BEdgeHealth: A Decentralized Architecture for Edge-Based IoMT Networks Using Blockchain

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

***Abstract*—The healthcare industry has witnessed significant transformations in e-health services by using mobile-edge computing (MEC) and blockchain to facilitate healthcare operations. Many MEC-blockchain-based schemes have been proposed, but some critical technical challenges still remain, such as low Quality of Services (QoS), data privacy, and system security vulnerabilities. In this article, we propose a new decentralized health architecture, called *BEdgeHealth* that integrates MEC and blockchain for data offloading and data sharing in distributed hospital networks. First, a data offloading scheme is proposed where mobile devices can offload health data to a nearby MEC server for efficient computation with privacy awareness. Moreover, we design a data-sharing scheme, which enables data exchanges among healthcare users by leveraging blockchain and interplanetary file system. Particularly, a smart contract-based authentication mechanism is integrated with MEC to perform decentralized user access verification at the network edge without requiring any central authority. The real-world experiment results and evaluations demonstrate the effectiveness of the proposed *BEdgeHealth* architecture in terms of improved QoS with data privacy and security guarantees, compared to the existing schemes.**

***Index Terms*—Blockchain, data offloading, data sharing, healthcare, mobile edge computing (MEC), security.**

I. Introduction

RECENT advances in mobile-edge computing (MEC) and Internet of Medical Things (IoMT) technologies have promoted smart e-health services [2] in the healthcare industry. In hospital networks, health data collected from mobile devices (MDs), i.e., smartphones, laptops, and tablets, can be offloaded to nearby MEC servers for low-latency data execution, which, thus, helps improve users' computation experience, enhance Quality of Services (QoS), and reduce computing burden on local devices. Furthermore, MEC also enables the ubiquitous sharing of health data acquired from the offloading phase among health users to support healthcare delivery [3]. For example, a doctor can exploit health data at the network edge to serve disease diagnosis and treatment, and patients can gain medical benefits such as medication advice in a low-latency fashion. Particularly, blockchain, a disruptive technology for enabling healthcare in recent years, is able to provide high security for health data sharing due to its unique features, such as decentralization, traceability, and immutability [4]. In fact, blockchain can ensure reliable data exchange among healthcare users (HUs), such as healthcare providers, insurance companies, and patients in smart healthcare environments, such as cooperative hospital networks [5], by using immutable data ledgers and smart contracts [4]. Enabled by the computing efficiency of MEC and the security features of blockchain, the combination of these two emerging technologies has been regarded as the key enabler for smart healthcare services, i.e., data offloading and data sharing, in distributed hospital settings [2].

However, realizing the potential of such a comprehensive system in healthcare still faces nontrivial challenges. First, how to offload IoT healthcare data to the MEC servers to support e-health applications while guaranteeing both high QoS (i.e., minimal service latency) and data privacy is a critical issue. Most traditional approaches [6], [7] only either focus on the QoS problem of network latency and energy usage or data privacy for healthcare offloading, while building a holistic framework with all these factors taken into consideration is vitally necessary. Second, some data sharing solutions in [8] rely on a centralized cloud architecture, which is prone to single-point failures and raises trust concerns from third party. The storage of electronic health records (EHRs) in central cloud also incurs high communication overhead, although it requires less data management efforts. Finally, some blockchain-based proposals for health data sharing have been introduced in recent works [9]–[11]. However, these schemes often use a classic interplanetary file system (IPFS), which relies on a global distributed hash table (DHT) for health data storage and sharing, leading to high data retrieval latency. To be clear, to access a record stored on an IPFS node, one needs to refer to a DHT to obtain the hash and then return to the IPFS node for retrieving data, which incurs high data sharing overhead. Moreover, the potential of blockchain for healthcare sharing scenarios with MEC in cooperative hospital networks has not been adequately investigated.

To fill these research gaps, this article proposes a new decentralized health architecture, which integrates a data offloading

Manuscript received October 3, 2020; accepted February 8, 2021. Date of publication February 12, 2021; date of current version July 7, 2021.This work was supported in part by the CSIRO Data61, Australia. This article was presented in part at the IEEE Global Communications Conference, Taiwan, 2020 [1]. *(Corresponding author: Dinh C. Nguyen.)*

Dinh C. Nguyen is with School of Engineering, Deakin University, Geelong, VIC 3220, Australia, and also with the Data61, CSIRO, Docklands, VIC 3008, Australia (e-mail: cdnguyen@deakin.edu.au).

Pubudu N. Pathirana is with School of Engineering, Deakin University, Geelong, VIC 3220, Australia (e-mail: pubudu.pathirana@deakin.edu.au).

Ming Ding is with the Data61, CSIRO, Docklands, VIC 3008, Australia (e-mail: ming.ding@data61.csiro.au).

Aruna Seneviratne is with the School of Electrical Engineering and Telecommunications, University of New South Wales, Kensington, NSW 2033, Australia (e-mail: a.seneviratne@unsw.edu.au).

Digital Object Identifier 10.1109/JIOT.2021.3058953

scheme and a data sharing scheme for distributed hospital networks with MEC and blockchain. Particularly, we design a new decentralized smart contract associated with IPFS running on top of the MEC network, which brings two key benefits. First, the smart contract is able to provide authentication and traceability in the data sharing [9]. Any upload events or user access behaviors in the hospital network will be authenticated and traced by the contract without the need of external authority. Furthermore, any modification or alternation on the data record will lead to a change on hash value, which can be also traced by the contract. Second, the combination of smart contract and IPFS helps accelerate the data retrieval rate. Here, we make an improvement in IPFS design by eliminating the global DHT. Instead, we store directly the hash values of data records in the smart contract at each of the MEC servers, which improves the data lookup and data retrieval rates in the data sharing.

A. Motivations and Contributions

First, the work in [12] makes us realize that the research on health data offloading and data sharing is of great practical significance in facilitating smart healthcare. Second, the existing studies on mobile health data offloading [13], [14] motivate us to focus on the MEC framework for highly efficient health data computation. Moreover, the analysis in [9] and [15] and the preliminary results from our recent work [10] reveal that blockchain can provide promising health data sharing solutions with improved QoS and enhanced security. Third, the in-depth discussion in [4] highlights the urgent need of developing a comprehensive architecture of data offloading and data sharing for improving the healthcare quality. Therefore, the above existing works strengthen our determination to implement a comprehensive healthcare architecture by using blockchain, IPFS, smart contract, and MEC technologies. The key contributions of this article are summarized as follows.

1) We propose a new *decentralized* health architecture for a cooperative hospital network, called *BEdgeHealth*, which integrates blockchain and MEC for data offloading and data sharing with user QoS and security awareness.
2) We propose a privacy-aware health data offloading scheme where MDs can offload data tasks to a nearby MEC server for efficient computation. An optimization algorithm is built on each MD to minimize the offloading cost of energy consumption, processing time, and memory usage with respect to system constraints.
3) We develop a data-sharing scheme enabled by the cooperation of blockchain, MEC, smart contract, and IPFS in the distributed hospital network. Particularly, a decentralized authentication mechanism associated with a distributed IPFS storage is built to implement access control and data management at the network edge without requiring third party, in order to enhance the sharing security and improve data retrieval rate.
4) We conduct real-world experiments to evaluate the effectiveness of the proposed *BEdgeHealth* architecture. The implementation results and discussions demonstrate the superior performance of our proposed approach over the existing works.

B. Organization

The remainder of this article is organized as follows. Section II discusses the literature works related to health data offloading and data sharing. In Section III, we introduce our decentralized health architecture where the network components and blockchain are explained. We then present our data offloading scheme in Section IV where the offloading model is formulated. Next, Section V introduces the health data sharing scheme using blockchain in distributed hospitals. The experimental results are provided in Section VI, while Section VII presents the security analysis and discussions. Finally, Section VIII concludes this article and highlights some possible future works.

II. Related Works

In this section, we survey the literature works related to health data offloading and sharing in healthcare.

A. Health Data Offloading

Many data offloading approaches have been proposed to support healthcare. In [6], mobile healthcare data can be offloaded to cloud for processing, analysis, and storage, but it remains high latency incurred by offloading data to remote clouds. Also, the offloading privacy is not considered, which puts sensitive health data at risks of external attacks. Another work in [12] proposed an IoT architecture for executing healthcare applications on clouds, but the optimization for memory usage of MDs in offloading has been overlooked. In fact, there is close relationship between memory usage and the data task as investigated in the recent work [16]. That is, the higher size of data tasks to be offloaded, the higher device memory required to handle the task. Therefore, it is important to consider device memory usage in the data task offloading. Other works in [7], [13], and [14] concentrated on offloading security issues in healthcare. For example, the research in [13] used a hash function and a key cryptosystem for data security. Also, the offloading privacy issues were also solved in [7] and [14] by using consensus algorithms and learning-based privacy preservation techniques with respect to response time and delay requirements. However, the above studies lack the joint consideration of all QoS constraints (i.e., network latency, energy consumption and memory usage) and privacy awareness, which are the important aspects of practical health data offloading systems [3].

