IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 June 2023, accepted 22 June 2023, date of publication 28 June 2023, date of current version 6 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3290139

RESEARCH ARTICLE

Energy-Efficient and Security-Aware Task Offloading for Multi-Tier Edge-Cloud Computing Systems

WALEED ALMUSEELEM
Faculty of Computing and Information Technology (FCIT), University of Tabuk, Tabuk 47713, Saudi Arabia
e-mail: Waleedalmuseelem@ut.edu.sa

ABSTRACT Recently, edge-cloud computing (ECC) has emerged as a new paradigm for alleviating the intensive overhead for mobile IoT applications. However, data security remains a significant concern that has not been adequately addressed. Moreover, the diversity of mobile devices leads to overloaded edge servers and thereby perpetually increasing the latency and limiting the gain of performance. Therefore, this paper proposes a new security, load balancing, and energy-aware task offloading framework for the ECC system environment to bypass potential security threats and the edge servers' balancing challenge. Specifically, a new layer of security based on an advanced encryption standard (AES) cryptographic method and fingerprint combination is introduced in order to protect the data from vulnerabilities during offloading. Moreover, to organize the load on edge servers, a new load-balancing algorithm is being developed. Subsequently, task offloading, data security, and load balancing are jointly formulated as an integer problem whose objective is to reduce the system's energy with latency constraints. Finally, extensive simulation results demonstrated that our model is scalable and can save about 19%, 17.5%, 20.3%, 14.4%, and 13% of system energy with respect to other benchmark solutions.

INDEX TERMS Cloud/edge computing, security, computation offloading, load balancing, optimization.

I. INTRODUCTION

The exponential expansion of wireless sensors and the Internet of Things (IoTs) has resulted in the eruption of a vast array of industrial IoT applications and complicated services that create vast quantities of data. Additionally, many industrial IoT applications and services demand increased computing capacity and energy efficiencies, such as augmented and virtual reality, real-time online gaming, the internet of vehicles (IoV), smart healthcare, image recognition, and smart healthcare [1], [2]. However, the power supplies and computational capabilities for these devices pose new obstacles to the emergence of these applications to meet the Quality-of-Service (QoS) [3], [4].

Consequently, the computation offloading concept is introduced as a prominent solution to alleviate these limitations, where the intensive and delay tasks could be transmitted and remotely executed at more powerful devices [5], [6]. Through the mobile cloud computing (MCC) paradigm, cloud computing has become a useful service for mobile devices, where it offers a range of adaptable processing, storage, and service capabilities while consuming less energy [7], [8]. MCC is nevertheless criticized as having high latency of communication and security issues, especially in delay-sensitive and real-time applications [9].

In order to address MCC's main limitations, the edge cloud computing (ECC) paradigm has just recently been introduced as a viable solution, where the placement and installation of their resources at the edge of mobile networks [10], [11]. For instance, real-time video analytic is considered as one of the killer application types that can efficiently utilize the ECC paradigm as it demeaned more processing and computation.

Several methods for offloading computation have been developed for mobile devices, some of which were handled

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

in a single-server environment, while others were dealt with in a multi-server environment [12], [13], [14]. However, the majority of the offloading methods in place right now let mobile devices send their tasks to the same edge server that's connected to them, and as a result, these edge servers are subject to higher loads, which invariably results in increased latency for mobile devices and limits performance gains. Furthermore, it may well be the case that some of these devices are not capable of completing computation tasks within a reasonable amount of time. This is because it takes overloaded edge servers to communicate and compute. In the meantime, data offloading is still fraught with security threats that have not been well handled. This motivates our study in this paper to propose an energy-efficient and data-secured framework for multi-user with multi-task offloading in a multi-tier environment. The main contribution is detailed and clarified in the next subsection.

A. CONTRIBUTIONS

The objective of this paper is to minimize the energy consumption of mobile devices while simultaneously protecting the data of applications from cyber-attacks before they are transferred to the MEC server. Therefore, in this study, to bypass potential security threats and the edge servers' balancing challenge, a new layer of security based on an advanced encryption standard (AES) cryptographic method and fingerprint combination is introduced in order to protect the data from vulnerabilities during offloading. Furthermore, a novel load-balancing algorithm is being developed to distribute loads among edge servers. The main contributions of this study can be summarized as follows:

- To protect the vulnerability of data during transmission, a new layer of AES cryptographic method is introduced, which is secured with a fingerprint-based security encryption and decryption key.
- To balance the load between edge servers, an efficient algorithm is proposed where mobile devices are re-distributed between edge servers according to their location, CPU cycles, current number of users, and available data rate, and each device is forced to be handed over to the most suitable available edge server.
- Task offloading, data security, and load balancing are collectively stated as an optimization problem to minimize the energy cost of mobiles while meeting the latency restriction.
- Finally, simulation outcomes proved that our model is scalable and can save about 19%, 17.5%, 20.3%, 14.4%, and 13% of system energy with respect to other benchmark solutions (i.e., local, edge, cloud, and work in [15] and [16]).

The remainder of this work is structured as follows. The Section II describes the related works. The system model is introduced in Section III which also explains the problem formulation for our suggested framework. Section IV demonstrates the discussion and experimental results. Section V presents the paper's conclusion.

II. RELATED WORK

Recently, MEC provides mobile IoT devices with extensive computation and storage capacity at the edge of the network to handle IoT device issues. In the majority of these studies, the issue of offloading computations in single-tier contexts has been discussed [13], while others have dealt with offloading computations in multi-tier environments [17]. An overview of common models is provided in this section.

A collaborative task offloading and resource allocation framework for MEC networks is investigated in [15], where the computation tasks of IoT devices can be partially performed locally, or at the edge or cloud nodes. In addition, an offloading scheme is developed based on the pipeline, in which the intensive task of both IoT devices and edge nodes can be respectively offloaded to a particular edge and cloud node, with the goal of minimizing the overall latency of IoT devices. Further, an efficient classic successive convex approximation-based scheme is proposed to derive the solution. Finally, their results indicated that using the pipeline strategy in the proposed scheme make it efficient and can achieve better performance than other offloading approaches.

An integrated architecture for enhancing cloud computing security is proposed in [18]. This architecture utilized blockchain technology to guarantee data integrity for all homophobic encryption schemes as well as decentralized data storage and exchange platform. In addition, a safe and transparent system for cloud-based data storage and exchange is developed. Whereas in cite, Zhongjin et al. proposed an energy-aware and secured task offloading framework for D2D communication. Specifically, they provided a technique for work offloading in D2D communication that takes both energy efficiency and security into account. In addition, the main objective of this study is to safeguard sensitive data and calculations from unauthorized access while conserving energy to extend the battery life of the associated devices.

Elsewhere, [19], [20], and [21] jointly considered task offloading and resource allocation for MEC networks. More specifically, He et al. [19] focused on enhancing the performance of task offloading and resource allocation in UAV-assisted VANETs by balancing the need to improve resource consumption, facilitating efficient task offloading, and assuring network security. They provided a system that strikes a trade-off between job offloading, resource allocation, and security guaranteeing. The objective of the current solution is likely to be to ensure that jobs are offloaded to UAVs quickly, that available computing resources are utilized effectively, and that sensitive data is protected from unwanted access. Finally, in terms of successful task processing ratios and task processing delays, their approach performs significantly better than other schemes. Meanwhile, in [20], a deep learning-based task offloading technique is proposed for MEC networks, where it makes dynamic decisions for tasks regarding where jobs should be offloaded for processing. The main objective is to optimize the performance of task offloading by ensuring tasks are processed efficiently and with the quickest response time possible. Whereas in [21],

an energy-efficient and deadline-aware algorithm is proposed for MEC networks, where each IoT device can offload its tasks to edge servers placed at the network's edge. In addition, this algorithm prioritizes tasks depending on their deadlines and the available energy of the mobile device and edge servers with the goal of maximizing the performance of task offloading by ensuring that tasks are processed efficiently and with the quickest reaction time feasible, while conserving energy. The simulations show that the proposed algorithm improves deadline satisfaction ratios by 85.4% and reduces average response times by 62.9%, while only 13.4% more energy is consumed by IoT devices.

