probable security-breach cost for the MEC techniques discussed in this article. The *TCO technique* and the *CMEC technique* do not consider the security aspects of MEC. Therefore, in both techniques, the offloaded tasks are prone to malicious attacks and the probable security breach cost is high. The *STO technique* achieves significant reduction in probable security-breach cost because it adopts cryptographic algorithm for security protections. However, this technique does not consider specific security requirements of different tasks. Our proposed MEC model performs best and achieves significant reduction in probable security-breach cost compared to the *STO technique*.

In Fig. 4(a), we present the probable security-breach cost for changing computation capacity of the MEC server. An increase in the computation capacity offloads more tasks to the MEC servers. On the other hand, the offloaded tasks are vulnerable to eavesdropping and malicious attacks. Therefore, the probable security-breach cost increases when computation capacity is higher.

Fig. 4(b) shows that the probable security-breach cost decreases when number of user devices increase. In this scenario, the computation capacity remains constant. If the number of user devices is high, the MEC servers have more options to choose for task offloading. Then the finite computation capacity is utilized to offload less vulnerable tasks and more vulnerable tasks are executed locally. In this model, local execution does not have probable security-breach cost. Hence, this cost is lower for a system with more user devices while the capacity of the MEC servers remains fixed.

We present the impact of the average computations per task on the probable security-breach cost in Fig. 4(c). When the value of the average computations is small, the MEC servers can offload more tasks. Therefore, the probable security-breach cost is higher. This cost does not vary significantly because the security overheads are mainly dependent on the data size, not on the computation requirement.

According to Fig. 4(d), the probable security-breach cost is higher when average input data size increases. This is because the increase in the data size not only causes delay, but also introduces energy overheads. Therefore, for some of the offloaded tasks, the system compromises stringent security protection to meet delay requirements and achieve energy savings.

In Fig. 5, average delay is presented as a function of different system parameters. Fig. 5(a) shows that the delay reduces sharply with increasing computation capacity. When the computing capacity increases from 12 GHz to 20 GHz, the delay reduces by 12%. Higher computation capacity enables more tasks to be offloaded at the MEC server and ensures faster execution. On the other hand, when computation capacity is fixed, an increase in the number of user devices does not allow to offload the additional tasks. Hence, those tasks are executed locally and increases average delay. For 16 GHz computation capacity, the upsurge in the number of user devices from 80 to 120 increases average delay by 8%.

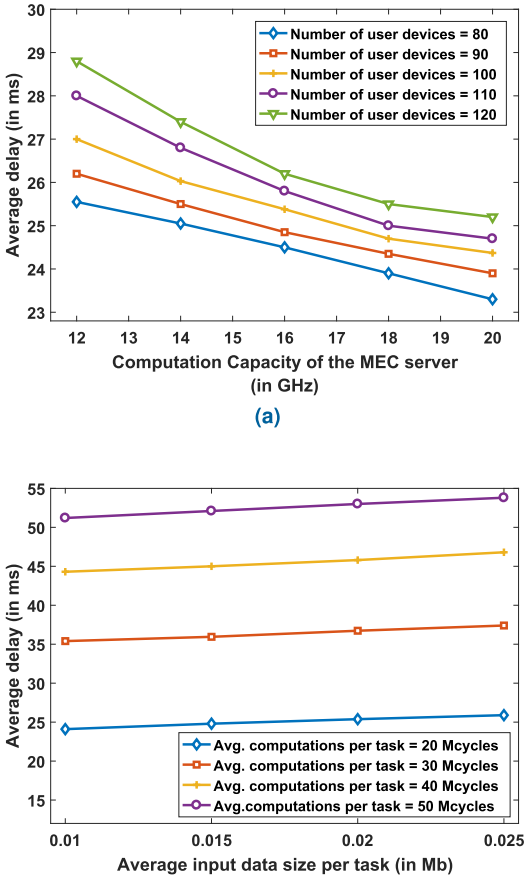

FIGURE 5. Average delay for changing (a) computation capacity of the MEC server and number of user devices; (b) average computations per task and average input data size.

From Fig. 5(b), we observe that average delay increases when either average computations per task or average input data size increases. As the average input data size for IoT tasks is small, the changes in average delay with respect to average computations is significantly dominant compared to the changes regarding input data size.