B. Health Data Sharing

Several solutions using blockchain have been introduced for health data sharing. Our preliminary work in [1] proposed a decentralized health data sharing based on blockchain and MEC for federated hospitals. The work [8] presented a privacy-preserved data sharing scheme enabled by the conjunction of a tamperproof consortium blockchain and cloud storage. Furthermore, Guo *et al.* [15] described a hybrid architecture of using both blockchain and edge-cloud nodes where smart contracts are employed to monitor access behaviors and transactions. Despite data privacy enhancements, such solutions [8], [15] mainly rely on central cloud servers for EHRs

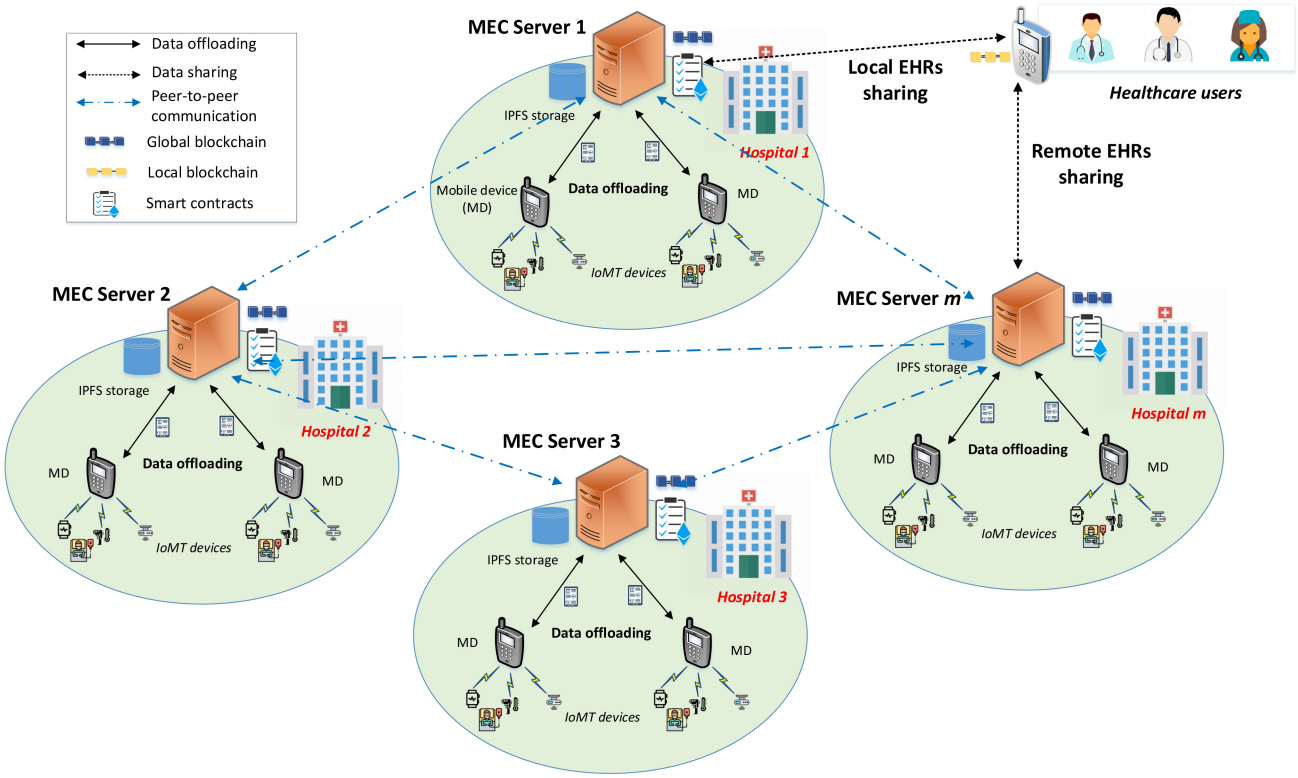

Fig. 1. Proposed healthcare architecture with MEC and blockchain.

storage, which remains single-point failure bottlenecks and incurs high communication overhead. Furthermore, the performances of smart health contract have not been evaluated. Another work in [9] employed an IPFS system with Ethereum blockchain for EHRs sharing over clouds, but the important performance metrics, such as data retrieval speed and security capability, have not been verified. Moreover, the work in [11] described a privacy-preserving EHRs sharing by using blockchain in the Ethereum. Blockchain stores hash values of EHRs while raw data are kept on the cloud server. The authors also constructed a stealth authorization framework to achieve privacy-preserving access authorization delivery over the hospital network. The study in [17] suggested an EdgeCare model that uses edge computing for healthcare management in decentralized health environments. The local authorities were employed to perform access verification for the data offloading and sharing, but healthcare storage has not been considered. Also, a privacy-preserving sharing scheme for IoTs including healthcare was also considered in [18] by using a classic IPFS system with a global DHT look-up solution that results in unnecessary communication overhead.

III. System Model

In this section, we present a new health architecture in hospital networks and then describe our blockchain design.

A. Proposed Healthcare Architecture With MEC and Blockchain

Here, we propose a decentralized health architecture as shown in Fig. 1. The proposed architecture consists of a network of cooperative hospitals linked by a blockchain ledger. Each hospital is controlled by an MEC server, which executes the data tasks offloaded by MDs and also communicates with other MEC servers for data sharing. Here, we consider a set of MEC servers as $\mathcal{M} = \{1, 2, \ldots, M\}$, each MEC server m is located at a hospital HP_m. We assume that each hospital has a set of MDs $\mathcal{N} = \{1, 2, \ldots, N\}$, which collect data from a set of patient $\mathcal{J} = \{1, 2, \ldots, J\}$ using IoMT devices. Then, these data are offloaded to the MEC server for computation via the data offloading scheme. Note that here we consider at each hospital HP_m, an MD n can collect health data from different patients. We also assume that there is a set of HUs $\mathcal{U} = \{1, 2, \ldots, U\}$, such as doctors and clinicians, who may be situated at any hospital and want to access data on the MEC network via the data-sharing scheme (HUs, users, and HUs are used interchangeably throughout this article). The details of each network components are explained as follows.

1) *MEC Servers:* Each MEC server acts as a coordinator that manages a group of MD devices in its hospital to provide low-latency computation services in the data offloading scheme. The MEC server also links with the other MEC servers from other hospitals in a peer-to-peer (P2P) manner to build a decentralized data-sharing network. We consider a realistic scenario that MEC servers may be semitrusted, which means that the MEC server may be curious about health data and misbehaves in the data sharing. To overcome these challenges, we store sensitive EHRs data in the IPFS, instead of in the MEC server's hard drive, while the hash value of data record is kept in the smart contract so that any

data retrieval behaviors are traced on the blockchain in a transparent manner.

2) *HUs:* HUs, such as doctors and clinicians, across federated hospitals may be interested in EHRs data for providing healthcare, e.g., medical diagnosis. Considering the realistic health settings that the HUs are not trusted, we develop an access control mechanism based on smart contracts to perform authentication of user access behaviors, which will be presented in Section V. Only authenticated users are able to retrieve data in IPFS, while malicious users are detected and prevented.
3) *MDs:* MDs, such as smartphones and laptops, are responsible for gathering data from IoMT sensors. MD can use its hardware to execute its data tasks or offload them to a nearby MEC server by data offloading. In this article, we use smartphones as MDs to collect data from wearable sensors and perform data computation.
4) *IPFS Storage:* As MEC servers are semitrusted, in our paper, raw EHRs collected from IoMT and data processed from the offloading phase are uploaded to the decentralized off-chain IPFS platform running on top of the MEC network. IPFS introduces low-latency and fast decentralized archiving with reliable P2P content delivery, which has been investigated in healthcare scenarios [19]; and thus, it is well suitable for our health applications. Especially, in our work, we make an improvement in the IPFS design by replacing the global DHT with smart contracts to manage the hash records of health data, aiming to solve communication latency issues remained in classic IPFS systems [9], [10].
5) *Smart Contract:* A smart contract can be regarded as a self-operating computer program, which is automatically executed when specific conditions are met [4]. Each contract has an account that contains data and codes with programmable logic functions. In our paper, we design a contract called access control smart contract (ACSC). Each MEC server holds a copy of smart contracts and any their new events (i.e., user access) are updated at other MEC servers via the global blockchain network. The details of our contract design will be explained in the following section.

B. Blockchain Design

Blockchain is the heart of our decentralized healthcare architecture. In this article, we suggest using a permissioned Hyperledger Fabric [20] blockchain platform based on practical Byzantine fault tolerance (PBFT) consensus to implement our healthcare system. The Hyperledger Fabric blockchain only allows authenticated users to join the network, and the validation is performed by only preselected nodes with high computing capability, i.e., MEC servers in our scenario, without mining requirements for lightweight entities such as mobile users. This would improve the transaction performance, i.e., low transaction latency, compared to permissionless blockchains such as Ethereum [20]. As shown in Fig. 1, we consider two types of blockchain for our healthcare architecture: 1) a global blockchain and 2) local blockchains.

1) *Global Blockchain:* It interconnects all MEC servers together for hospital communications under the control of all MEC servers. Once a data storage event at each MEC server occurs (triggered by the offloading process), this MEC server broadcasts an offloading transaction to the other MEC servers for global updates. Moreover, in the data-sharing phase, when a mobile user performs a data request to an MEC server, this MEC server also creates a sharing transaction and broadcasts it to other MEC servers in the hospital network.
2) *Local Blockchain:* Each hospital deploys a local hospital to link the local MEC server with its mobile users. This local blockchain is controlled by the local MEC server. When a mobile user performs a data request to the MEC server, he creates a sharing transaction and submits it to the local blockchain so that the MEC server can process the request and return the data. If the MEC server can look up data locally, the server will return immediately to the user. Otherwise, it asks the other MEC servers to find the address of the requested data and then responds the user.

Remark: It is worth noting that in the proposed healthcare architecture as illustrated in Fig. 1, blockchain is only used in the sharing scheme, while we assume that the offloading network is private and thus does not need to apply blockchain in the offloading scheme. In fact, this assumption holds in practical hospital settings [21] where each MD is managed by its healthcare provider such as a physician who uses the device to only collect data from his trusted patients and offload it to the MEC server in each medical test.

In the following sections, we will present our designs for a data offloading scheme and a data sharing scheme in detail.

IV. Health Data Offloading

In this section, we introduce the offloading scheme and then formulate the offloading model in detail.

A. Offloading Model

Here, we focus on formulating the offloading model for a representative hospital. Based on the QoS requirements (i.e., latency, energy consumption), each health data task can be executed at the MD or at the MEC server via task offloading. Accordingly, we introduce an offloading decision policy for each MD n denoted by a binary variable $x_n \in \{0, 1\}$. Here, $x_n = 0$ means that the MD n decides to process its data task locally, and $x_n = 1$ indicates that the MD n offloads its task to the MEC server. For the sake of simplicity, we assume that each MD n has a task Y_n to be executed, which can be defined by a variable $Y_n = (D_n, X_n^l, T_n^{\max})$, where D_n expresses the size of the input data (in bit), X_n^l denotes the required number of CPU cycles of the task, and $T_n^{\max}$ specifies the maximum permissible latency (in second) to accomplish the task Y_n. Motivated by the experimental results of our recent work [22], in this article, we propose an offloading architecture as shown in Fig. 2, which consists of two main modules: 1) task profile and 2) decision maker installed on each MD n.

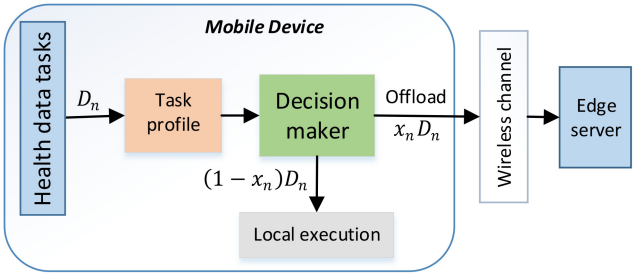

Fig. 2. Data offloading scheme.