Similar to the enumerated efforts, Alhelaly et al. [16] jointly optimized task offloading and resource allocation for multi-user multi-UAV-aided ECC systems. They proposed an efficient and scalable model that can support the network traffic without effect on the performance. Meanwhile, in a different contribution, task offloading and bandwidth allocation are jointly optimized [22], in which the processing requirements of tasks, the available resources at the edge servers, and the condition of the network are considered through an efficient approach. In addition, this approach enables real-time, dynamic decisions regarding where tasks should be handled and how bandwidth should be allocated. According to experimental findings, the proposed model significantly boosts the energy and time efficiency of the system.

Recently, [23] and [24] proposed a task offloading scheme for Internet of Vehicles systems. More specifically, in [23], an end-edge-cloud vehicle architecture for task computation offloading is proposed using three task computing approaches. In addition, an Asynchronous Advantage Actor-Critic (A3C)-based task offloading technique is designed to solve the formulated problem and finds optimal offloading decisions in the IoV's dynamic environment. Meanwhile, in [24], a distributed multi-hop task offloading decision model for task execution efficiency is developed, which primarily is consisting of two parts which are a candidate vehicle selection mechanism for screening nearby vehicles that can participate in offloading and a task offloading decision algorithm for determining the task offloading solution. Finally, according to the results, offloading in which all tasks are completed locally, as well as offloading from neighboring vehicles, is more efficient in terms of time delay performance, task number, task required computation power, and task size environment than offloading in which all tasks are completed under greedy or bat-based algorithms.

According to the above summary of related research, many methodologies and models have been presented for multi-user job offloading in MEC networks employing single or multiple edge servers with or without the cloud. According to our study, the highlighted studies failed to appropriately examine load balancing concerns across multi-edge nodes, which greatly affect consumers' energy expenses. Moreover, the lack of data protection for applications in multi-user, multi-tier environments, is considered another drawback that has not been addressed well.

III. SYSTEM MODEL

A. NETWORK MODEL

As part of this section, a model of the proposed multi-user, multi-tier ECC environment will be presented, where our goal is to minimize the system's energy consumption. This system is made up of three layers, as shown in Fig. 1, with a **M** set of mobile devices linked to a **k** set of edge servers. Moreover, each device has a set of **N** intensive tasks to complete. Further, the edge servers' set is managed through a centralized router, which is responsible for routing traffic, addressing load, and linking them with the remote cloud. $\mathcal{M} = \{1, 2, \ldots, M\}$, $\mathcal{N} = \{1, 2, \ldots, N\}$, and $\mathcal{K} = \{0, 1, 2, \ldots, K+1\}$ are used in this study to indicate the sets of mobile devices, tasks, and available servers for task execution, where 0 denotes the local execution and $K+1$ denotes the cloud execution. Mobile devices can access edge server resources and offload compute duties via wireless channels. Moreover, guided by the previous work [25], [26], [27], the set of mobile devices remains quasi-static during the task offloading period but may change between periods.

Suppose $a_{ijk} \in \{0, 1\}$ represents the decision to offload task j from device i to server k. It is specifically $(a_{ij0} = 1)$ indicates local execution, $(a_{ijK+1} = 1)$ indicates cloud execution, and $(a_{ijk} = 1, \forall k \in [1..K])$ indicates edge server execution. Additionally, each task can only be completed on one server, whether it is local, edge, or cloud-based. Thus, the following equation can be used to guarantee that these requirements are met:

$$\sum_{k=0}^{K+1} a_{ijk} = 1, \tag{1}$$

B. MODEL OF COMMUNICATION

Throughout this section, we describe our communication model, which includes **M** mobile devices and **k** edge servers. In addition, each device is expected to complete **N** intense tasks. Furthermore, these tasks can be represented with a 4-tuple value $\{\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}\}$ (i.e., input size, the output size, the CPU cycles, and the required deadline for each task), in which these values can be determined via a task execution careful profiling [28], [29].

Hence, orthogonal frequency division multiple access is used for simultaneous offloading in the same channel to reduce intracellular interference in uplink transmission [30]. Moreover, according to Shannon law, each device's available data transmission and receiving rate is as follows:

$$R_{ik}^{U} = B_{ik}^{U} \log_2\left(1 + \frac{p_i^T G^2}{\omega B_{ik}^{U}}\right), \tag{2}$$

$$R_{ik}^{D} = B_{ik}^{D} \log_2\left(1 + \frac{p_k^T G_2}{\omega D_{ik}^{D}}\right), \tag{3}$$

where B_{ik}^{U}, B_{ik}^{D}, p_i^T, p_k^T are denote the up-link, downlink bandwidth, and transmission power of the device and edge server, respectively. In addition, G and ω are indicate the channel gain and noise power of the edge server.

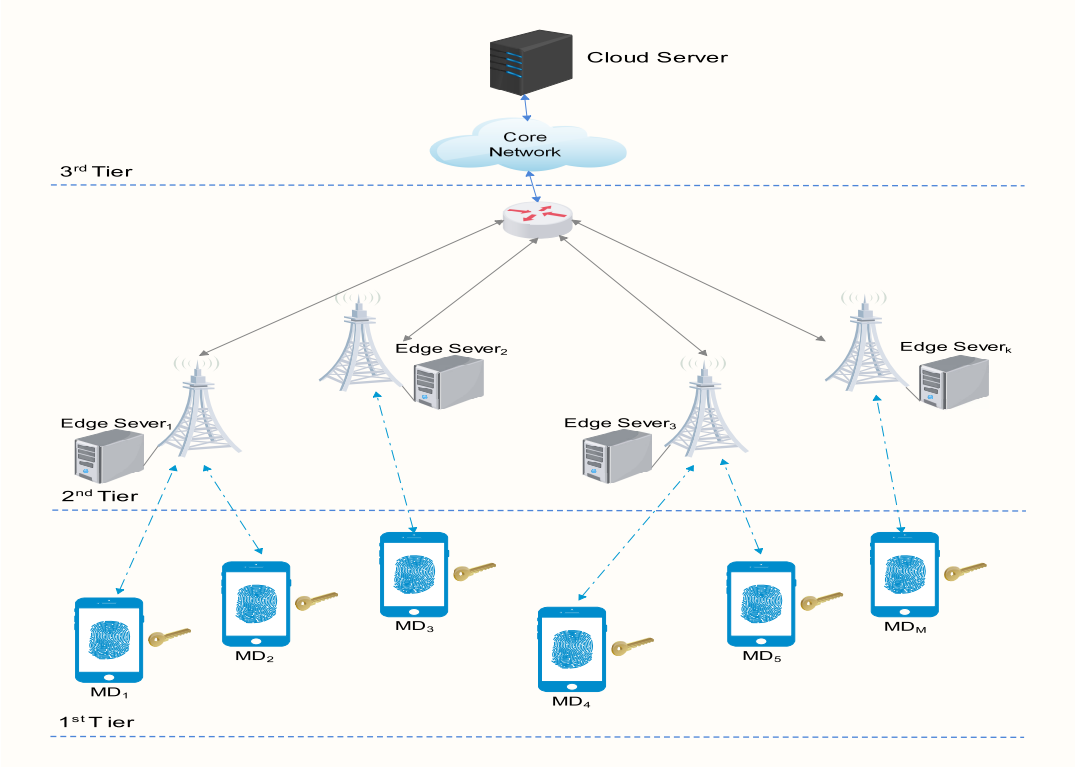

FIGURE 1. System model.