Fig. 6 shows the joint impact of computation capacity and the number of user devices on task offloading. MEC servers with higher computation capacity accommodates more tasks and increases the percentage of the offloaded tasks. An improvement in computation capacity from 12 GHz to 20 GHz upraises the task offloading percentage from 29% to 45% for a system with 80 user devices connected to each node. When the number of user devices become 100, the task offloading percentage declines to 22% and 31% for 12 GHz and 20 GHz computation capacity, respectively. This is because the servers with limited capacity cannot offload the tasks of the additional user devices.

In this article, the trade-off coefficient (η) is varied as 0.1, 0.2, and 0.5. These three values represent extremely

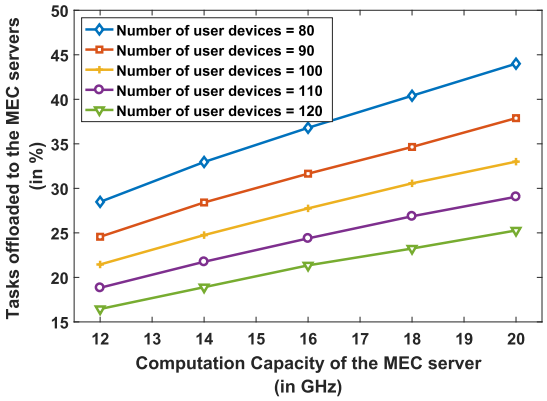

FIGURE 6. Task offloading percentage for changing computation capacity of the MEC servers and number of user devices.

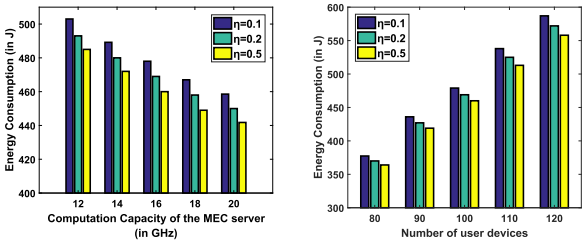

FIGURE 7. Energy consumption for different trade-off coefficients with changing (a) computation capacity of the MEC servers and (b) number of user devices.

secure, highly secure, and moderately secure conditions, respectively. When η possesses a low magnitude, the system shows more sensitivity to the security issues compared to energy consumption. Fig. 7 shows that an increase in the magnitude of η compromises security concerns and achieves further energy savings. Similarly, in Fig. 8, it is evident that the probable security-breach cost upraises when η increases. Stringent security can be achieved by reducing η and sacrificing energy efficiency. This observation remains valid for different computation capacity and the number of user devices.

In Fig. 9 and Fig. 10, we compare the simulation outcomes of the heuristic solution with the ILP solution. Fig. 9(a) and Fig. 9(b) present that, similar to the ILP solution, the energy consumption declines and the probable-security breach cost upraises with increasing computation capacity. However, the heuristic solution is marginally high compared to the ILP solution and the difference varies within 3% to 6%.

From Fig. 10(a) and Fig. 10(b), we observe that an increase in the number of user devices escalates the energy consumption and drops the probable security-breach cost. This remark upholds for both the ILP and the heuristic solution. The heuristic solution attains 2% to 5% higher value compared to the ILP solution. This difference is within an acceptable range, and the heuristic can be utilized to solve the optimization problem in polynomial time.

We have generated results utilizing ILP solver and validated our proposed heuristic comparing with these results. This validation is crucial because ILP solvers can not generate

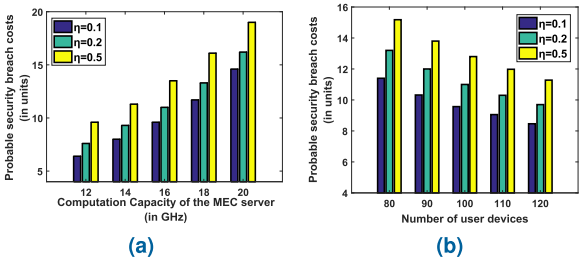

FIGURE 8. Probable security-breach costs for different trade-off coefficients with changing (a) computation capacity of the MEC servers and (b) number of user devices.