1) Task Profile: This module is responsible for collecting all task information, such as the data task Y_n, energy consumption (E_n), processing time (T_n), and memory usage (M_n), by using mobile performance measurement tools. Therefore, a task profile of an MD n can be formulated as a tuple $[Y_n, E_n, T_n, M_n]$, which is then stored in a database of the MD n for supporting the offloading decision process. Details of the modeling of each component in the task profile will be presented in the following section.

2) Decision Maker: This module receives task profile information collected by the profile module to make offloading decisions. Similar to [23], we employ an integer linear programming model to develop a decision making algorithm on MDs. By using profile information, the algorithm analyzes and makes decisions whether the data task should be offloaded to the MEC server or not. The main objective is to determine an optimal computation decision for each task to minimize the offloading cost of energy consumption, execution latency, and memory usage.

B. Offloading Formulation

In the literature, the existing schemes [7], [12]–[14] lack a joint consideration of important QoS metrics in health data offloading, including processing time, energy consumption, and memory usage. Moreover, data privacy issues in the offloading have not been solved [6]. Motivated by the limitations of current schemes, we, here, formulate the health data offloading problem by taking these factors into account. Two computation modes are considered, namely, local execution and offloading.

1) Local Execution: When an MD decides to execute the task Y_n locally ($x_n = 0$), it uses its resource to process healthcare data. We denote X_n^l and f_n^l as mobile CPU utilization for task n (in CPU/b) and mobile computational capacity (in CPU/s), respectively. Then, the local execution time can be calculated as

$$T_n^{\text{local}} = \frac{D_n X_n^l}{f_n^l}. \tag{1}$$

We also define E_n^{local} and M_n^{local} as battery consumption (in Mah) and memory usage (in MB), which can be measured by mobile measurement tools [23].

2) Offloading to the MEC Server: In the case of task offloading ($x_n = 1$), the data task needs to be encrypted for privacy before transmitting to the MEC server. Here, we employ an advanced encryption standard (AES) encryption function installed in each MD to encrypt healthcare data due to its less energy consumption and low latency [24]. Thus, the time required to accomplish a task in the offloading mode includes three parts: 1) the task encryption time at the MD n; 2) the time for transmitting the task to the MEC server; and 3) the task execution time at the MEC server. Then, let us denote X_n^{enc} and X_n^e as mobile CPU utilization for encrypting the task Y_n (in CPU cycles/bit) and edge CPU utilization (in CPU cycles/bit), respectively, and the total offloading time is specified as

$$T_n^{\text{offload}} = \frac{D_n X_n^{\text{enc}}}{f_n^l} + \frac{D_n}{R_n} + \frac{D_n X_n^e}{f_n^e} \tag{2}$$

where f_n^e is defined as the computational resource (in CPU cycles/second) allocated by the MEC server to accomplish the task Y_n, which should not exceed the available edge computational budget f^e. Also, R_n is the data transmission rate of the MD n (in bit/second).

Moreover, the energy consumption for the offloading is mainly characterized by the encryption energy and transmission energy [23]. Then, we define E_n^{enc} as battery consumption when encrypting the task Y_n at the MD n, which is measured by mobile measurement tools. Moreover, according to [22], the transmission energy is computed by

$$E_n^{\text{trans}} = p_n \frac{D_n}{R_n} \tag{3}$$

where p_n is the transmission power of MD n. Hence, the total offloading energy is specified by

$$E_n^{\text{offload}} = E_n^{\text{enc}} + E_n^{\text{trans}}. \tag{4}$$

The offloading process also incurs a memory usage cost for encryption, defined as M_n^{offload}, which can be also obtained through mobile measurement tools [23]. Accordingly, the total offloading time, energy consumption, and memory usage can be expressed, respectively, as

$$T_n = (1 - x_n) T_n^{\text{local}} + x_n T_n^{\text{offload}} \tag{5}$$

$$E_n = (1 - x_n) E_n^{\text{local}} + x_n E_n^{\text{offload}} \tag{6}$$

$$M_n = (1 - x_n) M_n^{\text{local}} + x_n M_n^{\text{offload}}. \tag{7}$$

3) Offloading Problem Formulation: Based on the above formulations, we derive the optimization problem to jointly optimizing time latency, energy consumption, and memory usage under system constraints as follows:

$$\begin{aligned}
(\text{P1}) : \min_{\mathbf{x}} \ & \sum_{n=1}^{N} (\alpha_t T_n + \alpha_e E_n + \alpha_m M_n) \\
\text{subject to} \ & (C1) : x_n \in \{0, 1\}, \forall n \in \mathcal{N} \\
& (C2) : T_n \le T_n^{\max}, n \in \mathcal{N} \\
& (C3) : R_n \ge R_n^{\min}, n \in \mathcal{N} \\
& (C4) : M_n \le M_n^{\max}, n \in \mathcal{N} \\
& (C5) : 0 < p_n \le P_n, n \in \mathcal{N} \\
& (C6) : 0 < f_n^e \le f^e, n \in \mathcal{N}
\end{aligned} \tag{8}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ is the offloading decision vector of all MDs $n \in \mathcal{N}$, and α_t, α_e, and α_m are the weights of time, energy, and memory cost, respectively. In the problem (*P1*), the constraint (C1) implies that each task can be either executed locally or offloaded to the MEC server. Furthermore,

the total task execution time should not exceed a maximum latency value (C2). Constraint (C3) is added to guarantee the reliability of the task offloading at an MD n, where $R_n^{\min}$ is the minimum transmission rate and specified by $R_n^{\min} = D_n/T_n^{\max}$. This condition ensures that the transmission time of MD n during the offloading process is not too long, and thus, the computation of the data task can be completed before the deadline $T_n^{\max}$. (C4) defines that the memory used for task computation must not exceed the available mobile memory. (C5) shows the transmission power constraint of each MD, and constraint (C6) shows that the MEC server must allocate a positive computing resource to each MD under an available computation budget f^e. In this article, we employed a particle swarm optimization (PSO) [25] model written in java to build the above offloading optimization algorithm in Android phones. The PSO algorithm has proven its superior advantages over its counterparts such as genetic algorithm in terms of extremely low computational cost and simple implementation on Android devices for mobile offloading applications such as healthcare [26].

C. Health Data Storage

The raw health data associated with the analyzed results from the data offloading scheme are stored in the IPFS system running on top of the MEC network. We, here, analyze a representative hospital HP_m where there is an MEC server m along with its network of MDs n ($n \in \mathcal{N}$) and patients PID_j ($j \in \mathcal{J}$). The data storage process is implemented through four steps.

1) The raw health data of the patient PID_j collected from MD n are offloaded to the MEC sever m for computation. For simplicity, here we assume that each patient j has a health data record. Then, the MEC server aggregates the raw data and the analyzed result and adds them into a data record identified by the patient ID j as
$$C_j = (\text{rawdata}||\text{analyzedResult}). \tag{9}$$
2) Instead of storing the data and analyzed results in the MEC server's hard drive, we encrypt these data as
$$C_j^{\text{enc}} \leftarrow \text{Enc}(C_j, \text{PKM}_m) \tag{10}$$
where PKM_m is the private key of the MEC m (its creation will be explained in the following section). Then, we put this data to the IPFS storage node located on top of the MEC server for data security.
3) Uploading the data to the IPFS storage would automatically return a cryptographic hash of its content by using a hash function H_{IPFS}
$$h_j = H_{\text{IPFS}}(C_j^{\text{enc}}, \text{timestamp}). \tag{11}$$
Here, we keep the hash value in the smart contract on the MEC server for fast data look-up, instead of relying on the classic DHT.
4) The storage of a data record in the IPFS node at the MEC server m is synchronized with other IPFS nodes at other hospitals via the P2P network for global data sharing. The MEC server m also adds (h_j, PID_j, PKM_m, timestamp) as a transaction including the EHRs' hash h_j and broadcasts it to the global blockchain network
$$\text{MEC}_m \rightarrow *:(h_j, \text{PID}_j, \text{PKM}_m, \text{timestamp}). \tag{12}$$
The other MEC servers will receive the transaction and extract the offloading information h_j, PID_j that is then stored on the database of the ACSC contract. The details of contract design will be presented in the following section.

V. Health Data Sharing

The health data records stored in the IPFS from the data offloading scheme are then used in the data-sharing scheme in federated hospitals. To provide authentication for data sharing, a smart contract-based access control solution is adopted at the network edge without requiring any central authority like previous works [8], [15]. This not only reduces authentication latency but also increases the reliability of data sharing. Moreover, the integration of smart contract into IPFS enables the replica of hash value of data records across the edge network, which helps accelerate the data retrieval rate while the data traceability is achieved. In the following, we explain our proposed data sharing scheme as shown in Fig. 1, including two key parts: 1) user authentication and 2) data retrieval.

A. User Authentication With MEC and Smart Contract

The user authentication process consists of three phases: 1) initialization phase; 2) user registration phase; and 3) data retrieval phase, which are explained as follows.

1) Initialization Phase: Each hospital is initialized by its MEC server. In this phase, a key is set up by the MEC server for data sharing establishment. More specifically, an MEC server m in each hospital implements the following steps to initialize its network.

1) The MEC server m selects an elliptic curve $E_p(a, b)$ over a prime finite field F_p and chooses a base point P over $E_p(a, b)$.
2) Then, it chooses a high entropy random integer x_h as a private key and derives $X_h = x_h.P$ as its elliptic curve cryptography public key. Based on that, we can express the private key and the public key of the MEC m as $\text{SKM}_m = x_h$ and $\text{PKM}_m = X_h$, respectively.
3) Next, it selects a secure one way hash function $\text{Hash}(.) : 0, 1^* - > Z_p^*$, where $Z_p^* = 1, 2, \ldots, p1$, for a given prime p, is a finite cyclic group of order $(p1)$. This group is commutative under multiplication mod p.
4) Based on that, each HU u in the hospital HP_m generates a high entropy random integer SKU_u as its private key. Afterward, a HU computes the public key $\text{PKU}_u = \text{SKU}_u.P$. Besides, each HU also has a unique identity ID_u for identification.

2) Registration Phase: This phase is invoked whenever a HU wants to register to the MEC server for the first time. To do so, the user joins the local blockchain network and follows the steps as follows.