C. MODEL OF COMPUTATION

This section outlines the computation model for our environment, which includes **M** mobile devices linked to **k** edge servers. In addition, each device has a **N** set of intensive tasks that should be completed. Therefore, both local and remote calculations are detailed in further detail in the subsections below.

1) LOCAL EXECUTION

The study considers the possibility that different devices have varying levels of processing power. As such, we can use equations 4 and 5 to calculate the amount of time and energy required to do all the computations locally on each device i:

$$T_{ij}^{L} = \frac{\gamma_{ij}}{f_i^{L}}, \tag{4}$$

$$E_{ij}^{L} = \psi_i \gamma_{ij}, \tag{5}$$

where ψ_i and f_i^{L} are denote the energy consumed per each cycle of CPU and device i computing capability.

2) REMOTE EXECUTION

In this subsection, we introduce the possibility of remotely executing the tasks of mobile devices, with each task being allocated and then executed by one of the available servers. Therefore, 6 and 7 can be used to estimate the time required for all computations to be done remotely: 6 and 7:

$$T_{ij}^{e} = T_{ij}^{off} + T_{ij}^{e\_ex} + T_{ij}^{don}, \tag{6}$$

$$T_{ij}^{c} = T_{ij}^{off} + \Delta + T_{ij}^{c\_ex} + T_{ij}^{don}, \tag{7}$$

where Δ denotes the propagation delay between edge server and cloud as well as $= T_{ij}^{off}$, $T_{ij}^{e\_ex}$, $T_{ij}^{c\_ex}$, and T_{ij}^{don} are indicate the offloading, edge execution, cloud execution and downloading time, respectively, that can be expressed as:

$$T_{ij}^{off} = \frac{\alpha_{ij}}{R_{ik}^{U}}, \tag{8}$$

$$T_{ij}^{e\_ex} = \frac{\gamma_{ij}}{f_i^{e}}, \tag{9}$$

$$T_{ij}^{c\_ex} = \frac{\gamma_{ij}}{f_i^{c}}, \tag{10}$$

$$T_{ij}^{don} = \frac{\beta_{ij}}{R_{ik}^{D}}, \tag{11}$$

where f_i^{e} and f_i^{c} are denote the assigned capabilities for device i at edge and cloud server.

Moreover, based on equation 12, the amount of energy necessary to perform all the computations remotely at one of the servers can be calculated as:

$$E_{ij}^{R} = p_i T_{ij}^{off}, \tag{12}$$

It should be noted that edge servers have computing resources denoted by F_k, which will be allocated to all devices connected by the offloading process and thereby

should be limited as follows:

$$\sum_{i=1}^{M}\sum_{i=1}^{N}\sum_{i=1}^{K} a_{ijk} \leq F_K, \tag{13}$$

D. MODEL OF LOAD BALANCING

Designing a load-balancing process between edge servers that jointly considers computation and communication resources is a complex task that requires balancing multiple competing objectives, while also taking into account real-time considerations, scalability, energy efficiency, privacy, security, and QoS guarantees. Fig.2 shows an example of mobile device Distribution on edge servers. As shown in the figure, we can observe the following limitations:

1) Resource allocation: One of the main challenges in designing a load-balancing process is to find an efficient and fair way to allocate resources between edge servers. This includes balancing the load between edge servers based on computation and communication requirements, while also taking into account factors such as energy consumption and network congestion.
2) Real-time considerations: Load balancing in edge server networks must be done in real-time, which means that the algorithm must be able to quickly respond to changes in the network, such as new user connections or changes in resource availability.
3) Scalability: As the number of edge servers and users in the network increases, the load-balancing process must be able to scale to handle the increased demand.
4) Energy efficiency: The load balancing algorithm must be designed to minimize energy consumption, as edge servers are typically powered by batteries and need to be energy-efficient to ensure long-term operation.
5) Complexity: The load-balancing algorithm must be able to handle the complexity of the network and make decisions based on a large amount of data and variables.
6) QoS guarantee: The load balancing algorithm must guarantee the Quality of Service (QoS) of the users, ensuring that the users' needs are met.

Therefore, we must spread the load between edge servers by compelling mobile devices to transfer to the best available edge server [31]. The technique for doing this is described in the following text.

First, a summary of mobile devices is transmitted to the central control by each edge server. This summary includes the number of linked mobile devices, data rate, CPU cycles assigned to each mobile device, and the number of mobile devices that can be reallocated to other edge servers. Following that, the central control manager cycles through the mobile devices, compelling them to give over to the most appropriate edge servers. The steps required to execute our suggested algorithm for load balancing are depicted in Algorithm 1.

Consider a scenario with three edge servers and twelve mobile devices, with seven, four, and one MD connected to edge server$_1$, edge server$_2$, and edge server$_3$, respectively, to illustrate how the algorithm operates. In addition, MD_3, MD_4, MD_5, MD_6, and MD_{10} are co-located at the edge servers boundary, allowing them to be assigned as required. Moreover, the bandwidth of the channel is assumed to be 20 MHz for the three edge servers and each edge server has a computing capability of 20 GHz. The main objective of this study is to spread the load between edge servers to optimize the quality of service and latency, as described in the following sections.

Algorithm 1 Load Balancing Algorithm

1: **Input**: Every mobile device i is assigned to edge server k.
2: **Output**: Mobile devices are forced and connected to the most suitable edge server through the hand-over process.
3: **for all** edge server k **do**
4: $\iota \leftarrow$ Count the number of connected mobile devices.
5: $\chi \leftarrow$ Calculate the data rate for each mobile user i and its assigned capabilities.
6: $\varkappa \leftarrow$ Locate the mobile devices that are reallocatable.
7: Regarding the values of ι, χ, and $\varkappa$, each mobile device is forced to be handed over to the most suitable edge server.
8: **end for**

As shown in Table 1, the central control manager initially obtains a summary of mobile device information from edge servers. According to this table, MD_3 is connected to edge server$_1$ and can be reassigned to edge server$_2$. Similar to MD_4, MD_5, MD_6, and MD_{10}, edge server$_1$ and edge server$_2$ are linked to MD_4, MD_5, MD_6, and MD_{10}, and they can be reallocated to edge server$_2$ and edge server$_3$. Consequently, it can iterate through these mobile devices and then find the most appropriate edge server to connect to MD_3 is chosen, where the upload and calculation durations for the available edge servers are received (5.3 seconds for edge server$_1$ and 3.7 seconds for edge server$_2$). The optimal edge server (in this case, edge servers$_2$) is then chosen, and the rate of data and computing capabilities allocated to each user is modified. This procedure is repeated till all mobile devices have been assigned to the most suitable edge servers. In practice, software-defined network controllers and the standard OpenFlow protocol can be utilized at the backbone router to load balance edge servers, where it has a global perspective of the network and can make more accurate and efficient load balancing decisions [32].