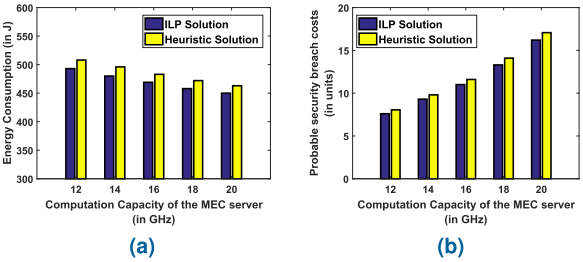

FIGURE 9. Comparison between the ILP solution and the heuristic solution for the changing computation capacity of the MEC sever.

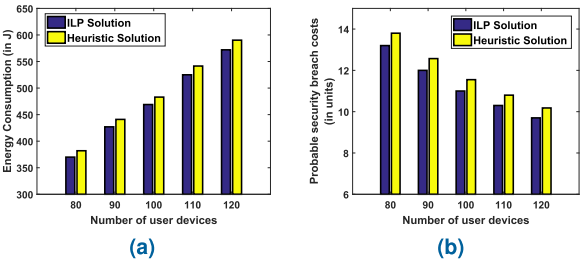

FIGURE 10. Comparison between the ILP solution and the heuristic solution for the changing number of user devices.

polynomial-time solutions for a larger scenario. On the other hand, ultra dense wireless networks are expected to have up to 1 million users per km^2. In this study, we consider $0.04km^2$ coverage area for every BS. Hence, 40,000 mobile devices under the coverage area of a BS is equivalent to 1 million users per km^2. To show the effectiveness of our proposed strategy in ultra dense scenario, we present energy consumption and probable security breach cost with respect to the number of user devices per BS in Fig. 11. The energy consumption cost increases exponentially with increasing user devices. The security breach cost decreases because the MEC servers have more options to choose less vulnerable tasks. These results are consistent with our previous results in Fig. 3 and Fig. 4.

In Table 4, we present the summary of the numerical results and compare our proposed technique with the existing methods [4], [5], [13], [28]. Among the existing MEC techniques, the TCO technique is the best in terms of energy-efficiency and the STO technique performs best according to security aspects. For the default simulation parameters, our proposed technique reduces energy consumption by 8% and probable

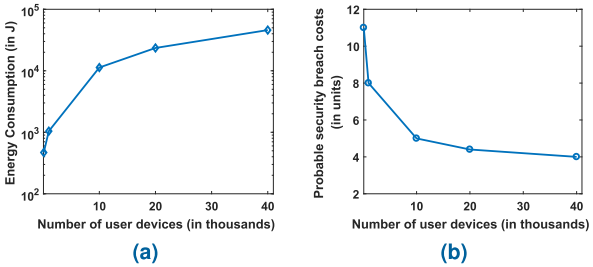

FIGURE 11. The impact of the ultra dense scenario on the proposed technique.

TABLE 4. Summary of the results.

MEC Technique	Energy Consumption (in J)	Probable Security-Breach Cost (in units)
TCO Technique [4]	508	17.60
CMEC Technique [5], [28]	518	16.90
STO Technique [13]	540	12.94
Proposed Technique	469	11.00

security-breach cost by 60% compared to the TCO technique. On the other hand, in comparison with the STO technique, our proposed model achieves 15% energy savings and 18% savings in probable security-breach cost.

In this study, our proposed model and other comparing methods consider static-geographical positions of the user devices. If user mobility is taken into account, the associated overheads will be higher because of caching and computing redundancy. Therefore, the integration of the mobility management into these studies will improve the performance gain further. Compared to other methods, the proposed model is more suitable to incorporate mobility management. This is because our model considers the cooperation between the edge nodes, which is ignored in most of the existing studies. Hence, the inclusion of mobility management into our model is expected to achieve further improvement in terms of energy savings and probable security-breach cost. We look forward to address this in our future work.

VII. CONCLUSION

In this article, we designed an MEC technique for IoT applications based on the security and energy related issues. The proposed model combines caching, cooperative task offloading, and security services assignment to achieve not only stringent security protection, but also energy savings. We developed a constrained optimization model, which reduced the overall cost combining probable security-breach cost and energy consumption. Furthermore, a heuristic solution is designed to solve the optimization problem in polynomial time. Numerical results present that our proposed model can reduce the probable security-breach cost up to 60% and energy consumption up to 15% compared to the current MEC techniques. In the future, we will explore the mobility management and incentive mechanisms for security services in MEC with caching facilities.