1) Each HU u submits a transaction T_{reg_u} to the MEC server m as T_{reg_u} for registration

$$T_{\text{reg}_u} = (\text{PKU}_u||\text{ID}_u||\text{timestamp}). \quad (13)$$

2) The MEC server m decodes T_{reg_u} to obtain the user's public key as $\text{PKU}_u \leftarrow T_{\text{reg}_u}$.getSenderPublicKey().
3) Then, the MEC server decodes the transaction to get the user ID. First, it decodes the transaction T_{reg_u}, and then finally obtains the user ID as a unique address ADD_u

$$\text{deT}_u \leftarrow \text{abiDecoder.decodeMethod}(T_{\text{reg}_u}) \quad (14)$$

$$\text{ADD}_u \leftarrow \text{web3.eth.getData}(\text{deT}_u([\text{ID}_u])). \quad (15)$$

4) The MEC server checks user information and stores $\{\text{PKU}_u, \text{ADD}_u\}$ as the user identification information in the contract database.
5) The MEC server then calculates the hash value of the register transaction T_{reg_u} as

$$H_{\text{HU}_u} \leftarrow \text{Hash}(T_{\text{reg}_u}, \text{SKM}_m) \quad (16)$$

which is then published to the local hospital blockchain network for tracing. The MEC server m also broadcasts its public key PKM_m to the user that is necessary for the future user data access.

3) User Authentication Phase: It is supposed that a HU u wants to access the patient's EHRs stored on the MEC network for their medical tasks. To obtain the EHRs of the target patient, the HU u needs to know his patient identity PID_j so that the MEC server can locate the address of this patient in the hospital network during EHRs sharing. The data retrival process is presented by the following key steps.

1) A HU u prepares a data retrieval request T_{req_u} involved a target patient ID PID_j and the address of patient's hospital HP_w ($w \in \mathcal{M}$). Thus, the target patient address on the hospital network can be expressed as $P_{\text{addr}} =< \text{PID}_j, HP_w >$, i.e., the fifth patient in the third hospital. Then, the request T_{req_u} can be specified by

$$T_{\text{req}_u} \leftarrow (\text{PKU}_u||\text{ID}_u||P_{\text{addr}}||\text{timestamp}) \quad (17)$$

where each compoment in T_{req_u} is formatted with an index in the array $index = [1-4]$, i.e., the index of PKU_u is 1. This format is necessary for transaction decoding later.
2) To ensure privacy, the user request should be encrypted with the MEC m's public key PKM_m (obtained from the registratrion phase) as $T_{\text{enc}_u} \leftarrow \text{Enc}(T_{\text{req}_u}, \text{PKM}_m)$ and submits it to the MEC m.
3) At the edge side, the MEC server m decrypts the user request T_{enp_u} as $T_{\text{dec}_u} \leftarrow \text{Dec}(T_{\text{req}_u}, \text{SKM}_m)$. To provide security for the EHRs sharing, user authentication is highly essential. To do so, the MEC server extracts the user's public key from the request as

$$\text{Pub}_u \leftarrow T_{\text{dec}_u}.\text{getSenderPublicKey}(). \quad (18)$$

It also decodes the transaction T_{dec_u} to obtain the request information ReqInf_u

$$\text{deT}_{\text{dec}_u} \leftarrow \text{abiDecoder.decodeMethod}(T_{\text{dec}_u}) \quad (19)$$

$$\text{ReqInf}_u \leftarrow \text{web3.eth.getData}(\text{deT}_{\text{dec}_u}[\text{DataIndex}]) \quad (20)$$

and then obtains the user identity Iden_u as

$$\text{Iden}_u = \text{ReqInf}_u(\text{Index}[\text{index}_{\text{Iden}}]). \quad (21)$$

4) The MEC server will check and authenticate the received user identification information $<Pub_u, \text{Iden}_u>$, and then put them into user mapping as

$$\text{UMAP}_{PK_u} = Map < Pub_u => \text{PKU}_u > \quad (22)$$

$$\text{UMAP}_{\text{ID}_u} = Map < \text{Iden}_u => \text{ID}_u > \quad (23)$$

by using the smart contract (see in Algorithm 1). If both $\text{UMAP}_{PK_u} \rightarrow true$ and $\text{UMAP}_{\text{ID}_u} \rightarrow true$, the user request is validated successfully, otherwise, a penalty is issued for access prevention.
5) In the case of successful request validation, the MEC server m will calculate the signature of T_{dec_u} as

$$Sig_u \leftarrow \text{Hash}(T_{\text{dec}_u}, \text{SKM}_m). \quad (24)$$

Finally, the MEC server will issue a certificate Cert_u as

$$\text{Cert}_u = \{Sig_u, \text{PKU}_u, \text{ID}_u, \text{timestamp}\} \quad (25)$$

which is then sent to HU u via the local blockchain for successful authentication proof.

B. Health Data Retrieval With MEC and Blockchain

After successful authentication, the MEC server m will locate the requested EHRs based on the patient information $< \text{PID}_j, HP_w > \leftarrow \text{ReqInf}_u(\text{Index}[\text{indexPID}_u])$ that is defined in the authentication phase. In fact, the patient and the user may be located in the same hospital or in different hospitals. For example, a patient may only stay at a hospital for his treatment, but in some cases (e.g., seeking treatment for a new disease), he wants to visit a different doctor in another hospital. Motivated by this realistic scenario, here we consider two cases: 1) the patient and the user are in the same hospital and 2) the patient and the user are in different hospitals.

Case 1 (Patient and the User Are in the Same Hospital): In this case, the MEC server finds that the HU and the patient are in the same hospital by checking HP_w information. It is supposed that the HU u communicates with the MEC server m to request the data record of the patient PID_j in the same hospital, then the data retrieval process is implemented by the following steps.

1) The MEC server m first verifies the request information PID_j by refering to the ACSC contract to perform mapping between the patient record stored in the contract PID_j^{sc} and information in the request $\text{PID}_j^{\text{req}}$

$$\text{UMAP}_{\text{PID}_j} = \text{Map} < \text{PID}_j^{sc} = > \text{PID}_j^{\text{req}} >. \quad (26)$$

If $\text{UMAP}_{\text{PID}_j} \rightarrow true$, the request information is verified for ready data retrieval.
2) Based on the received patient information, the ACSC contract will extract the hash value h_j that represents the patient j's health record. Then, the contract sends a request to the IPFS storage for data retrieval using the hash by a command: $C_j^{\text{enc}} = \text{GET}_{\text{IPFS}}(h_j)$, i.e.,: `ipfs get /ipfs/Qmd84db7be0690ebb015f1cD9d9491cE18076c`.

3) Since the data stored on the IPFS were encrypted in the storage process (see in Section IV-C), we need to decrypt to obtain the real data as

$$C_j \leftarrow \text{Dec}(C_j^{\text{enc}}, \text{SKM}_m). \quad (27)$$

The MEC m then returns the data C_j via a secure channel to the requestor.

4) Finally, the MEC server m adds a conjunction of $(\text{PKU}_u, h_j, \text{PKM}_m, \text{timestamp})$ as a transaction and broadcasts it to the global blockchain network

$$\text{MEC}_m \rightarrow *: (\text{PKU}_u, h_j, \text{PKM}_m, \text{timestamp}). \quad (28)$$

Case 2 (Patient and the User Are in Different Hospitals): In this case, the MEC server will seek the address of the patient in the MEC network. As all the patient addresses (PID_j, HP_w) are stored on the ACSC contract replicated across the global hospital network, an MEC server at a hospital can know exactly where the requested patient data are currently located. Hence, an MEC server only needs to send the data request to the destination MEC server (using HP_w information) that contains the requested data for data retrieval, without broadcasting the request to all MEC servers. This strategy not only saves data lookup time but also saves network bandwidth and potentially reduces the traffic congestion on the global blockchain network. The data retrieval process in this case is summarized as the following steps.

1) The MEC server m first also verifies the request information PID_j from the user PKU_u by refering to the ACSC contract, and then performs mapping to verify that the patient information PID_j is correct. Then, it also identifies which MEC server is currently storing the requested data by checking HP_w information. Here, we assume that an MEC server MEC_y, $(y\#m, y \in \mathcal{M})$ is holding the requested data.
2) After identifying the destination MEC server y, the MEC server m will send a transaction for data retrieval request

$$\text{MEC}_m \rightarrow \text{MEC}_y : (\text{PID}_j, \text{PKU}_u, \text{PKM}_m, \text{time}). \quad (29)$$

3) Based on the patient information PID_j, the ACSC contract in the MEC y obtains the hash value h_j. Next, the contract sends a request to the IPFS node in MEC y by a command: $C_j^{\text{enc}} = \text{GET}_{\text{IPFS}}(h_j)$, which is then decrypted to obtain the real data

$$C_j \leftarrow \text{Dec}(C_j^{\text{enc}}, \text{SKM}_y). \quad (30)$$

4) The MEC server y then transmits the collected data C_j to the MEC server m so that the server m returns it to the requestor. Finally, the MEC server y adds a conjunction of $(\text{PKU}_u, h_j, \text{PKM}_y, \text{timestamp})$ as a transaction and broadcasts it to the global blockchain network

$$\text{MEC}_y \rightarrow *: (\text{PKU}_u, h_j, \text{PKM}_y, \text{timestamp}). \quad (31)$$

Finally, all MEC servers update the user access events and achieve a synchronization over the data sharing across the hospital network.

The data retrieval process for two cases are shown in Fig. 3, and the data sharing is summarized in Algorithm 1. Here, lines (5–8) present the preprocessing steps when a HU u submits a request to the MEC server. Then, the MEC server will authenticate the request using ACSC, by verifying the identification information $(\text{PKU}_u, \text{ID}_u)$ (lines 9–18). For the request authenticated by the contract, the MEC server specifies whether the patient and the user are in the same hospital or in the different hospitals. The MEC server will refer to the IPFS storage for data retrieval based on patient information PID_j before returning to HU u (in lines 19–38).