E. MODEL OF SECURITY

During offloading, the mobile devices may opt to wirelessly transmit information about the processing burden to ECC servers. This may raise the system's susceptibility to various forms of attack. In addition, the confidentiality of sensitive and private data may be compromised, hence diminishing the advantages of ECC [33]. To avoid these dangers, we want an efficient and secure layer that encrypts offloaded data, so shielding it from attackers. In this paper, the AES method

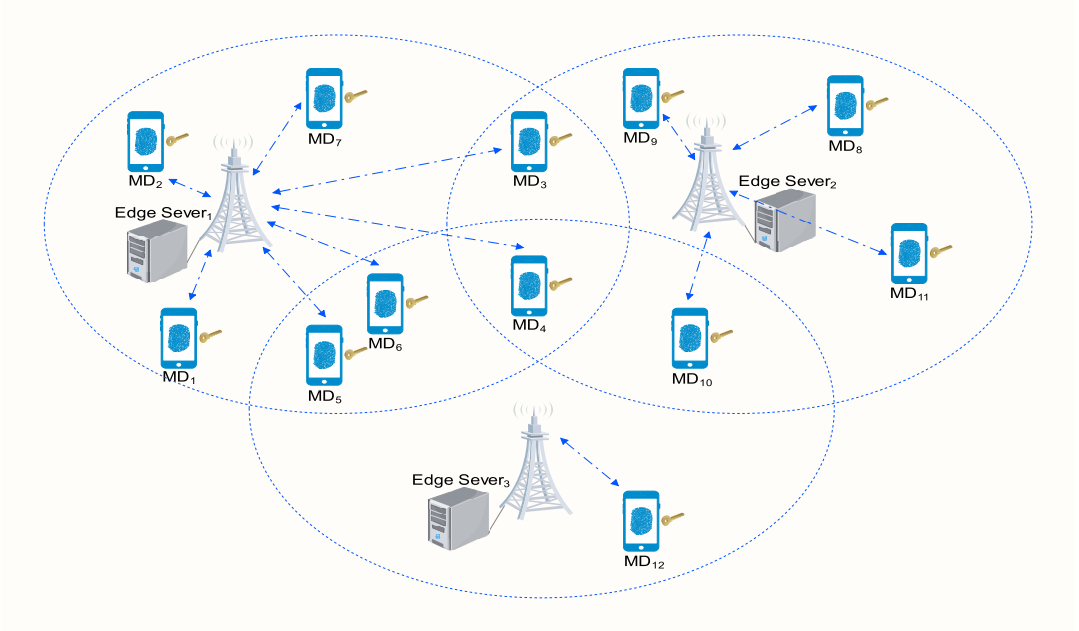

FIGURE 2. Example of mobile devices distribution.

TABLE 1. Proposed algorithm illustration.

Mobile Devices	Size of Data	CPU Cycle	Transferring Time	Execution Time at edge server	Best edge server
MD_3	10 MB	10 Gigacycle	edge $server_1 \rightarrow 5.3s$ edge $server_2 \rightarrow 3.7s$	edge $server_1 \rightarrow 3.5s$ edge $server_2 \rightarrow 2.5s$	edge $server_2$
MD_4	20 MB	25 Gigacycle	edge $server_1 \rightarrow 8.9s$ edge $server_2 \rightarrow 8.9s$ edge $server_3 \rightarrow 2.8s$	edge $server_1 \rightarrow 7.6s$ edge $server_2 \rightarrow 7.6s$ edge $server_3 \rightarrow 2.5s$	edge $server_3$
MD_5	15 MB	20 Gigacycle	edge $server_1 \rightarrow 5.6s$ edge $server_3 \rightarrow 3.4s$	edge $server_1 \rightarrow 5.0s$ edge $server_3 \rightarrow 3.0s$	edge $server_3$
MD_6	15 MB	10 Gigacycle	edge $server_1 \rightarrow 4.5s$ edge $server_3 \rightarrow 4.5s$	edge $server_1 \rightarrow 2.0s$ edge $server_3 \rightarrow 2.0s$	edge $server_1$
MD_{10}	5 MB	30 Gigacycle	edge $server_2 \rightarrow 2.2s$ edge $server_3 \rightarrow 1.5s$	edge $server_2 \rightarrow 9.1s$ edge $server_3 \rightarrow 6.0s$	edge $server_3$

is combined with fingerprint images in order to protect the data and decrease its complexity over time.

The Advanced Encryption Standard (AES) is a symmetric method that has been shown to be effective in numerous applications [34], [35]. However, its elementary algebraic structure and identical replication method for encrypting each block in the signal make it vulnerable to a variety of attacks. In addition, the security key is frequently referred to as the strength-determining component of the algorithm. Incorporating fingerprint images into the classic AES algorithm increases resilience, which is assessed by key efficiency and indistinguishability of ciphertext.

A fingerprint is a pattern of ridges and valleys on the surface of the finger that is unique and recognizable. These patterns are formed during fetal development and remain unchanged throughout a person's life [36], [37]. Fingerprints are used for various purposes such as identification, authentication, and forensic analysis. Fingerprints are considered to be one of the most accurate forms of biometric identification as they are unique to each individual and do not change over time. In this paper, cryptographic keys will be generated based on fingerprint images to encrypt and decrypt data. The methods necessary to include the fingerprint into our AES scheme are detailed below [38], [39]. Each user develops fingerprint images using wearable devices, imbuing the images with randomness and individuality. These images are preprocessed to enhance the quality of the fingerprint images, reduce noise, and make the features of the fingerprints more prominent. This improves the accuracy and reliability of the fingerprint recognition system. Then, extract the unique characteristics of a fingerprint that can be used to identify an individual using one of the correlation-based methods [40]. These characteristics, known as minutiae, include ridge endings, bifurcations, and dots. Based on these

features, the procedures described in [41] are subsequently used to generate cryptographic keys. Using the produced keys, the encryption and decryption processes are performed.

For mobile device i, we refer to a binary decision of security as $\varepsilon_i \in \{0, 1\}$, in which $\varepsilon_i = 0$ refers to unsecured data will be sent to the server and $\varepsilon_i = 1$ indicates that secured data will be sent to the server. During computational tasks, the privacy of data is determined by the user's behavior, which means that security decisions are made manually by each user according to the requirements they have. As a result of applying the security layer, there is an additional overhead in energy and time as follows:

$$t_{ij}^{en+de} = \frac{\varrho_{ij}}{f_i^L} + \frac{\varsigma_{ij}}{f_i^e}, \tag{14}$$

$$e_{ij}^{en} = \psi_i \varsigma_{ij}, \tag{15}$$

where ϱ_{ij} and ς_{ij} indicate the cycles of CPU to encrypt and decrypt data if mobile device i [42], [43]. Following the application of the security decision, the time and energy overhead to remotely send the task j of a mobile device i at a cloud server or edge server can be calculated as follows:

$$T_{ij}^{se} = \left[\varepsilon_i(t_{ij}^{en+de} + T_{ij}^{off}) + (1 - \varepsilon_i)T_{ij}^{off}\right], \tag{16}$$

$$E_{ij}^{se} = \left[\varepsilon_i(e_{ij}^{en} + E_{ij}^{R}) + (1 - \varepsilon_i)E_{ij}^{R}\right], \tag{17}$$