REFERENCES

[1] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.
[2] P. Liu, G. Xu, K. Yang, K. Wang, and X. Meng, "Jointly optimized energy-minimal resource allocation in cache-enhanced mobile edge computing systems," *IEEE Access*, vol. 7, pp. 3336–3347, 2019.
[3] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, and V. C. M. Leung, "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4283–4294, Jun. 2019.
[4] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.
[5] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for IoT applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1159–1166, May 2019.
[6] IoT Analytics. (2014). *Why the Internet of Things is Called Internet of Things: Definition, History, Disambiguation*. [Online]. Available: https://iot-analytics.com/internet-of-things-definition/
[7] J. Ni, X. Lin, and X. S. Shen, "Efficient and secure service-oriented authentication supporting network slicing for 5G-enabled IoT," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 644–657, Mar. 2018.
[8] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultradense IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4317–4329, Jun. 2019.
[9] P. Brown. (2018). *Is Battery Life Hindering Growth Internet Things Devices*. [Online]. Available: https://electronics360.globalspec.com/article/13112/is-battery-life-hin dering-the-growth-of-internet-of-things-devices
[10] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.
[11] X. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[12] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.
[13] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 100, pp. 531–541, Nov. 2019.
[14] K. Li, M. Tao, and Z. Chen, "Exploiting computation replication for mobile edge computing: A fundamental computation-communication tradeoff study," 2019, *arXiv:1903.10837*. [Online]. Available: http://arxiv.org/abs/1903.10837
[15] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.
[16] Statista. *Internet Things (IoT) Connected Devices Installed Base Worldwide From 2015 to 2025 (in Billions)*. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
[17] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, "D2D computation offloading optimization for precedence-constrained tasks in information-centric IoT," *IEEE Access*, vol. 7, pp. 94888–94898, 2019.
[18] J. A. Khan, C. Westphal, and Y. Ghamri-Doudane, "Information-centric fog network for incentivized collaborative caching in the Internet of everything," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 27–33, Jul. 2019.
[19] Y. Lan, X. Wang, D. Wang, Z. Liu, and Y. Zhang, "Task caching, offloading, and resource allocation in D2D-aided fog computing networks," *IEEE Access*, vol. 7, pp. 104876–104891, 2019.
[20] J. Yao and N. Ansari, "Caching in energy harvesting aided Internet of Things: A game-theoretic approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3194–3201, Apr. 2019.
[21] Z. Zhao, Y. Shi, B. Diao, and B. Wu, "Optimal data caching and forwarding in industrial IoT with diverse connectivity," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2288–2296, Apr. 2019.