Algorithm 1 Health Data Sharing With MEC and Blockchain

1: **Input:** $\text{PID}_j, HP_w, \text{PKU}_u, \text{ID}_u, T_{\text{req}_u}$
2: **Output:** $Auth_u, C_j$
3: **Initialization:** *(by the user HU u)*
4: Encrypt the request: T_{enc_u} and submits it to the MEC m
5: **Pre-processing the request** *(by MEC)*
6: The MEC m decrypts the user request T_{enp_u} as $T_{\text{dec}_u} \leftarrow \text{Dec}(T_{\text{req}_u}, \text{SKM}_m)$
7: Obtain the user's public key: $Pub_u \leftarrow T_{\text{dec}_u}.getSenderPublicKey()$
8: Decode T_{dec_u} and get the user identity Iden_u
9: **Authentication** *(by the ACSC contract)*
10: **if** $msg.sender == \text{MEC}_m$ **then**
11: $PKcheck = policy[EHRresource][action].\text{PKU}_u$
12: $Idencheck = policy[EHRresource][action].\text{ID}_u$
13: **if** $PKcheckIdencheck \rightarrow true$ **then**
14: $Auth_u \leftarrow AccessResult(msg.sender, Accepted, true, time)$
Accept the user request
15: **else**
16: $Auth_u \leftarrow AccessResult(msg.sender, Denied, false, time)$
Refuse the user request
17: **end if**
18: **end if**
19: **while** $Auth_u \rightarrow true$ **do**
20: **if** $\text{HP}_w == HP_w^{sc}$ **then**
21: (***Case 1**: The patient and the user are in the same hospital*)
22: **if** $\text{PID}_j^{sc} == \text{PID}_j$ **then**
23: Get the data on IPFS: $C_j^{\text{enc}} = \text{GET}_{\text{IPFS}}(h_j)$
24: Decrypt to obtain the real data $C_j \leftarrow \text{Dec}(C_j^{\text{enc}}, \text{SKM}_m)$
25: **end if**
26: The MEC m returns the data C_j to the HU_u
27: The MEC m adds a transaction to the global blockchain network: $\text{MEC}_m \rightarrow *:(\text{PKU}_u, h_j, \text{PKM}_m, \text{timestamp})$
28: **else if** $\text{HP}_w != HP_w^{sc}$ **then**
29: (***Case 2**: The patient and the user are in different hospitals*)
30: **if** $\text{PID}_j^{sc} == \text{PID}_j$ **then**
31: Communicate with the MEC y: $\text{MEC}_m \rightarrow \text{MEC}_y:(\text{PID}_j, \text{PKU}_u, \text{PKM}_m, \text{timestamp})$
32: Get the data on IPFS: $C_j^{\text{enc}} = \text{GET}_{\text{IPFS}}(h_j)$
33: Decrypt to obtain the real data $C_j \leftarrow \text{Dec}(C_j^{\text{enc}}, \text{SKM}_y)$
34: **end if**
35: The MEC y returns data C_j to the MEC m and then the HU u
36: The MEC y adds a transaction to the global blockchain network: $\text{MEC}_y \rightarrow *: (\text{PKU}_u, h_j, \text{PKM}_y, \text{timestamp})$
37: **end if**
38: **end while**

VI. Experimental Results and Evaluations

In this section, we present experiments and perform implementation evaluations in detail.

A. Experiment Settings

We implemented a testbed to evaluate the proposed *BEdgeHealth* architecture. We employed three computers (Microsoft Windows 10 64-b, Intel core i7 at 3.4 GHz, 8 GB of RAM) to work as MEC servers where each MEC server represents a hospital. Each server will connect with a network

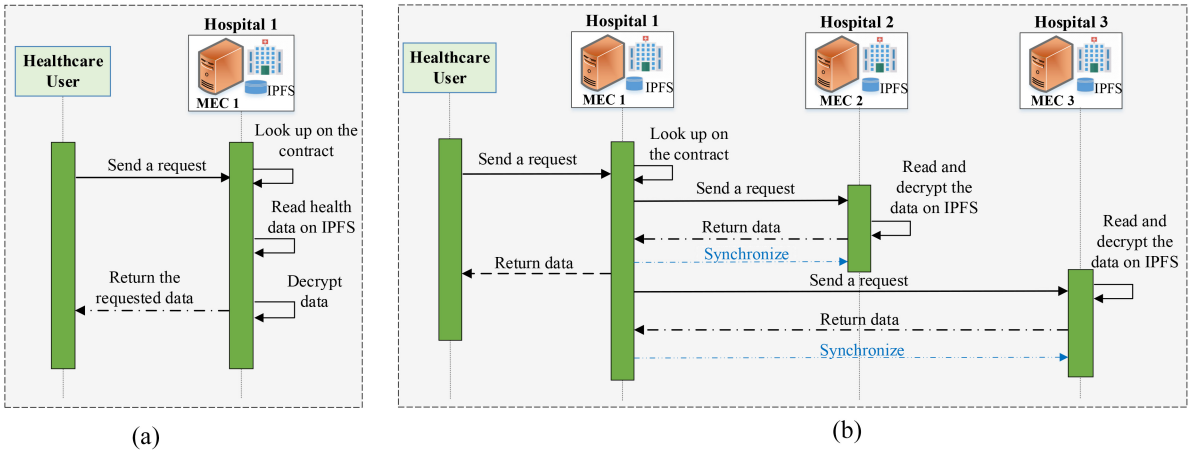

Fig. 3. Proposed health data sharing procedure with MEC and IPFS. (a) Case 1: Patient and the user are in the same hospital. (b) Case 2: Patient and the user are in the different hospitals.

of MDs via a Cisco access point. Here, we used Sony Android mobile phones as MDs for data offloading and data sharing tasks with Qualcomm Snapdragon 845 processor, 2-GB memory, and a battery capacity of 2870 mAh. In our study, healthcare data and programming code are necessary. For a specific case study, we used Biokin sensors [10] developed by our lab as IoMT devices to collect simultaneously human motion data (including acceleration and gyroscope time-series data) with the size from 100 KB to 5 MB. These data are stored in separate files to be executed by an analysis program integrated in both Android phones and the MEC server. By using a programmed data analysis algorithm, we can estimate the movement disorder of a patient to serve doctors during clinical decisions [10]. The experiment parameters are set up as follows: task CPU workload $X_n^l = [0.8 - 1.5]$ Gcyles, maximum latency threshold $T_n^{\max} = 10$ s, device computing capability $f_n^l = 1$ GHz, and edge computing capability $f^e = 5$ GHz. The maximum transmission power P_n of MDs is set to 20 mW, and the minimum transmission rate requirement $R_n^{\min} = 0.5$ Mb/s. For mobile performance evaluations, we employed a Firebase Performance Monitoring[1] service to measure processing time, battery consumption, and memory usage. The mobile application for the offloading optimization was implemented using Android studio 3.5.

For blockchain deployment, two Hyperledger Fabric platforms version 1.3 were used to build the global blockchain on the MEC system and the local blockchain on the MEC-device system. The PBFT consensus was implemented by MEC servers, while devices only joined the blockchain network for data request. We followed the instructions in the official Hyperledger Fabric tutorial to install required files and docker images [27]. The smart contract was implemented in docker to serve user authentication and data retrieval [28]. We also installed the JavaScript version of the IPFS platform in the MEC system [29]. Each of three MEC servers holds an IPFS node, which is embedded with the Fabric blockchain to perform data storage and data sharing. To highlight the merits of the proposed *BEdgeHealth* architecture, we compare our scheme with the related works using different performance metrics, which are presented as follows.

[1]https://firebase.google.com/docs/perf-mon

B. Evaluation of Data Offloading Performance

We compare our edge offloading scheme with two baselines: 1) local execution [10] (only executing data on devices) and 2) cloud computation [23] (offloading to the cloud server) to prove the advantages of our scheme. A set of health data files with different sizes (200–1000 KB) collected from sensors is used in our evaluations. We implement each test ten times to obtain average values, and evaluate the offloading performance of these schemes on processing time, energy consumption, and memory usage. We present two settings with different weight values to evaluate the tradeoff among these three performance metrics, as shown in Figs. 4 and 5.

Fig. 4 shows the offloading performances with $\alpha_e = 1/3, \alpha_t = 1/3$, and $\alpha_m = 1/3$. For the processing time, it consists of execution time in the local case, and encryption time, offloading time, and remote execution time in the edge and cloud offloading cases. From Fig. 4(a), we can see that our edge offloading scheme consumes less energy when processing the health data tasks. As an example, offloading a 200-KB file consumes in the edge scheme less 11% energy than that in the case of local computation and less 5% energy than that in the cloud scheme. Especially, the energy usage of the edge scheme becomes more efficient when the data size increases. For instance, the edge offloading scheme can save 21.3% energy when executing a 1000-KB file, compared to the local scheme.

Fig. 4(b) indicates the average processing time of three schemes, with a significant performance improvement in the proposed edge scheme. For example, executing a 200-KB file by the edge scheme only consumes 1.1 s, whereas it requires about 1.3 and 1.5 s in the cloud and local schemes, respectively. This leads to a 10%–18% time saving of data execution by using our edge scheme. Furthermore, the proposed edge scheme saves up to 31% and 15% time when computing a 1000-KB file, compared to the local and cloud schemes, respectively. We also found with the selected human motion

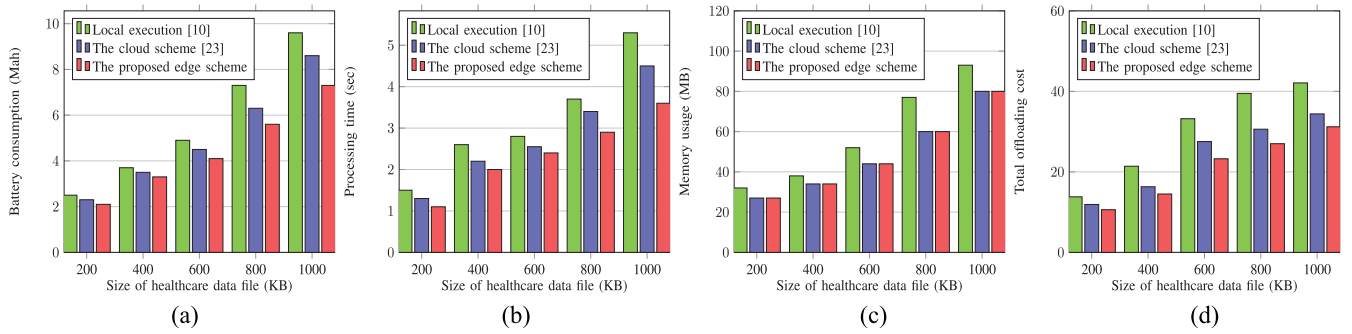

Fig. 4. Comparison results for health data offloading with $\alpha_e = 1/3, \alpha_t = 1/3$, and $\alpha_m = 1/3$. (a) Battery consumption. (b) Processing time. (c) Memory usage. (d) Total offloading cost.