Finally, the total time and energy required for performing tasks are calculated based on computation, communication, load balancing, and security models:

$$T_{ij} = \left[a_{ij0}T_{ij}^{L} + a_{ijK+1}(T_{ij}^{se} + \delta + T_{ij}^{c\_ex}) + \sum_{k=1}^{K} a_{ijk}(T_{ij}^{se} + T_{ij}^{e\_ex})\right], \tag{18}$$

$$E_{ij} = \left[a_{ij0}E_{ij}^{L} + \sum_{k=1}^{K+1} a_{ijk}E_{ij}^{se}\right], \tag{19}$$

F. FORMULATION OF PROBLEM

In this section, we build our model for a multi-tier ECC system with the goal of reducing energy use while meeting latency requirements. Our optimization problem is therefore formulated as follows:

$$\min_{a} \left[\sum_{i=1}^{N}\sum_{j=1}^{M} E_{ij}\right]$$

$$\text{s.t} \left[E_{ij} - E_{ij}^{L}\right] \le 0, \quad \forall_k \in [1..K] \qquad C1$$

$$\left[T_{ij} - T_{ij}^{L}\right] \le 0, \quad \forall_k \in [1..K] \qquad C2$$

$$\sum_{k=0}^{K+1} a_{ijk} = 1, \quad \forall_{i,j} \qquad C3$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K+1} a_{ijk}f_i^e \le F_k, \quad \forall_k \in [1..K] \qquad C4$$

$$a_{ijk} \in \{0, 1\}, \quad \forall_{i,j} \qquad C5$$

$$\varepsilon_i \in \{0, 1\}, \qquad C6$$

(20)

wherein the first two constraints (C1 and C2) are responsible for energy and delay bounds, and the third constraint (C3) makes sure that each task is only done once. In addition, the fourth constraint (C3) is concerned with the capability bound of edge servers. Finally, the last two constraints ensure task offloading and security decision binarization.

The problem's solution can be derived by obtaining optimal task offloading and security values. However, the values for the security decision can be determined based on the data's sensitivity. We presumptively assume that the user set this. This problem, therefore, belongs to the integer linear class, where both the objective function and constraints are linear. Moreover, it is classified as NP-hard and has a non-convex feasible set [44]. As a result, the binary relaxation approach can be used to transform this problem into a convex one, in which variables are relaxed into real variables, and a new formulation is presented in equations 21 [45]. It is also possible to derive an efficient offloading decision by using branch and bound methods [46].

$$\min_{a} \left[\sum_{i=1}^{N}\sum_{j=1}^{M} E_{ij}\right]$$

$$\text{s.t} \left[E_{ij} - E_{ij}^{L}\right] \le 0, \quad \forall_k \in [1..K] \qquad C1$$

$$\left[T_{ij} - T_{ij}^{L}\right] \le 0, \quad \forall_k \in [1..K] \qquad C2$$

$$\sum_{k=0}^{K+1} a_{ijk} = 1, \quad \forall_{i,j} \qquad C3$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K+1} a_{ijk}f_i^e \le F_k, \quad \forall_k \in [1..K] \qquad C4$$

$$a_{ijk} \in [0, 1], \quad \forall_{i,j} \qquad C5$$

$$\varepsilon_i \in \{0, 1\}, \qquad C6$$

(21)

IV. PERFORMANCE RESULTS AND DISCUSSION

During the simulations, we use a personal computer with an Intel® Core(TM) i7-10750H CPU running at 2.6 GHz and 16 GB of RAM, and Windows 10 Home 64-bit. In addition, MatLab(R2019b) is installed and utilized to simulate the proposed framework and solve our model efficiently. A multi-tier ECC environment is considered that consists of one cloud server, five edge servers, 100 mobile devices, and three independent tasks per device that each device can perform locally or remotely at the edge or cloud server. Each user's fingerprint image is obtained from a publicly accessible and widely used database, namely FVC 2000, 2002, and 2004 [47]. The data in all three FVC databases are identical [47]. Each database has four distinct datasets (DB1, DB2, DB3 and DB4). Each dataset has 110 fingerprints from eight different fingers. This study makes use of eighty fingerprint pictures (of 10 individuals with 8 images per

TABLE 2. Simulation parameters.

Parameter	Value
Edge Servers K	5
MD M	100
Tasks N	3
Bandwidth	20 MHz
Noise of background	-100 dBm
Data size	$(0, 10)$MB Unif-Dist
Transmission power	100 mW
Cycles of CPU	500 cycles/bit
Energy consumption per CPU cycle	$(0, 20 \times 10^{-11})$ J/cycle
MD Capability	$\{0.4, 0.5, \ldots, 1.0\}$ GHz
Cloud Capability	500 GHz
Server capability	100 GHz

individual). All fingerprint images are 8-bit grayscale TIF files with a resolution of 500 dpi. In contrast, these databases employ various scanners and collection methods. To recover the major features from these preprocessed images, the minutiae extraction method outlined in [48] is applied. These features are then used to construct the cryptographic key that is used to secure the AES cryptography technique used prior to data transmission. In addition, on the edge and cloud server, CPU capabilities are set to 100GHz and 500GHz, respectively, while on the mobile device, they are uniformly distributed within $\{0.4, 0.5, \ldots, 1.0\}$. Moreover, it is estimated that 500 CPU cycles are required per byte for each task with data sizes uniformly distributed within a range of 0 to 10 MB. According to [27], local computing energy consumes $(0, 20 \times 10^{-11})$ Joule per cycle in a uniform fashion. A uniformly distributed pseudo-random function is used to determine the value of decision security with regard to user behavior since user behavior varies from application to application. The channel bandwidth is set to 20 MHz and transmission power and background noise of mobile devices are set to 100 mW and -100 dBm. Table 2 lists the other simulation settings that are required for the computation and communication tasks to be carried out by the model. Using these specifications, the simulation is executed 50 times and the average values are obtained.

A. EFFECT OF LOAD BALANCING ADDITION

This subsection evaluates the performance of applying our load-balancing algorithm within a scenario without load balancing consideration. Fig. 3, shows the allocated users at each edge server in the two scenarios. As seen thence, in the latter scenario, GBS_1 and GBS_4 are underloaded, while GBS_2 and GBS_3 are overloaded. Moreover, our proposed algorithm roughly balances the load by redistributing most of the allocatable users to the appropriate GBS. Further, UAVs will act as MEC nodes for the still-overloaded GBSs (i.e., GBS_2) thereby improving the overall system performance as we showed later.

B. EFFECT OF SECURITY ADDITION

In this subsection, the effect of security layer addition is illustrated, where the performance of our proposed model

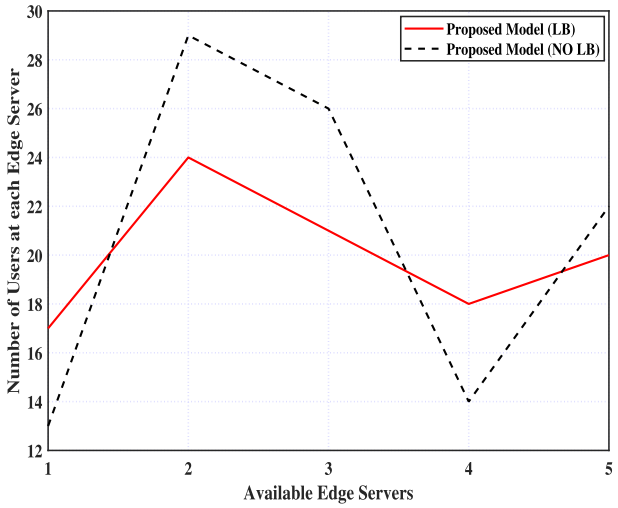

FIGURE 3. Users distribution at edge servers With and without load balancing layer addition (M = 100, N = 3, K = 5).

with and without the security layer is evaluated. Fig. 4 shows the energy consumption for performing the computation tasks under a different number of users using our proposed model with and without the addition of a security layer. It's observed from the figure that, the consumed energy of our secured model is a little greater than the unsecured model, for a small number of users. In addition, this cost is growing as the number of users increases. This result is traced to the consumed energy demands that are associated with the process of encryption and decryption. However, enhancing data security is a critical point that should be considered compared with this overhead consumption.