[22] M. I. A. Zahed, I. Ahmad, D. Habibi, Q. V. Phung, and M. M. Mowla, "Proactive content caching using surplus renewable energy: A win–win solution for both network service and energy providers," *Future Gener. Comput. Syst.*, vol. 105, pp. 210–221, Apr. 2020.
[23] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 116–122, Jun. 2018.
[24] B. Huang, Z. Li, P. Tang, S. Wang, J. Zhao, H. Hu, W. Li, and V. Chang, "Security modeling and efficient computation offloading for service workflow in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 755–774, Aug. 2019.
[25] M. I. Aziz Zahed, I. Ahmad, D. Habibi, and Q. V. Phung, "Content caching in industrial IoT: Security and energy considerations," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 491–504, Jan. 2020.
[26] W. Jiang, K. Jiang, X. Zhang, and Y. Ma, "Energy optimization of security-critical real-time applications with guaranteed security protection," *J. Syst. Archit.*, vol. 61, no. 7, pp. 282–292, Aug. 2015.
[27] D. He, S. Chan, and M. Guizani, "Security in the Internet of Things supported by mobile edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 56–61, Aug. 2018.
[28] Y. Dong, S. Guo, J. Liu, and Y. Yang, "Energy-efficient fair cooperation fog computing in mobile edge networks for smart city," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7543–7554, Oct. 2019.
[29] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.
[30] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.
[31] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 459–474, Jun. 2019.
[32] V. Balasubramanian, F. Zaman, M. Aloqaily, S. Alrabaee, M. Gorlatova, and M. Reisslein, "Reinforcing the edge: Autonomous energy management for mobile device clouds," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 44–49.
[33] V. Balasubramanian, N. Kouvelas, K. Chandra, R. V. Prasad, A. G. Voyiatzis, and W. Liu, "A unified architecture for integrating energy harvesting IoT devices with the mobile edge cloud," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 13–18.
[34] S. K. Mishra, D. Puthal, B. Sahoo, S. Sharma, Z. Xue, and A. Y. Zomaya, "Energy-efficient deployment of edge dataenters for mobile clouds in sustainable IoT," *IEEE Access*, vol. 6, pp. 56587–56597, 2018.
[35] V. Balasubramanian, F. Zaman, M. Aloqaily, I. A. Ridhawi, Y. Jararweh, and H. B. Salameh, "A mobility management architecture for seamless delivery of 5G-IoT services," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[36] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.
[37] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.
[38] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "Joint resource allocation for software-defined networking, caching, and computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 274–287, Feb. 2018.
[39] X. Liu, J. Zhang, X. Zhang, and W. Wang, "Mobility-aware coded probabilistic caching scheme for MEC-enabled small cell networks," *IEEE Access*, vol. 5, pp. 17824–17833, 2017.
[40] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.
[41] H. M. Al-Kadhim and H. S. Al-Raweshidy, "Energy efficient and reliable transport of data in cloud-based IoT," *IEEE Access*, vol. 7, pp. 64641–64650, 2019.
[42] Y. Wu, J. Shi, K. Ni, L. Qian, W. Zhu, Z. Shi, and L. Meng, "Secrecy-based delay-aware computation offloading via mobile edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4201–4213, Jun. 2019.
[43] T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for IoT devices," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4162–4175, Jun. 2019.
[44] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017.
[45] R. Huo, F. R. Yu, T. Huang, R. Xie, J. Liu, V. C. M. Leung, and Y. Liu, "Software defined networking, caching, and computing for green wireless networks," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 185–193, Nov. 2016.
[46] A. Khreishah, H. Bany Salameh, I. Khalil, and A. Gharaibeh, "Renewable energy-aware joint caching and routing for green communication networks," *IEEE Syst. J.*, vol. 12, no. 1, pp. 768–777, Mar. 2018.
[47] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.
[48] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.
[49] A. Pal, H. K. Rath, S. Shailendra, and A. Bhattacharyya, "IoT standardization: The road ahead," in *Internet of Things-Technology, Applications and Standardization*. London, U.K.: IntechOpen, 2018, pp. 53–74, doi: 10.5772/intechopen.75137.
[50] Z. Li, J. Ge, C. Li, H. Yang, H. Hu, B. Luo, and V. Chang, "Energy cost minimization with job security guarantee in Internet data center," *Future Gener. Comput. Syst.*, vol. 73, pp. 63–78, Aug. 2017.

M. ISHTIAQUE A. ZAHED received the B.Sc. and M.Sc. degrees in electrical and electronic engineering from the Bangladesh University of Engineering and Technology (BUET), in 2011 and 2015, respectively. He is currently pursuing the Ph.D. degree with the School of Engineering, Edith Cowan University, Australia. He served as a Faculty Member with the School of Engineering, East Delta University, Bangladesh, from 2012 to 2016. His research interests include green communications, the IoT, wireless communications, and network security.

IFTEKHAR AHMAD (Member, IEEE) received the Ph.D. degree in communication networks from Monash University, Australia, in 2007. He is currently an Associate Professor with the School of Engineering, Edith Cowan University, Australia. His research interests include 5G technologies, green communications, QoS in communication networks, software-defined radio, wireless sensor networks, and computational intelligence.

DARYOUSH HABIBI (Senior Member, IEEE) received the B.E. degree (Hons.) in electrical engineering and the Ph.D. degree from the University of Tasmania, Australia, in 1989 and 1994, respectively. His employment history includes Telstra Research Laboratories, Flinders University, Intelligent Pixels Inc., and Edith Cowan University, where he is currently a Professor, the Pro Vice-Chancellor, and the Executive Dean of engineering. His current research interests include engineering design for sustainable development, smart energy systems, environmental monitoring technologies, and reliability and quality of service in communication systems and networks. He is a Fellow of Engineers Australia and the Institute of Marine Engineering, Science, and Technology.

QUOC VIET PHUNG (Member, IEEEE) received the Ph.D. degree in communication networks from Edith Cowan University, Australia, in 2010. He is currently a Postdoctoral Research Fellow with the School of Engineering, Edith Cowan University. His current research interests include 5G technologies, green communications, wireless sensor networks, and computational intelligence.