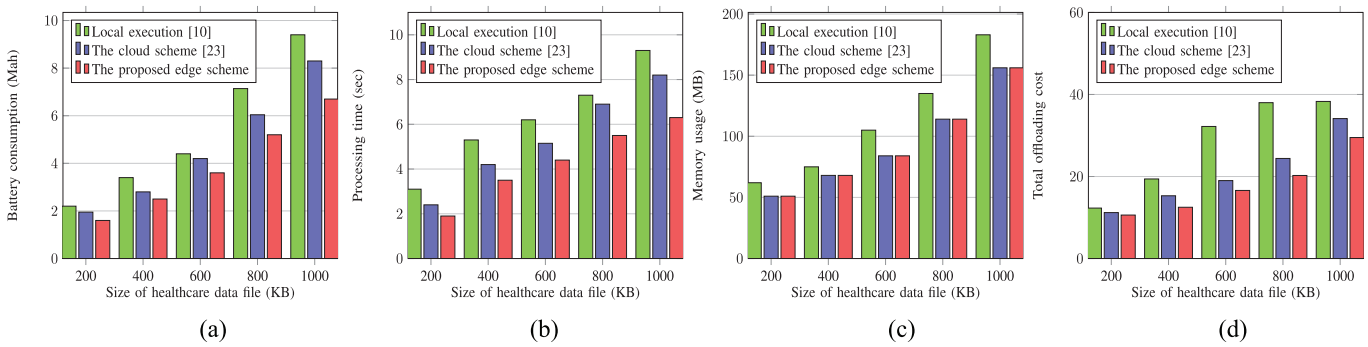

Fig. 5. Comparison results for health data offloading with $\alpha_e = 2/3, \alpha_t = 1/6$, and $\alpha_m = 1/6$. (a) Battery consumption. (b) Processing time. (c) Memory usage. (d) Total offloading cost.

data set, although data encryption is integrated in offloading, the edge-cloud offloading schemes still achieve better offloading performances than the local scheme, showing the efficiency of the proposed encryption technique.

Moreover, the memory performance is illustrated in Fig. 4(c). The edge and cloud schemes have the same memory usage due to using the same encryption function for the offloading. However, these schemes achieve greater memory performances, with 5% and 9% memory savings compared to the local scheme when executing a 200- and 1000-KB file, respectively. Note that the above implementation results were obtained from the proposed offloading application with human motion data and current hardware settings of devices and MEC servers. Different mobile applications with other health data types, such as videos and different hardware settings, can achieve different offloading performances. However, in general, the proposed edge offloading scheme yields the best performances with reduced time latency, energy consumption, and better memory savings among three schemes for any task size. As a result, our edge scheme shows an significant improvement in the offloading performance with lower offloading cost, compared to the cloud scheme and the local scheme, as indicated in Fig. 4(d).

Next, we investigate the offloading performances of three schemes with $\alpha_e = 2/3, \alpha_t = 1/6$, and $\alpha_m = 1/6$, as illustrated in Fig. 5. Based on the objective function in the optimization problem (*P1*), a larger α_e value will give more penalty on the energy consumption. Particularly, the performance gap between the edge scheme and other schemes in this setting is larger than that in the previous setting in

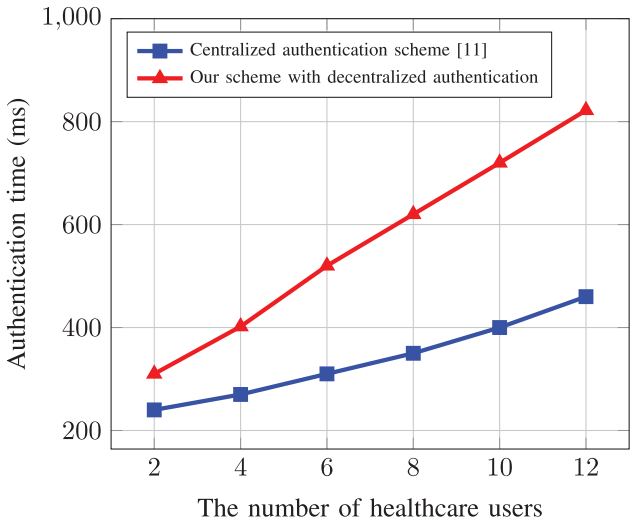

Fig. 6. Authentication latency with different numbers of HUs.

Fig. 4 due to the increased latency and memory cost and reduced energy consumption. Moreover, our scheme achieves the best performance with lowest offloading cost among three schemes. For instance, when the task size is 1000 KB, the offloading cost of our edge scheme is 29.5, compared to 34.2 in the cloud scheme and 38.3 in the local scheme. The experimental results from two tradeoff settings in Figs. 4 and 5 demonstrate the advantage of our proposed edge scheme over the baselines, showing its efficiency in health offloading applications.

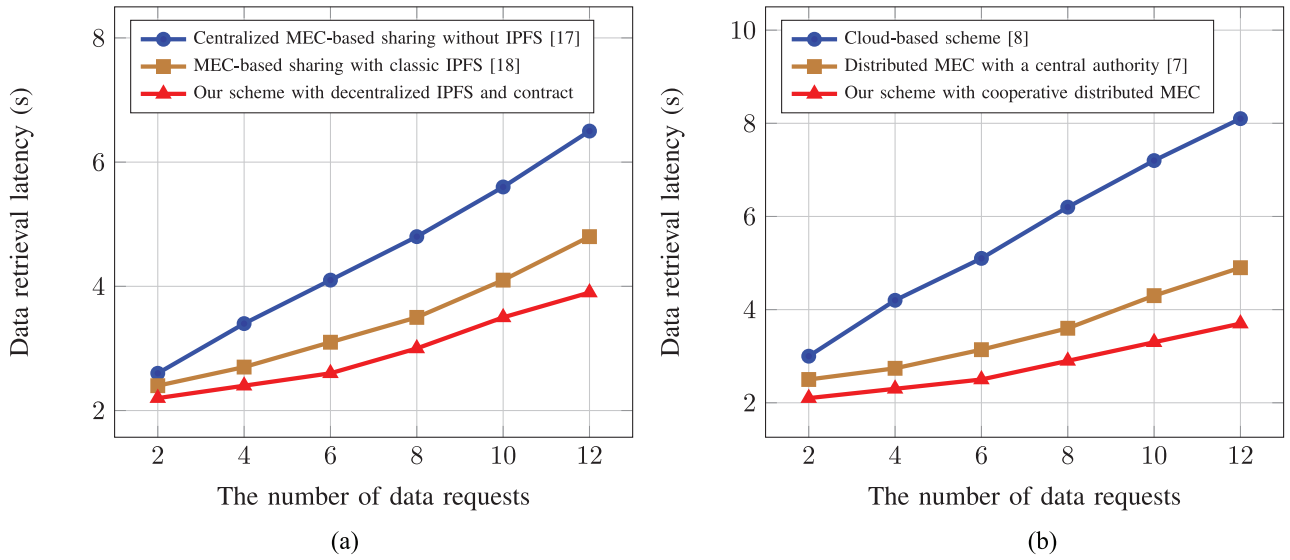

Fig. 7. Data retrieval rate under different numbers of data requests. (a) Data retrieval latency under blockchain designs. (b) Data retrieval latency under network designs.

TABLE I
AUTHENTICATION COST TEST

Authentication functions	Computation Cost
Hash function	5.6 ms
Request encryption	12.3 ms
Transaction decoding	3.2 ms
Request decryption	4.1 ms
User verification	22.5 ms
PBFT consensus commitment	30 ms

C. Evaluation of Data Sharing Performance

For data sharing, we investigated the authentication cost, data retrieval latency, request acceptance probability, and blockchain performance.

1) Authentication Cost: We calculate the computation cost for the authentication process. Here, a user leverages his smartphone to submit a request with 160 b to its MEC server for data retrieval in the blockchain network. The MEC server performs some functions, such as encryption, decryption, and transaction decoding, for user verification. As shown in Table I, the computation costs are small, and the latency of PBFT consensus is acceptable and thus suitable for time-sensitive healthcare applications.

Besides, we also compare the authentication latency of our proposed scheme with a centralized scheme [11] with the different number of HUs. In the proposed scheme, we organize the access authentication at the network edge where each MEC server authenticates its users by the distributed smart contract. Meanwhile, the scheme [11] relies on a central authority to implement its user authentication. As shown in Fig. 6, our scheme exhibits a lower latency compared to the baseline [11]. This is because that the use of decentralized smart contract enables fast authentication at the network edge without passing a remote authority, which thus reduces communication overhead in the authentication process.

2) Data Retrieval Latency: We investigated the data retrieval latency of our proposed model from blockchain design and network design perspectives as shown in Fig. 7. We used smartphones to send data requests continuously to the MEC servers to record the results. In terms of blockchain design, we use two existing works for comparison. The first one is a centralized edge-based health sharing scheme without IPFS [17], which used a centralized MEC server to serve a large hospital network and health data were stored in a classic database. The second one is an edge blockchain health sharing scheme with classic IPFS [18] that integrated blockchain and edge computing without IPFS design improvement.

From Fig. 7(a), we can see that when the number of requests increases, the baseline [17] has the highest data retrieval latency due to the queuing latency in the centralized MEC server. The baseline [18] used a classic IPFS storage with a global DHT look-up solution that results in unnecessary communication overhead. In contrast, our scheme provides a fully decentralized solution with distributed MEC and smart contracts, which allows to implement request verification and data look-up at the network edge without using the global DHT. As a result, our scheme can achieve a minimal data retrieval latency.