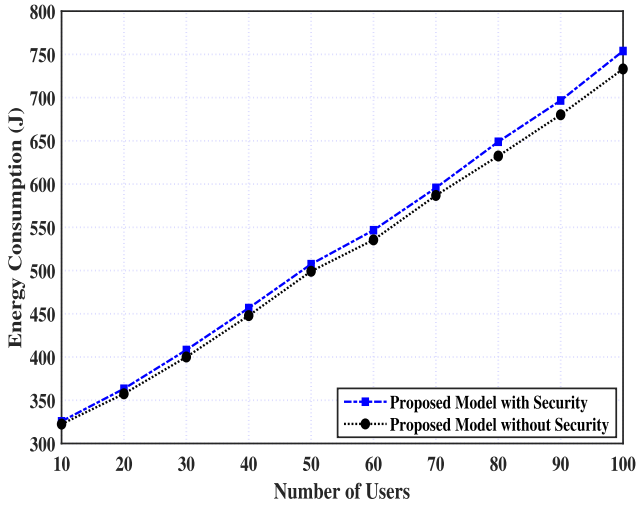

FIGURE 4. Energy consumption for the proposed mode with and without security layer addition (M = 100, N = 2, K = 5).

C. PERFORMANCE MEASUREMENTS WITH OTHER MODELS

As part of this subsection, five different approaches are compared to verify the performance of our model:

- **Local Execution:** This approach performs all the tasks at the device's resources without offloading.

- **Edge Execution:** This approach offloads and performs all the tasks at the connected edge server.
- **Cloud Execution:** This approach offloads and performs all the tasks at the cloud node.
- **Task Offloading Approach in [15]:** This approach performs the tasks at one of the available servers based on the work in [15].
- **Task Offloading Approach in [16]:** This approach performs the tasks at one of the available servers based on the work in [16].

We evaluate the performance of the proposed model against the five different offloading approaches and investigate how changes in user numbers, data sizes, tasks and edge servers affect performance. First, the consumed energy for performing the tasks over a different number of users is presented in Fig. 5. It is deduced from this figure that the proposed model outperforms the other approaches. In addition, the gap between the approaches is not large especially for a small number of users. However, this gap is increasing as the number of users increases. Moreover, the consumed energy for the edge and cloud execution approaches exceeds the local execution approach (i.e., when the number of users is greater than 60). This result is attributed to the communication channels' resources have competed between users which leads to consuming more energy through the transmission. Additionally, GBSs' capabilities are enough for handling many users, thereby suggesting the redistribution of users among GBSs and utilizing the UAVs as MEC nodes have a profound impact on the overall system performance.

Second, the consumed energy for performing the tasks over a different data size is presented in Fig. 6. It is shown from this figure that the amount of energy consumed by the four approaches (i.e., cloud, edge, and work in [15] and [16]) is almost equal and less than the local execution approach, whereas the proposed model is superior to them. In addition, with the increase in the size of data, the energy consumed by edge and cloud execution approaches increases rapidly and exceeds the energy consumed by the local execution approach as well. This result is because increasing data sizes leads to longer communications and thereby consumes more energy. However, by smartly adapting our model, we can not only balance the load among GBSs, but also derive the best decisions as well as handle computation tasks efficiently, thus consuming less energy.

Similarly, the consumed energy for performing the tasks over a different number of tasks is presented in Fig. 7. From this figure, we deduce that our model can maintain a lower energy consumption than the other approaches. In addition, with a growing number of tasks, the consumed energy of the edge execution approach grows rapidly, even exceeding the consumed energy of the local execution approach. This result is traced to the GBSs' computing resources are shared between all the tasks and multiple of these tasks will be performed simultaneously, which affects the performance.

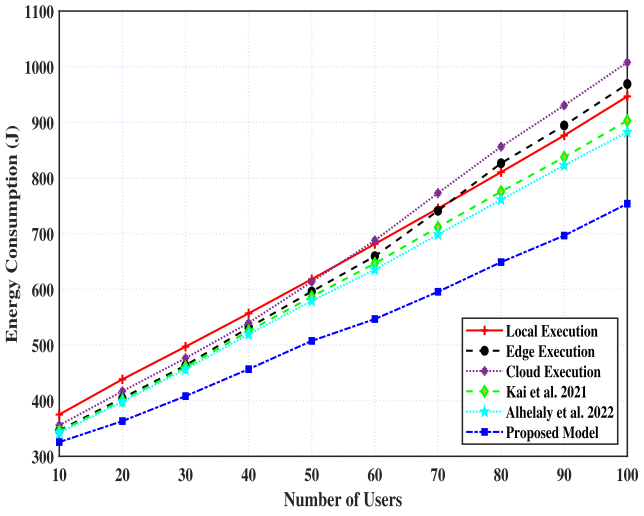

FIGURE 5. Energy consumption versus different number of mobile users (N = 3, K = 5).

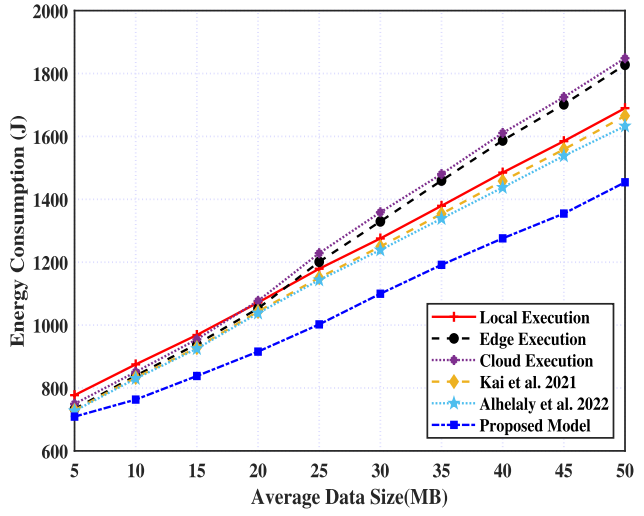

FIGURE 6. Energy consumption versus different data size (M = 100, N = 3, K = 5).

Furthermore, the consumed energy for performing the tasks over a different number of edge servers is presented in Fig. 8. Based on this figure, it is evident that the local execution approach does not depend on the number of edge servers, whereas the energy consumed by the other approach steadily decreases with a growing number of edge servers. Further, our proposed model can also be more energy efficient and outperform other approaches. This is due to the fact that with a growing number of edge servers, the consumed energy will be decreased as the user will be assigned more resources, while these resources are not utilized by the local execution approach.