Next, we evaluated the data retrieval latency from a network design perspective as shown in Fig. 7(b). We leveraged a cloud-based scheme [8] and a distributed MEC with a central authority [7] as the baselines for comparison. For cloud computing implementation, we employed Amazon cloud services to communicate with smartphones [10]. The results in Fig. 7(b) clearly show a significant improvement in our decentralized scheme with a much lower retrieval latency. This is because our scheme combines MEC, distributed smart contracts, and decentralized IPFS for fast data retrieval, without passing any external authority during the data sharing. Meanwhile, the work in [8] relies on a remote cloud model, which remains high latency due to excessive communication overhead. Also, the work in [7] used a central authority for request verification that consumes a certain overhead for communication between the MEC servers and the authority in the request verification. The above experiment results demonstrate a lower retrieval

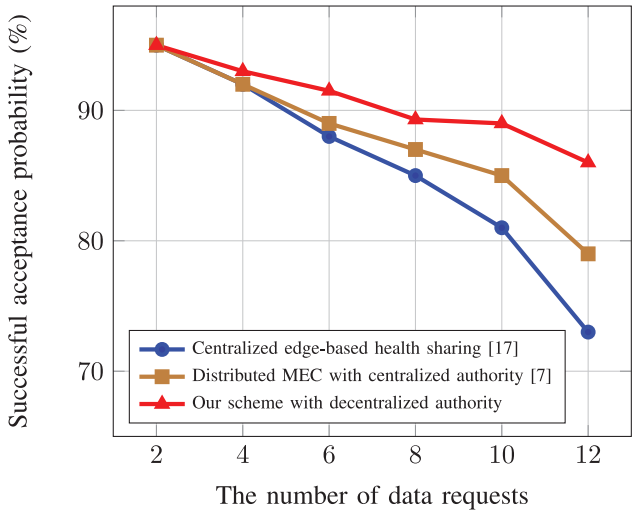

Fig. 8. Request acceptance probability under different numbers of data requests.

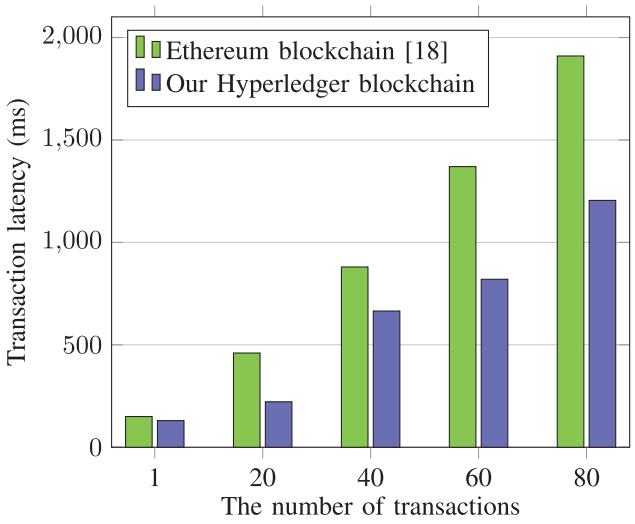

Fig. 9. Comparison on transaction latency of Hyperledger and Ethereum blockchains.

latency cost of our approach in comparison with the existing works.

3) Request Acceptance Probability: We added a time threshold δ_u in the request transaction T_{req_u} to set up a new transaction involving the latency condition: $T_{\text{req}_u} \leftarrow (\text{PKU}_u||\text{ID}_u||P_{\text{addr}}||\delta_u||\text{timestamp})$. This threshold represents the maximum latency that a request needs to be responded by returning the data to the requestor before the δ_u deadline (successful request); otherwise, the request is regarded as a failed one. Here, we introduce an acceptance probability function, which is defined as the number of successful requests per the total requests. As shown in Fig. 8, the request acceptance probability of our scheme is higher than that of the other baselines [7] and [17]. Although the probability reduces when the number of requests increases due to the longer queue time, our decentralized scheme still yields the best performance. This can be explained by the significant processing time saving achieved in our scheme thanks to an optimized edge computing and decentralized smart contract design.

4) Blockchain Performance: Next, we evaluate the performance of our permissioned Hyperledger Fabric blockchain and compare it with the popular permissionless Ethereum blockchain used in [18]. We run the ACSC contract on three computers and use smartphones to send transaction continuously to the computers to measure the average transaction latency in the local blockchain. As indicated in Fig. 9, our proposed Hyperledger Fabric blockchain exhibits a much lower transaction latency, compared to the Ethereum blockchain under the varying number of transactions. This experiment result shows the suitability of using Hyperledger Fabric blockchain for time-sensitive healthcare applications like our scenario.

VII. Security Analysis and Discussions

A. Security Analysis

In this section, we present the attack model and then analyze theoretically some key security features achieved by our blockchain design.

1) Attack Model: We consider two types of attacks: 1) internal attack and 2) external attack.

a) Internal attack: According to our design description in the previous sections, the MEC servers and HUs in this article are considered semitrusted in the data sharing, while the MDs are regarded as fully trusted entities in the private offloading. Under the semitrusted assumption, the MEC servers would be honest but curious about the health data and thus can infer some sensitive information from transactions on the blockchain.

b) External attack: During data offloading and sharing, external attackers can gain access to obtain health information. For example, an adversary can attack the MEC server to obtain patient information offloaded from MDs. Also, the data sharing may be vulnerable due to the data threats on the communication between the MEC server and HUs.

Next, we present the main security properties of our proposed *BEdgeHealth* scheme, and explain how these properties make our scheme resilient to security threats.

2) Security Analysis: Our scheme is able to provide four important security features, including data privacy, authentication, traceability, and confidentiality.

a) Data privacy: Our scheme can preserve data privacy in both data offloading and data sharing schemes. In data offloading, the health data are actually encrypted by an AES encryption function in MDs when performing the offloading. Moreover, the health data offloaded from MDs are encrypted by using the private key of the MEC server ($C_j^{\text{enc}} \leftarrow \text{Enc}(C_j, \text{PKM}_m)$). An external adversary may not decrypt the data due to the lack of MEC server's private key. Thus, the health data privacy is preserved. In the data sharing, the data request is also encrypted ($T_{\text{enc}_u} \leftarrow \text{Enc}(T_{\text{req}_u}, \text{PKM}_m)$) so that private user information is protected against threats. Moreover, the data transactions stored in the blockchain cannot be changed or modified by using immutable ledgers. Unlike recent works [9], [10], [15] with a centralized IPFS on a third-party cloud that may remain data leakage, our architecture provides strong data control in the MEC network under the

management of blockchain without a third party. This would eliminate single-point failures and avoid illegal data usage for better data privacy control.

b) Authentication: In our *BEdgeHealth* scheme, the data sharing is authenticated in a decentralized manner with the help of the distributed ACSC contract. In our design, the smart contract works independently with the MEC server, which means the authentication policies cannot be controlled by the misbehaving MEC server but managed by the global blockchain network. Any contract updates are reflected on the blockchain network and monitored by all MEC servers. This would avoid the risks of contract modifications caused by internal attacks, and hence, ensure reliable contract operations. In the data sharing, the data retrieval is executed only if the user information $\text{PKU}_u, \text{ID}_u$ is authenticated by the contract.

c) Traceability: This is ensured by the fact that our *BEdgeHealth* framework is running on a blockchain network where any data access events and user behaviors are traced by all entities that cannot achieved in the existing works [7], [8], [11]. A user request to our *BEdgeHealth* system is registered by an MEC server and broadcast to all other entities in both global blockchain and local blockchains. As a result, all MEC servers and users have acknowledged a data access event when it occurs. Furthermore, one can easily trace the origin where data are modified or updated through transaction logs. Moreover, we store health data on IPFS associated with smart contracts, instead of in the MEC server's hard drive to achieve traceability over data usage. To be clear, the raw data are stored in IPFS while the hash value of data is kept in the smart contract. Hence, any modification or change behaviors on a data record will lead to a change in its hash value, and thus, the smart contract can detect for prevention. Moreover, since all activities at an MEC server are recorded and synchronized on the global blockchain, other MEC servers can also trace them in a transparent manner.

d) Confidentiality: In *BEdgeHealth*, the confidentiality of health communications is guaranteed by exploiting the standard cryptographic primitives. We employ key-based encryption solutions coupled with digital signatures for our offloading and sharing schemes. An external attack cannot gain access to the health communication due to the lack of private keys of entities (i.e., MEC servers or health users). Even if an external adversary tries to modify or change the communication protocol, the message digest on blockchain would detect such malicious actions. In fact, to modify the metadata on blockchain, an attack needs an extensive computation capability to gain the power control from all MEC servers, which may be nearly impossible to achieve in practice [2]. Hence, the user confidentiality can be preserved.

Based on the above implementation results and discussions, we summarize the key features of our proposed *BEdgeHealth* architecture and compare with the related works in Table II. The comparison results demonstrate the advantages of our scheme over the conventional schemes, showing the usability of our *BEdgeHealth* architecture in practical e-health applications.

TABLE II
COMPARISON OF SECURITY FEATURES BETWEEN OUR PROPOSED SCHEME AND THE EXISTING WORKS

Features	**Schemes**					
	[7]	[8]	[11]	[17]	[18]	Our scheme
Decentralization		✓	✓	✓	✓	✓
Data privacy	✓	✓	✓	✓	✓	✓
Decentralized storage					✓	✓
Authentication	✓	✓	✓	✓		✓
Traceability				✓	✓	✓
Confidentiality	✓	✓	✓	✓	✓	✓
Offloading and sharing services						✓

B. Discussion

In this article, we present a decentralized *BEdgeHealth* architecture in hospital networks with a data-sharing scheme and a data offloading scheme, and their performances are also verified via real-world experiments. By using MEC, our scheme is able to provide efficient health data offloading services at the network edge, proximity to MDs and IoMT networks, for user QoE improvement. Moreover, the health data can be stored and shared over the distributed hospital networks using blockchain without the need of central authority. We incorporate smart contracts with IPFS for enabling reliable access authentication and fast data retrieval in order to facilitate the data exchange among HUs. However, the integration of smart contracts in IPFS possibly introduces new challenges, including security vulnerabilities, which can include timestamp dependence, mishandled exceptions, reentrancy attacks on smart contracts [30], [31], and fake authorization caused by misbehaving MEC servers. These security issues may result in privacy leakage or system logic modifications in the data storage process on IPFS; and thus, more research efforts should be made for the full realization of smart contract-based IPFS in health data-sharing applications. For example, developing an incentive mechanism may be very useful to solve fake authentication issues [32]. More specifically, an incentive scheme can be designed and deployed across the MEC network, aiming to incentivize the MEC server with efficient authentication and punish the MEC server with misbehaving authentication. In this way, we can mitigate the possibility of fake authentication among MEC servers.

At present, the health data offloading scheme in our *BEdgeHealth* architecture has been launched commercially at Royal Victorian Eye and Ear Hospital, Melbourne city, Australia for supporting the clinical assessments of cerebellar disease [33]. This project aims to use Biokin sensors [10] associated with a Biokin mobile app to collect real-world health data from patients diagnosed with cerebellar disease and then offload to the MEC server for medical analysis. Besides, our health data sharing with blockchain has been implemented in the testbed environment at networked sensing and control (NSC) lab, Deakin University on the Hyperledger blockchain platform. We set up MEC servers located in different hospitals in Melbourne city and connect with an MEC server at our university, aiming to implement a health sharing among doctors working in cerebellar disease. The implementation results

achieved in this work demonstrate the practicality and feasibility of our proposed *BEdgeHealth* model in real-world health applications.

VIII. Conclusion

This article has proposed a new decentralized health architecture, called *BEdgeHealth*, which employs MEC and blockchain for health data offloading and sharing in distributed hospital networks. We have first proposed a privacy-aware data offloading scheme where MDs can offload IoMT health data to the nearby MEC server under system constraints. Then, a new data-sharing scheme is introduced by using blockchain and smart contracts to enable secure data exchange among HUs in different hospitals. To realize access management, we have developed an ACSC contract that enables decentralized user authentication at the network edge without requiring central authority, which would ensure authentication reliability and reduce network latency. We have implemented various real-world experiments to verify the effectiveness of the proposed *BEdgeHealth* architecture. The implementation results have demonstrated the significant advantages of the proposed offloading scheme over the other baseline methods in terms of reduced time latency, energy consumption, and better memory usage. Moreover, the data-sharing scheme can achieve fast data retrieval with improved blockchain performance, compared to the existing works. The evaluations also prove the high system security of our design, showing the feasibility of the proposed model for healthcare applications.

Future work is in progress to extend our blockchain-MEC model to many other healthcare systems, including speech and video data management and secure real-time health monitoring systems.

References

[1] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain and edge computing for decentralized EMRs sharing in federated healthcare," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taiwan, 2020, pp. 1–6.

[2] M. A. Rahman *et al.*, "Blockchain-based mobile edge computing framework for secure therapy applications," *IEEE Access*, vol. 6, pp. 72469–72478, 2018.

[3] P. Verma and S. K. Sood, "Fog assisted-IoT enabled patient health monitoring in smart homes," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1789–1796, Jun. 2018.

[4] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 135, pp. 62–75, Jun. 2019.

[5] A. Yazdinejad, G. Srivastava, R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and M. Aledhari, "Decentralized authentication of distributed patients in hospital networks using blockchain," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 8, pp. 2146–2156, Aug. 2020.

[6] M. Asif-Ur-Rahman *et al.*, "Toward a heterogeneous MIST, fog, and cloud-based framework for the Internet of healthcare things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4049–4062, Jun. 2019.

[7] R. Saha, G. Kumar, M. K. Rai, R. Thomas, and S.-J. Lim, "Privacy ensured e-healthcare for fog-enhanced IoT based applications," *IEEE Access*, vol. 7, pp. 44536–44543, 2019.

[8] J. Liu, X. Li, L. Ye, H. Zhang, X. Du, and M. Guizani, "BPDS: A blockchain based privacy-preserving data sharing for electronic medical records," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[9] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[10] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for secure EHRs sharing of mobile cloud based e-health systems," *IEEE Access*, vol. 7, pp. 66792–66806, 2019.

[11] S. Jiang, H. Wu, and L. Wang, "Patients-controlled secure and privacy-preserving EHRs sharing scheme based on consortium blockchain," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[12] R. M. Abdelmoneem, A. Benslimane, E. Shaaban, S. Abdelhamid, and S. Ghoneim, "A cloud-fog based architecture for IoT applications dedicated to healthcare," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[13] D. Giri, M. S. Obaidat, and T. Maitra, "SecHealth: An efficient fog based sender initiated secure data transmission of healthcare sensors for e-medical system," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.

[14] M. Min *et al.*, "Learning-based privacy-aware offloading for healthcare IoT with energy harvesting," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4307–4316, Jun. 2019.

[15] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 44–51.

[16] H. H. Elazhary and S. F. Sabbeh, "The W5 framework for computation offloading in the Internet of Things," *IEEE Access*, vol. 6, pp. 23883–23895, 2018.

[17] X. Li, X. Huang, C. Li, R. Yu, and L. Shu, "EdgeCare: Leveraging edge computing for collaborative data management in mobile healthcare systems," *IEEE Access*, vol. 7, pp. 22011–22025, 2019.

[18] P. C. M. Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "A trustworthy privacy preserving framework for machine learning in industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6092–6102, Sep. 2020.

[19] R. Kumar, N. Marchang, and R. Tripathi, "Distributed off-chain storage of patient diagnostic reports in healthcare system using IPFS and blockchain," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, 2020, pp. 1–5.

[20] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2017, pp. 1–6.

[21] T. Banerjee, M. Enayati, J. M. Keller, M. Skubic, M. Popescu, and M. Rantz, "Monitoring patients in hospital beds using unobtrusive depth sensors," in *Proc. 36th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2014, spp. 5904–5907.

[22] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2536–2549, Dec. 2020.

[23] I. Elgendy, W. Zhang, C. Liu, and C.-H. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Trans. Cloud Comput.*, early access, Jun. 18, 2018, doi: 10.1109/TCC.2018.2847347.

[24] B. S. Kumar, V. R. Raj, and A. Nair, "Comparative study on AES and RSA algorithm for medical images," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, 2017, pp. 0501–0504.

[25] K. Tamilarasi and A. Jawahar, "Medical data security for healthcare applications using hybrid lightweight encryption and swarm optimization algorithm," *Wireless Pers. Commun.*, vol. 114, pp. 1865–1886, Mar. 2020.

[26] S. K. ElKady and H. M. Abdelsalam, "A modified particle swarm optimization algorithm for solving capacitated maximal covering location problem in healthcare systems," in *Applications of Intelligent Optimization in Biology and Medicine*. Cham, Switzerland: Springer, 2016, pp. 117–133.

[27] *Hyperledger Fabric*. Accessed: Jun. 2020. [Online]. Available: https://www.hyperledger.org/use/fabric

[28] *Docker Software*. Accessed: Jun. 2020. [Online]. Available: https://www.docker.com/get-started

[29] *Interplanetary File System (IPFs)*. Accessed: Jun. 2020. [Online]. Available: https://docs.ipfs.io/

[30] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50759–50779, 2019.

[31] M. Wohrer and U. Zdun, "Smart contracts: Security patterns in the ethereum ecosystem and solidity," in *Proc. Int. Workshop Blockchain Orient. Softw. Eng. (IWBOSE)*, 2018, pp. 2–8.

[32] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, and M. K. Khan, "PoRX: A reputation incentive scheme for blockchain consensus of IIoT," *Future Gener. Comput. Syst.*, vol. 102, pp. 140–151, Jan. 2020.

[33] D. Phan, N. Nguyen, P. N. Pathirana, M. Horne, L. Power, and D. Szmulewicz, "A random forest approach for quantifying gait ataxia with truncal and peripheral measurements using multiple wearable sensors," *IEEE Sensors J.*, vol. 20, no. 2, pp. 723–734, Jan. 2020.

Dinh C. Nguyen (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Engineering, Deakin University, Geelong, VIC, Australia.

He is currently working on blockchain and reinforcement learning for Internet of Things and 5G networks. His research interests focus on blockchain, deep reinforcement learning, mobile edge/cloud computing, network security, and privacy.

Mr. Nguyen was a recipient of the prestigious Data61 Ph.D. Scholarship, CSIRO, Australia.

Pubudu N. Pathirana (Senior Member, IEEE) was born in 1970 in Matara, Sri Lanka, and was educated with Royal College Colombo, Colombo, Sri Lanka. He received the B.E. degree (First Class Hons.) in electrical engineering 1996, the B.Sc. degree in mathematics in 1996, and the Ph.D. degree in electrical engineering in 2000 from the University of Western Australia, Perth, WA, Australia, all sponsored by the government of Australia on EMSS and IPRS scholarships, respectively.

He was a Postdoctoral Research Fellow with Oxford University, Oxford, U.K., a Research Fellow with the School of Electrical Engineering and Telecommunications, University of New South Wales, Kensington, NSW, Australia, and a Consultant with the Defence Science and Technology Organization, Canberra, ACT, Australia, in 2002. He was a Visiting Professor with Yale University, New Haven, CT, USA, in 2009. He is currently a Full Professor and the Director of NSC group with the School of Engineering, Deakin University, Geelong, VIC, Australia. His current research interests include bio-medical assistive device design, human motion capture, mobile/wireless networks, rehabilitation robotics, and radar array signal processing.

Ming Ding (Senior Member, IEEE) received the B.S. and M.S. degrees (First-Class Hons.) in electronics engineering, and the Doctor of Philosophy (Ph.D.) degree in signal and information processing from Shanghai Jiao Tong University, Shanghai, China, in 2004, 2007, and 2011, respectively.

From April 2007 to September 2014, he worked with the Sharp Laboratories of China, Shanghai, as a Researcher/Senior Researcher/Principal Researcher. He is currently a Senior Research Scientist with Data61, CSIRO, Sydney, NSW, Australia. He has authored over 140 papers in IEEE journals and conferences, all in recognized venues, and around 20 3GPP standardization contributions, as well as a Springer book *Multipoint Cooperative Communication Systems: Theory and Applications*. Also, he holds 21 U.S. patents and co-invented another 100+ patents on 4G/5G technologies in CN, JP, KR, and EU. His research interests include information technology, data privacy and security, machine learning, and AI.

Dr. Ding is an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE WIRELESS COMMUNICATIONS LETTERS. He has served as a Guest Editor/Co-Chair/Co-Tutor/TPC member of many IEEE top-tier journals/conferences and received several awards of his research work and professional services.

Aruna Seneviratne (Senior Member, IEEE) received the B.Sc. (Hons) degree from Middlesex Polytechnic, London, U.K., and the Ph.D. degree from the University of Bath, Bath, U.K. He is currently a Foundation Professor of telecommunications with the University of New South Wales, Kensington, NSW, Australia, where he holds the Mahanakorn Chair of Telecommunications. He has also worked with a number of other Universities in Australia, U.K., and France, and industrial organizations, including Muirhead, Standard Telecommunication Laboratories, Avaya Labs, and Telecom Australia (Telstra). In addition, he has held visiting appointments with INRIA, Paris, France. Most recently, his team has been working on using these technologies in behavioral biometrics, optimizing the performance of wearables, and the IoT system verification. His current research interests are in physical analytics: technologies that enable applications to interact intelligently and securely with their environment in real time.

Dr. Seneviratne was awarded a number of fellowships, including one at British Telecom and one at Telecom Australia Research Laboratories.