Finally, Fig. 9 shows the total consumed energy for performing the tasks over five different types applications (see Table 3). Based on the results, the proposed approach consumed the least amount of energy with respect to other approaches. This is due to the fact that the proposed model

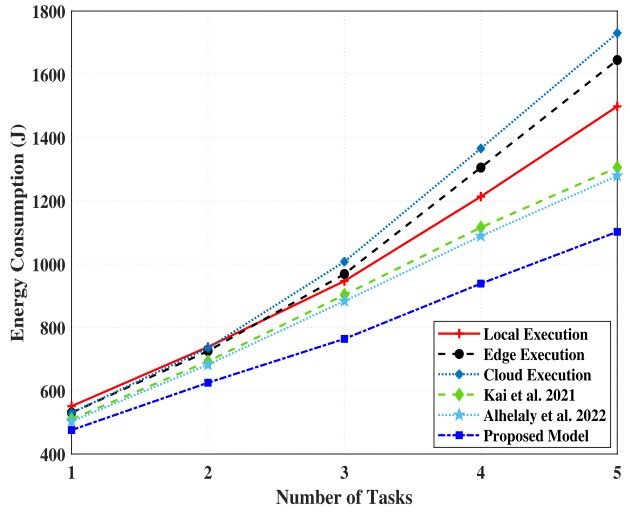

FIGURE 7. Energy consumption versus different tasks (M = 100, K = 5).

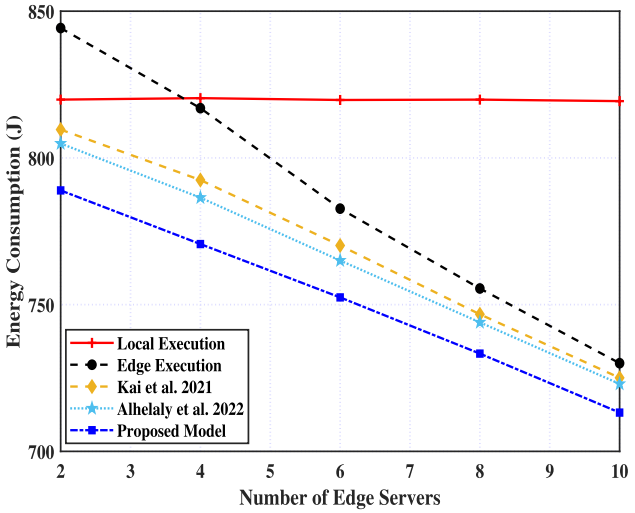

FIGURE 8. Energy consumption versus different edge servers (M = 100, N = 3).

can smartly select the appropriate execution location based on the environmental settings.

D. DISCUSSION

Our model offers several significant contributions, and as part of this subsection, we report the main contributions of the proposed model, the differences between it and the other schemes reported in Section II, as well as the performance enhancements it offers over the other schemes.

As listed in Table 2, most of the reported computation offloading models only let mobile devices send their tasks to the same edge server that's connected to them, and as a result, these edge servers are subject to higher loads, which invariably results in increased latency for mobile devices and limits performance gains. Furthermore, it may well be the case that some of these devices are not capable of completing computation tasks within a reasonable amount of time. This

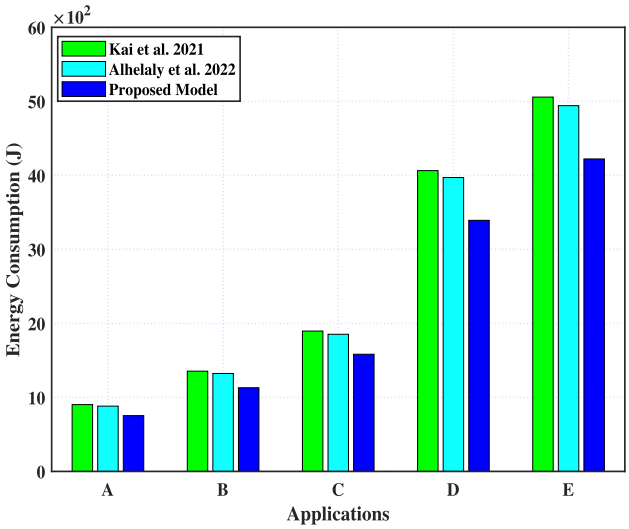

FIGURE 9. Energy consumption versus different applications' types (M = 100, N = 3, K = 5).

TABLE 3. Applications complexity.

Application	Label	CPU Cycle/Byte
Health Monitoring	A	500
Automatic Number Plate Reading	B	960
x264 CBR encode	C	1900
Traffic Management	D	5900
Augmented Reality	E	12000

is because it takes overloaded edge servers to communicate and compute. In the meantime, data offloading is still fraught with security threats that have not been well handled. In contrast, the work in this study addresses a multi-tier edge cloud computing environment with considering the load balancing and security issue, in which a new layer of AES cryptographic method is introduced, which is secured with a fingerprint-based security encryption and decryption key. In addition, an efficient algorithm is proposed where mobile devices are re-distributed between edge servers according to their location, CPU cycles, current number of users, and available data rate, and each device is forced to be handed over to the most suitable available edge server and thereby balance the load between edge servers. Finally, simulation outcomes proved that our model is scalable and can save about 19%, 17.5%, 20.3%, 14.4%, and 13% of system energy with respect to other benchmark solutions (i.e., local, edge, cloud, and work in [15] and [16]).

V. CONCLUSION

This research proposed an energy-efficient and secure task offloading Framework for a multi-tier edge-cloud computing system using a new layer of AES cryptographic approach with a fingerprint-based security encryption and decryption key. In addition, to balance the load between edge servers,

an efficient algorithm is proposed where mobile devices are re-distributed between edge servers according to their location, CPU cycles, current number of users, and available data rate, and each device is forced to be handed over to the most suitable available edge server. Moreover, task offloading, data security, and load balancing are jointly formulated as an integer problem whose objective is to reduce the system's energy with latency constraints. Finally, extensive simulation results demonstrated that our model is scalable and can save about 19%, 17.5%, 20.3%, 14.4%, and 13% of system energy with respect to other benchmark solutions (i.e., local, edge, cloud, and work in [15] and [16]).

As part of future work, mobile devices' mobility will be handled, so that they can dynamically change their locations and move between edge servers during offloading. Moreover, an automated decision of security will be improved by developing an intelligent technique based on user behavior. Finally, deep learning techniques will be applied to model problems' solutions and address the complexity of making decisions on ECC large scale.

REFERENCES

[1] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of Things applications: A systematic review," *Comput. Netw.*, vol. 148, pp. 241–261, Jan. 2019.

[2] M. Khayyat, A. Alshahrani, S. Alharbi, I. Elgendy, A. Paramonov, and A. Koucheryavy, "Multilevel service-provisioning-based autonomous vehicle applications," *Sustainability*, vol. 12, no. 6, p. 2497, Mar. 2020.

[3] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 179–198, 1st Quart., 2013.

[4] Y. G. Kim, J. Kong, and S. W. Chung, "A survey on recent OS-level energy management techniques for mobile processing units," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 10, pp. 2388–2401, Oct. 2018.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[6] M. Khayyat, I. A. Elgendy, A. Muthanna, A. S. Alshahrani, S. Alharbi, and A. Koucheryavy, "Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks," *IEEE Access*, vol. 8, pp. 137052–137062, 2020.

[7] A. Boukerche, S. Guan, and R. E. D. Grande, "Sustainable offloading in mobile cloud computing: Algorithmic design and implementation," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–37, Jan. 2020.

[8] F. J. G. Peñalvo, A. Sharma, A. Chhabra, S. K. Singh, S. Kumar, V. Arya, and A. Gaurav, "Mobile cloud computing and sustainable development: Opportunities, challenges, and future directions," *Int. J. Cloud Appl. Comput.*, vol. 12, no. 1, pp. 1–20, Oct. 2022.

[9] M. Maray and J. Shuja, "Computation offloading in mobile cloud computing and mobile edge computing: Survey, taxonomy, and open issues," *Mobile Inf. Syst.*, vol. 2022, pp. 1–17, Jun. 2022.

[10] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[11] I. A. Elgendy and R. Yadav, "Survey on mobile edge-cloud computing: A taxonomy on computation offloading approaches," in *Security and Privacy Preserving for IoT and 5G Networks*. Cham, Switzerland: Springer, 2022, pp. 117–158.

[12] Z. Bingjie, Y. Yanhong, and C. Shaozhong, "A review of computing offloading schemes for multi-access edge computing," *J. Front. Comput. Sci. Technol.*, vol. 4, no. 3, pp. 1–19, 2021.

[13] 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.

[14] N. A. Abu-Taleb, F. H. Abdulrazzak, A. T. Zahary, and A. M. Al-Mqdashi, "Offloading decision making in mobile edge computing: A survey," in *Proc. 2nd Int. Conf. Emerg. Smart Technol. Appl. (eSmarTA)*, Oct. 2022, pp. 1–8.

[15] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 2, pp. 624–634, Jun. 2021.

[16] S. Alhelaly, A. Muthanna, and I. A. Elgendy, "Optimizing task offloading energy in multi-user multi-UAV-enabled mobile edge-cloud computing systems," *Appl. Sci.*, vol. 12, no. 13, p. 6566, Jun. 2022.

[17] Q. Pham, F. Fang, V. N. Ha, Md. J. Piran, M. Le, L. B. Le, W. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[18] R. Awadallah, A. Samsudin, J. S. Teh, and M. Almazrooie, "An integrated architecture for maintaining security in cloud computing based on blockchain," *IEEE Access*, vol. 9, pp. 69513–69526, 2021.

[19] Y. He, D. Zhai, F. Huang, D. Wang, X. Tang, and R. Zhang, "Joint task offloading, resource allocation, and security assurance for mobile edge computing-enabled UAV-assisted VANETs," *Remote Sens.*, vol. 13, no. 8, p. 1547, Apr. 2021.

[20] S. Yang, G. Lee, and L. Huang, "Deep learning-based dynamic computation task offloading for mobile edge computing networks," *Sensors*, vol. 22, no. 11, p. 4088, May 2022.

[21] S. Azizi, M. Othman, and H. Khamfroush, "DECO: A deadline-aware and energy-efficient algorithm for task offloading in mobile edge computing," *IEEE Syst. J.*, vol. 17, no. 1, pp. 952–963, Mar. 2023.

[22] S. Song, S. Ma, X. Zhu, Y. Li, F. Yang, and L. Zhai, "Joint bandwidth allocation and task offloading in multi-access edge computing," *Exp. Syst. Appl.*, vol. 217, May 2023, Art. no. 119563.

[23] C. Chen, H. Li, H. Li, R. Fu, Y. Liu, and S. Wan, "Efficiency and fairness oriented dynamic task offloading in Internet of vehicles," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1481–1493, Sep. 2022.

[24] C. Chen, Y. Zeng, H. Li, Y. Liu, and S. Wan, "A multihop task offloading decision model in MEC-enabled Internet of vehicles," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3215–3230, Feb. 2023.

[25] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[26] 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.

[27] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[28] X. Lyu and H. Tian, "Adaptive receding horizon offloading strategy under dynamic environment," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 878–881, May 2016.

[29] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors," *Sensors*, vol. 19, no. 5, p. 1105, Mar. 2019.

[30] S. Deb and P. Monogioudis, "Learning-based uplink interference management in 4G LTE cellular systems," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 398–411, Apr. 2015.

[31] F. Gunnarsson, A. Centonza, M. Fadaki, and L. Lindbom, "Methods and apparatus for hetergeneous network handover," U.S. Patent 10 299 178, May 21, 2019.

[32] Y.-D. Lin, C.-H. Hsu, M.-T. Thai, C.-T. Wang, Y.-J. Lu, and Y.-T. Chiang, "SAMF: An SDN-based framework for access point management in large-scale Wi-Fi networks," *J. Commun. Softw. Syst.*, vol. 13, no. 4, pp. 189–196, Jan. 2018.

[33] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.

[34] J. Daemen and V. Rijmen, *The Design of Rijndael, AES—The Advanced Encryption Standard*. Berlin, Germany: Springer, 2013.

[35] S. S. Rekha and P. Saravanan, "Low-cost AES-128 implementation for edge devices in IoT applications," *J. Circuits, Syst. Comput.*, vol. 28, no. 4, Apr. 2019, Art. no. 1950062.

[36] K. Bahmani, R. Plesh, P. Johnson, S. Schuckers, and T. Swyka, "High fidelity fingerprint generation: Quality, uniqueness, and privacy," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 3018–3022.

[37] Y.-H. Jo, S.-Y. Jeon, J.-H. Im, and M.-K. Lee, "Security analysis and improvement of fingerprint authentication for smartphones," *Mobile Inf. Syst.*, vol. 2016, pp. 1–11, Mar. 2016.
[38] R. Srividya and B. Ramesh, "Implementation of AES using biometric," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 9, no. 5, p. 4266, Oct. 2019.
[39] E. Rahmawati, M. Listyasari, A. S. Aziz, S. Sukaridhoto, F. A. Damastuti, M. M. Bachtiar, and A. Sudarsono, "Digital signature on file using biometric fingerprint with fingerprint sensor on smartphone," in *Proc. Int. Electron. Symp. Eng. Technol. Appl. (IES-ETA)*, Sep. 2017, pp. 234–238.
[40] G. Aguilar, G. Sanchez, K. Toscano, M. Salinas, M. Nakano, and H. Perez, "Fingerprint recognition," in *Proc. IEEE 2nd Int. Conf. Internet Monitor. Protection (ICIMP)*, Jul. 2007, p. 32.
[41] G. Panchal and D. Samanta, "A novel approach to fingerprint biometric-based cryptographic key generation and its applications to storage security," *Comput. Electr. Eng.*, vol. 69, pp. 461–478, Jul. 2018.
[42] I. A. Elgendy, W. Zhang, C. Liu, and C.-H. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 79–87, Jan. 2021.
[43] C. Paar and J. Pelzl, *Understanding Cryptography, A Textbook for Students and Practitioners*. Bochum, Germany: Springer Universität, 2009.
[44] E. Munapo, "Solving the binary linear programming model in polynomial time," *Amer. J. Operations Res.*, vol. 6, no. 1, pp. 1–7, 2016.
[45] M. Anthony, E. Boros, Y. Crama, and A. Gruber, "Quadratic reformulations of nonlinear binary optimization problems," *Math. Program.*, vol. 162, nos. 1–2, pp. 115–144, Mar. 2017.
[46] S. Boyd and J. Mattingley, "Branch and bound methods," Stanford Univ., Stanford, CA, USA, Appl. Notes EE364b, 2007, p. 7.
[47] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Cham, Switzerland: Springer, 2009.
[48] M. Hammad and K. Wang, "Fingerprint classification based on a Q-Gaussian multiclass support vector machine," in *Proc. Int. Conf. Biometrics Eng. Appl.*, Apr. 2017, pp. 39–44.

WALEED ALMUSEELEM was born in Riyadh, Saudi Arabia, in 1981. He received the M.S. and Ph.D. degrees in computer science and technology from the College of Computer Science and Technology, Wuhan University of Technology, China, in 2012 and 2017, respectively. He is currently an Assistant Professor with the Faculty of Computer and Information Technology, University of Tabuk, Saudi Arabia. His research interests include cloud computing security and data privacy.